Multi-Objective Optimization for Road Vertical Alignment Design

by

Ayazhan Akhmet

B.Sc., Nazarbayev University, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

June 2021

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

Multi-Objective Optimization for Road Vertical Alignment Design

submitted by Ayazhan Akhmet in partial fulfilment of the requirements of the degree of Master of Science

Dr. Warren Hare, Irving K. Barber School of Arts and Science
Supervisor

Dr. Yves Lucet, Irving K. Barber School of Arts and Science
Supervisory Committee Member

Dr. Solomon Tesfamariam, School of Engineering
Supervisory Committee Member

Dr. Amir Ardestani-Jaafari, Faculty of Management
University Examiner

Abstract

This thesis defines a multi-objective optimization model that seeks to find road profiles that would be optimal for the manufacturers in terms of the road construction cost and at the same time for the users in terms of the vehicle operating costs, specifically in terms of the fuel consumption cost. The research implements and validates the formula for the fuel consumption cost. It further presents and examines a variety of well-known methods: three classical scalarization techniques (the ε-constraint method, weighted sum method, and weighted metric methods) and two widely-used evolutionary methods (NSGA-II and FP-NSGA-II). Moreover, to accelerate the performance of the chosen scalarization approaches, a warm start strategy is proposed.

Numerical experiments are performed on 30 road samples for Caterpillar 793D off-highway trucks to determine the most robust approach for the proposed multi-objective optimization problem. The results are analyzed using the commonly-used performance indicators, namely, hypervolume (to assess the convergence of solutions), spacing (to assess the diversity of solutions), and CPU time (to assess the speed).

The research determines that the most promising and recommended method for the proposed problem is the ε-constraint method (successfully solved approximately 75% of test problems) followed by the weighted sum method (successfully solved approximately 50% of test problems). Moreover, the research shows that the warm start strategy improves the performance of the scalarization techniques.

Lay Summary

A high-quality road infrastructure is important in ensuring the economic growth of the country. But the expenses for designing new roads are not cheap. Along with large capital investments for road construction, roads also incur future user costs. In this thesis, we present an optimization framework, that takes into account the cost of building the road and the cost of using the road, as well as strategies for solving it. Our experiments determine that the most recommended method for our problem is the ε-constraint method with a warm start.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I take this opportunity to sincerely thank everyone who supported and guided me in this journey.

First and foremost, I express a deep sense of gratitude and appreciation to my supervisor, Dr. Warren Hare, for leading me throughout the course of this research. This work has not been realized without his constant encouragement, invaluable advice, and cordial support.

Acknowledgment is extended to my graduate committee member, Dr. Yves Lucet, for giving detailed feedback and wholesome insights. The course of numerical optimization that was taught by him broadened my knowledge and established a solid foundation for my research.

I am very grateful to my graduate committee member, Dr. Solomon Tesfamariam, for giving constructive reviews that put me on a right track at the start of my thesis.

Special thanks to our industrial partner, Softree Technical Systems Inc., for supplying us with practical datasets, as well as with insightful feedback.

I gratefully acknowledge the funding provided by The University of British Columbia (UBC) and the Natural Sciences and Engineering Research Council (NSERC).

Deserving a special mention, I express my sincere appreciation to the Computer-Aided Convex Analysis (CA^2) laboratory for providing persistent access to computers and services.

I owe a great debt of gratitude to my family and my friends for their love, faith, and understanding. Especially, I would like to thank my labmates, Nusrat Suzana Momo and Khandoker Md Ayman, for sharing great ideas and having happy times together as graduate students.

Above all, I thank God for giving all grace, good health, and strength to complete this work.

Dedication

Dedicated to the individuals whom you can tell your night dreams.

Chapter 1

Introduction

1.1 Motivation and objective

The era of road design began about 2000 years ago when the Romans initiated the development of techniques for building durable roads to ease the mobility of armies within their empire [Abr13]. Since then, the road design process has been influenced by the idea that better roads result in lower road user costs. Well-designed roads increased the probability that the Roman armies would reach their destinations with lower fuel (analogously, food and water supplies), lower equipment repair costs, fewer accidents, as well as in less time. Today, the goal of road design is still analogous.

Even though we attempt to serve several objectives while constructing a new road, we often consider only the minimization of the road construction cost and neglect the basic importance of user costs affecting the country's economy. This is similar to how people often choose what to eat and drink without considering the simple nutritional benefits of food options. Therefore, we should increase consideration of the road user costs while planning a road.

According to the most recent report for the Canadian Automobile Association (CAA), the poor roads for Canada results in higher vehicle operating costs of about $3 billion per year [CPC21]. The statistics by the provinces are given in Figure 1.1. These expenses encompass several cost components such as fuel consumption, tire usage, depreciation, repair, and maintenance of the vehicle. Among all these costs, the fuel consumption cost is the major component that is sensitive to road slope, rolling resistance, and air drag. This cost is very significant for heavy trucks that are used in forestry or mining, which is our main consideration in this research.

This thesis defines a multi-objective optimization model for designing a road that would be optimal for the manufacturers in terms of the road construction cost and at the same time for the users in terms of the vehicle operating costs, specifically in terms of the fuel consumption cost. It further presents and examines a variety of methods for solving this multi-objective optimization problem.

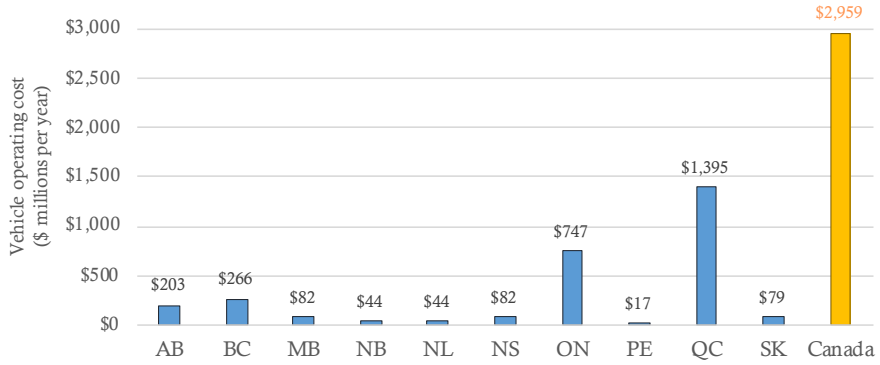


Figure 1.1: Statistics showing the annual vehicle operating cost due to poor roads, by Canadian provinces and territories ($ millions per year) [CPC21]. Note that these costs are not normalized by the population.

One may wonder why we focus on the multi-objective optimization model instead of defining a single-objective optimization problem that aggregates both objective functions into one scalar function. Firstly, as the fuel price varies from region to region and varies over the lifetime of the mine, we decide to focus on the fuel amount used in mL, whereas the road construction cost is estimated in dollars because it is a short-term event. Secondly, the total fuel consumption cost also depends on the lifetime usage of the road which is information that may not be available, for example, in a new mine where the exact amount of minerals are unknown. Therefore, for our problem, these two cost functions cannot be added together into a single-objective problem.

1.2 Research approach

Generally, the road design procedure can be considered as finding an optimal road geometry for a given ground profile. Due to computational barriers, many studies split this process into horizontal alignment and vertical alignment [MR12, BHLH17]. In this thesis, we assume that the horizontal alignment is predetermined, and we consider only the vertical alignment design that follows this fixed horizontal alignment. We optimize the vertical alignment by performing optimal earthwork operations to construct an optimal road under some design requirements.

In our research, we examine how road vertical alignment affects the road

user costs, particularly the fuel consumption cost, and we model a multi-objective approach that takes in both the cost of road construction and the cost of road usage.

In any optimization problem, a decision-maker first needs to identify the core decision variables that will impact the overall quality of decisions. The objective functions represent this quality and are then used to make the most suitable decision. Thus, we first present the functional form of the road construction cost and road user cost in the context of road vertical alignment design.

Then, we review and examine several well-known multi-objective optimization methods. We also present an approach (called warm start strategy) for our problem that can improve the performance of some chosen methods. We test the performance of these selected methods on 30 road samples and attempt to determine the most promising approach for our model.

1.3 Thesis outline

Chapter 2 serves as a brief introduction to the road vertical alignment design followed by the key concepts of multi-objective optimization necessary for this research.

Chapter 3 formulates the functional forms of the road construction cost and road user costs for our multi-objective optimization model in the context of road vertical alignment design.

Chapter 4 reviews several approaches for solving the proposed multi-objective problem. It provides brief descriptions of the algorithms along with their pseudocodes.

Chapter 5 describes the experimental framework performed to assess the performance of the selected methods and presents an analysis of the numerical results. It examines the most promising and appropriate method for our proposed model.

The final chapter, Chapter 6, summarizes all the main points of the study and highlights some suggestions for possible future work.

Chapter 2

Preliminaries

In this chapter, we review some essential preliminary material to set the foundation of this research.

In Section 2.1, we present a brief introduction to the road vertical alignment design, and in Section 2.2, we summarize the basic concepts of multi-objective optimization.

2.1 Vertical alignment design

Definition 2.1 (Ground profile and road profile [MR12])**.** The *ground profile* is the vertical profile of the ground prior to the construction of the road. The *road profile* is the vertical profile of the ground after the construction of the road.

Whereas the horizontal alignment design examines the projection of the three-dimensional ground profile on a horizontal plane, in the vertical alignment design we focus on the projection on a vertical plane. In other words, the vertical alignment is a longitudinal cross-section of the horizontal alignment. It is used to determine the pavement elevation, road grade, and other road characteristics.

To model the vertical alignment, we split a road into smaller units known as *sections*. We index them by the set $\mathcal{S} = \{1, 2, \ldots, m\}$. We represent the road with a *spline* which is defined as a piecewise quadratic function given in the following form

$$P_i(x) = a_{i,1} + a_{i,2}x + a_{i,3}x^2, \tag{2.1}$$

where $a_{i,1}, a_{i,2}, a_{i,3}$ are the coefficients of the quadratic polynomial spanning along the section $i \in \mathcal{S}$. Each polynomial piece is referred as a *segment* (also known as a *spline segment*).

For instance, Figure 2.1 illustrates a graphical example of the vertical alignment in profile view. The y-axis shows the ground elevation (or height of the ground), while the x-axis represents the horizontal distance of the road in meters. In this example, the road is divided into 5 sections and is

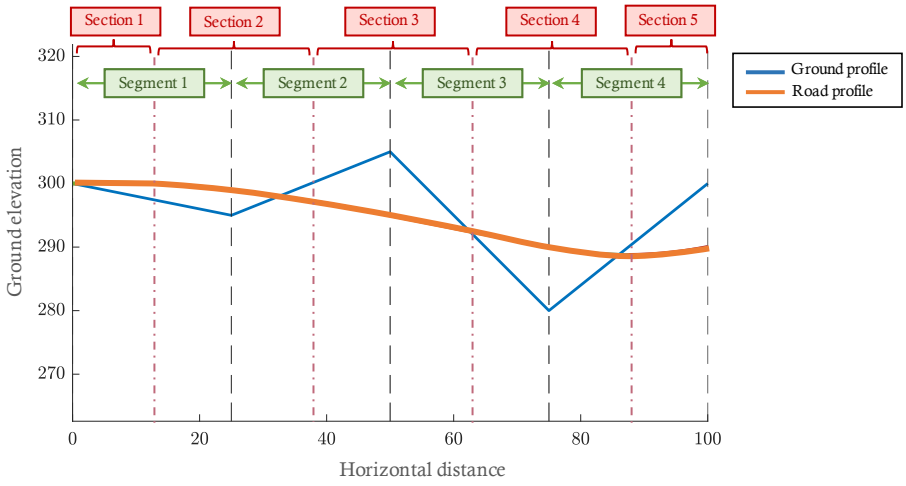


Figure 2.1: Graphical illustration of vertical alignment.

approximated using 4 (spline) segments. The ground profile and road profile are given with the blue and orange lines, respectively. The road profile is constructed by changing the height of the ground elevation at different points and by moving the earth between sections.

In other words, the input of the vertical alignment design is the ground profile, and the output is the road profile provided as a quadratic spline. Note that the vertical alignment is determined for the fixed horizontal alignment.

For further details on the vertical alignment design, we refer to [MR12, BHLH17].

2.2 Multi-objective optimization

Many real-life optimization problems in the agricultural, construction, and manufacturing sectors involve multiple objectives. These objectives are often conflicting in nature in a way that a solution that minimizes one of the objectives might worsen another one. Therefore, all objectives need to be handled at the same time. This creates the field of *Multi-Objective Optimization (MOO)*.

2.2.1 Problem formulation

We consider a MOO problem given in the form

$$\begin{aligned} &\text{minimize} && \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{subject to} && \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a decision variable, $f_j : \mathbb{R}^n \to \mathbb{R}$ for $j \in \mathcal{J} = \{1, ..., k\}$ with $k \geq 2$ are objective functions, and $\mathcal{X} \subset \mathbb{R}^n$ is called a *feasible set*. The image of the feasible set is called the *objective function space*.

2.2.2 Pareto optimality

In MOO, we need to select the 'best' solutions from the domain of feasible solutions. Thus, we need the concept of Pareto dominance to say which solution is better than others.

Definition 2.2 (Pareto dominance [JMC09])**.** Consider the problem (2.2). Suppose that $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Then we say that $\mathbf{x}$ *Pareto dominates* $\mathbf{y}$, denoted by $\mathbf{x} \prec \mathbf{y}$, if

1. $f_j(\mathbf{x}) \leq f_j(\mathbf{y})$ for all $j \in \mathcal{J}$, and
2. $f_j(\mathbf{x}) < f_j(\mathbf{y})$ for at least one $j \in \mathcal{J}$.

Example 2.3. Suppose that we have $\mathbf{f} = [f_1, f_2]$ where f_1 and f_2 are the objective functions that are being minimized. And suppose that we have four solutions $a, b, c,$ and d, as illustrated in Figure 2.2. Here, c minimizes

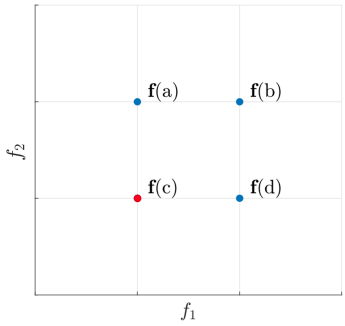


Figure 2.2: Pareto dominance example with four solutions $a, b, c,$ and d, where a is a non-dominated solution (red point) and $a, b,$ and d are dominated solutions (blue points).

both f_1 and f_2 better than b. Thus, both cases in Definition 2.2 hold, and $c \prec b$. Also, $c \prec a$, since c minimizes f_2 better than a and minimizes f_1 as good as a. In similar way, $c \prec d$, $a \prec b$, and $d \prec b$, while a and d are incomparable according to the definition. Among all these four solutions, there is no solution that Pareto dominates the solution c. Therefore, we refer to c as a non-dominated solution, whereas $a, b,$ and d are dominated ones.

We can now define the concept of solution in the multi-objective optimization context using the Pareto dominance relation.

Definition 2.4 (Pareto optimal set [JMC09, ABC$^+$20])**.** For a set of solutions $\mathcal{P} \subset \mathcal{X}$ of the problem (2.2), *Pareto optimal set* $\mathcal{P}^*$ is a set of solutions that are not Pareto dominated by any other solution of $\mathcal{P}$, that is,

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{P} \mid \nexists \mathbf{y} \in \mathcal{P} : \mathbf{y} \prec \mathbf{x}\}.$$

Definition 2.5 (Pareto front [JMC09, ABC$^+$20])**.** For the Pareto optimal set $\mathcal{P}^*$ of the problem (2.2), *Pareto front* $\mathcal{PF}^*$ is the image of the solutions of the set $\mathcal{P}^*$, that is,

$$\mathcal{PF}^* = \Big\{ \big(f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})\big) \mid \mathbf{x} \in \mathcal{P}^* \Big\}.$$

Instead of finding a single optimal solution as in a single-objective case, the aim of MOO is to estimate a Pareto optimal set that makes all the objectives have as good outcomes as possible. It attempts to converge closer to the true Pareto optimal set and to discover a diverse set of solutions that are well-distributed among themselves.

In the next definition, we define some bounds of the Pareto front $\mathcal{PF}^*$.

Definition 2.6 (Ideal point and nadir point [JMC09])**.** For the Pareto optimal set $\mathcal{P}^*$ of the problem (2.2), the *ideal point* $z^* = (z_1^*, \ldots, z_k^*)^T$ of the Pareto front $\mathcal{PF}^*$ is defined as

$$z_i^* = \min_{\mathbf{x} \in \mathcal{P}^*} f_i(\mathbf{x})$$

for $i = \{1, 2, \ldots, k\}$. Similarly, the *nadir point* $z^{nad} = (z_1^{nad}, \ldots, z_k^{nad})^T$ of the Pareto front $\mathcal{PF}^*$ is defined as

$$z_i^{nad} = \max_{\mathbf{x} \in \mathcal{P}^*} f_i(\mathbf{x})$$

for $i = \{1, 2, \ldots, k\}$.

For instance, Figure 2.3 illustrates the Pareto front for the case of two objectives f_1 and f_2. The ideal point and nadir point are given with blue and red dots, respectively. The ideal point represents the simplest point that Pareto dominates the entire Pareto front, whereas the nadir point represents the simplest point that is Pareto dominated by the entire Pareto front. These two points give a rectangular region that contains the entire Pareto front.

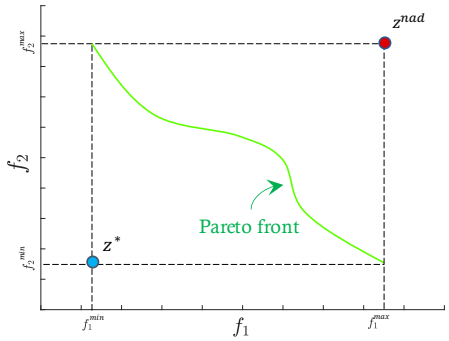


Figure 2.3: Graphical illustration of the ideal and nadir points of the Pareto front for bi-objective problem.

The notion of the ideal point is used in the descriptions of some selected MOO methods in Chapter 4, whereas the nadir point is used when computing the hypervolume indicator in Chapter 5.

2.2.3 Pareto front filtering

Algorithm 2.1 outlines the pseudocode for identifying the Pareto optimal solutions for the given solution set $\mathcal{P}$.

The main idea of the algorithm is to classify the solutions into two categories, namely, dominated and non-dominated. The set $\mathcal{I}$ saves the labels for each solution of the set $\mathcal{P}$ in a way that it labels the dominated solutions with 0 and the non-dominated ones with 1.

At the beginning of the algorithm, we assume that all the elements of the set $\mathcal{P}$ are non-dominated. It starts with the first element of the set $\mathcal{P}$, and at Steps 5-6 it identifies the solutions that are dominated by the selected element and marks them with a label 0. Then, at Step 7, it moves to the next element of the set $\mathcal{P}$ that has a label 1 and repeats the process. In other words, the algorithm selects the next element which is a 'potential' non-dominated solution and identifies the solutions that it Pareto dominates. In this way, in each iteration, we ignore the already found dominated solutions

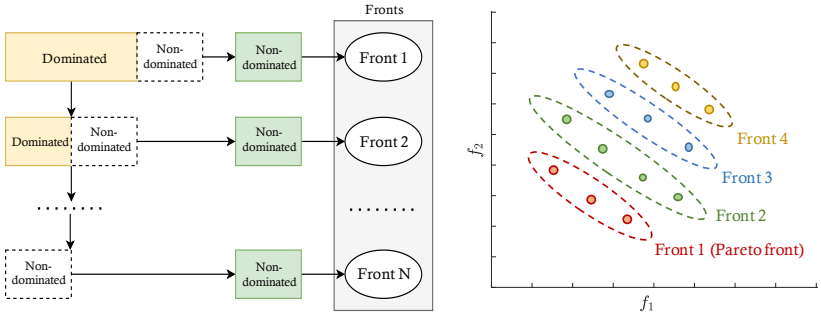
Algorithm 2.1: Pareto optimal set filtering

Input : Set of solutions $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$
Output: Pareto optimal set $\mathcal{P}^*$

```
1) Declare a set I of size n
2) I = 1
3) i = 1
4) while i ≤ n do
5)     for j such that I(j) = 1 do
6)         if j ≠ i and p_i ≺ p_j then I(j) = 0
       end
7)     Increment i until I(i) ≠ 0
   end
8) P* = P(I)
```

and perform comparisons among the 'potential' non-dominated solutions. The algorithm stops when it reaches the last element of the set $\mathcal{P}$, and it returns the set of non-dominated solutions, that is, Pareto optimal set $\mathcal{P}^*$.

By applying Algorithm 2.1 on the set $\mathcal{P}$, we find the Pareto optimal set composing the so-called Front 1. By removing the solutions of Front 1 from the set $\mathcal{P}$ and applying Algorithm 2.1 on the new set, we identify Front 2, and so on. This strategy is called a *non-dominated sorting* and is illustrated in Figure 2.4(a) [DPAM02]. Figure 2.4(b) shows an example of fronts for



(a) Graphical illustration of non-dominated sorting.

(b) Graphical illustration of the fronts.

Figure 2.4: Graphical illustration of non-dominated sorting and fronts.

bi-objective problem.

The concept of the non-dominated sorting is used later in Chapter 4 while describing the genetic algorithms selected for this thesis.

Chapter 3

Model

In this chapter, we present the objective functions of our MOO model, namely, road construction cost in Section 3.1 and vehicle operating cost in Section 3.2.

3.1 Road construction cost

In the road vertical alignment design, the aim of the road construction process is to transform the given ground profile into a road profile using earthwork operations. Thus, the effect of earthwork cost on the road construction cost is significant [Fwa89]. The earthwork cost consists of the following three components.

1. *Excavation cost*, denoted by C_{cut}, is the cost of cutting earth from a section;
2. *Hauling cost*, denoted by C_{haul}, is the cost of transporting earth between sections;
3. *Embankment cost*, denoted by C_{fill}, is the cost of filling a section with earth.

In this thesis, the road construction cost is retrieved from the Mixed-Integer Linear Programming (MILP) model with a single material pavement proposed by [MR12, BHLH17]. This cost function is considered as a black-box, that is, the internal structure and code implementation of the cost function is not known. Below we present the general functional form for the road construction cost.

3.1.1 Variables and parameters

Decision variables

A *decision variable* is a variable whose optimal value is determined during the optimization process. The decision variables in the road construction problem that are relevant to this research are listed below.

– $a_{i,1}, a_{i,2}, a_{i,3} \in \mathbb{R}$: the coefficients of the quadratic polynomial defining the spline spanning along section $i \in \mathcal{S}$.

Design parameters

A *design parameter* is a predefined parameter used as an input for the optimization problem. The design parameters are listed below.

– $p_i \in \mathbb{R}^+$: the cost per cubic unit of earth that is excavated from section $i \in \mathcal{S}$.

– $c_{ij} \in \mathbb{R}^+$: the cost per cubic unit of earth that is hauled from section $i \in \mathcal{S}$ to section $j \in \mathcal{S}$ $(j \neq i)$.

– $q_i \in \mathbb{R}^+$: the cost per cubic unit of earth that is embanked to section $i \in \mathcal{S}$.

3.1.2 Formula

The road construction cost, denoted by $C_{construct}$, includes the total cost of excavation, hauling, and embankment, as follows.

$$C_{construct} = C_{cut} + C_{haul} + C_{fill}, \tag{3.1}$$

It estimates the total earthwork cost performed to construct a new road. For more details on the road construction cost, we refer to [MR12, BHLH17].

3.2 Vehicle operating cost

Vehicle operating costs consist of the daily costs spent by the users while operating vehicles on a given road profile. Fuel consumption, oil usage, tire wear, repairs, and maintenance of the vehicle are all examples of vehicle operating costs that differ with vehicle use [TV03]. Among them, fuel consumption is the component that is most impacted by the vertical alignment of the road, thus, is chosen as the second objective function of the model [JSJ06, TV03].

3.2.1 Literature review

Our goal is to derive an equation to quantify the fuel consumption cost based on road alignment. However, there is no clear relationship between

fuel consumption and road alignment. A number of studies have been conducted to identify the effect of various parameters on fuel consumption. We begin by discussing some of the previous methods to estimate fuel consumption.

In 1987, it was proposed to estimate fuel consumption using only one variable, namely, vehicle running speed [JSJ06]. After conducting several experiments, Jha [JSJ06] advanced the equation by including the road grade in calculations. Moreover, the author introduced a model for predicting fuel usage for the entire year taking into account the interest rate affecting the fuel price.

Akçelik et. al. [ASB12] proposed the model for estimating the amount of fuel consumed by taking into consideration the physics laws. It relies on the engine-generated tractive force, which is required to drive the vehicle, and considers the main external resistance forces affecting the vehicle. The model is analyzed using the software called SIDRA INTERSECTION and is calibrated for different vehicle parameters [ASB12]. The user can estimate the fuel consumption cost for any vehicle type by providing the model with the needed vehicle parameters.

In 2013, Swedish National Road and Transport Research Institute proposed a model for estimating the fuel consumption cost generated with a help of simulation software called VETO [CHE13]. Carlson et. al. [CHE13] simulated the traffic flow of different vehicle types for a systematic variation of roads and speed conditions. By generating a dataset, the influence of road variables on fuel usage was analyzed to produce the regression model for the fuel consumption cost. This model has a high degree of explanation, with $R^2 = 0.99$. However, this model is overly complex and is only valid for specific vehicle categories [CHE13].

Another model created based on the collected data is introduced by Kubler [Kub15]. Kubler [Kub15] collected the data for fuel usage in various environmental settings for Caterpillar 785D off-highway trucks. The uniqueness of this model is that it depends only on one variable, namely on road grade. Thus, the model is very simple with the high coefficient of determination of $R^2 = 0.89$, and it considers the fuel usage for loaded trucks and unloaded trucks separately. Nevertheless, as the authors mentioned, this regression model is valid only for the Caterpillar 785D off-highway trucks [Kub15].

Soofastaei et. al. [SAKK16] outlines the model to evaluate an approximate amount of fuel burnt by a haul truck to move one tonne of mined material in an hour. According to their model, the fuel consumption cost depends on the vehicle rimpull, which is defined as an amount of force ex-

erted by tires to the road surface. As the authors mentioned, the vehicle rimpull can be estimated using the so-called Rimpull Table which is unique for all vehicle types [SAKK16].

Another regression model with a coefficient of determination $R^2 = 0.84$ is outlined by Svenson and Fjeld [SF16]. They conducted an experiment with a test truck to collect data for a different road grade, road curvature, surface roughness, and vehicle payloads in dry summer and wet fall weathers [SF16]. Then, the regression analysis was performed on the dataset to identify the relationship between the fuel consumption cost and road parameters. Nevertheless, the model is highly dependent on the surface roughness which is not easily accessed in real life.

Based on the investigations of different approaches to estimate fuel consumption cost, this thesis uses the model proposed by [ASB12] that can be adjusted for different environmental settings and can be applied for different vehicle types. Using this model, the user can estimate the fuel usage by providing all the needed vehicle parameters with the environmental factors affecting the vehicle.

3.2.2 Variables and parameters

Decision variables

The decision variables defining the fuel consumption cost are listed below.

- $a_i \in \mathbb{R}$: the engine-induced acceleration ($\mathrm{m/s^2}$) for section $i \in \mathcal{S}$.
- $s_i \in \mathbb{R}$: the road grade or slope ($\mathrm{m/100\,m}$) of the spline segment spanning along section $i \in \mathcal{S}$.
- $v_i \in \mathbb{R}^+$: the vehicle running speed (m/s) for section $i \in \mathcal{S}$.

Design parameters

The design parameters required for the vehicle operating cost can be classified into two categories: vehicle parameters and environment parameters.

Vehicle parameters are the predefined characteristics of the vehicle, such as vehicle mass, horsepower, idle fuel rate, and others. The vehicle parameters are listed below.

- $A \in \mathbb{R}^+$: the projected frontal area ($\mathrm{m^2}$) of the vehicle.

- $M \in \mathbb{R}^+$: the vehicle total mass (kg).
- $P_e \in \mathbb{R}^+$: the vehicle engine power (kW).
- $p_{tract} \in \mathbb{R}^+$: the percent mass acting on tractive axle (%) of the vehicle.
- $\alpha \in \mathbb{R}^+$: the constant idle fuel rate (mL/s) of the vehicle.
- $\beta_1 \in \mathbb{R}^+$: the fuel usage per unit of energy (mL/kJ) (also called as the energy efficiency parameter).
- $\beta_2 \in \mathbb{R}^+$: the fuel usage per unit of energy-acceleration (also called as the energy-acceleration efficiency parameter) (mL/(kJ.m/s^2)).
- $\eta \in \mathbb{R}^+$: the power transmission efficiency (%) of the vehicle.

Environment parameters refer to the road resistance, gravity, air density, and other environmental factors affecting the vehicle motion. The environment parameters are listed below.

- $c_{air} \in \mathbb{R}^+$: the air drag coefficient.
- $c_{roll} \in \mathbb{R}^+$: the rolling coefficient depending on the road surface type.
- $c_{tire1}, c_{tire2} \in \mathbb{R}^+$: the rolling coefficient depending on vehicle's tire type such as bias ply or radial.
- $g \in \mathbb{R}^+$: the acceleration due to gravity (m/s^2).
- $r \in \mathbb{R}^+$: the road resistance (%).
- $\Delta d_i \in \mathbb{R}^+$: the distance or length (m) of section $i \in \mathcal{S}$.
- $\mu \in \mathbb{R}^+$: the coefficient of friction between tires and pavement.
- $\rho_{air} \in \mathbb{R}^+$: the air density (kg/m^3).

3.2.3 Formula

The paper by [ASB12] proposed the model for estimating the value of fuel consumed (mL) for a simulation distance Δd_i for section $i \in \mathcal{S}$. The model has the following three components.

1. *Fuel consumption due to idling*, denoted by $C_{idle,i}$, is an estimated fuel amount used to maintain engine operation at constant idle fuel rate α for section $i \in \mathcal{S}$;

2. *Fuel consumption due to tractive force*, denoted by $C_{tract,i}$, is an estimated fuel usage to increase tractive force $R_{tract,i}$ that moves the vehicle along section $i \in \mathcal{S}$;

3. *Fuel consumption due to engine-induced acceleration*, denoted by $C_{accel,i}$, is an estimated fuel amount used by the engine to accelerate the vehicle along section $i \in \mathcal{S}$. Note that the engine-induced acceleration refers to the acceleration produced while pushing the gas pedal of the vehicle, and it ignores the acceleration produced by the environmental factors such as gravitational force.

For each section $i \in \mathcal{S}$, the cost function $C_{fuel,i}$ is defined as

$$C_{fuel,i} = \begin{cases} C_{idle,i} + C_{tract,i} + C_{accel,i} & \text{for } a_i > 0 \\ C_{idle,i} + C_{tract,i} & \text{for } a_i \leq 0 \end{cases} \tag{3.2}$$

where

$$C_{idle,i} = \alpha \, \Delta d_i / v_i, \tag{3.3}$$

$$C_{tract,i} = \beta_1 \, \Delta d_i \, R_{tract,i} / 1000, \tag{3.4}$$

$$C_{accel,i} = \beta_2 \, M \, \Delta d_i \, a_i^2 / 1000. \tag{3.5}$$

Note that the vehicle parameters α, β_1, and β_2 can be retrieved from [AB03, ASB12]. We provide sample values in Table A.1 and A.2 (in Appendix A).

For each section $i \in \mathcal{S}$, we first need to estimate four parameters, namely, vehicle running speed v_i, vehicle tractive force $R_{tract,i}$, engine-induced acceleration a_i, and road grade s_i.

Vehicle running speed

The vehicle running speed v_i is a rate at which the vehicle travels the section $i \in \mathcal{S}$. With the purpose of simplicity of the model, we assume that the vehicle running speed is constant and predefined by the user.

Vehicle tractive force

The vehicle tractive force $R_{tract,i}$ (N) is an amount of force required to drive the vehicle along the section $i \in \mathcal{S}$. It is directly proportional to the engine power P_e generated by the vehicle and is inversely proportional to the vehicle running speed v_i, as follows.

$$R_{tract,i} = \min \Big(\frac{3600 \, \eta \, P_e}{v_i}, \; R_{tract}^{max} \Big), \tag{3.6}$$

where R_{tract}^{max} is the constant maximum tractive force (N) which can be produced with no harm to the vehicle. Note that we include R_{tract}^{max} in Equation (3.6) to ensure that the vehicle tractive force does not approach infinity at low vehicle speeds [RLD+01]. Moreover, it can be quantified as

$$R_{tract}^{max} = M\, g\, \mu\, p_{tract}.$$

According to Equation (3.6), the vehicle tractive force is always positive. This implies that the vehicle is moving without any stops. So, this model does not consider the case when the vehicle is parked with the engine running. We ignore this case because it indeed does not affect the road geometry.

Engine-induced acceleration

To estimate the engine-induced acceleration, we assume that the vehicle tractive force $R_{tract,i}$ (N) is the sum of inertia force (N), air drag force $R_{air,i}$ (N), grade resistance force $R_{grade,i}$ (N), and rolling resistance $R_{roll,i}$ (N), as follows [RLD+01]

$$R_{tract,i} = M\, a_i + R_{air,i} + R_{grade,i} + R_{roll,i}, \tag{3.7}$$

where $R_{air,i} = c_{air}\, A\, \rho_{air}\, v_i^2/2,$
$R_{grade,i} = M\, g\, s_i,$
$R_{roll,i} = M\, g\, c_{roll}\, (c_{tire1} + 3.6\, c_{tire2}\, v_i)/1000.$

For more details on the external resistance forces $R_{air,i}$, $R_{grade,i}$ and $R_{roll,i}$, see [RLD+01].

By rearranging Equation (3.7), the engine-induced acceleration a_i is defined as

$$a_i = \frac{R_{tract,i} - R_{air,i} - R_{grade,i} - R_{roll,i}}{M}.$$

The engine-induced acceleration a_i is positive when the vehicle tractive force $R_{tract,i}$ is more than the external resistance forces. In other words, we increase the tractive force to overcome all external resistance forces exposed by nature to accelerate the vehicle. As a result, the engine induces an acceleration and uses more fuel. However, when the produced tractive force by the vehicle is less than the external resistance forces, the vehicle is decelerating, and so, no extra fuel is consumed by the engine. This process is visualized in Figure 3.1.

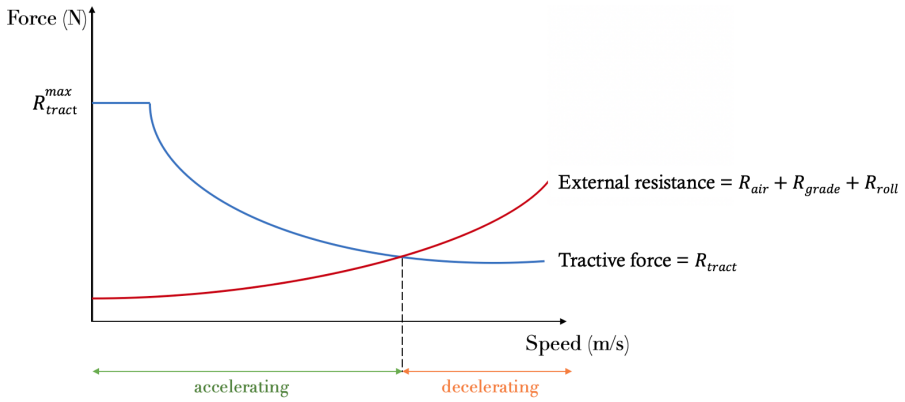


Figure 3.1: Graphical illustration of tractive force and external resistance forces.

Road grade

The road grade s_i is a slope of the spline segment spanning along the section $i \in \mathcal{S}$. Using Equation (2.1), the road grade s_i is

$$s_i(x) = \frac{\mathrm{d}P_i(x)}{\mathrm{d}x} = a_{i,2} + a_{i,3}x. \tag{3.8}$$

Total vehicle operating cost

Along the given vertical alignment, the vehicle can move in two directions as follows.

1. *Left-to-right traffic flow* is the vehicle motion from left to right along the given vertical alignment;

2. *Right-to-left traffic flow* is the vehicle motion from right to left along the given vertical alignment.

The road vertical alignment, which is optimal for one of the directions of the traffic flows, might be inefficient for the other direction of the traffic flow by consuming significantly more fuel. With this in mind, we need to consider both directions of the traffic flow at the same time while optimizing the total vehicle operating cost.

Suppose that for the given vertical alignment, we label the left-to-right traffic flow with sections $\mathcal{S} = \{1, 2, \ldots, n\}$, while for the right-to-left traffic flow we use its reverse order $\mathcal{S}' = \{n, n-1, \ldots, 1\}$. In addition, suppose

that the road grade is s_i for section $i \in \mathcal{S}$ in the left-to-right traffic flow. Then, in the right-to-left traffic flow, the road grade will be $-s_i$ for section $i \in \mathcal{S}$.

For instance, in Figure 3.2, we present a road profile with 5 sections and illustrate the road grades s_i for each section $i \in \{1, 2, \ldots, 5\}$ in two traffic flow directions.

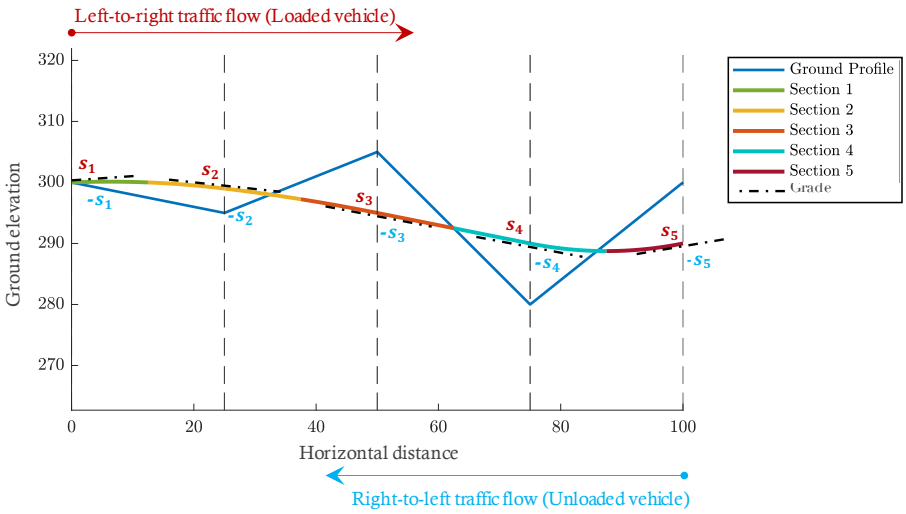


Figure 3.2: Graphical illustration of traffic flows for the road with 5 sections and grades s_1, s_2, s_3, s_4, s_5.

In this model, we assume that the leftmost endpoint of the vertical alignment (corresponding to Station 0) is a mining site or a forestry loading site. Thus, we force the ground profile and road profile to have the same ground elevations at the point associated with Station 0. With this in mind, we also assume that the vehicle is loaded when moving from left to right, but is unloaded while moving from right to left.

Ultimately, the total vehicle operating cost spent for traveling the whole alignment in both directions of the traffic flow is

$$C_{fuel} = \sum_{i \in \mathcal{S}} C_{fuel,i} + \sum_{i \in \mathcal{S}'} C_{fuel,i}. \tag{3.9}$$

Unit testing

In order to ensure that the code for our model was high-quality, the unit testing was done along the way. The unit testing included the following small roads:

1. Linear flat ground with 5 sections;
2. Linear inclined ground with 5 sections;
3. Linear declined ground with 5 sections;
4. Quadratic ground with a sag with 5 sections;
5. Quadratic ground with a crest with 5 sections;
6. Quadratic ground with a sag and crest with 5 sections;
7. Quadratic ground with a crest and sag with 5 sections.

Validation

The current model is validated on the dataset found in [SAKK16]. The dataset was collected from a surface coal mine with Caterpillar 793D off-highway trucks (CAT 793D). In the validation test, the vehicle parameters for CAT 793D are retrieved from the official website of Caterpillar manufacturer. As a result, the coefficient of determination for the proposed model is $R^2 = 0.93$ for this dataset.

Chapter 4

Methods

In this chapter, we discuss the MOO methods used in this thesis.

In Section 4.1, we give a brief survey on some MOO techniques. In Sections 4.2-4.5, we present the algorithm descriptions of the selected MOO methods followed by the warm start strategy in Section 4.6 that can improve the performance of some MOO methods.

4.1 Literature review

In the literature, there are plenty of MOO techniques, and we start by discussing some of the well-known approaches for solving MOO problems.

One of the widely used approaches is scalarization. Therein, the objective functions are merged (or reformulated as constraints) to form a single-objective optimization problem which is then solved by known methods. Below we first discuss some of the classical and commonly used scalarization techniques.

The weighted sum method is one of the well-known and classical scalarization approaches. The method combines all the objectives into a scalar single objective by pre-multiplying each objective function with predefined weights. A positive fact for this technique, which was first proposed by Gass and Satty [GS55], is its simplicity and easiness of implementation. However, the main limitation of the weighted sum method is the incapability of finding solutions in nonconvex regions of the Pareto front.

Goal programming is also one of the classical scalarization approaches, which was extended from linear programming and was firstly invented by Charnes and Cooper [CCF55, CC61]. The method does not attempt to optimize the objectives directly, instead, it declares the objective functions as the problem constraints restricted with the specific goals, and then it optimizes deviations from these goals. Even though the method is very simple and easy to implement, it highly relies on the choice of the goals which are predefined by the user. Similar approaches that use this idea are the goal attainment method, interactive goal programming, and lexicographical goal programming [JMC09].

The weighted metric method is another scalarization approach that is similar to the weighted sum method but uses another means of aggregation of the objective functions [Asg98, Deb01]. It seeks the nearest possible solution to a user-defined reference point, which is generally chosen as the ideal point, using either Manhattan, Euclidean, or Chebyshev metric. Depending on the metric type, the method can succeed in finding solutions in nonconvex parts of the Pareto front.

Benson's method is another example of a scalarization approach. It is similar to the weighted metric method but it chooses a reference point to be a feasible non-Pareto optimal solution instead of the ideal point, and then it aims to find the farthest solution from the selected reference point [Deb01]. This method can also uncover the nonconvex portion of the Pareto front. In addition, the method does not require the users to choose the reference point. Instead, it searches and selects the reference point on its own by defining some extra constraints that restrict the search region. Nevertheless, those extra constraints may complicate the optimization process [Deb01].

The ε-constraint method is also one of the commonly used scalarization techniques. This method was firstly proposed by Chankong and Haimes [CH83]. In this approach, one of the objectives is optimized while the others are converted into the constraints which are restricted by some allowable thresholds ε. The main advantage of this method is that it works for both convex and nonconvex Pareto fronts, and it can better approximate the Pareto front by varying ε values. On the other hand, in practice, it may be difficult to specify the values of ε. We purpose one method to select ε values in Subsection 4.2.2 herein.

The key benefit of all scalarization techniques is that once the given MOO problem has been reformulated as a single-objective optimization problem, we can apply all known theoretical results and existing numerical algorithms dedicated to this single-objective case. On the other hand, the scalarization approaches generally tend to find a single solution for each scalarization, thus, the quality of approximation of the Pareto front relies on the number of scalarizations performed.

The MOO approaches that can handle this drawback of scalarization are multi-objective evolutionary approaches. For instance, NSGA-II uses a genetic algorithm for producing a set of solutions in each iteration, instead of finding a single solution. In this way, the method attempts to approximate the entire Pareto front in a single run. There are a number of MOO methods integrated with evolutionary strategies, such as Vector Evaluated Genetic Algorithm (VEGA) [Sch85], Strength Pareto Evolutionary Algorithm 2^{nd} version (SPEA-II) [EML01], Non-Dominated Sorting Genetic Algorithm 2^{nd}

version (NSGA-II) [DPAM02], Non-Dominated Sorting Genetic Algorithm 3rd version (NSGA-III) [DJ13], and Front Prediction based Non-Dominated Sorting Genetic Algorithm 2nd version (FP-NSGA-II) [FTG15].

Another drawback of the scalarization approaches is related to the heterogeneous structure of the optimization problem. Consider the case when one of the objectives is a computationally expensive function that requires a lot of time to evaluate, while the other objectives are given analytically and are easy to assess. Then it would be beneficial to use this heterogeneous structure of the optimization problem. However, scalarization techniques may break the heterogeneous structure of the given problem, leaving us with a high computational effort that is mainly exposed by one expensive objective function.

To take advantage of the heterogeneity, Thomann and Eichfelder [TE19] introduced a trust-region algorithm for a multi-objective setting. The method employs a basic trust-region approach by defining a local region within which the model can be trusted and the objectives can be approximated with suitable models. In each iteration, the search direction is determined with a help of local ideal points [TE19]. Nevertheless, as the authors stated, this approach is developed for smooth MOO problems. Thus, it is not applicable for our proposed problem since the vehicle operating cost presented in Chapter 3 is nonsmooth. A similar approach for the case when all of the objectives are expensive functions is presented in [RK14], and another one for non-expensive objectives with Taylor models is proposed in [VOS14].

To handle the nonsmooth MOO problems, one can employ the multi-objective proximal bundle method introduced by Mäkelä et. al. [MKO14]. The main idea is to transform all the objectives and constraints into one objective using a so-called improvement function. Then, it solves a nonsmooth unconstrained single-objective optimization problem using the proximal bundle method [MKO14]. It finds a descent direction that improves all the objectives of the original problem at each iteration, and it gathers subgradients from the previous iterations into a bundle. In that way, it approximates the whole subdifferential instead of using only one arbitrary subgradient at each point. Recently, Mäkelä and Montonen [MM20] advanced the multi-objective proximal bundle method with a new version of the improvement function that can improve the algorithm performance. Nevertheless, as the authors stated, more numerical testing is needed.

In this thesis, as a starting point of the MOO for vertical alignment design, we focus on three classical scalarization methods (ε-constraint method, weighted sum method, and weighted metric method) and two multi-objective evolutionary methods (NSGA-II and FP-NSGA-II).

4.2 ε-constraint method

In the ε-constraint method, we optimize one of the objectives while we use the others as constraints that are bounded by some threshold ε. For simplicity, we consider a bi-objective case of the problem (2.2). We reformulate the given MOO problem to a single-objective optimization problem of the form

$$\begin{aligned} \text{minimize} \quad & f_2(\mathbf{x}) \\ \text{subject to} \quad & f_1(\mathbf{x}) \leq \varepsilon, \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{4.1}$$

In the above formulation, the value of ε needs to be in the range between some user-defined lower bound $\underline{f}$ and upper bound $\bar{f}$ of the objective f_1, that is, $\varepsilon \in [\underline{f}, \bar{f}]$. Since we aim to approximate the whole Pareto front, we choose $\underline{f} = \min_{\mathbf{x} \in \mathcal{X}} f_1(\mathbf{x})$ and $\bar{f} = f_1(\mathbf{x}^*)$ where $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x} \in \mathcal{X}} f_2(\mathbf{x})$.

In order to find a set of Pareto optimal solutions, we need to solve the problem (4.1) for different values of $\varepsilon \in [\underline{f}, \bar{f}]$. While the idea is straightforward, it raises a complex query. What values of ε should we consider to succeed in approximating the entire Pareto front? In the following subsections, we address this question.

4.2.1 Algorithm

Algorithm 4.1, inspired by [JMC09], outlines the pseudocode of the ε-constraint method for bi-objective optimization problem.

Algorithm 4.1: Bi-objective ε-constraint method

Input : Objectives $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$, objective bounds $\underline{f}, \bar{f} \in \mathbb{R}$, and increment $\delta \in \mathbb{R}^+$

Output: Set of solutions $\mathcal{P}$

1) Declare a matrix D with 4 columns
2) $\varepsilon = \underline{f}$
3) **while** $\varepsilon \leq \bar{f}$ **do**
4) Approximately solve the problem (4.1) with ε to find $\mathbf{x}_\varepsilon \in \mathbb{R}^n$
5) $D(i, :) = [\, \varepsilon,\ \mathbf{x}_\varepsilon,\ f_1(\mathbf{x}_\varepsilon),\ f_2(\mathbf{x}_\varepsilon)\,]$
6) $\varepsilon = \varepsilon + \delta$

end

7) $\mathcal{P} = D(:, 2)$

The general idea of the method is to solve the problem (4.1) for various values of ε which are iteratively increased by a predefined δ. Note that the choice of δ highly affects the performance of the algorithm. For instance, if δ is too small, the algorithm may converge to the same optimal solution multiple times, and thus, it may lead to a computational burden. On the other hand, if δ is too large, the algorithm may miss some Pareto-optimal solutions. The algorithm presented in the next subsection seeks to address this drawback.

4.2.2 Dynamic ε selection

In Algorithm 4.2, we provide the pseudocode for the bi-objective ε-constraint method with dynamic ε selection. The main idea of the dynamic ε selection is to update the ε value by analyzing the obtained solutions. This is done in the following way.

We create a matrix D to save all results found during the algorithm

Algorithm 4.2: Bi-objective ε-constraint method with dynamic ε selection

Input : Objectives $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$, objective bounds $\underline{f}, \bar{f} \in \mathbb{R}$, and tolerance $tol_\varepsilon \in \mathbb{R}^+$

Output: Set of solutions $\mathcal{P}$

1) Declare a matrix D with 4 columns
2) Approximately solve the problem (4.1) with $\varepsilon = \underline{f}$ to find $\mathbf{x}_\varepsilon \in \mathbb{R}^n$
3) $D(1, :) = [\, \varepsilon,\, \mathbf{x}_\varepsilon,\, f_1(\mathbf{x}_\varepsilon),\, f_2(\mathbf{x}_\varepsilon)\,]$
4) Approximately solve the problem (4.1) with $\varepsilon = \bar{f}$ to find $\mathbf{x}_\varepsilon \in \mathbb{R}^n$
5) $D(2, :) = [\, \varepsilon,\, \mathbf{x}_\varepsilon,\, f_1(\mathbf{x}_\varepsilon),\, f_2(\mathbf{x}_\varepsilon)\,]$
6) $i = 1$
7) **while not** *stopping criteria* **do**
8) $\quad \varepsilon = (D(i,1) + D(i+1,1))/2$
9) $\quad$ Approximately solve the problem (4.1) with ε to find $\mathbf{x}_\varepsilon \in \mathbb{R}^n$
10) $\quad D(end+1, :) = [\, \varepsilon,\, \mathbf{x}_\varepsilon,\, f_1(\mathbf{x}_\varepsilon),\, f_2(\mathbf{x}_\varepsilon)\,]$
11) $\quad$ Sort D in ascending order of $D(:, 1)$
12) $\quad$ Find i such that $norm(D(i, 3:4) - D(i+1, 3:4))$ is maximum and $D(i,1) - D(i+1,1) > tol_\varepsilon$
13) $\quad$ **if** $i = \varnothing$ **then** stop

end

14) $\mathcal{P} = D(:, 2)$

process: ε values and the corresponding solutions $\mathbf{x}_\varepsilon$ with the objective function values $f_1(\mathbf{x}_\varepsilon)$ and $f_2(\mathbf{x}_\varepsilon)$.

First, we need to determine the region where the true Pareto front lies using the objective bounds $\underline{f}$ and $\bar{f}$. We solve the problem (4.1) with the value of ε set to these user-defined objective bounds and record the results into the matrix D. We set $i = 1$ and define our search region to be the space between $D(i, 1)$ and $D(i + 1, 1)$.

At each iteration, we select ε in the middle of the search region. Then, we optimize the problem (4.1) with the selected ε and record the output into D. By performing these steps, in each iteration, we divide one search region into two 'potential' search regions from which we select the next ε.

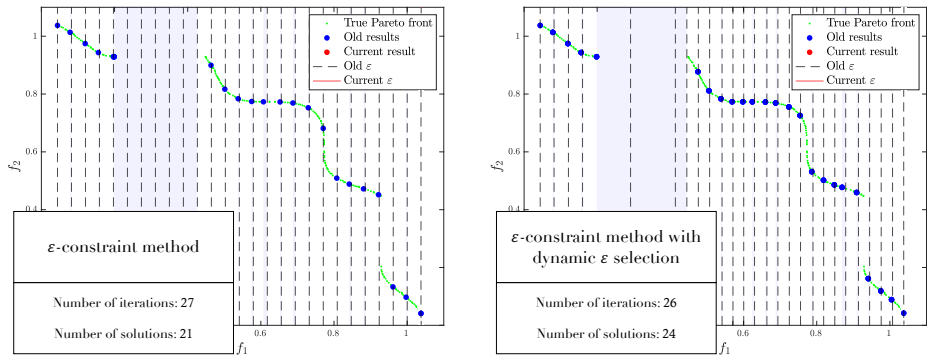
To identify the next search region, we sort the matrix D in ascending order of its column that stores the values of ε. Then, we find the biggest region i where no solution is found yet and which is greater than the predefined tolerance tol_ε. If no such region is found, then the algorithm terminates. For our research, we choose the tolerance tol_ε to be 5% of the region between the objective bounds $\underline{f}$ and $\bar{f}$.

We repeat this process until some stopping criteria are met. These stopping criteria are defined by the user. For example, it might be the maximum number of found solutions or some other tolerance restrictions.

To show the effect of the dynamic selection approach, we present an example in Figure 4.1 that illustrates the approximated Pareto fronts for the TNK test problem that is presented in [TWFT95]. It visualizes the results found at each iteration by the ε-constraint method (Figure 4.1(a)) and the ε-constraint method with dynamic ε selection (Figure 4.1(b)). The light green dots are the solutions of the true Pareto front, the red dot represents the value of ε from the current iteration, and the blue dots correspond to the values of ε from the previous iterations.

Watching Figure 4.1, the ε-constraint method spends several number of iterations searching in the region where no solutions of the true Pareto front exist. To put it in another way, the dynamic ε selection approach helps the algorithm to identify the regions where predominantly to search for solutions.

In this thesis, we use the ε-constraint method with the dynamic ε selection approach due to its effectiveness in the ε selection. In the coming chapters, for simplicity, we refer to this method as the ε-constraint method.



(a) ε-constraint method.

(b) ε-constraint method with dynamic ε selection.

Figure 4.1: TNK problem solved by ε-constraint method and ε-constraint method with dynamic ε selection.

4.3 Weighted sum method

We consider the problem given in (2.2) with k objectives. In the weighted sum approach, we aggregate all the objectives by pre-multiplying each objective f_i by user-defined weights w_i and form the following problem

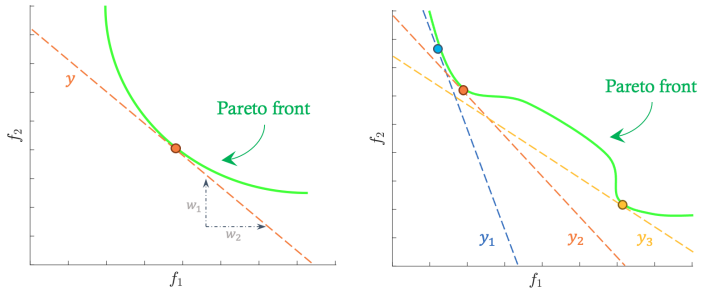
$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{k} w_i\, f_i(\mathbf{x}) \\ &\text{subject to} && \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{4.2}$$

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

As the ε-constraint method, this approach is very simple. To show how the weighted sum method performs to find a solution on the Pareto front, we consider a bi-objective case of the problem (2.2). Then, the weighted sum method aims to minimize

$$y = w_1\, f_1(\mathbf{x}) + w_2\, f_2(\mathbf{x})$$

subject to some given constraints. As y is a linear combination of the objectives f_1 and f_2, we can represent it in the objective function space as a straight line which has the slope defined by the weights w_1 and w_2 as



(a) Convex case. (b) Nonconvex case.

Figure 4.2: Graphical illustration of the weighted sum method for convex and nonconvex Pareto fronts.

shown in Figure 4.2(a). By varying the weight combinations, we can represent different y lines that touch the Pareto front, and for the convex Pareto front, we have enough room to quantify such points with different weights. However, for the nonconvex case presented in Figure 4.2(b), we can see that there are points of the nonconvex region of the Pareto front that cannot be reached for any combinations of the weight.

4.3.1 Algorithm

Algorithm 4.3 presents the pseudocode for the weighted metric method for the bi-objective optimization problem.

The algorithm idea is similar to the ε-constraint method. It finds a set of Pareto optimal solutions by iteratively updating the weights by a fixed increment δ. As before, this approach of choosing the weights may increase the computational time. To alleviate this drawback as in Subsection 4.2.2, we introduce the dynamic weight selection which assists the algorithm with the choice of weights. We discuss the proposed approach in the next subsection below.

4.3.2 Dynamic weight selection

Algorithm 4.4 outlines the pseudocode of the weighted sum method integrated with the dynamic weight selection approach. The main idea of the algorithm is the same as in Subsection 4.2.2.

Algorithm 4.3: Bi-objective weighted sum method

Input : Objectives $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ and increment $\delta \in (0, 1)$
Output: Set of solutions $\mathcal{P}$

1) Declare a matrix D with 4 columns
2) $w_1 = 0$ and $w_2 = 1 - w_1$
3) $\mathbf{w} = (w_1, w_2)^T$
4) **while** $w_1 \leq 1$ **do**
5) Approximately solve the problem (4.2) with $\mathbf{w}$ to find $\mathbf{x_w} \in \mathbb{R}^n$
6) $D(i, :) = [\, \mathbf{w},\, \mathbf{x_w},\, f_1(\mathbf{x_w}),\, f_2(\mathbf{x_w})\,]$
7) $w_1 = w_1 + \delta$ and $w_2 = 1 - w_1$
8) $\mathbf{w} = (w_1, w_2)^T$

end
9) $\mathcal{P} = D(:, 2)$

The procedure determines which weight combinations should be considered in order to cover a representative portion of the Pareto front. As previously, this is accomplished by archiving and analyzing all of the results obtained during the algorithm's execution. We record the weight vector $\mathbf{w}$, the corresponding solutions $\mathbf{x_w}$, and the objective function values $f_1(\mathbf{x_w})$ and $f_2(\mathbf{x_w})$.

Firstly, the method determines the bounds where each of the objectives is optimal using the weights $\mathbf{w} = (0, 1)^T$ and $\mathbf{w} = (1, 0)^T$. In this way, we identify the region where the true Pareto front lies. Then, it follows the same procedure presented in Subsection 4.2.2. It chooses the weights depending on the region where no solution is found yet.

The algorithm stops either when it completes the search for solutions or when it satisfies some stopping condition defined by the user. Note that for this thesis, we select the tolerance $tol_{\mathbf{w}}$ to be 5% of the allowable range for the weight $\mathbf{w}$, which are 0 and 1. In other words, we choose $tol_{\mathbf{w}}$ to be equal to 0.05.

Similar to the dynamic ε selection approach presented in Subsection 4.2.2, the proposed method can also perform better than the weighted sum method outlined in Algorithm 4.3. Thus, as before, we choose to use the weighted sum method with the dynamic weight selection due to its effectiveness in weight adjustments. In the coming chapters, we simply refer to this method as the weighted sum method.

Algorithm 4.4: Bi-objective weighted sum method with dynamic weight selection

Input : Objectives $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ and tolerance $tol_{\mathbf{w}} \in (0, 1)$
Output: Set of solutions $\mathcal{P}$

1) Declare a matrix D with 4 columns
2) Approximately solve the problem (4.2) with $\mathbf{w} = (0, 1)^T$ to find $\mathbf{x_w} \in \mathbb{R}^n$
3) $D(1, :) = [\, \mathbf{w},\, \mathbf{x_w},\, f_1(\mathbf{x_w}),\, f_2(\mathbf{x_w}) \,]$
4) Approximately solve the problem (4.2) with $\mathbf{w} = (1, 0)^T$ to find $\mathbf{x_w} \in \mathbb{R}^n$
5) $D(2, :) = [\, \mathbf{w},\, \mathbf{x_w},\, f_1(\mathbf{x_w}),\, f_2(\mathbf{x_w}) \,]$
6) $i = 1$
7) **while not** *stopping criteria* **do**
8) $\quad \mathbf{w} = (D(i, 1) + D(i + 1, 1))/2$
9) $\quad$ Approximately solve the problem (4.2) with $\mathbf{w}$ to find $\mathbf{x_w} \in \mathbb{R}^n$
10) $\quad D(end + 1, :) = [\, \mathbf{w},\, \mathbf{x_w},\, f_1(\mathbf{x_w}),\, f_2(\mathbf{x_w}) \,]$
11) $\quad$ Sort D in ascending order of $D(:, 1)$
12) $\quad$ Find i such that $norm(D(i, 3 : 4) - D(i + 1, 3 : 4))$ is maximum and $D(i, 1) - D(i + 1, 1) > tol_{\mathbf{w}}$
13) $\quad$ **if** $i = \varnothing$ **then** stop
end
14) $\mathcal{P} = D(:, 2)$

4.4 Weighted metric method

We consider the problem (2.2) with k objectives. In the weighted metric method, we aim to find the closest feasible solution to an ideal point z^* using the l_p-metric for $p \in [\, 1, \infty \,]$. For $p \in [\, 1, \infty \,)$, this is done by solving a single-objective optimization problem of the form

$$\begin{aligned} &\text{minimize} \quad \left(\sum_{i=1}^{k} w_i \, |f_i(\mathbf{x}) - z_i^*|^p \right)^{1/p} \\ &\text{subject to} \quad \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{4.3}$$

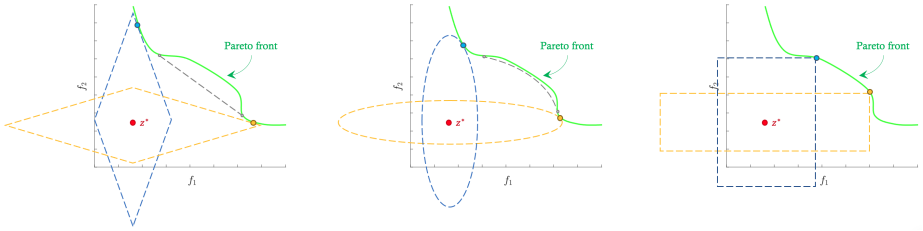
and for $p = \infty$, by solving the problem of the form

$$\begin{aligned} &\text{minimize} \quad \max_{i=\{1,\ldots,k\}} \{w_i \, |f_i(\mathbf{x}) - z_i^*|\} \\ &\text{subject to} \quad \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{4.4}$$

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

The method can perform differently depending on the l_p-metric. The well-known metric types are Manhattan metric ($p = 1$), Euclidean metric ($p = 2$), and Chebyshev metric ($p = \infty$).

To show how the metric types differ from each other and how they affect the performance of the weighted metric method, we consider the bi-objective optimization problem. Figure 4.3 visualizes how the weighted metric method with different metric types works.



(a) Manhattan metric. (b) Euclidean metric. (c) Chebyshev metric.

Figure 4.3: Graphical illustration of the weighted metric method with different metric types

As can be seen from Figure 4.3(a), the weighted metric method with $p = 1$ is similar to the weighted sum method discussed in Section 4.3, and thus, it also cannot approximate the points of the nonconvex region of the Pareto front [JMC09]. With $p = 2$, the method can perform better compared to $p = 1$ as presented in Figure 4.3(b). On the other hand, it increases the degree of the polynomial that is being optimized in the problem (4.3), and this may complicate the optimization process. Lastly, the weighted metric method with $p = \infty$ can reach the solutions lying on the nonconvex portion of the Pareto front as illustrated in Figure 4.3(c). However, note that the reformulated problem given in (4.4) is nonsmooth.

4.4.1 Algorithm

Algorithm 4.5 presents the pseudocode of the weighted metric method with the l_p-metric. The procedure is very similar to the weighted sum method outlined in Algorithm 4.3. The only difference is that the user needs to input the metric type with which the method approximates the Pareto front.

Algorithm 4.5: Bi-objective weighted metric method

Input : Objectives $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$, increment $\delta \in (0, 1)$, and metric type $p \in [\,1, \infty\,]$

Output: Set of solutions $\mathcal{P}$

1) Declare a matrix D with 4 columns
2) $w_1 = 0$ and $w_2 = 1 - w_1$
3) $\mathbf{w} = (w_1, w_2)^T$
4) **while** $w_1 \leq 1$ **do**
5) Approximately solve the problem (4.3) with $\mathbf{w}$ and l_p-metric to find $\mathbf{x_w} \in \mathbb{R}^n$
6) $D(i, :) = [\,\mathbf{w},\, \mathbf{x_w},\, f_1(\mathbf{x_w}),\, f_2(\mathbf{x_w})\,]$
7) $w_1 = w_1 + \delta$ and $w_2 = 1 - w_1$
8) $\mathbf{w} = (w_1, w_2)^T$

end

9) $\mathcal{P} = D(:, 2)$

4.4.2 Dynamic weight selection

Like other scalarization techniques, in order to find several solutions of the Pareto optimal set, we need to run the weighted metric method with different weight combinations. Unfortunately, this can produce some computational effort. Therefore, as before, we apply the dynamic weight selection approach to the weighted metric method. This approach follows the same logic discussed in Subsection 4.3.2, and thus, we skip the pseudocode for this method.

In this thesis, we use the weighted metric method with dynamic weight selection to which in the coming chapters we simply refer as the weighted metric method.

4.5 Genetic algorithms

The genetic algorithm (GA) is an optimization method based on the concepts of natural selection and genetics. The main idea is to imitate an evolutionary process on a set of feasible solutions. It removes the poor solutions and generates a set of better solutions. The pseudocode is retrieved from [AH17] and is presented in Algorithm 4.6.

In GA, solutions are referred as *individuals*, while we call a set of solutions as a *population*. Moreover, we call new individuals created during the

Algorithm 4.6: GA

Input : Objective $f : \mathbb{R}^n \to \mathbb{R}$, a feasible region $\mathcal{X} \in \mathbb{R}^n$, an initial population $P^0 = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, and mutation probability $\gamma \in (0, 1)$

Output: Final population P^{final}

1) $t = 0$
2) **while not** *stopping criteria* **do**
3) Evaluate the fitness of all $\mathbf{x} \in P^t$ using $f(\mathbf{x})$ ▷ `Fitness`
4) Select 2 parents from P^t and go to 5) ▷ `Selection`
or select 1 survivor from P^t and go to 7)
5) Generate a new offspring $\mathbf{x}^{new}$ from 2 parents ▷ `Crossover`
6) Mutate $\mathbf{x}^{new}$ with the probability γ ▷ `Mutation`
7) **if** $\mathbf{x}^{new} \notin \mathcal{X}$ **then**
Return to 4)
else
Add $\mathbf{x}^{new}$ to P^{t+1}
if $|P^{t+1}| < N$ **then** return to 4)
end
8) $t = t + 1$
end
9) $P^{final} = P^t$

evolutionary process as *offspring*.

Given an initial population P^0, GA begins by assigning each individual a fitness score that is used to select survivors and parents. The fitness determines how strong an individual $\mathbf{x}$ is based on the function value $f(\mathbf{x})$. The likelihood that an individual will be selected to reproduce the next population depends on the individual's fitness score.

The randomly selected parents create offspring using the crossover. As the offspring in nature are never exact copies of their parents, they go under the mutation step to create more variation. Then, the feasible offspring are added to the next population P^{t+1}, while the infeasible ones are rejected and we go back to the selection step. Note that the child population P^{t+1} must have as many individuals as its parent population P^t. This evolutionary process stops when some stopping condition is satisfied, and the algorithm returns the final population with better solutions. A common stopping criterion is to set the maximum number of generations or to set the maximum function evaluations [AH17].

The GA is very flexible so that the user can choose crossover and mutation techniques. Some of them can be found in [AH17]. In this thesis, the version of GA that we used was NSGA-II taken from [DPAM02], and all parameters were left at their default values. The details of the parameter configurations are given in Section 5.2.

In the GA, the definition of fitness is indeed very flexible. For a single-objective optimization problem, the fitness of the given population can be represented by the function values evaluated for this population. In the multi-objective case, we seek to quantify the fitness score in a way that for the given population it shows the goodness of an individual for all the objectives at the same time. Therefore, in the coming GA methods, we use the non-dominated sorting approach presented in Chapter 2 for quantifying the fitness score of the individuals.

4.5.1 NSGA-II

One of the well-known benchmark GA methods developed for the multi-objective setting is NSGA-II which has succeeded in achieving the uniformly distributed and well-approximated Pareto front for bi-objective problems [FTG15]. The main goal of NSGA-II is to find a set of Pareto-optimal solutions using the GA and non-dominated sorting. In Algorithm 4.7, we

Algorithm 4.7: NSGA-II

Input : Objectives $\mathbf{f} = \{f_1, f_2, \ldots, f_k\}$, a feasible region $\mathcal{X} \in \mathbb{R}^n$, an initial population $P^0 := \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, and mutation probability $\gamma \in (0, 1)$

Output: Final population P^{final} of non-dominated solutions

1) $t = 0$
2) **while not** *stopping criteria* **do**
3) Generate a new population Q^t using $GA(\mathbf{f}, \mathcal{X}, P^t, \gamma)$
4) $R^t = P^t \cup Q^t$
5) Apply non-dominated sorting on R^t
6) Calculate crowding distances for all $\mathbf{x} \in R^t$
7) Generate P^{t+1} by choosing N solutions of R^t based on the fitness and crowding distance
8) $t = t + 1$

end

9) $P^{final} = P^t$

provide the pseudocode for NSGA-II [DPAM02].

To better understand the algorithm, we discuss Steps 3-7 in more detail below.

At Step 3, we create a child population Q^t from the parent population P^t using the GA. Then, at Step 4, the parent population P^t and child population Q^t are combined to produce a population R^t of size $2N$. In this way, we preserve the best solutions of the current population in the next generation.

At Step 5, we evaluate the fitness of all individuals of the population R^t by ranking them based on the Pareto dominance as stated above. Then, at Step 6, the crowding distance is calculated for each individual of the population R^t. As shown in Figure 4.4, the crowding distance of an individual p_i is a perimeter of the rectangle formed by two closest neighbors p_{i-1} and p_{i+1} in the objective function space. It shows how close an individual is to its neighbors in the objective function space. Note that for the less crowded solutions, the crowding distance is higher. Therefore, in order to reach a better diversity in solution, the solutions with higher crowding distances are preferable.

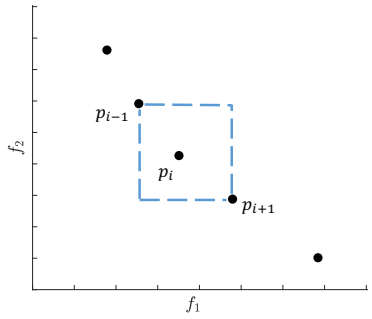


Figure 4.4: Graphical illustration of the crowding distance.

Lastly, at Step 7, we generate the next population P^{t+1} by selecting only the best N solutions of R^t based on the rank and highest crowding distances. NSGA-II terminates when some stopping condition is satisfied and returns the final population of non-dominated solutions.

4.5.2 FP-NSGA-II

The paper by [PF03] showed that the performance of NSGA-II decreases for the optimization problems with higher dimensions and more variables. Thus, Fettaka et. al. [FTG15] proposed a new hybrid algorithm that applies

NSGA-II with the prediction algorithm FP-NSGA-II to improve convergence to the true Pareto optimal set. The pseudocode is given in Algorithm 4.8 [FTG15].

The method is similar to NSGA-II, except it has an additional step for the predictive operator to produce better offspring. This step is called *front prediction*. The general idea of the front prediction is to identify the direction in the decision variable space toward the true Pareto optimal solutions [FTG15]. This is done in the following way.

Given the population P^t, we apply the Pareto front filtering, which was discussed in Chapter 2, on P^t and find the Pareto optimal set, which forms the solutions of the first front F_1^t. In the same way, by applying the Pareto front filtering once again on $P^t \setminus F_1^t$, we identify the solutions on the second front F_2^t.

Algorithm 4.8: FP-NSGA-II

Input : Objectives $\mathbf{f} = \{f_1, f_2, \ldots, f_k\}$, a feasible region $\mathcal{X} \in \mathbb{R}^n$, an initial population $P^0 := \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, and mutation probability $\gamma \in (0, 1)$

Output: Final population P^{final} of non-dominated solutions

1) $t = 0$
2) **while not** *stopping criteria* **do**
3) Determine Front in P^t. Label this F_1^t
4) Create $\tilde{P}^t = P^t \setminus F_1^t$
5) Determine Front in $\tilde{P}^t$. Label this F_2^t. (Note, if $\tilde{P}^t = \emptyset$, then $F_2^t = \emptyset$)
6) **if** $F_2^t \neq \emptyset$ **then**
7) **foreach** $\mathbf{x} \in F_1^t$ **do**
8) Determine $\mathbf{y}^* = \text{argmin}_{\mathbf{y} \in F_2^t}\ norm(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}))$
9) Calculate $step = \mathbf{x} - \mathbf{y}^*$
10) $\mathbf{x}_{pred} = \mathbf{x} + step$
11) Add $\mathbf{x}_{pred}$ to P^t
end
end
12) $P^{t+1} = NSGA\text{-}II(\mathbf{f}, \mathcal{X}, P^t, \gamma)$
13) $t = t + 1$
end
14) $P^{final} = P^t$

If the second front F_2^t is not empty, then at Step 8, for each solution $\mathbf{x}$ in F_1^t, we determine the solution $\mathbf{y}^*$ in F_2^t which is the nearest to $\mathbf{x}$ in the objective function space. Then, at Steps 9-10, we calculate a vector difference between $\mathbf{x}$ and $\mathbf{y}^*$ and refer to it as a step direction with which we extrapolate the predicted solutions $\mathbf{x}_{pred}$. At Step 11, we add the predicted solution $\mathbf{x}_{pred}$ to the population P^t.

Then, we continue with the NSGA-II applied to the population P^t. Note that, Steps 6-12 of the algorithm increase the population size of P^t. In other words, suppose that for each t, we generate N_{pred}^t number of the new predicted points. Then, the size of the population P^t that is passed to NSGA-II is $N + N_{pred}^t$. This implies that FP-NSGA-II has N_{pred}^t more f-calls than NSGA-II in each iteration.

Lastly, Algorithm 4.8 terminates when some predefined stopping condition is satisfied and returns the final population of non-dominated solutions.

To give some idea of how NSGA-II and FP-NSGA-II are performing, we imitate one simulation from the paper by [FTG15] and present in Figure 4.5 that illustrates the Pareto fronts found by the methods for the TNK problem. The light green dots are the true Pareto front, while the results for NSGA-II and FP-NSGA-II are given by a red cross symbol and blue circles, respectively.

Watching Figure 4.5, we can see that FP-NSGA-II performs better than NSGA-II in every generation. This example supports the results presented in the paper by [FTG15] that compares both methods for a fixed number of generations.

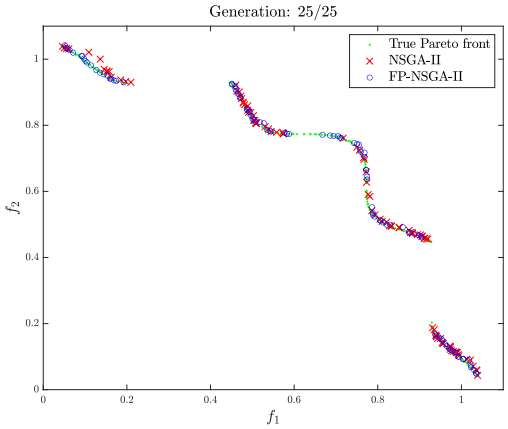


Figure 4.5: TNK problem solved by NSGA-II and FP-NSGA-II.

4.6 Warm start strategy

Referring to the scalarization techniques discussed in Sections 4.2-4.4, we notice that all of them solve a new single-objective optimization problem in each iteration. Depending on the original problem, these single-objective problems can be classified as smooth/nonsmooth, convex/nonconvex, linear/quadratic/nonlinear, differentiable/nondifferentiable, etc. Based on this information, the user can use any appropriate optimization solver.

Generally, before the optimization solver starts an optimization process, the user can set a *starting point* that indicates the initial values for the decision variables. This starting point represents the best guess where we expect to find an optimal point. If no starting point is provided by the user, then the optimization solver takes a default starting point.

For most optimization methods, the better the starting point, the faster the convergence. With this in mind, we implement the *warm start strategy*.

The main idea of the warm start strategy is to provide the optimization solver with 'good' starting points so that it can converge to the optimal solution faster. Figure 4.6 visualizes the warm start strategy for the vertical alignment setting.

Consider the bi-objective optimization problem with objectives $C_{construct}$ and C_{fuel} proposed in Chapter 3. Recall that in the vertical alignment design we input a ground profile to construct the road profile as an output. Moreover, as mentioned in the previous sections, the first step of all of the selected scalarization techniques is to identify the region where the true Pareto front lies by optimizing each objective separately. So, as shown in Figure 4.6, we start by minimizing one of the objectives, $C_{construct}$, for a given ground profile and some default starting point that is set by the

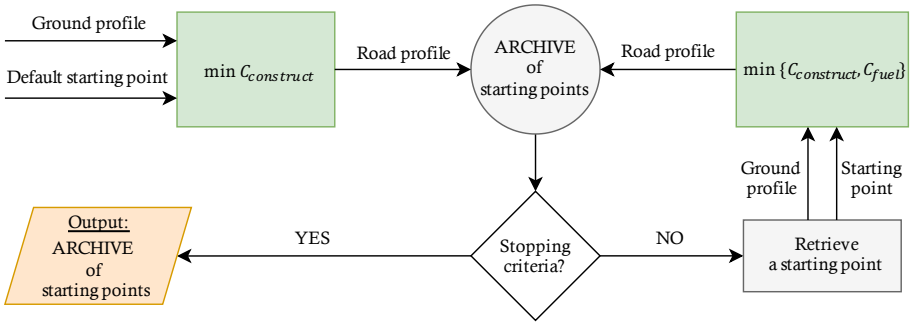


Figure 4.6: Graphical illustration of the warm start strategy.

optimization solver. Then, we save the resulting road profile to use it later as a starting point for the next optimization problem.

We record all road profiles found during the algorithm process into an archive of starting points. If the stopping criteria defined by the user are not met, then we retrieve one starting point from this archive in order to use it as a warm start in the next optimization procedure. In particular, for the ε-constraint method, our choice of the starting point depends on the value of the current ε. We select the road profile that was obtained for the ε value that is closest to the current ε. Similarly, for the weighted sum method and weighted metric methods, we use the weight values to determine the potential starting point.

With the selected starting point and the given ground profile, we move to the next optimization problem and save the resulting road profile into the archive. In this way, we use a warm start in each iteration of the scalarization techniques to select the potentially 'good' starting points that can accelerate the convergence process.

Chapter 5

Numerical results

In this chapter, we describe the numerical experiments that were performed to analyze the efficiency of the chosen MOO methods for solving the proposed bi-objective optimization problem.

In Section 5.1, we discuss the well-known performance indicators that are used to compare the results found by the MOO methods. Then, in Section 5.2, we outline how the experiment is conducted and give some overview on the experimental setup and test problems. In Section 5.3, we present all the numerical results analyzed by the selected performance indicators.

5.1 Performance indicators

To identify the most promising technique for solving the proposed MOO problem, we want to compare the performance of the selected methods in a quantitative manner. In the single-objective minimization problem, we say that the method with the smaller objective function value is better. In the multi-objective case, it is not straightforward what the appropriate quality measures are for the approximated Pareto fronts. In order to compare several sets of the solutions found by different MOO methods, we use *performance indicators* that attempt to quantify the quality of the found solution set using a scalar value. In the coming subsections, we review several performance indicators found in the literature and then discuss the selected ones more closely.

5.1.1 Literature review

There are numerous studies conducted to develop the performance indicators that aim to assess the quality of Pareto front estimates. They attempt to examine how close the approximation of the Pareto front is to the true Pareto front and how diverse the found solutions are. As the authors [ZTL$^+$03, CSQ12] mentioned, there is no performance indicator that surpasses other indicators and is universal for all the MOO problems. Below

we discuss some of the well-known and widely-used performance indicators along with their strengths and weaknesses.

Hypervolume is the most well-known performance indicator introduced by Zitzler [Zit99]. It measures the volume of space covered by the points of the obtained Pareto front and bounded by some predefined reference point in the objective function space. A positive fact for the hypervolume is that it can capture the information about the convergence and diversity of the found solution set without knowing the true Pareto optimal set. On the other hand, the measure is sensitive to the choice of the reference point [CSQ12]. We attempt to address this for our problem in Subsection 5.1.2.

Generational distance [VV99] is another commonly used performance indicator that evaluates how close the found Pareto optimal solutions are to the true Pareto optimal set. It calculates the average Euclidean distance in the objective function space from each obtained solution to the closest point on the true Pareto front which must be known beforehand [ABC$^+$20]. A weakness of this indicator becomes apparent by considering the case when the method finds only one solution and this solution lies on the true Pareto front. Then, the generational distance will be 0 implying that the found result is an ideal Pareto front approximation, which is clearly not correct.

Inverted generational distance [CC05], averaged Hausdorff distance [SELC12], and modified inverted generational distance [IMTN15] are similar variations of the generational distance that can handle its drawback but still require the true Pareto front to be known.

Spread, also known as the Δ-metric [DPAM02], is also a common performance indicator. It is a measure showing the diversity and uniformity of the solutions along the true Pareto front. However, the spread's equation proposed by [DPAM02] is applicable only for bi-objective problems. In [ZJZ$^+$06], the spread indicator is advanced for problems with more than two objectives. The main disadvantage of this performance indicator is that it needs the true Pareto optimal set to be predetermined before the assessment.

Similar to the spread, one can use another performance indicator called spacing [Sch95, DPAM02]. Spacing also analyzes the diversity and uniformity of the found Pareto optimal set in the objective function space but does not need any information about the true Pareto front. Nevertheless, as it only considers the distance between a point and its neighbor, it does not assess the convergence aspect, that is, it does not examine how close the found solutions are to the true Pareto optimal set.

Another common performance indicator is the final number of non-dominated solutions. As its name suggests, it counts the number of non-dominated solutions of the found solution set and does not need the infor-

mation about the true Pareto optimal set. However, this indicator might be inapplicable in general [ABC$^+$20]. For instance, suppose that we have method A that determines an approximated Pareto optimal set with the hundred non-dominated solutions, whereas another method B finds only one solution that can Pareto dominate all the solutions found by A. Then, according to the final number of non-dominated solutions, we reach an incorrect conclusion that method A is better than B.

Error ratio is another performance indicator that deals with the cardinality of the found Pareto optimal solutions [VV99]. It identifies what portion of the found solutions belongs to the true Pareto optimal set. This indicator may suffer from rounding errors during the classification process. Moreover, the true Pareto front must be available in advance.

Lastly, the CPU time plays a crucial role while comparing the performance of the optimization methods. It records the time required to find the approximation of the Pareto optimal set with appropriate quality. Note that the quality of the solutions can be defined by the above-stated performance indicators.

More information on other performance indicators can be found in the existing surveys [ZTL$^+$03, CSQ12, ABC$^+$20].

In this thesis, we employ the performance indicators that can be evaluated for unknown true Pareto front: hypervolume, spacing, and CPU time. The reason why we choose these indicators is that using the CPU time we can identify the fastest MOO method, and using the other two indicators we can define the conditions for good quality results. While the spacing indicator can purely examine the diversity and uniformity of the obtained solutions, the hypervolume indicator can assess both convergence and diversity of the results. In this way, we attempt to determine the most promising MOO method that can find relatively good results for the proposed MOO problem in road vertical alignment design.

In the next subsections, we discuss the selected performance indicators in more detail.

5.1.2 Hypervolume

Hypervolume is a measure of the region in the objective function space dominated by the Pareto optimal solutions p_i and bounded from top by a reference point as shown in Figure 5.1 [FTG15, ABC$^+$20]. We choose the reference point to be the point slightly worse than the nadir point of the obtained Pareto front. For consistency in calculations and fairness in comparisons, the reference point must be the same for all the compared

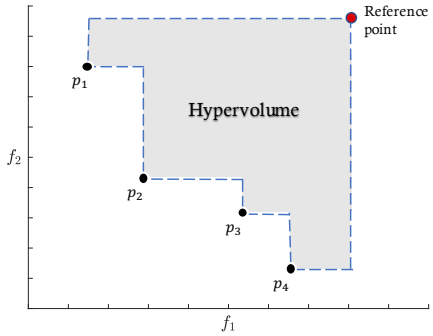


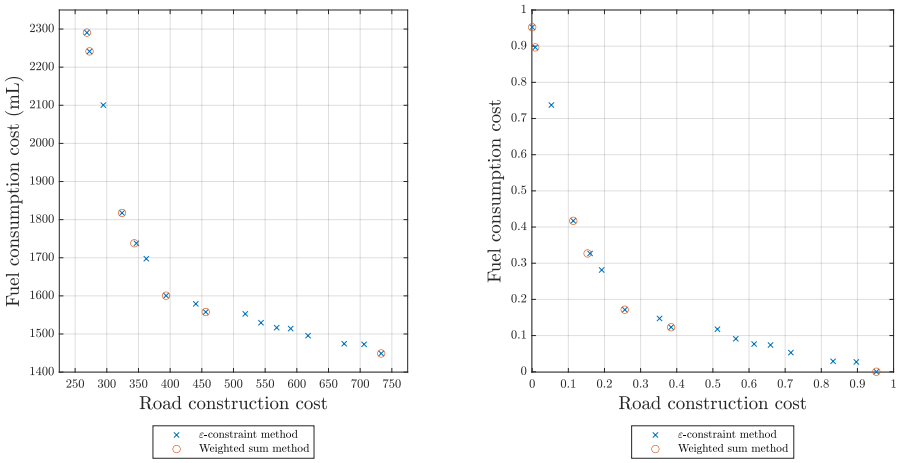
Figure 5.1: Graphical illustration of the hypervolume indicator for bi-objective problem. Hypervolume is the area of the shaded region.

MOO methods. Also, note that the reference point is not the nadir point itself, but is related to it. If we choose the nadir point as the reference point for all the compared methods, then the space covered by the found edge points would be 0, that is the hypervolume value for those edge points would be 0. This implies that the method could not find those solutions at the edges of the Pareto front, which is actually not true. This is the main reason why we do not choose the nadir point as the reference point for calculating the hypervolume indicator.

Hypervolume indicator can tell the user about the closeness of the found solutions to the true Pareto front and, at the same time, about the diversity of the solutions [FTG15, ABC$^+$20]. Moreover, there is no need to know the true Pareto front to calculate this metric. The MOO method with a higher hypervolume is considered to be better than the others.

Suppose that we optimize a small road with 22 sections using the ε-constraint method and weighted sum method. As a result, we obtain the Pareto fronts shown in Figure 5.2(a). Choosing the reference point as the point that is 5% worse than the nadir point, the hypervolume indicators are 3.33×10^5 for the ε-constraint method and 3.16×10^5 for the weighted sum method. Visually and numerically, we can conclude which method performs better. Nonetheless, these values for hypervolume are quite large. This indeed may complicate the analysis process later.

To alleviate these issues, we normalize the obtained solutions into 1-to-1 box as shown in Figure 5.2(b). Then, by selecting the reference point at a point $(1, 1)^T$, we find that the hypervolume indicators are 0.80 for the ε-constraint method and 0.75 for the weighted sum method. Note that the hypervolume for an ideal case will be 1. In this way, we can easily identify



(a) Original objective function space.

(b) Normalized objective function space.

Figure 5.2: Example of the found Pareto fronts for the road with 22 sections.

what portion of the 1-to-1 box is covered without visualizing the results, and we can come to the same conclusion on which method performs better for the given test problem.

Note that while normalizing the results into the 1-to-1 box, we do not put the solutions located at the edges at the points $(1,0)^T$ and $(0,1)^T$, instead, we keep 5% of space farther from the edges of the plot. This is done in order to track whether the MOO method could identify those solutions or not. If we put them at the corners and the method indeed finds them, then the space covered by those solutions would be still 0.

There are several approaches to estimate the hypervolume indicator for the given Pareto front. In this thesis, we calculate the hypervolume indicator using the algorithm proposed by [Fle03] and coded by [Kru11].

5.1.3 Spacing

Spacing is a measure showing how evenly and uniformly the resulting Pareto optimal solutions are distributed among themselves. For the approximated Pareto front $\mathcal{PF}^*$, the spacing indicator Δ' is calculated as follows [JOZF14, ABC+20].

$$\Delta' = \sum_{i=1}^{|\mathcal{PF}^*|-1} \frac{|d_i - \bar{d}|}{|\mathcal{PF}^*| - 1}, \tag{5.1}$$

where d_i are the Euclidean distances between consecutive solutions in the objective function space as shown in Figure 5.3, and $\bar{d}$ is the average of all d_i for $i \in [\,1, |\mathcal{PF}^*| - 1\,]$. Equation (5.1) is applicable only for bi-objective optimization problems. For more than two objectives, refer to [ZJZ+06].

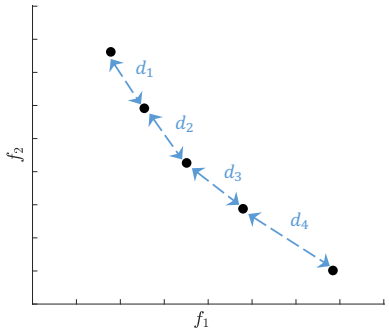


Figure 5.3: Graphical illustration of the spacing indicator for bi-objective problem. $d_1, d_2, d_3,$ and d_4 are the Euclidean distances between consecutive solutions.

Note that the spread indicator for the ideally distributed dataset is 0, while for the clustered distribution the value increases. Moreover, this indicator does not give any information about how good the found results are compared to the true Pareto front. It just assesses how well the obtained results are distributed among themselves.

5.1.4 CPU time

CPU time (sec.) is a measure indicating an approximate performance time required to obtain an estimate for the Pareto optimal set with some good quality under some timeout limit. We define good quality results with some successful test condition(s). In this way, we aim to know whether the chosen MOO method works well for the proposed MOO model. In this thesis, the successful test conditions are defined using the hypervolume and/or spacing indicators.

Using the CPU time, we implement the so-called *performance profiles* developed by Dolan and Moré [DM02] to identify which method performs better and finds good quality results. The performance profile considers the

number of problems resolved by the method for the given successful test condition(s), as well as the CPU time required to solve them.

5.2 Experiment description

The experiment is performed on 30 test problems, that is, on 30 road samples with distinct ground profiles. The road samples vary in length (number of stations). The data are provided by our industrial partner Softree Technical Systems Inc. For each problem, we test the following 10 methods.

1. ε-constraint method;
2. ε-constraint method with warm start;
3. Weighted sum method;
4. Weighted sum method with warm start;
5. Weighted metric method ($p = 1$);
6. Weighted metric method ($p = 1$) with warm start;
7. Weighted metric method ($p = \infty$);
8. Weighted metric method ($p = \infty$) with warm start;
9. NSGA-II;
10. FP-NSGA-II.

Each method is run until either the problem is *solved*, or the allocated timeout limit is hit. The timeout limit is presented in Table 5.1.

The visualizations of the found Pareto fronts for all 30 test problems appear in Appendix B.

Table 5.1: Timeout limits that are set for the experiment.

Stations	Roads	Time (sec.)
10-20 stations	8	300
20-30 stations	7	300
30-50 stations	4	1800
50-80 stations	4	3600
80-100 stations	7	10800

Note that for the proposed MOO problem we skip the weighted metric method with $p = 2$ since the reformulated problem given in (4.3) with the Euclidean metric becomes Mixed-Integer Polynomial Program (MIPP) of degree higher than 2 that is also nonsmooth and nonconvex. For our MOO model, this MIPP problem was not solved by the tested solvers (SCIP, BARON, and BMIBNB) in the given timeout limits.

Before conducting the main experiment on the 30 test problems, we implemented all the chosen scalarization approaches with/without the warm start strategy as described in the pseudocodes provided in Chapter 4. For the NSGA-II method, we used the free-accessed code written in MATLAB by [Abr19], while for the FP-NSGA-II method, we have accordingly modified the code for NSGA-II using the pseudocode given in [FTG15]. All the selected MOO algorithms were unit tested on the 14 benchmark test problems with two objectives (ZDT1-ZDT6, SRN, TNK, and other test problems given in [iJ15]).

Even though all the selected scalarization methods are deterministic, we run each of them 3 times on all 30 test problems to record the minimum CPU time spent. In this way, we aim to determine the best CPU time recorded for the least number of external operations performed in the background by the computer.

The methods such as NSGA-II and FP-NSGA-II are stochastic, and therefore, they tend to find a different set of solutions in each run. With this in mind, we run these methods 5 times each and record all the results found. The parameter settings used for NSGA-II and FP-NSGA-II are similar to those used by [DPAM02, FTG15] and are summarized in Tables 5.2-5.3.

Table 5.2: Parameter configurations for NSGA-II and FP-NSGA-II similar to those used by [DPAM02, FTG15]. Any other parameters were left at their default values.

Setting	Value
Selection	Binary tournament selection
Crossover method	Simulated binary crossover
Crossover rate	0.9
Distribution index for crossover	20
Mutation method	Polynomial mutation
Mutation rate	1/Number of decision variables
Distribution index for mutation	20
Constraint handling strategy	Constrained tournament method

Table 5.3: Population size used for NSGA-II and FP-NSGA-II.

Stations	Population size
10-20 stations	100
20-30 stations	100
30-50 stations	200
50-80 stations	500
80-100 stations	1000

We perform the experiment using the vehicle characteristics for the CAT 793D off-highway truck (as in Subsection 3.2.3) that can be retrieved from the official website of Caterpillar manufacturer. Note that the model is applicable for any heavy truck type. Moreover, we assume that the vehicle running speed is constant and is 10 m/s.

In this thesis, the optimization process is carried out using the well-known CPLEX solver retrieved from the academic edition of the IBM ILOG CPLEX Optimizer 12.10.0 [Cpl09]. This solver is accessed through a free toolbox YALMIP [Löf04] used for modeling and optimization in the programming platform MATLAB. For this thesis, we use the R2019b version of the MATLAB [MAT19]. The technical characteristics of the computer on which the experiment was performed are a Dell workstation with an Intel(R) Core(TM) i7 2.8GHz processor, a 16 GB of Random Access Memory (RAM), and a 64-bit Windows 10 operating system.

5.3 Experimental results

We first focus on the effect of the warm start strategy on the convergence speed of the selected scalarization techniques. Then, we aim to determine which is the most promising MOO method for solving the proposed optimization problem in the vertical alignment design.

5.3.1 Effect of warm start strategy on speed

As mentioned in Section 4.6, the main idea of implementing the warm start strategy on the scalarization techniques is to speed up the optimization process by converging to the optimal solution faster.

Table 5.4 presents the approximate percentage value by which the warm start strategy accelerates the convergence speed of the chosen scalarization techniques. In the brackets, we state the standard deviations of the results.

Table 5.4: Average speed improvement (stdev) due to the warm start strategy.

Stations	ε-constraint method	Weighted sum method	Weighted metric method ($p = 1$)	Weighted metric method ($p = \infty$)
10-20 stations	15.1% (2.0)	6.8% (2.2)	15.2% (1.6)	6.8% (0.1)
20-30 stations	17.3% (1.5)	18.9% (0.7)	21.7% (0.8)	6.2% (0.5)
30-50 stations	18.0% (0.4)	14.0% (2.8)	11.3% (3.6)	*
50-80 stations	13.9% (0.1)	7.6% (0.2)	*	*
80-100 stations	13.9% (0.5)	7.6% (0.3)	8.2% (0.8)	*
Average	15.7% (0.9)	11.0% (1.2)	14.1% (1.7)	6.5% (0.3)

* Terminated due to the timeout limit.

Moreover, we ignore the test problems that terminated with the timeout limit. If every problem belonging to the same entry terminated due to the timeout limit, then we mark such entries with a star symbol. As a result, we can see that the warm start strategy indeed improves the CPU time of the selected MOO methods by approximately 5-15%.

In the next subsection, we present results for the other performance indicators and discuss how the warm start strategy has affected the quality of the results.

5.3.2 Examination of performance indicators

To present the results obtained for the hypervolume and spacing indicators for all 30 test problems and all 10 MOO methods, we use the box-and-whisker plots given in Figures 5.4-5.5.

Each box-and-whisker plot shows the distribution of numerical results for each MOO method by visualizing the minimum, lower quartile, median, upper quartile, and maximum of the selected indicator. The box-and-whisker plots label the mean values with a cross symbol and the outliers with the dots outside the box. The ends of the box are the upper and lower quartiles, whereas a line inside the box is the median. The whiskers are the two vertical lines that extend outside the box to the maximum and minimum observations.

For Figure 5.4, the higher hypervolume values are preferable, and the ideal result would be 1. For Figure 5.5, the lower spacing values are desired, and the ideal result would be 0.

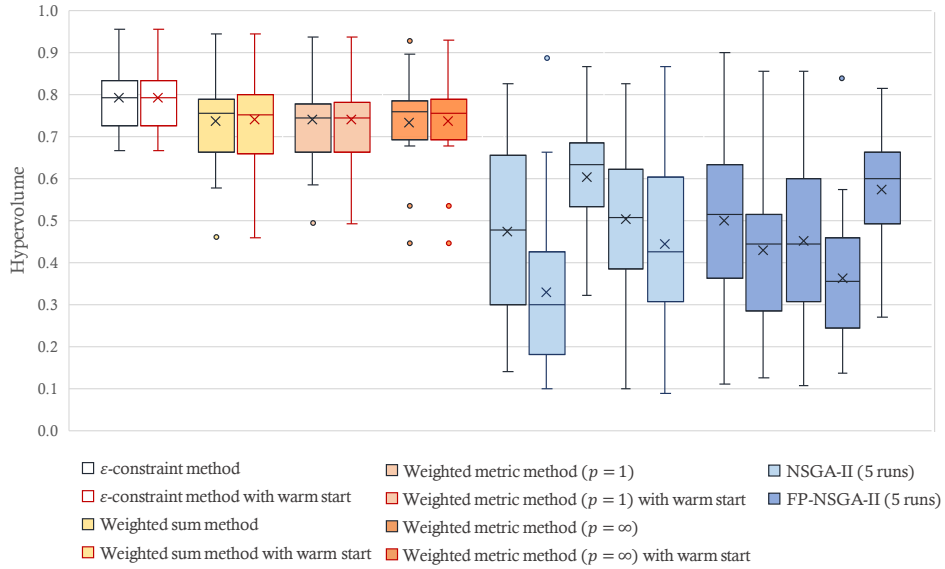


Figure 5.4: Results for hypervolume indicator (higher is better). In a box-and-whisker plot: the ends of the box are the upper and lower quartiles; a line inside the box is the median; a cross is the mean; the whiskers are the two lines that extend outside the box to the highest and lowest observations.

First of all, we focus on scalarization techniques. As we can see from both plots, the scalarization techniques with the warm start strategy perform as well as (or slightly better in some cases) the versions without the warm start. Combining this information with the timing results in Subsection 5.3.1, it is clear that using the warm start is a good choice.

The cases where the warm start strategy slightly improves the performance of some of the MOO methods can be explained as follows. Among 30 test problems, the scalarization techniques solved the bulk of them within a time range that is less than the timeout limit. Nonetheless, there were still several test problems that required the entire timeout limit to terminate. So, the warm start strategy showed the improvement only in those instances by accelerating the optimization process and performing more iterations to find more solutions before reaching the given timeout limit.

Now, we focus our attention on the numerical results found by the NSGA-II and FP-NSGA-II methods given in Figures 5.4-5.5. For each of them, we illustrate the results for all 5 runs. In all runs, both methods obtain a wide spread of results for the hypervolume and spacing indicators. This

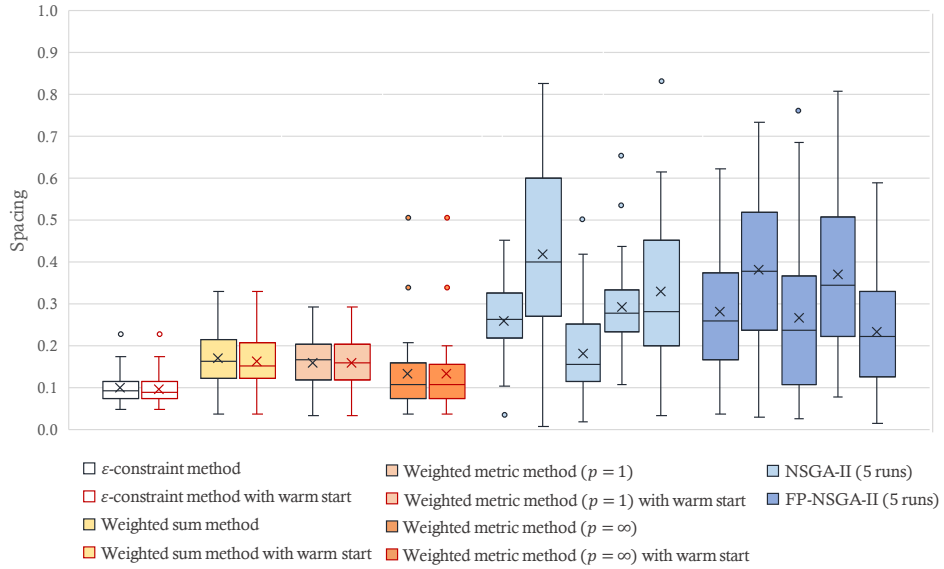


Figure 5.5: Results for spacing indicator (lower is better). In a box-and-whisker plot: the ends of the box are the upper and lower quartiles; a line inside the box is the median; a cross is the mean; the whiskers are the two lines that extend outside the box to the highest and lowest observations.

implies that these selected GA methods are unstable for the proposed MOO problem. A possible reason for their non-stable performance can be the vast amount of equality constraints required to define the objective function for the road construction cost. These constraints are retrieved from the blackbox model described in [MR12, BHLH17]. Generally, the equality constraints reduce the feasible search space which, in turn, complicates the selection of feasible solutions for the chosen GA methods.

Overall, based on the results shown in Figures 5.4-5.5, the ε-constraint method with the warm start is a candidate for the promising MOO method for our proposed problem followed by the weighted metric method ($p = \infty$) with the warm start.

5.3.3 Comparison of time

Now, we analyze the CPU time performance of the chosen MOO methods by visualizing the performance profiles for different successful test condition(s).

Before discussing the results, we briefly present an interpretation of the

performance profiles. On its y-axis, we have the percentage of test problems that passed the specified successful test condition(s), whereas, on its x-axis, we have the performance ratio showing the ratio of time cost. For example, an (x, y) point of $(5, 0.3)$ for an examined solver means that given 5 times as long as the fastest solver, the examined solver successfully solved 30% of problems. As such, the method with the highest speed of convergence would be found at $\tau = 0$ with the highest value of the y-axis, while the method solving the most number of test problems with good quality would be found close to the top part of the plot. So, to determine the fastest and efficient method, we need to look for the graph closest to the top-left region of the performance profile.

Moving on to the discussion of the results, we make the following important notes. For the scalarization techniques, we focus only on their warm start versions, because, as discussed before, they obtain as good (or slightly better) results as their original versions within a shorter amount of time. For the GA methods, we consider only the best run out of 5 runs for each test problem.

Based on the box-and-whisker plots presented in Figures 5.4-5.5, we first choose to examine the performance of the methods for some non-strict successful test conditions: hypervolume ≥ 0.75 and spacing ≤ 0.25 (both of them implied at the same time). In this way, we aim to identify the most suitable and promising method for our MOO problem that determines average quality results for most of the test problems.

Figure 5.6 shows the performance profile for the selected successful test conditions. As can be seen from Figure 5.6, the ε-constraint method with the warm start successfully solves the most number of test problems (23 problems out of 30) with relatively good time performance. Nonetheless, the fastest method is the weighted sum method with the warm start, as it solves half of the test problems the fastest. Moreover, we did an additional test by removing the 'best' method (that is, ε-constraint method with the warm start) from the performance profile to verify that the weighted sum method is the second 'best' method. This was done to check that there is no switching phenomenon (see [GS16]). Lastly, for Figure 5.6, note that even the best runs of the GA methods show poor results for the proposed optimization model since they use the entire timeout limit to stop.

In Figure 5.7, we examine the CPU time performance for other different successful test conditions.

For the hypervolume ≥ 0.75 (Figure 5.7(a)), we obtain almost similar results as in Figure 5.6, while for the spacing ≤ 0.25 (Figure 5.7(b)), we reveal the similar pattern as in Figure 5.6 but with more number of the

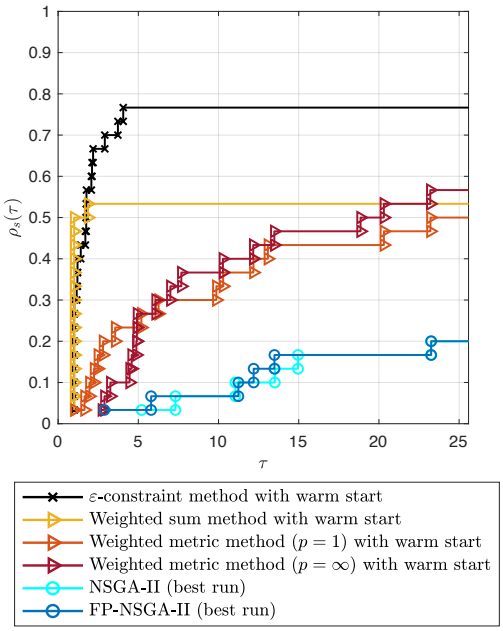
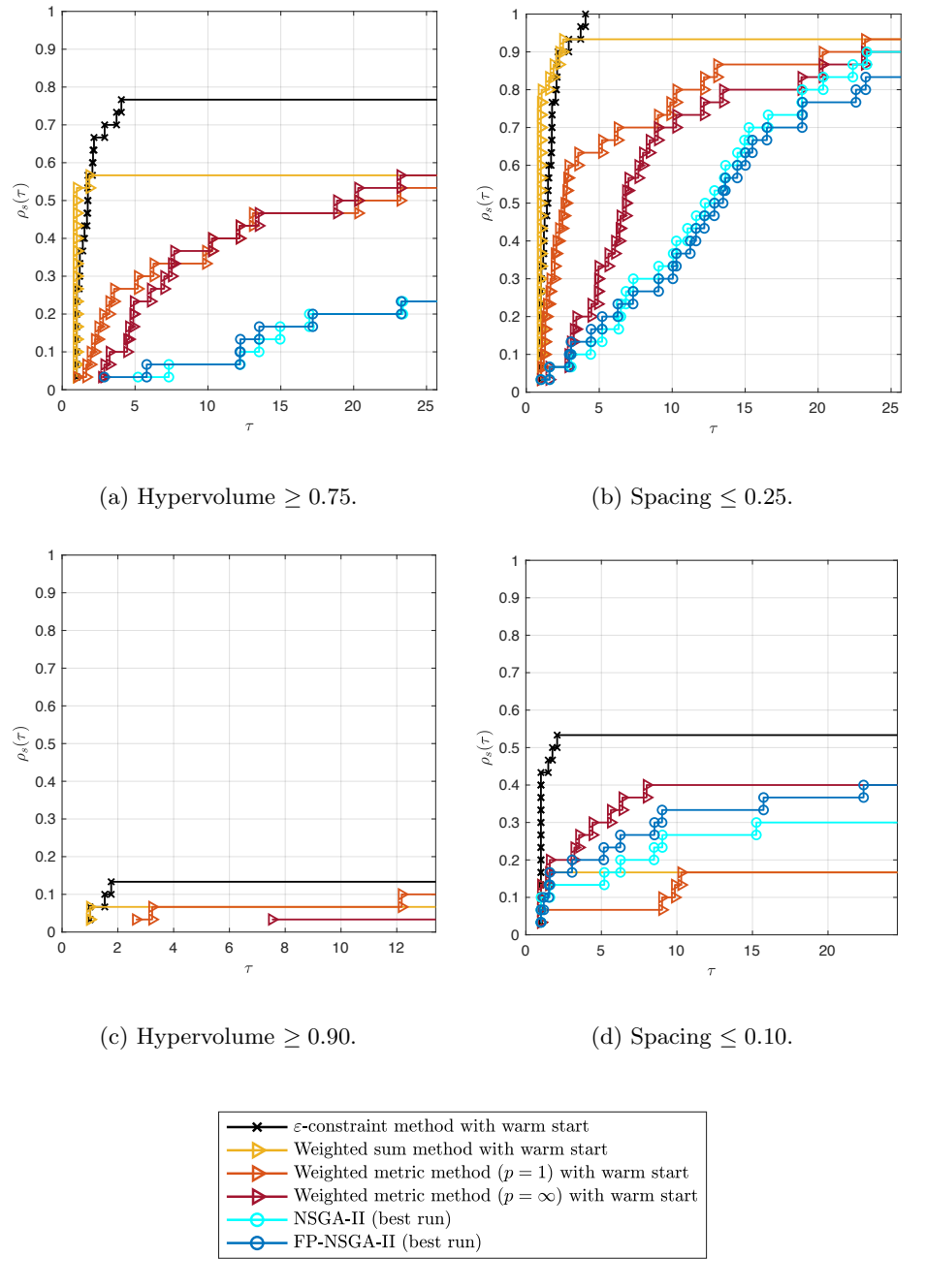


Figure 5.6: Performance profile with the successful test conditions of hypervolume ≥ 0.75 and spacing ≤ 0.25.

successfully solved test problems. This implies that the compared MOO methods can reach diversity easier than convergence. For the hypervolume ≥ 0.90 (Figure 5.7(c)), only a small portion of the test problems passed this strict condition, and most of the methods could not reach the high level of convergence. For the spacing ≤ 0.1 (Figure 5.7(d)), we reveal that the ε-constraint method with the warm start performs best by solving almost half of the problems with a high level of diversity. Moreover, we can notice that the GA methods obtain better results for diversity than the weighted sum method with the warm start and the weighed metric method ($p = 1$) with the warm start. From these two plots, we can again conclude that for the given MOO model the diversity and uniformity of the solutions can be obtained easier than the convergence.

Overall, Figure 5.7 supports the conclusions of Figure 5.6 that the ε-constraint method with the warm start provides the most robust convergence.



(a) Hypervolume ≥ 0.75.

(b) Spacing ≤ 0.25.

(c) Hypervolume ≥ 0.90.

(d) Spacing ≤ 0.10.

Figure 5.7: Performance profiles with different successful test conditions.

5.3.4 Summary of results

We tested 10 MOO methods on 30 road samples and analyzed the obtained results using the performance indicators. We find the following results.

- The warm start strategy accelerates the performance of the scalarization techniques by approximately 5% to 15%.
- The warm start versions of the scalarization methods obtain similar (or slightly better) results as their original versions.
- The most promising and recommended MOO method for the proposed optimization model is the ε-constraint method with the warm start strategy.
- The second best MOO method is the weighted sum method with the warm start strategy. It shows the best performance in time but has a notably lower success rate than the ε-constraint method.
- The GA methods perform unstably and poorly for our MOO problem.
- The diversity and uniformity of the solutions can be reached easier than the convergence.

Chapter 6

Conclusion

6.1 Conclusion

The cost of road design is significant as it needs large capital investments for building new roads and incurs future user costs as well. This thesis proposes a multi-objective approach that seeks to find road profiles that are optimal for the manufacturers in terms of the road construction cost and at the same time for the users in terms of the vehicle operating costs.

In our model, we use the road construction cost defined by the MILP model given in [MR12, BHLH17]. As the vehicle operating cost, we employ the fuel consumption cost proposed by [ASB12]. It estimates fuel usage based on the vehicle parameters and environmental factors that affect vehicle motion. Moreover, the fuel consumption model was validated on the dataset given in [SAKK16] with the coefficient of determination $R^2 = 0.93$.

In this thesis, we examine and test several well-known MOO methods on 30 road samples to determine the most promising approach for optimizing our model. We focus on three classical scalarization techniques (the ε-constraint method, weighted sum method, and weighted metric methods) and two commonly-used GA methods (NSGA-II and FP-NSGA-II). Moreover, we propose the warm start strategy to accelerate the performance of the chosen scalarization techniques. The results were assessed with hypervolume indicator (to test the convergence of solutions), spacing indicator (to test the diversity of solutions), and CPU time (to test the speed).

As a result, we determine that the warm start versions of the scalarization techniques obtain similar (or slightly better) results as their original versions, as well as in less time. Moreover, the most robust and recommended MOO method for our proposed problem is the ε-constraint method with the warm start strategy. We also identify that the compared GA methods perform poorly and unstably for our problem.

6.2 Future work

Below we outline several suggestions for future work as an improvement/extension of this research.

6.2.1 Model extension

- Our model considers only bi-objective optimization problem. One could include more objectives such as tire wear, travel time, etc.
- Our model assumes that the vehicle running speed is constant and user-defined. One could advance the model by introducing a new decision variable that can control the vehicle running speed.
- Our model focuses only on the vertical alignment design. One could extend the existing model to the horizontal alignment design and could optimize the horizontal and vertical alignments at the same time.

6.2.2 Methods extension

- Our dynamic ε/weight selection approach for the scalarization techniques uses Euclidean distance to calculate the range of search regions. One could improve this approach by implementing different distance metrics so that it would select the search regions more efficiently.
- Our tested GA methods suffer from the vast amount of equality constraints that significantly reduce the feasible search space. One could advance them by introducing an efficient way of handling the equality constraints or could implement some other GA methods.
- Our experiment tests only several commonly-used MOO methods. One could choose to employ other MOO methods such as the derivative-free MOO methods, heterogeneous MOO methods, or some other MOO methods that can address nonsmooth/nonconvex/nonlinear problems.

Bibliography

[AB03] R. Akçelik and M. Besley. Operating cost, fuel consumption, and emission models in aaSIDRA and aaMOTION. In *25th Conference of Australian Institutes of Transport Research (CAITR 2003)*, pages 1–15. University of South Australia Adelaide, Australia, 2003. → pages 16, 65

[ABC+20] C. Audet, J. Bigeon, D. Cartier, S. L. Digabel, and L. Salomon. Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 2020. → pages 7, 41, 42, 43, 44

[Abr13] S. Abrams. The unseen history of our roads. Road & Track. `https://www.roadandtrack.com/car-culture/a4447/the-road-ahead-road-evolution.html`, 2013. Accessed: 2021-05-15. → pages 1

[Abr19] A. F. Abril. Real coded (integer handling) NSGA-II. MATLAB Central File Exchange. `https://www.mathworks.com/matlabcentral/fileexchange/71609-real-coded-integer-handling-nsga-ii.html`, 2019. Accessed: 2021-01-05. → pages 47

[AH17] C. Audet and W. Hare. *Derivative-free and blackbox optimization*. Springer, 2017. → pages 32, 33, 34

[ASB12] R. Akçelik, R. Smit, and M. Besley. Calibrating fuel consumption and emission models for modern vehicles. In *IPENZ Transportation Group Conference, Rotorua, New Zealand*, pages 1–13, 2012. → pages 13, 14, 15, 16, 56

[Asg98] M. J. Asgharpour. *Multiple criteria decision making, volume 1*. Tehran University Publishing, Tehran, 1998. → pages 22

[BHLH17] V. Beiranvand, W. Hare, Y. Lucet, and S. Hossain. Multi-haul quasi network flow model for vertical alignment optimization.

Engineering Optimization, 49(10):1777–1795, 2017. → pages 2, 5, 11, 12, 51, 56

[CC61] A. Charnes and W. W. Cooper. *Management models and industrial applications of linear programming*. John Wiley & Sons, LTD, New York, 1961. → pages 21

[CC05] C. A. C. Coello and N. C. Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190, 2005. → pages 41

[CCF55] A. Charnes, W. W. Cooper, and R. O. Ferguson. Optimal estimation of executive compensation by linear programming. *Management Science*, 1(2):138–151, 1955. → pages 21

[CH83] V. Chankong and Y. Y. Haimes. *Multiobjective decision making theory and methodology*. Elsevier Science, New York, 1983. → pages 22

[CHE13] A. Carlson, U. Hammarström, and O. Eriksson. Models and methods for the estimation of fuel consumption due to infrastructure parameters. *MIRAVEC Deliverable D2.1*, 2013. → pages 13

[CPC21] CPCS. Cost of poor roads in Canada - final report for Canadian Automobile Association (CAA). In *Solutions for Growing Economies*, pages 1–16, 2021. → pages 1, 2

[Cpl09] IBM ILOG Cplex. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009. → pages 48

[CSQ12] S. Cheng, Y. Shi, and Q. Qin. On the performance metrics of multiobjective optimization. In *International Conference in Swarm Intelligence*, pages 504–512. Springer, 2012. → pages 40, 41, 42

[Deb01] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, LTD, New York, 2001. → pages 22

[DJ13] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2013. → pages 23

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. → pages 45

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. → pages 9, 23, 34, 35, 41, 47

[EML01] Z. Eckart, L. Marco, and T. Lothar. Improving the strength pareto evolutionary algorithm for multiobjective optimization. *EUROGEN, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 1–21, 2001. → pages 22

[Fle03] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 519–533. Springer, 2003. → pages 44

[FTG15] S. Fettaka, J. Thibault, and Y. Gupta. A new algorithm using front prediction and NSGA-II for solving two and three-objective optimization problems. *Optimization and Engineering*, 16(4):713–736, 2015. → pages 23, 34, 35, 36, 37, 42, 43, 47

[Fwa89] T. F. Fwa. Highway vertical alignment analysis by dynamic programming. *Transportation Research Record*, 1239(1-9):2–3, 1989. → pages 11

[GS55] S. Gass and T. Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2(1-2):39–45, 1955. → pages 21

[GS16] N. Gould and J. Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS)*, 43(2):1–5, 2016. → pages 52

[iJ15] Metaheuristic Algorithms in Java. Problems included in jMetal. `http://www.jmetal.sourceforge.net/problems.html`, 2015. Accessed: 2021-02-15. → pages 47

[IMTN15] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima. Modified distance calculation in generational distance and inverted generational distance. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 110–125. Springer, 2015. → pages 41

[JMC09] A. L. Jaimes, S. Z. Martınez, and C. A. C. Coello. An introduction to multiobjective optimization techniques. *Optimization in Polymer Processing*, pages 29–57, 2009. → pages 6, 7, 21, 24, 31

[JOZF14] S. Jiang, Y. S. Ong, J. Zhang, and L. Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404, 2014. → pages 44

[JSJ06] M. K. Jha, P. M. Schonfeld, and J. C. Jong. *Intelligent road design*, volume 19. WIT press, 2006. → pages 12, 13

[Kru11] J. W. Kruisselbrink. Hypervolume computation. MATLAB Central File Exchange. `https://www.mathworks.com/matlabcentral/fileexchange/30785-hypervolume-computation.html`, 2011. Accessed: 2021-02-14. → pages 44

[Kub15] K. A. C. Kubler. Optimisation of off-highway truck fuel consumption through mine haul road design. Master's thesis, University of Southern Queensland, Queensland, Australia, 2015. → pages 13

[Löf04] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference*, pages 284–289, Taipei, Taiwan, 2004. → pages 48

[MAT19] MATLAB. *9.7.0.1190202 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2019. → pages 48

[MKO14] M. Mäkelä, N. Karmitsa, and W. Outi. Multiobjective proximal bundle method for nonsmooth optimization. Technical report, TUCS, 2014. → pages 23

[MM20] M. Mäkelä and O. Montonen. New multiobjective proximal bundle method with scaled improvement function. In *Numerical Nonsmooth Optimization*, pages 461–479. Springer, 2020. → pages 23

[MR12] F. Md Rahman. Optimizing the vertical alignment under earthwork block removal constraints in road construction. Master's thesis, University of British Columbia (Okanagan), Kelowna BC, Canada, 2012. → pages 2, 4, 5, 11, 12, 51, 56

[PF03] R. C. Purshouse and P. J. Fleming. Evolutionary many-objective optimisation: an exploratory analysis. In *The 2003 Congress on Evolutionary Computation, 2003 (CEC'03)*, volume 3, pages 2066–2073. IEEE, 2003. → pages 35

[RK14] J. H. Ryu and S. Kim. A derivative-free trust-region method for biobjective optimization. *SIAM Journal on Optimization*, 24(1):334–362, 2014. → pages 23

[RLD+01] H. Rakha, I. Lucic, S. H. Demarchi, J. R. Setti, and M. V. Aerde. Vehicle dynamics model for predicting maximum truck acceleration levels. *Journal of Transportation Engineering*, 127(5):418–425, 2001. → pages 17

[SAKK16] A. Soofastaei, S. M. Aminossadati, M. S. Kizil, and P. Knights. The influence of rolling resistance on haul truck fuel consumption in surface mines. *Tribology International Journal*, 2(1):215–228, 2016. → pages 13, 14, 20, 56

[Sch85] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 93–100. Lawrence Erlbaum Associates. Inc., Publishers, 1985. → pages 22

[Sch95] J. R. Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1995. → pages 41

[SELC12] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello. Using the averaged hausdorff distance as a performance measure in

evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522, 2012. → pages 41

[SF16] G. Svenson and D. Fjeld. The impact of road geometry and surface roughness on fuel consumption of logging trucks. *Scandinavian Journal of Forest Research*, 31(5):526–536, 2016. → pages 14

[TE19] J. Thomann and G. Eichfelder. A trust-region algorithm for heterogeneous multiobjective optimization. *SIAM Journal on Optimization*, 29(2):1017–1047, 2019. → pages 23

[TV03] R. J. Thompson and A. T. Visser. Mine haul road maintenance management systems. *Journal of the Southern African Institute of Mining and Metallurgy*, 103(5):303–312, 2003. → pages 12

[TWFT95] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino. Ga-based decision support system for multicriteria optimization. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 2, pages 1556–1561. IEEE, 1995. → pages 26

[VOS14] K. D. V. Villacorta, P. R. Oliveira, and A. Soubeyran. A trust-region method for unconstrained multiobjective problems with applications in satisficing processes. *Journal of Optimization Theory and Applications*, 160(3):865–889, 2014. → pages 23

[VV99] D. A. Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations.* PhD thesis, School of Engineering of the Air Force Institute of Technology, Dayton, Ohio, 1999. → pages 41, 42

[Zit99] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: methods and applications.* PhD thesis, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, 1999. → pages 41

[ZJZ+06] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *2006 IEEE International Conference on Evolutionary Computation*, pages 892–899. IEEE, 2006. → pages 41, 45

[ZTL+03] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003. → pages 40, 42

Appendices

Appendix A

Tables

Table A.1: Parameters α, β_1 and β_2 retrieved from [AB03].

Parameter	Symbol	Unit	Light vehicle	Heavy vehicle
Idle fuel rate	α	mL/s	0.375	0.556
Energy efficiency parameter	β_1	mL/kJ	0.009	0.008
Energy-acceleration efficiency parameter	β_2	mL/(kJ.m/s^2)	0.003	0.002

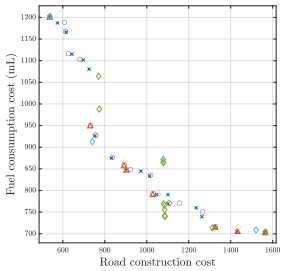
Table A.2: Idle fuel rates for various vehicle types. Source: Argonne National Laboratory, www.anl.gov, 2014.

Vehicle type	Class	Fuel type	Idle fuel rate (gal/hr)	
			No load	With load
Medium heavy truck	6	Gasoline	0.84	-
Delivery truck	5	Diesel	0.84	1.10
Tow truck	6	Diesel	0.59	1.14
Medium heavy truck	6-7	Diesel	0.44	-
Transit bus	7	Diesel	0.97	-
Combination truck	7	Diesel	0.49	-
Bucket truck	8	Diesel	0.90	1.50
Tractor-semitrailer	8	Diesel	0.64	1.15

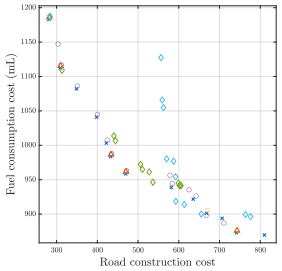
Appendix B

Figures

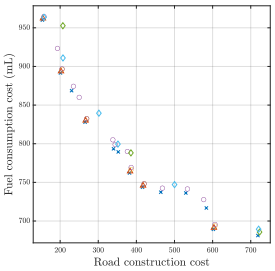
In this appendix, we illustrate the Pareto fronts for all 30 test problems. In each problem, we plot only the non-dominated solutions found by the scalarization methods with the warm start and the best run of GA methods.



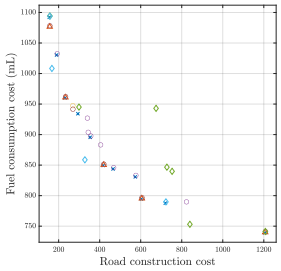
(a) Test problem 1.



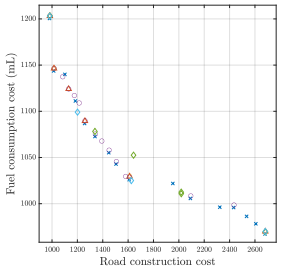
(b) Test problem 2.



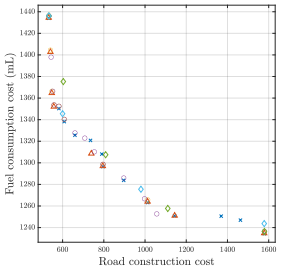
(c) Test problem 3.



(d) Test problem 4.



(e) Test problem 5.



(f) Test problem 6.

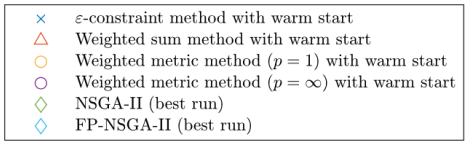
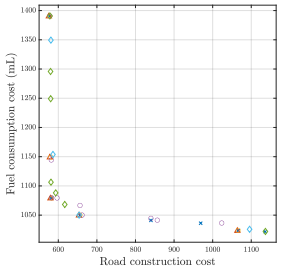
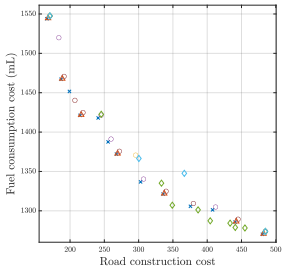


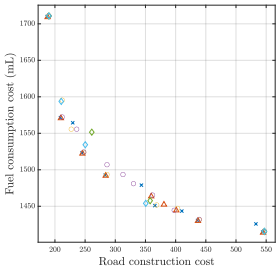
Figure B.1: Resulting Pareto fronts for 30 test problems.



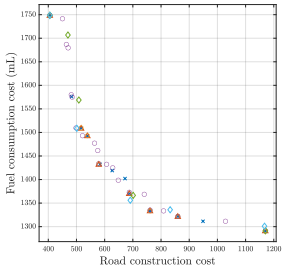
(g) Test problem 7.



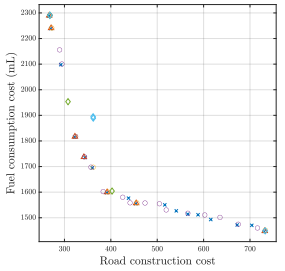
(h) Test problem 8.



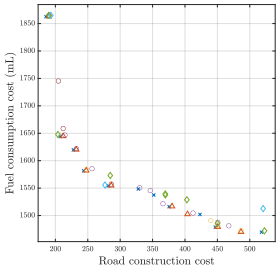
(i) Test problem 9.



(j) Test problem 10.



(k) Test problem 11.



(l) Test problem 12.

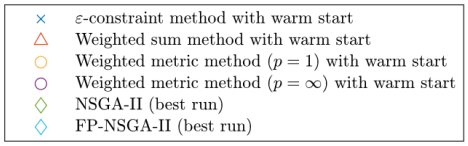
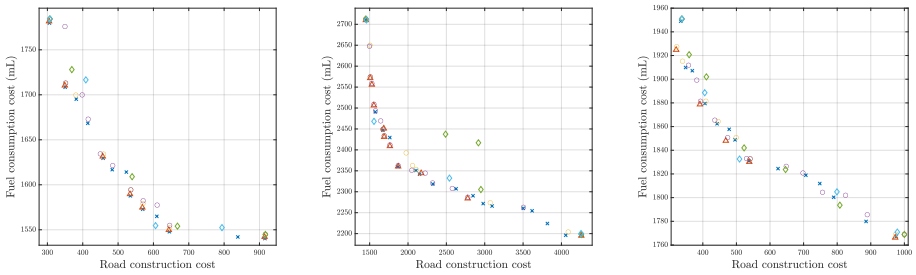
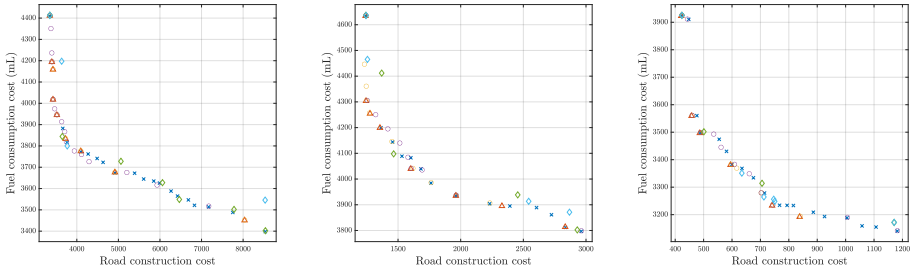


Figure B.2: Resulting Pareto fronts for 30 test problems (cont.).



(m) Test problem 13. (n) Test problem 14. (o) Test problem 15.



(p) Test problem 16. (q) Test problem 17. (r) Test problem 18.

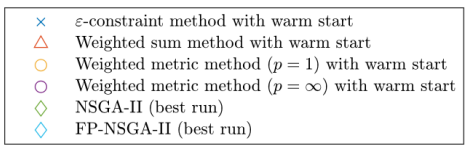
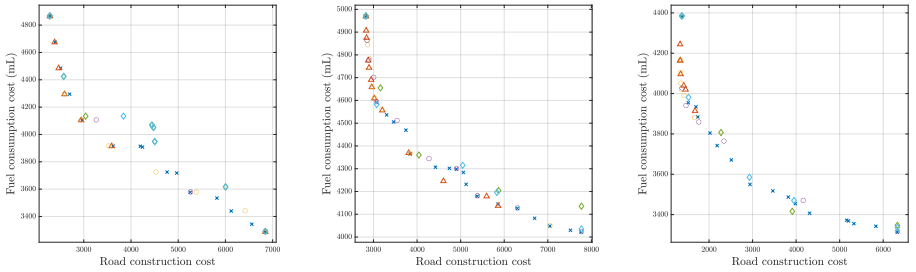
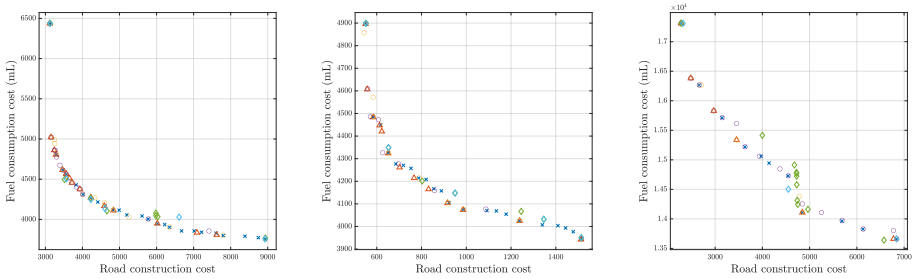


Figure B.3: Resulting Pareto fronts for 30 test problems (cont.).



(s) Test problem 19. (t) Test problem 20. (u) Test problem 21.



(v) Test problem 22. (w) Test problem 23. (x) Test problem 24.

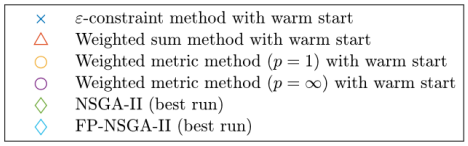
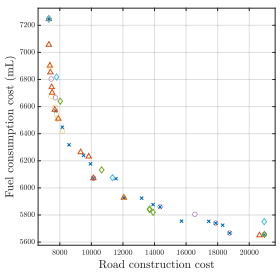
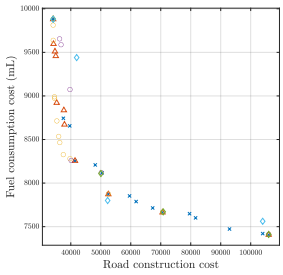


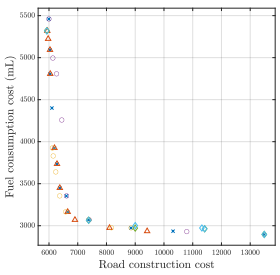
Figure B.4: Resulting Pareto fronts for 30 test problems (cont.).



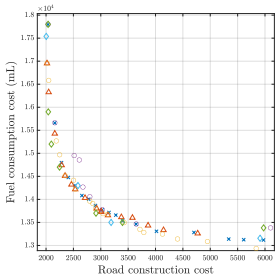
(y) Test problem 25.



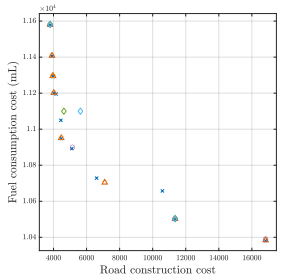
(z) Test problem 26.



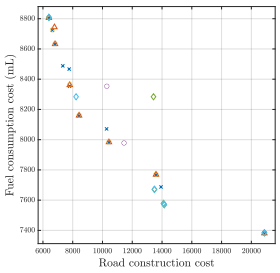
(aa) Test problem 27.



(ab) Test problem 28.



(ac) Test problem 29.



(ad) Test problem 30.

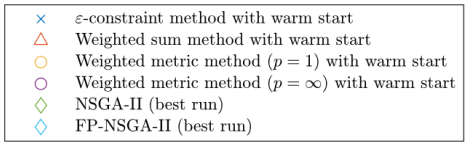


Figure B.5: Resulting Pareto fronts for 30 test problems (cont.).