Bus route design and frequency setting for public transit systems

Günther Hüsselmann

Dissertation presented for the degree of
Doctor of Philosophy in Industrial Engineering
in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof JH van Vuuren
Co-supervisor: Prof SJ Andersen

April 2022

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date:April 2022

Abstract

The availability of effective public transport systems is increasingly becoming an urgent problem in urban areas worldwide due to the traffic congestion caused by private vehicles. The careful design of such a transport system is important because, if well designed, such a system can increase the comfort of commuters and ensure that they arrive at their destinations timeously. A well-designed public transport system can also result in considerable cost savings for the operator of the system.

The problem considered in this dissertation is that of designing three mathematical models for aiding a bus company in deciding upon efficient bus transit routes (facilitated by the first two models) and setting appropriate frequencies for buses along these routes (facilitated by the third model). The design criteria embedded in the first model (for designing bus routes) are the simultaneous pursuit of minimising the expected average passenger travel time and minimising the system operator's cost (measuring the latter as the sum total of all route lengths in the system). The first model takes as input an origin-destination demand matrix for a specified set of bus stops, along with the corresponding road network structure, and returns as output a set of bus route solutions. The decision maker can then select one of these route sets subjectively, based on the desired trade-off achieved between the aforementioned transit system design criteria. This bi-objective minimisation problem is solved approximately in three distinct stages — a solution initialisation stage, an intermediate analysis stage and an iterative metaheuristic search stage during which high-quality trade-off solutions are sought. A novel procedure is introduced for the solution initialisation stage aimed at effectively generating high-quality initial feasible solutions. Two metaheuristics are adopted for the solution implementation, namely a dominance-based multi-objective simulated annealing algorithm and an improved non-dominated sorting genetic algorithm.

The second model is a novel approach towards establishing high-quality bus routes resembling a reference set of bus routes (typically the currently operational bus routes) to varying degrees, providing the decision maker with bus route design alternatives that may be implemented incrementally in order to limit the disruption experienced by passengers in the bus transit network. The objectives pursued in this model are the simultaneous minimisation of the expected average passenger travel time and the minimisation of a reference-route-to-design-route similarity measure. The second model takes the same input as the first model above, with the addition of a reference route set with which to compare alternative design routes in terms of similarity, and provides as output a set of trade-off solutions according to this model's design criteria. The same three-stage approximate solution methodology described above is adopted for this model, and the same two metaheuristic implementations are utilised to solve instances of this new model.

In the third model, high-quality bus frequencies are sought for each bus route in pursuit of minimising the expected average travel time for passengers (including waiting time, transfer time and travel time) and simultaneously minimising the total number of buses required by an opera-

tor to maintain the specified frequencies. The third model takes as input all the data required by the first model, along with a route set for which frequencies should be set, and returns as output a set of bus frequencies at which buses should operate along the various routes, based on a desired trade-off between the aforementioned two design criteria. The solution approach adopted for this bi-objective minimisation problem again conforms to the three aforementioned distinct stages, with the exception that only a non-dominated sorting genetic algorithm is designed for solving it.

The first and third models are finally applied to a special case study involving real data in order to showcase the practical applicability of the modelling approach.

Opsomming

Die beskikbaarheid van doeltreffende openbare vervoerstelsels word wêreldwyd toenemend 'n dringende probleem in stedelike gebiede as gevolg van die verkeersopeenhopings wat deur private voertuie veroorsaak word. Die noukeurige ontwerp van so 'n vervoerstelsel is belangrik, want as dit goed ontwerp is, kan so 'n stelsel die gemak van pendelaars verhoog en verseker dat hul betyds by hul bestemmings aankom. 'n Goed-ontwerpte openbare vervoerstelsel kan ook aansienlike kostebesparings vir die stelseloperateur tot gevolg hê.

Die probleem wat in hierdie proefskrif oorweeg word, is die ontwerp van drie wiskundige modelle om 'n busonderneming daartoe in staat te stel om besluite oor doeltreffende busvervoerroetes (die eerste twee modelle) en die geskikte frekwensies vir busse langs hierdie roetes (die derde model) te neem. Die ontwerpkriteria in die eerste model (vir die ontwerp van busroetes) is die gelyktydige strewe daarna om die verwagte gemiddelde reistyd van passasiers te minimeer en die koste van die stelseloperateur te minimeer (laasgenoemde gemeet as die somtotaal van alle roetelengtes in die stelsel). Die eerste model neem as toevoer 'n oorsprong-bestemming aanvraagmatriks vir 'n spesifieke stel bushaltes, tesame met die ooreenstemmende padnetwerkstruktuur, en lewer as afvoer 'n versameling busroetestelle. Die besluitnemer kan dan een van hierdie roetestelle subjektief kies, gebaseer op die gewenste afruiling tussen die bogenoemde ontwerpkriteria. Hierdie twee-doelige minimeringsprobleem word in drie verskillende fases benaderd opgelos — 'n oplossingsinisialiseringsfase, 'n intermediêre analise-fase en 'n iteratiewe metaheuristiese soekfase waartydens afruilingssoplossings van hoë gehalte gesoek word. 'n Nuwe prosedure word vir die oplossingsinisialiseringsfase daargestel wat daarop gemik is om aanvanklike haalbare oplossings van hoë gehalte op 'n doeltreffende wyse te genereer. Twee meteheuristieke word vir die oplossing van die model gebruik, naamlik 'n dominansie-gebaseerde meer-doelige gesimuleerde temeperingsalgoritme en 'n verbeterde nie-gedomineerde sorteer-genetiese algoritme.

Die tweede model is 'n nuwe benadering om busroetes van hoë gehalte te vestig wat in verskillende mates ooreenkomste met 'n verwysingstel busroetes (tipies die huidige stel operasionele roetes) toon, en bied die besluitnemer alternatiewe vir busroetes wat geleidelik geïmplementeer kan word om die ontwrigting van passasiers in die busvervoernetwerk te beperk. Die doele wat in hierdie model nagestreef word, is die gelyktydige minimering van die verwagte gemiddelde passasier se reistyd en die minimering van 'n verwysingsroete-na-ontwerp-roete ooreenkomsmaatstaf. Die tweede model neem dieselfde toevoere as die eerste model hierbo, met die byvoeging van 'n verwysingsroete waarmee alternatiewe ontwerproetestelle in terme van ooreenkoms vergelyk kan word, en bied as afvoer 'n stel afruilingsoplossings volgens die model se ontwerpkriteria. Dieselfde drie-fase benaderde oplos-singsmetode hierbo beskryf, word op hierdie model toegepas, en dieselfde twee metaheuristiese implementerings word gebruik om gevalle van hierdie nuwe model op te los.

In die derde model word busfrekwensies van hoë gehalte vir elke busroete gesoek om die verwagte gemiddelde reistyd van passasiers (insluitend wagtyd, oorklimtyd en werklike reistyd) te mi-

nimeer en terselfdertyd die totale aantal busse wat 'n operateur benodig, te minimeer terwyl die gespesifiseerde frekwensies gehandhaaf word. Die derde model neem dieselfde toevoerdata as die eerste model, tesame met 'n roete waarvoor frekwensies vasgestel moet word, en lewer as afvoer 'n stel busfrekwensies waarteen busse langs die verskillende roetes ontplooi moet word, gebaseer op 'n gewenste afruiling tussen die bogenoemde twee ontwerpkriteria. Die oplossingsbenadering wat op hierdie tweedoelige minimeringsprobleem toegepas word, volg weer die bogenoemde drie fases, met die uitsondering dat slegs 'n nie-gedomineerde sorteer-genetiese algoritme ontwerp word om dit op te los.

Die eerste en derde modelle word uiteindelik op 'n spesiale gevallestudie toegepas wat op werklike data gebaseer is om sodoende die praktiese toepaslikheid van die modelleringsbenadering te illustreer.

Acknowledgements

The author wishes to acknowledge the following people and institutions for their various contributions towards the completion of this work:

- the *Department of Transport* for providing funding for the author's years of research,
- the *Stellenbosch Smart Mobility Lab* (SSML) for giving the author exposure to the field of transportation,
- the *Stellenbosch Unit for Operations Research in Engineering* (SUnORE) for its support and its pursuit of excellence, inspiring the author to work hard and always strive for more, as well as for the use of its computational facilities,
- Prof SJ Andersen for his continual support, encouragement and for providing opportunities throughout the author's engineering career,
- Prof JH van Vuuren for his continual support, insight and clear guidance during the research carried out towards this dissertation — his high standard of excellence and quality has inspired the author to strive to produce work of the same calibre,
- the author's friends, for their encouragement and great support through the years,
- Ellen and Nicky Hüsselmann, the author's parents, for their long-standing support, love, and for believing in the author,
- the Lord Jesus Christ, who gave the author the ability and the strength to complete the work.

Table of Contents

List of Reserved Symbols

Symbols in this dissertation conform to the following font conventions:		
$\boldsymbol{A}$	Symbol denoting a **matrix**	(Roman boldfaced capitals)
$\mathcal{A}$	Symbol denoting a **set**	(Calligraphic capitals)

Variables

Symbol	Meaning
$A_{\max}$	The maximum number of attempts allowed per epoch (DBMOSA algorithm)
$B_{\min}$	The minimum number of accepted moves per epoch (DBMOSA algorithm)
β	The cooling parameter (DBMOSA algorithm)
ζ	The reheating parameter (DBMOSA algorithm)
ϕ	The service regularity parameter (UTFSP model)
$C_{\max}$	The maximum number of epochs that may pass without accepting new solutions (DBMOSA algorithm)
$H_{\max}$	The maximum number of reheatings allowed per epoch (DBMOSA algorithm)
$K_{\max}$	The maximum number of epochs allowed during the entire search (DBMOSA algorithm)
$L_{\max}$	The maximum number of iterations allowed per temperature (DBMOSA algorithm)
$L_{\max}$	The required number of generations (NSGA II)
$m_{\min}$	The minimum number of bus stops along a route
$m_{\max}$	The maximum number of bus stops along a route
N	The population size (NSGA II)
r	The number of routes in the desired route set
T_0	The initial starting temperature (DBMOSA algorithm)
T_c	The temperature during epoch c (DBMOSA algorithm)

Matrices

Symbol	Meaning
$\boldsymbol{W}$	The weight matrix of the transport network
$\boldsymbol{D}$	The OD demand matrix

Sets

Symbol	Meaning
$\mathcal{A}$	The archive containing the non-dominated front (DBMOSA algorithm)
$\mathcal{C}$	The set of all feasible shortest paths in the transport network
$\mathcal{F}$	The frequency set (decision variable)
$\mathcal{F}_k$	Non-dominated front k obtained when sorting solutions (NSGA II)
$\mathcal{P}_O$	The set of objective function values achieved by the members of $\mathcal{P}_S$
$\mathcal{P}_S$	The set of non-dominated solutions
$\boldsymbol{P}_i$	The population during generation i (NSGA II)
$\boldsymbol{Q}_i$	The offspring set during generation i (NSGA II)
$\mathcal{R}$	The transit route set (decision variable)
$\mathcal{S}$	The set of shortest paths between all pairs of vertices in the transport network
θ	The set of discretised frequencies

Graph symbols

Symbol	Meaning
$\alpha_{v_i,v_j}(\mathcal{R})$	The length of a shortest path from v_i to v_j along a transit route set $\mathcal{R}$
c_a	The cost of traversing arc a (measured in minutes)
$\mathcal{E}_R$	The edges contained in the route R
$\mathcal{E}_\mathcal{R}$	The edges contained in the route set $\mathcal{R}$
$\mathcal{E}'_\mathcal{R}$	The arcs of the expanded transport network induced by the route set $\mathcal{R}$
f_a	The bus frequency operating along arc a
f_i	Frequency i in the frequency set $\mathcal{F}$
$G_\mathcal{R}$	The subgraph of the transport network induced by the route set $\mathcal{R}$
$G'_\mathcal{R}$	The subgraph of the expanded transport network induced by the route set $\mathcal{R}$
h_a	The demand flowing through arc a
H_{ir}	The demand flowing from vertex i to destination r
R_i	Route i in the route set $\mathcal{R}$
$u_{v_i,v_j}(\mathcal{R},\mathcal{F})$	The expected travel time from v_i to v_j along a transit route set $\mathcal{R}$ based on the optimal strategies transit assignment
$\mathcal{V}_R$	The vertices contained in the route R

List of Acronyms

AETT: Average expected travel time

AMALGAM: A multi-algorithm genetically adaptive multi-objective

ATT: Average travel time

CPU: Central processing unit

DBMOSA: Dominance-based multi-objective simulated annealing

DP: Disruption proportion

DRS: Dominance resistant solutions

DSS: Decision support system

FNSA: Fast non-dominated sorting algorithm

GA: Genetic algorithm

GPU: Graphical processing unit

HV: Hypervolume

ICFSGP: Initial candidate frequency set generation procedure

ICRSGP: Initial candidate route set generation procedure

KSP-ICRSGP: K shortest paths initial candidate route set generation procedure

KSP-MMV-ICRSGP: K shortest paths maximising missing vertices initial candidate route set generation procedure

LLH: Low-level heuristic

MOP: Multi-objective optimisation problem

NAP: Network analysis procedure

NSGA II: Non-dominated sorting genetic algorithm II

OD: Origin-destination

OR: Operations research

QI: Quality indicator

RSD: Replace subsets deterministic

RSDKSP: Replace subsets deterministic K shortest path

RSS: Replace subsets stochastic

RSSKSP: Replace subsets stochastic K shortest path

SA: Simulated annealing

SOP: Single-objective optimisation problem

SP-ICRSGP: Shortest paths initial candidate route set generation procedure

SP-MMV-ICRSGP: Shortest paths maximising missing vertices initial candidate route set generation procedure

SSML: Stellenbosch Smart Mobility Lab

SSMV: Stochastic maximising missing vertices

SUnORE: Stellenbosch Unit for Operations Research in Engineering

TAP: Transit assignment problem

TBR: Total buses required

TRT: Total route time

UTFSP: Urban Transit Frequency Setting Problem

UTNDP: Urban Transit Network Design Problem

UTRP: Urban Transit Routing Problem

UTRFSP: Urban Transit Routing and Frequency Setting Problem

UTSP: Urban Transit Scheduling Problem

UTTP: Urban Transit Timetabling Problem

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Contents

1.1 Background

Transport is increasingly becoming an urgent problem in urban areas. The need for transport seems to be growing, and this results in increasing travel times for commuters [115]. Most of the urban population worldwide has encountered the frustration of sitting in traffic, particularly in large cities of developing countries where public transportation systems are challenged in terms of reliability, accessibility, cost and safety [106, 139]. In general, many commuting vehicles carry only their drivers and this exacerbates the problem of traffic in urban regions, especially around central business districts [81, 84].

There are different approaches towards alleviating traffic congestion problems. One such approach is to encourage the use of public transport — the situation where more people are transported in fewer vehicles. But this endeavour is often not as simple as it sounds, due to negative experiences of commuters in respect of public transport [113]. The current trend indicates that an acceleration in private vehicle usage is leading to increased congestion [106, 113]. The passenger transport modal split of Stellenbosch, a university town in the Western Cape Province of South Africa, indicates for example, that 86.6% of passengers are transported by light vehicles, 7.5% by minibus taxi, 4.5% by bus and 1.4% by heavy vehicles [138]. Embracing public transport therefore holds great potential. Various developed countries have adopted this transport medium and are mastering it in terms of both method of implementation and adoption of technological advancements, but the concept is no good if people do not buy into public transport [84].

One of the problems that arises when using public transport in a developing country is its often haphazard availability when it is needed. This creates considerable frustration when commuters are hindered by a lack of transport or a lack of effective routes by which to transport them

to their destinations. This can leave commuters feeling helpless and not catered for, especially those who are not privileged enough to own personal vehicles [113].

In the context of Stellenbosch University, a very unique problem arises in this general context. The town was originally designed for a small population, but with the ever-increasing size of the university population and the town's attractiveness from a tourism perspective, transport demand has also increased in recent years [81]. This problem induces considerable frustration in terms of traffic congestion, parking deficits, illegal parking, long walking distances to and from parking spaces and the time delays associated with each of these.

Students are especially influenced by this phenomenon as they need to commute between lecture venues and their living quarters, often living large distances away from campus and not all owning personal vehicles. Moreover, if students do happen to own personal vehicles, there is often not enough parking space for them close to campus. These challenges may, of course, be addressed by the introduction of adequate public transport [81].

The *Matie Bus* is a service initiative launched by Stellenbosch University which caters for students [155]. This service offers various routes, with some of them fixed, as can be seen in Figure 1.1. Problems that arise with the implementation of this fixed routing system is that the routes might not meet students' destination requirements sufficiently, that the buses do not have enough capacity to satisfy transport demand, and that the buses are not at the required locations at the required times.

These problems may be addressed by adopting an optimised system design approach if time-stamped *origin-destination* (OD) transport demand data of students are known *a priori*. If the demand can thus be predicted and optimal bus routes generated to maximise the throughput of passengers, minimise travel times and minimise transport service provider operating costs, the entire system would gain in terms of efficiency of service.

The challenge, however, is to put an appropriate technology platform in place to facilitate such an optimised bus service. This routing problem cannot merely be solved by solving a mathematical equation; there are many decision variables to consider and the establishment of such a system will require the aid of computers [115].

Solving this problem involves three fundamental aspects. The first is that once the demand is known or has been estimated, the bus service optimisation process becomes a vehicle routing problem — a notoriously difficult combinatorial optimisation problem in the operations research literature. The second, and more challenging, problem is making sense of past OD pair data (under the assumption that past travel patterns are indicative of future transport demand) and finding a way to group or cluster these data in a logical and helpful manner so that they can be used to solve the vehicle routing aspect of the problem. The third is the challenge that comes with collecting the OD demand data. Traditional methodologies of collecting transport demand data are becoming obsolete as the age of big data is upon us [48]. Whereas old methods relied on the manual collection of data by conducting surveys and inputting the data manually into databases, data are today widely available and easily accessible. The challenge, however, is making sense of these data and converting them into application-specific information.

Recently, there has been an increase in on-board mobile communication device sensing for transit data collection. This approach holds advantages over survey- and automated fare collection-based methods. The principle behind *Wi-Fi* and *Bluetooth* sensors is that multiple sensors are used to record the unique *media access control* (MAC) address of each wireless communication device at points along the travel route. The re-identification of passengers over time along routes by multiple sensors allows for the reconstruction of entire trips and also the computation of travel speed [48].

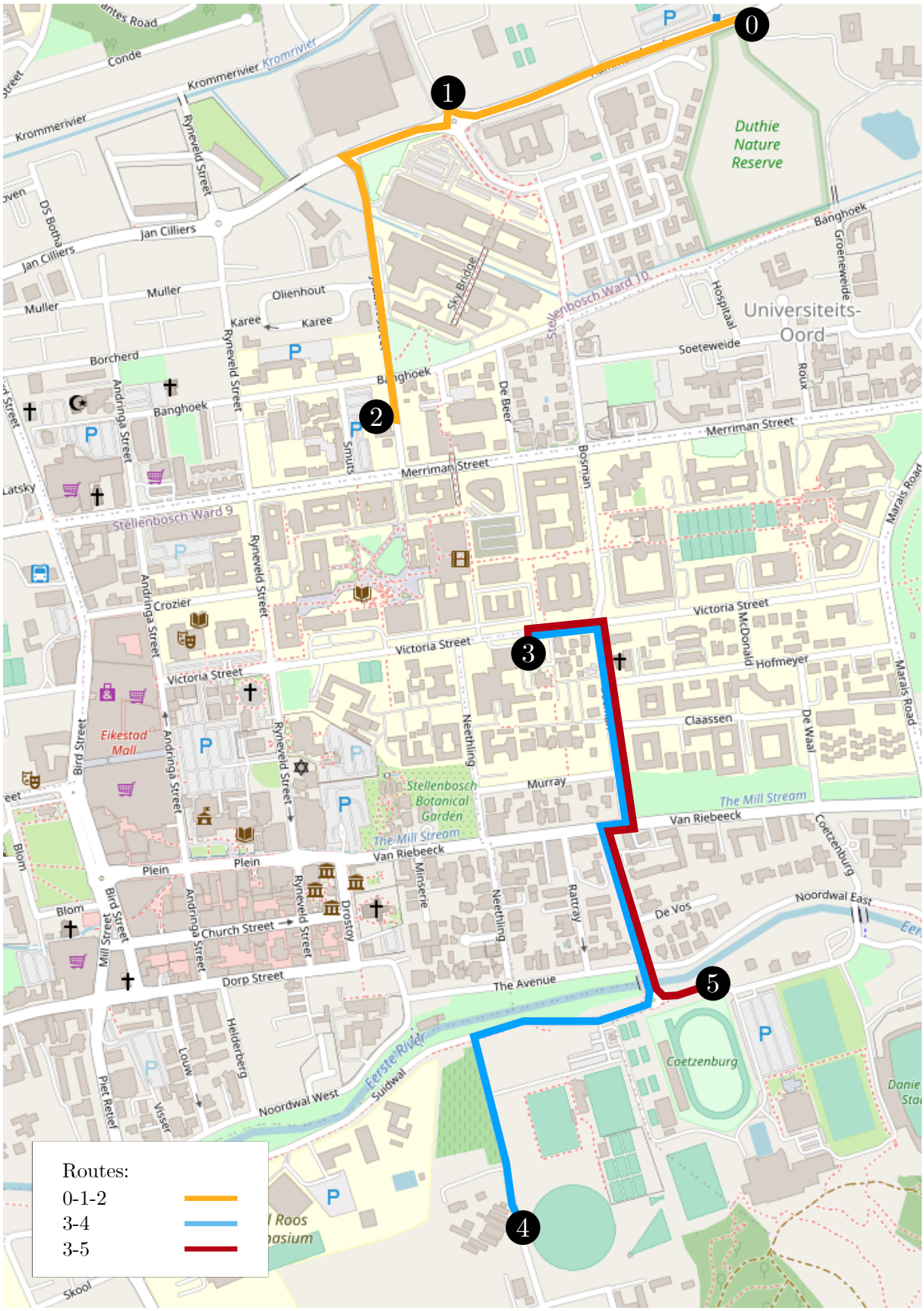


FIGURE 1.1: *Matie Bus routes for its day shuttle service [155].*

The *Stellenbosch Smart Mobility Lab* (SSML)[1] has recently established Bluetooth sensors [98], such as those shown in Figure 1.2, across the campus of Stellenbosch University to measure student movements. The SSML and the *Stellenbosch Unit for Operations Research in Engineering* (SUnORE)[2] are, furthermore, in the process of collaborating to discover possibilities of transport improvements in the Stellenbosch area with respect to the Stellenbosch University Matie Bus service provided to students.

(a) *Outside of sensor* (b) *Inside of sensor*

FIGURE 1.2: *A SMATS Wi-Fi and Bluetooth sensor attached to a lamp post [147].*

The bus service optimisation process described above is indeed a difficult problem with many decision variables and exhibiting a range of technical complexity, but if this problem can be solved in the context of the Matie Bus service, it may be very beneficial to students in terms of saving time and avoiding having to walk long distances.

1.2 Problem statement

The problem considered in this dissertation is that of designing three computerised *models* that can aid a bus company in making high-quality decisions with respect to transit route network design and appropriate bus frequencies that are associated with each transit route in order to satisfy demand. The first model, aimed at route design, should be able to take as input an OD demand matrix, representing the demand of the passengers, and the current road network structure. The second model should take the same input as the first, as well as a reference route set[3] so that the designed routes may be compared with a reference route set. The third model, aimed at setting bus frequencies for the transit routes, should be able to take the same input as the first model, together with a set of bus routes for which frequencies are to be set. A decision maker is required to provide the models with the relevant input data pertaining to a problem instance, upon which the first model should generate a set of high-quality bus routes achieving different trade-offs between operator cost[4] minimisation and passenger cost[4] minimisation. The second model should provide as output trade-off solutions between passenger cost minimisation

[1]The SSML is an *intelligent transportation systems* laboratory situated in the Department of Civil Engineering at Stellenbosch University and was established in June 2014. Its goal is to carry out research and implement intelligent transportation systems [153].

[2]The SUnORE research group is situated in the Department of Industrial Engineering at Stellenbosch University and was established in January 2015. Its goal is to support high-quality decision making through mathematical modelling [154].

[3]A reference route set, in this context, refers to the current bus routes in operation, if the network exists. If the network does not exist, this problem is not applicable.

[4]The notion of cost is to be contextualised in terms of the expenditure of valuable resources such as time and number of buses.

and a difference measurement compared with the reference route set. The third model should provide a set of high-quality bus frequencies for the bus route sets returned by the first model, also aimed at achieving different trade-offs of the aforementioned objectives. Both models should provide output for the decision maker to choose from, based on a subjectively preferred trade-off between operator cost and passenger cost[4] minimisation.

1.3 Dissertation objectives

The following eight objectives are pursued in this dissertation:

I To *conduct* a systematic study of the available literature with respect to:
   (a) *solution methodologies* for solving combinatorial optimisation problems,
   (b) models for *urban transit network design problems* (UTNDPs) in general,
   (c) models for *route design* and *frequency setting* of the UTNDP in particular, and
   (d) *solution approaches* adopted when solving UTNDP model instances.

II To *formulate* three vehicle-routing models capable of supporting the decision making process related to minimising passenger and operator cost in urban transit network design. The first model should take as input OD data of prospective passengers and provide as output a set of bus routes from which a decision maker can select a solution for implementation. An extension should also be proposed to standard urban transit network design models that allows for the incremental redesign of a transit network system. This model extension should take as input a reference bus route set and pursue a passenger cost minimisation objective while not deviating too much from the reference route set. Lastly, a frequency setting model should be formulated that can aid in the decision making process of selecting appropriate frequencies at which buses should operate along each route. The latter model should take as input a user-specified route set, along with the OD data of prospective passengers and the road network.

III To *design* an algorithm capable of suggesting high-quality bus routes and frequencies as solutions to the models of Objective II. In the routing context, an acceptable trade-off between minimising the *average travel time* (ATT) of passengers and minimising the total route traversal time, or *total route time* (TRT) should be pursued, based on the first vehicle routing model of Objective II. The same algorithm should be able to solve instances of the urban transit network design problem in which incremental redesign should be accommodated, where the TRT objective is replaced by minimising a novel design-route-to-reference-route similarity objective function. In the frequency setting context, an acceptable trade-off between minimising the *average expected travel time* (AETT) of passengers and minimising the *total buses required* (TBR) should be pursued, based on the frequency setting model of Objective II.

IV To *implement* the algorithms of Objective III in an applicable software platform. The algorithm for route setting should be flexible in terms of taking as input a user-specified data set of OD pairs together with the corresponding map of locations and current bus stops, and then provide as output a set of bus routes that achieves an acceptable trade-off between minimising the ATT of passengers and minimising the total bus route traversal time. The same algorithm should be able to return acceptable trade-off solutions between the ATT and the novel similarity objective function. The algorithm for frequency setting should also be flexible, taking information related to the OD demand matrix, the road

network with the current corresponding bus stop locations and a bus route network as input, and then producing bus frequencies for each route in the route set. These frequencies should achieve an acceptable trade-off between minimising the AETT of passengers and minimising the TBR.

V To *verify* and *validate* the algorithmic implementations of Objective IV according to generally accepted modelling guidelines.

VI To *apply* the algorithms of Objective IV to a special case study involving a real bus service.

VII To *evaluate* the effectiveness of the models and associated algorithmic implementations of Objectives III–IV in terms of their capability to identify high-quality solutions in the context of the case study of Objective VI and benchmark problems in the literature.

VIII To *recommend* sensible follow-up work related to the work documented in this dissertation which may be pursued in future.

1.4 Dissertation scope

Due to the complexity of the UTNDP described in §1.1, the scope of this dissertation is limited by the following assumptions:

Past travel patterns. The assumption is made that past travel passenger patterns are indicative of future transport demand.

Traffic considerations. For the scope and purposes of serving as a concept demonstrator, traffic considerations over time are excluded from the modelling approach underlying the models put forward in this dissertation.

Routing objective. The ATT per passenger and the total route traversal time are the transport system performance measures minimised simultaneously within the modelling approach adopted when designing bus routes in this dissertation. The aim is to vary bus routes so as to reduce the cost of inconvenience to passengers in terms of travel and transfer time, and the route maintenance cost to the operator in terms of the lengths of routes.

Incremental redesign objective. The ATT per passenger and the novel similarity measurement objective are the transport system performance measures minimised simultaneously within the modelling approach adopted when incrementally redesigning bus routes in this dissertation. The aim is to vary bus routes so as to reduce the cost of inconvenience to passengers in terms of travel and transfer time, and the route disruption experienced by passengers as a result of limiting the changes made to the transit system compared with a reference route set, when redesigning the routes.

Frequency setting objective. The AETT per passenger and the TBR are the transport system performance measures minimised simultaneously within the modelling approach adopted when choosing frequencies for bus routes. The aim is to vary bus frequency assignments to routes so as to reduce the cost of inconvenience to passengers in terms of travel, transfer and waiting time, and the route maintenance cost to the operator in terms of the number of buses required to maintain the routes.

Design parameters. During the design of the bus routes, the physical layout of the bus routes is considered, as well as the frequencies with which buses are required to traverse these routes, but not the time-tables according to which they operate.

Public transport attitude of customers. It is assumed that potential passengers will make use of the bus service when it is available to them, as opposed to walking, if this requires less time, based solely on a rational choice. This simplifies the modelling process. When frequencies are set, the option of walking is nevertheless taken into consideration.

Demand pattern. It is assumed that a fixed demand is applicable throughout the entire day so as to render the modelling approach adopted in this dissertation less complex.

Unmet demand. It is assumed that zero, one and two transfers between bus routes are acceptable for passengers, but whenever passengers have to make more than two transfers in order to travel to their destination, the situation is considered as unmet demand.

1.5 Research methodology

This section contains a brief description of the research methodology adopted in this dissertation. The research is carried out in four distinct phases. These phases are a literature review, a model development phase, a model solution implementation phase and a model solution validation stage.

The literature review of the first stage is focussed on the research areas identified in Objective I of §1.3. In order for the reader to form a clear understanding of certain modelling prerequisites introduced later in the dissertation, a review of the literature related to methods for solving combinatorial optimisation problems is conducted, in fulfilment of Objective I(a). Models of the UTNDP in general, the broader transit planning process, the main components and key characteristics of UTNDP models, and the classification of different types of UTNDPs are also studied in fulfilment of Objective I(b). These areas are covered in order to facilitate a general understanding of what typical UTNDPs entail, the different variations of UTNDPs that are available in the literature, the planning process followed when modelling UTNDP instances, and also the main components and key characteristics that constitute UTNDP instances so that a complete picture can be formed as to which inputs are required and how all the various parts of the problem are connected to one another. A more focused approach is adopted by reviewing the literature concerned with the route design and frequency setting aspects of the UTNDP, exploring separate and simultaneous solution approaches to both, in fulfilment of Objective I(c). Furthermore, UTNDP models previously adopted in the literature are reviewed in order to establish a wider understanding of the nature of these models and the different approaches adopted towards solving the models, in fulfilment of Objective I(d).

During the model development phase of the research, Objectives II–III are pursued. A bi-objective UTNDP model for designing bus routes is derived, in partial fulfilment of Objective II, which can aid operational managers of a bus transport service in terms of the establishment of appropriate routes for a fleet of available buses. The objective during the design of such a set of bus routes is to achieve a suitable trade-off between minimising passenger cost and minimising the cost of operating a bus transport service (both measured in terms of time) subject to satisfying passenger demand. This demand is assumed to have been estimated *a priori* in the form of OD demand volumes for passengers. An extension is also proposed to the aforementioned bi-objective UTNDP model in which its operator cost objective is replaced with a novel objective function for measuring the degree of similarity exhibited between a desirable route set and a reference route set. The goal of this is modelling approach is to determine a set of trade-off route sets exhibiting varying degrees of similarity (measured as a proportion) with respect to the reference route set and also varying passenger cost (measured in minutes). An operator may subjectively select one of these trade-off route sets for implementation according to the degree of

change it is willing to tolerate in terms of existing route disruption as perceived by passengers. A bi-objective UTNDP model for setting bus frequencies along candidate or established routes is further derived in final fulfilment of Objective II. This model may aid operational managers of a bus transport service in decisions related to the appropriate assignment of a number buses to each route so that the operating frequency is sufficient for meeting passenger demand timeously. The objective during the setting of these frequencies is to achieve a suitable trade-off between minimising the passenger cost (again measured in time) and minimising the cost of operating a bus transport service (measured in number of buses required) subject to satisfying passenger demand.

The design of appropriate algorithms for solving the aforementioned models approximately is pursued next in fulfilment of Objective III. The first two algorithms are designed to solve the route setting aspect of the UTNDP. These algorithms take a matrix representation of the road network available, the coordinates associated with each bus stop and an estimated passenger OD demand matrix as input for a specified transit network and suggest high-quality bus routes as output. A metaheuristic solution approach is adopted in both algorithms, involving a trajectory-based search in the first algorithm and a population-based search in the second algorithm. Both of these metaheuristics are equipped with an overarching hyperheuristic that aids in facilitating the management of lower-level operators utilised. The relative performances of the two different types of metaheuristics are compared, and the possibility of combining the two approaches is explored. The same two algorithms may also be used during an incremental redesign of a given transit network (serving the purpose of reference route set), this time returning as output route sets exhibiting varying degrees of similarity to the reference route set. Once a suitable route set has been established, another algorithm is designed for establishing high-quality frequencies of buses operating along each route, taking as input all the input of the route design problem mentioned above, as well as the route set decided upon. A population-based search approach is adopted for this purpose.

During the model implementation phase, the aforementioned solution methodologies for each of the mathematical models of Objective II are implemented in a suitable software environment. These implementations are capable of making high-quality recommendations aimed at pursuing acceptable trade-offs between minimising passenger cost and minimising operator cost based on the demand patterns estimated *a priori*. The implementations are generic in the sense that it is possible to provide different scenario input data sets to the algorithms for which new high-quality routing solutions and frequency settings can be computed. These algorithmic implementations stand in fulfilment of Objective IV.

The fourth phase of the research is the model solution validation phase carried out in pursuit of Objectives V–VII. The algorithmic implementations are first verified and then validated according to generally accepted modelling guidelines in the context of well-known benchmark test problem instances. This verification and validation process stands in fulfilment of Objective V.

A special case study is conducted thereafter during which both the routing and frequency setting models are solved by means of the implemented algorithms for the Matie Bus service of Stellenbosch University with the aim of recommending efficiency improvements for this student transport service, in fulfilment of Objective VI. This is achieved by taking Stellenbosch University student OD demand data as input to the algorithmic implementations and generating routes, as well as the associated frequencies at which buses should operate along these routes, as a recommendation to the Matie Bus service. This recommendation is aimed at achieving a desirable trade-off between minimising the travel time of student passengers and minimising the Matie Bus operating cost.

In pursuit of Objective VII, an evaluation is carried out in respect of the practical applicability of the models and their associated algorithmic implementations of Objectives III–VI, assessed in terms of their capability of generating high-quality solutions in the context of the case study of Objective VI.

Recommendations are finally made with respect to sensible follow-up work which may be carried out in future, in fulfilment of Objective VIII.

1.6 Dissertation organisation

Apart from the current stand-alone Chapter 1, this dissertation is partitioned into four parts, namely a literature study part, a mathematical modelling part, a case study part and a conclusion part. The literature review part consists of three chapters, numbered 2 to 4. Chapter 2 is devoted to a review of mathematical prerequisites for the work presented in the remainder of the dissertation, covering topics in graph theory, multi-objective optimisation, methods for non-dominated sorting of solution sets, and multi-objective optimisation solution quality evaluation. A reader who is versed in the aforementioned concepts of Chapter 2 may profitably skip the chapter, and only refer back to it when necessary. Next, Chapter 3 deals with popular solution approaches in the literature for solving combinatorial optimisation problems — the type of problems considered in this dissertation — in fulfilment of Objective I(a) of §1.3. Chapter 4 is devoted to a literature review on the UTNDP, covering the topics mentioned briefly in §1.5, in fulfilment of Objectives I(b)–(d) of §1.3.

The next part, called the mathematical modelling part, is the heart of this dissertation, in which mathematical models for the routing, incremental redesign, and frequency setting applications of the UTNDP are derived, implemented, verified and validated. These chapters are numbered Chapters 5–7. Chapter 5 contains derivations of the mathematical models, both for the route design aspects and the frequency setting aspect of the UTNDP, in fulfilment of Objective II of §1.3. Chapter 6 contains a description of the design and implementation of various algorithms for solving the models of Chapter 5, in fulfilment of Objectives III and IV of §1.3. In Chapter 7, the solution approaches of Chapter 6 are verified and validated based on well-known benchmark problem instances, in fulfilment of Objective V of §1.3.

The third part of this dissertation contains Chapter 8, a special case study in which the model implementations are applied to the route design problem for a real bus service, along with the setting of appropriate frequencies at which buses should operate along each route, in fulfilment of Objective VI of §1.3.

The fourth and final part is the conclusion, containing Chapters 9 and 10. A summary of the dissertation contents is presented in Chapter 9, along with a critical evaluation of the dissertation contributions, while recommendations for future work related to this dissertation are presented in Chapter 10, in fulfilment of Objectives VII and VIII of §1.3.

Part I

Literature study

CHAPTER 2

Prerequisites

Contents

This chapter is devoted to a discussion on certain basic concepts and terminologies that are required in later chapters of this dissertation. First, a short overview is given in §2.1 of a well-known approach towards optimisation within the context of the broader field of *operations research* (OR). An introduction to certain basic concepts in graph theory is next provided in §2.2, along with three well-known algorithms for finding shortest paths in weighted graphs. An introduction to basic notions in multi-objective optimisation then follows in §2.3. Methods for determining sets of non-dominated solutions within the objective space of a multi-objective optimisation problem are explored in §2.4, after which a discussion follows in §2.5 on the sorting of solutions into non-dominated fronts. The importance of, and methods for, the evaluation of the quality of solutions sets are considered in §2.6. Finally, the chapter closes in §2.7 with a brief summary of its contents.

2.1 An operations research approach towards optimisation

The notion of optimisation is often encountered when dealing with situations in which a best solution is sought to a particular problem at hand. Such a solution is called an *optimal solution*. This usually means that the solution achieves the largest or smallest value among all the candidate solutions in respect of a certain objective function. Optimisation is therefore the process of finding a best solution among all possible alternatives [29]. Winston [171] describes OR as a scientific approach towards decision making, in which one seeks to best design and/or operate a system, usually under certain resource constraints. A system, in this context, refers to an organisation of interdependent components that work together to realise one or more objectives. The scientific approach towards decision making traditionally involves the use of mathematical model formulations of real world problems [171]. It should be noted, however, that this view

of OR is a very narrow one, and by no means encapsulates the full scope of what OR is [165]. It was common during the 1950s and 1960s, as OR models were emerging, to define OR as mainly revolving around the construction of mathematical models and searching for optimal or near-optimal solutions to these models, but since then has evolved into a massive field which cannot be fully described in a concise manner, because of the many multi-disciplinary sub-fields that it straddles [165].

Another definition, given by the Association of European Operational Research Societies [159], is that "Operational research may be described as a scientific approach to the solution of problems in the management of complex systems. In a rapidly changing environment an understanding is sought which will facilitate the choice and the implementation of more effective solutions which, typically, may involve complex interactions among the elements of the system, for instance, people, materials and money. Operational Research has been used intensively in business, industry and government." This modern definition leaves space for the myriad different applications and approaches towards describing OR. Nonetheless, the use of mathematical models in OR still finds application in many real-world problems today, making it as relevant as ever [165].

OR models typically consist of three basic elements, namely variables, objective functions and constraints. Variables represent objects or quantities that can be manipulated to quantify some measure of a desired objective or outcome. Objective functions measure the effectiveness of some combination of variable values, and are the functions that should be optimised. A model may have multiple objective functions that have to be optimised simultaneously. Examples of quantities represented by objective functions are profit, revenue and cost. Constraints are equations or inequalities that should be satisfied by variable value combinations when attempting to find an optimal solution. These constraints are often called feasibility conditions, and a solution satisfying these conditions is said to be a feasible solution, while the violation of any of the conditions leads to a so-called infeasible solution [29].

In certain cases, mathematical modelling may lead to models that are too complex to solve, even by advanced computers, and this situation often leads to the intentional creation of a *heuristic* procedure as an alternative solution methodology [29]. Ceder [29] provided an outline of a framework for developing OR models in aid of solving deterministic optimisation problems, a modelling paradigm generally adopted. This framework is depicted in the flow diagram in Figure 2.1, and consists of eight steps.

The first step is to define the problem considered and the main objectives that are to be pursued, and secondly, to identify the underlying system's limitations and constraints. The third step involves the formulation of a mathematical model based on the output of the first two steps. Once the mathematical model has been established, the fourth step consists of performing a test to ascertain whether feasible solutions exist for the given model. If no feasible solution can be found, the analyst returns to step one and re-evaluates the system objectives and constraints, considering possible changes to or relaxations of some constraints. If, however, a feasible solution is found, the analyst proceeds to step five, during which a software package is selected for implementing the solution approach, or alternatively, an algorithm is developed afresh that can solve the OR model. The sixth step involves examination of the complexity of the algorithm, and determining whether the model can be solved by a computer within a reasonable amount of run time. If the model is deemed too complex, a heuristic procedure has to be constructed that is able to solve the problem without guaranteeing optimality; otherwise the model is solved exactly. After the appropriate solution methodology has been established, a sensitivity analysis is conducted as the seventh step, during which the sensitivity of the objective function is tested with respect to different input data. The final step involves a presentation to the relevant

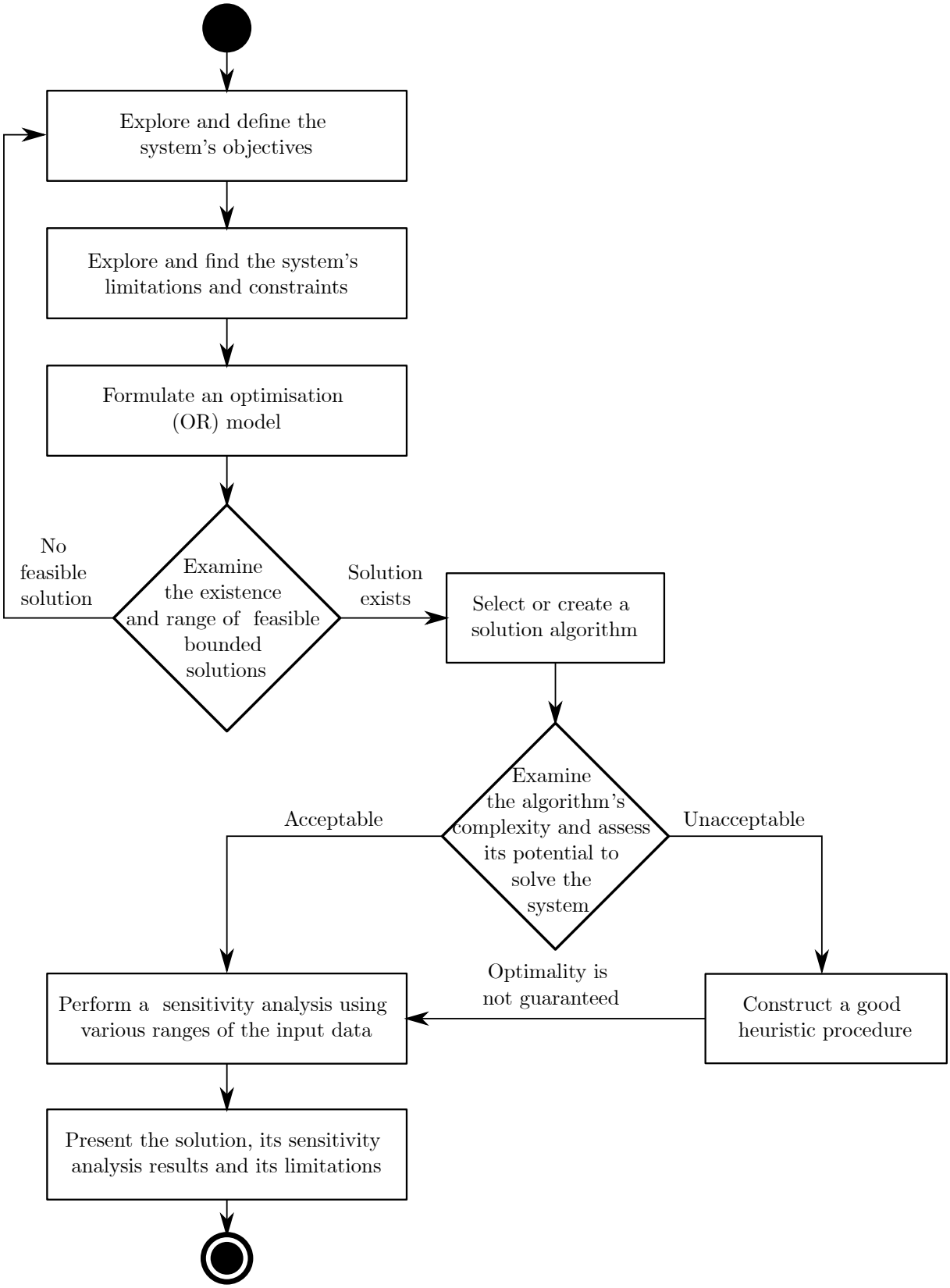


FIGURE 2.1: *Schematic framework of the OR process when solving a deterministic optimisation problem analytically [29].*

decision maker of the results, along with the sensitivity analysis, solution(s) and the model's limitations [29].

2.2 Graph theoretic preliminaries

The transit route design problem may be described succinctly in terms of a number of concepts from the realm of *graph theory*. Graphs are used extensively to model the structure embodied in and the interaction between networks of entities [73]. Some basic graph theoretic concepts and terminology are described, after which two famous shortest path finding algorithms are reviewed.

2.2.1 Basic concepts and terminology

A *graph* G is a finite, non-empty set of *vertices* together with a set of distinct unordered vertex pairs, called *edges*. The collection of all the vertices of G is denoted by $\mathcal{V}(G)$ and is called the *vertex set* of G, while the collection of all the edges of G is denoted by $\mathcal{E}(G)$ and is called the *edge set* of G. An edge e, consisting of two vertices u and v in G, is denoted by $e = \{u, v\}$ and is said to *join* the two vertices, which are called *adjacent* in G. Furthermore, if $e = \{u, v\}$ is an edge of G, then e is said to be *incident* with the vertices u and v. The cardinality of $\mathcal{V}(G)$ is called the *order* of G while the cardinality of $\mathcal{E}(G)$ is called the *size* of G [73].

A *subgraph* H of a graph G, written $H \subseteq G$, is a graph embedded within G, all of whose vertices belong to $\mathcal{V}(G)$ and all of whose edges belong to $\mathcal{E}(G)$. A *spanning subgraph* of a graph G is a subgraph of G containing all the vertices of G. The subgraph of G *induced* by some subset $\mathcal{E}' \subseteq \mathcal{E}(G)$ is that subgraph of G containing all the edges of $\mathcal{E}'$ as well as the vertices incident with these edges [73].

A *u-v path* in a graph G is an ordered sequence of distinct vertices of G, each successive pair of which are adjacent vertices in G, and which starts at the vertex u and ends at the vertex v. A graph G is *connected* if there exists a u-v path in G for every pair of vertices $u, v \in \mathcal{V}(G)$ [73].

A graph is usually represented graphically by drawing its vertices as dots or circles, and its edges as lines or curves joining pairs of these vertices. Figure 2.2 contains an example of a graphical representation of the graph G_1 with vertex set $\mathcal{V}(G_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and edge set $\mathcal{E}(G_1) = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_5\}, \{v_3, v_5\}, \{v_4, v_5\}, \{v_6, v_7\}\}$ [73]. This graph has order 7 and size 7. The vertices v_1 and v_2 are adjacent in G_1 (because they are both incident with the edge $\{v_1, v_2\}$, which joins them), while the vertices v_1 and v_3 are not adjacent in G_1. An example of a path in G_1 is the v_3-v_4 path v_4, v_1, v_2, v_3. The graph G_1 is not connected, because there is (for example) no v_3-v_6 path in G_1.

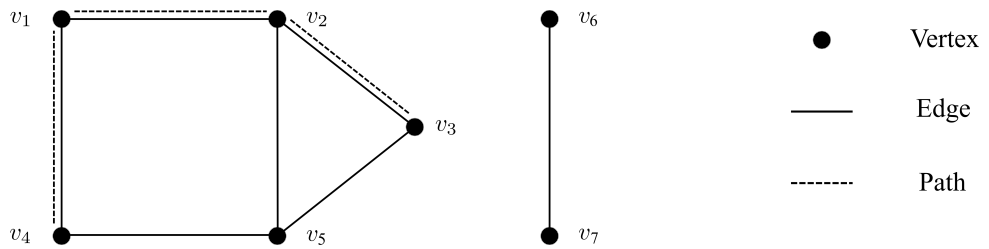
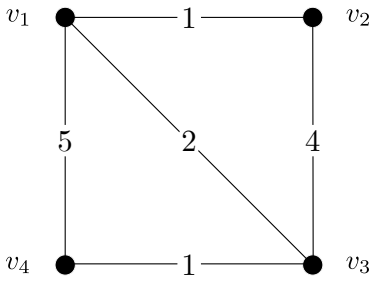


FIGURE 2.2: *Graphical representation of a graph G_1.*

A *weighted graph* is obtained by associating a real value with each of its edges. A weighted graph G_2 of order 4 and size 5 is illustrated graphically in Figure 2.3(a). Such a graph may be represented on a computer by means of a *weight matrix*. The weight matrix of a weighted graph G of order n, denoted by $\boldsymbol{W}(G)$, is an $n \times n$ symmetric matrix containing zeros on its main diagonal and which contains as entry in row i and column j the weight w_{ij} associated with the edge $\{v_i, v_j\}$ of G if such an edge exists in G, or the symbol ∞ otherwise. The weight matrix of the graph G_2 in Figure 2.3(a) is shown in Figure 2.3(b).



(a) G_2

	v_1	v_2	v_3	v_4
v_1	0	1	2	5
v_2	1	0	4	∞
v_3	2	4	0	1
v_4	5	∞	1	0

(b) $\boldsymbol{W}(G_2)$

FIGURE 2.3: *Graphical representation of a weighted graph G_2 and its corresponding weight matrix $\boldsymbol{W}(G_2)$.*

The *length* of a path in a weighted graph G is the sum of the weights of all the edges between pairs of adjacent vertices along the path. A *shortest path* between two vertices u and v in a connected graph G is a u-v path of minimum length in G. This minimum length is called the *distance* between u and v in G. The distance between the vertices v_2 and v_4 in the weighted graph G_2 of Figure 2.3(a), for example, is 4, which is the length of the shortest v_2-v_4 path v_2, v_1, v_3, v_4 in G_2.

Within the context of transit network design, the vertices in a weighted graph model of the network represent vehicle stops, while edges represent shortest road links between these stops. The weights of the edges may represent the lengths or traversal time durations of these shortest road links. A route in a transit network is simply a path in the weighted graph model.

2.2.2 Shortest paths in weighted graphs

When considering public transit networks, modelled as weighted graphs, finding shortest paths between all vertex pairs is a critical operation. In this section, two well-known shortest path finding algorithms are reviewed, namely Dijkstra's algorithm and Floyd's algorithm. Furthermore, Yen's algorithm is also reviewed, used to find the K shortest loop-less paths in a weighted graph.

Dijkstra's algorithm

Dijkstra's algorithm is a well-known algorithm for finding shortest paths from a fixed vertex x in a weighted graph G, all of whose edge weights are positive, to all other vertices in G [45]. The algorithm functions by assigning (and repeatedly updating) a label $l(v)$ to each vertex v of G. At any point during execution of the algorithm, the label represents the length of a shortest path from x found to v so far. These labels are decreased as shorter and shorter paths are found from x to v. At initialisation of the algorithm, the labels are assigned the values $l(x) = 0$ and $l(v) = \infty$ for every other vertex v of G. During execution of the algorithm, a variable parent(v)

is maintained for each vertex $v \neq x$, containing the vertex directly preceding v on a shortest x-v path discovered thus far [73].

The algorithmic progression is governed by a so-called *current vertex*. Initially, the current vertex is taken as x. Thereafter, any vertex adjacent to the current vertex and with the smallest label becomes the new current vertex during the next iteration of the algorithm. The label of the current vertex is made permanent (*i.e.* is not updated during any subsequent iterations of the algorithm). Each vertex u with a non-permanent label adjacent to the current vertex v is considered in turn, and the present label $l(u)$ is compared with $l(v) + w(vu)$. If this value is smaller than $l(u)$, where $w(vu)$ represents the weight of the edge vu in G, then the label $l(u)$ is replaced by $l(v) + w(vu)$. If the label of u is thus updated, the variable parent(u) is assigned the value v. This process is repeated until all vertices have received permanent labels (*i.e.* have had a turn to be designated the current vertex). At termination of the algorithm, all vertices will have permanent labels representing their distances from x. For a fixed vertex, the algorithm has an $\mathcal{O}(n^2)$ time complexity, where n denotes the order of the weighted input graph. Therefore, the process of computing shortest paths between all pairs of vertices of such a weighted input graph has an $\mathcal{O}(n^3)$ time complexity [73].

A pseudo-code description of Dijkstra's algorithm is presented in Algorithm 2.1. The set $\mathcal{S}$ in the pseudo-code description of the algorithm represents the collection of vertices of the input graph that have not yet received permanent labels. This set is updated during each iteration of the algorithm by removing the current vertex from it. The algorithm therefore terminates once $\mathcal{S}$ is empty.

Algorithm 2.1: Dijkstra's shortest path algorithm adapted from [73]

Input : A weighted graph G of order n with only positive weights; a vertex $x \in V(G)$.
Output: A set of shortest paths from vertex x to every other vertex v in G.

```
1  S ← V(G)
2  foreach v ∈ V(G)\{x} do l(x) ← ∞
3  l(x) ← 0, parent(x)← x
4  while |S| ≠ 1 do
5      u ← v where l(x) = min {l(w) w ∈ S}
6      foreach v ∈ S do
7      if uv ∈ E(G) and l(v) > l(u) + w(uv) then
8          l(v) ← l(u) + w(uv)
9          parent(v) ← u
10     S ← S\{u}
```

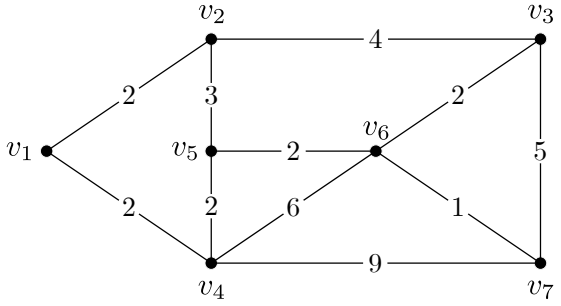


FIGURE 2.4: *Graphical representation of a weighted graph G_3 [73].*

TABLE 2.1: *The sequential steps followed when applying Algorithm 2.1 to the weighted graph G_3 in Figure 2.4, when $x = v_1$. In each ordered pair, the first entry represents the label l of the vertex in the column heading, whereas the second entry represents the parent of the vertex in the column heading within a current shortest path from v_1 to that specific vertex [73].*

v_1	v_2	v_3	v_4	v_5	v_6	v_7	S
$(0, v_1)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$
—	$(2, v_1)$	$(\infty, -)$	$(2, v_1)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$\{v_2, v_3, v_4, v_5, v_6, v_7\}$
—	—	$(6, v_2)$	$(2, v_1)$	$(5, v_2)$	$(\infty, -)$	$(\infty, -)$	$\{v_3, v_4, v_5, v_6, v_7\}$
—	—	$(6, v_2)$	—	$(4, v_4)$	$(8, v_4)$	$(11, v_4)$	$\{v_3, v_5, v_6, v_7\}$
—	—	$(6, v_2)$	—	—	$(6, v_5)$	$(11, v_4)$	$\{v_3, v_6, v_7\}$
—	—	—	—	—	$(6, v_5)$	$(11, v_4)$	$\{v_6, v_7\}$
—	—	—	—	—	—	$(7, v_6)$	$\{v_7\}$

The working of the algorithm is illustrated by computing shortest paths from the vertex $x = v_1$ in the weighted graph G_3 of Figure 2.4 to all other vertices in the graph [73]. The progression of the algorithm is presented in Table 2.1. From the seventh column of Table 2.1, for example, it is deduced that the distance between the vertices v_1 and v_7 in G_3 is 7. A shortest v_1-v_7 path in G_3 is traced back from v_7 as v_1, v_4, v_5, v_6, v_7.

Floyd's algorithm

Floyd's algorithm [59] is another well-known procedure for finding shortest distances between all the pairs of vertices in a weigthed graph, which may or may not be connected, and may contain edges with negative weights. Dijkstra's algorithm, on the other hand, only applies to graphs in which all edge weights are positive [45]. A prerequisite for Floyd's algorithm, however, is that no negative cycles are allowed (*i.e.* cycles are not allowed to have lengths that are negative). The reason for this is that when a negative cycle is traversed repeatedly, no lower bound exists for a shortest walk along the negative cycle [73].

Consider a weighted graph G of order n, with an associated weight matrix $\boldsymbol{D}^{(0)} = [d_{ij}^{(0)}]$. Suppose that a matrix $\boldsymbol{D}^{(k-1)}$ is known, in which the (i, j)-th entry $d_{ij}^{(k-1)}$ denotes the length of a shortest v_i-v_j path, making use of only the first $k-1$ vertices $v_1, \ldots, v_{k-1}$ of G as internal vertices of the path. Intuitively, $d_{ij}^{(k)} \leq d_{ij}^{(k-1)}$ for all $i, j, k \in \{1, \ldots, n\}$, due to the additional vertex yielding more flexibility in finding shorter v_i-v_j paths when only considering the first k vertices $v_1, \ldots, v_k$, instead of the first $k-1$ vertices $v_1, \ldots, v_{k-1}$ of G as internal path vertices. If no v_i-v_j path, using only the first k vertices of G, is shorter than one using only the first $k-1$ vertices, then it is true that $d_{ij}^{(k)} = d_{ij}^{(k-1)}$. If, however, there is a v_i-v_j path, denoted by P, that utilises only the first k vertices of G, and P is shorter than any v_i-v_j path that utilises only the first $k-1$ vertices of G, then the vertex v_k appears in the path P, and therefore $d_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ because of the hereditary property of shortest paths [73].

When these two cases above are combined, it is possible to construct a matrix $\boldsymbol{D}^{(k)}$, whose (i, j)-th element is

$$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}, \tag{2.1}$$

where $d_{ij}^{(k)}$ denotes the length of a shortest path v_i-v_j, when only utilising the first k vertices $v_1, \ldots, v_k$ of G as internal vertices, using only the information contained in $\boldsymbol{D}^{(k-1)}$. Let the weight matrix, $\boldsymbol{D}^{(0)}$, be the initial matrix. Then a sequence $\boldsymbol{D}^{(0)}, \ldots, \boldsymbol{D}^{(n)}$ of $n \times n$ matrices

can be formed by (2.1). The final matrix in this sequence, $\boldsymbol{D}^{(n)}$, will contain the shortest distances between each v_i-v_j vertex pair of G, due to the fact that the matrix entries represent shortest v_i-v_j path lengths, and in the final case, every vertex v may be utilised as internal path vertex. The procedure described above leads naturally to the pseudo-code description given in Algorithm 2.2.

Algorithm 2.2: Floyd's shortest path algorithm adapted from [73]

Input : A weight matrix $\boldsymbol{D}^{(0)} = [d_{ij}^{(0)}]$ of a weighted graph G of order n which may contain no negative cycles.

Output: A matrix $\boldsymbol{D}^{(n)} = [d_{ij}^{(n)}]$ of shortest distances between all pairs of vertices of G.

```
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            d_ij^(k) = min{ d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1) }
return [D^(n)]
```

The working of Algorithm 2.2 is illustrated by means of an example [73] in which the shortest distances between all pairs of vertices in the graph G_3 of Figure 2.4 are sought. Floyd's algorithm is used to compute the sequence of matrices $\boldsymbol{D}^{(0)}, \ldots, \boldsymbol{D}^{(7)}$, given by

$$\boldsymbol{D}^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & \infty & 3 & \infty & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 & 5 \\ 2 & \infty & \infty & 0 & 2 & 6 & 9 \\ \infty & 3 & \infty & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix}, \quad \boldsymbol{D}^{(1)} = \begin{bmatrix} 0 & 2 & \infty & 2 & \infty & \infty & \infty \\ 2 & 0 & 4 & 4 & 3 & \infty & \infty \\ \infty & 4 & 0 & \infty & \infty & 2 & 5 \\ 2 & 4 & \infty & 0 & 2 & 6 & 9 \\ \infty & 3 & \infty & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix},$$

$$\boldsymbol{D}^{(2)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 5 & \infty & \infty \\ 2 & 0 & 4 & 4 & 3 & \infty & \infty \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 5 & 3 & 7 & 2 & 0 & 2 & \infty \\ \infty & \infty & 2 & 6 & 2 & 0 & 1 \\ \infty & \infty & 5 & 9 & \infty & 1 & 0 \end{bmatrix}, \quad \boldsymbol{D}^{(3)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 5 & 8 & 11 \\ 2 & 0 & 4 & 4 & 3 & 6 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 5 & 3 & 7 & 2 & 0 & 2 & 12 \\ 8 & 6 & 2 & 6 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 12 & 1 & 0 \end{bmatrix},$$

$$\boldsymbol{D}^{(4)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 8 & 11 \\ 2 & 0 & 4 & 4 & 3 & 6 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 6 & 9 \\ 4 & 3 & 7 & 2 & 0 & 2 & 11 \\ 8 & 6 & 2 & 6 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 11 & 1 & 0 \end{bmatrix}, \quad \boldsymbol{D}^{(5)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 11 \\ 2 & 0 & 4 & 4 & 3 & 5 & 9 \\ 6 & 4 & 0 & 8 & 7 & 2 & 5 \\ 2 & 4 & 8 & 0 & 2 & 4 & 9 \\ 4 & 3 & 7 & 2 & 0 & 2 & 11 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 11 & 9 & 5 & 9 & 11 & 1 & 0 \end{bmatrix},$$

$$\boldsymbol{D}^{(6)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 7 \\ 2 & 0 & 4 & 4 & 3 & 5 & 6 \\ 6 & 4 & 0 & 6 & 4 & 2 & 3 \\ 2 & 4 & 6 & 0 & 2 & 4 & 5 \\ 4 & 3 & 4 & 2 & 0 & 2 & 3 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 7 & 6 & 3 & 5 & 3 & 1 & 0 \end{bmatrix} \text{ and } \boldsymbol{D}^{(7)} = \begin{bmatrix} 0 & 2 & 6 & 2 & 4 & 6 & 7 \\ 2 & 0 & 4 & 4 & 3 & 5 & 6 \\ 6 & 4 & 0 & 6 & 4 & 2 & 3 \\ 2 & 4 & 6 & 0 & 2 & 4 & 5 \\ 4 & 3 & 4 & 2 & 0 & 2 & 3 \\ 6 & 5 & 2 & 4 & 2 & 0 & 1 \\ 7 & 6 & 3 & 5 & 3 & 1 & 0 \end{bmatrix}.$$

It should be noted that $\boldsymbol{D}^{(0)}$ corresponds to the weight matrix of the graph G_3 and that the first row of entries in $\boldsymbol{D}^{(7)}$ corresponds to the permanent labels found by means of Dijkstra's algorithm in Table 2.1. Floyd's algorithm can also be modified so as to retain shortest paths corresponding to each vertex pair.

Yen's algorithm

Yen's algorithm [179], or Yen's K shortest loop-less path finding algorithm as it is commonly known, is an algorithm searching not only for the shortest loop-less path in a weighted graph G between a source vertex v_s and a target vertex v_t, but also the second shortest path, the third shortest path, and so on, until the K shortest paths have been found. A set of K shortest loop-less paths is then returned by the algorithm. A potential application may entail having to find the next shortest route if, for some reason, any of the shorter routes are not available for traversal, such as in a transportation network, for instance, where some natural disaster could have rendered certain routes unusable, and finding a new shortest path is still important, given the circumstances. Yen's algorithm is known as a deviation algorithm [118], because new paths are generated by deviating slightly from other paths.

Consider a connected graph G of order n and having positive weights assigned to each edge, denoted by d_{ij} for the edge joining vertex v_i and vertex v_j, with $i \neq j$. If no edge exists between vertices v_i and v_j, then $d_{ij} = \infty$. Let $v_s, (i), \ldots, v_t$ be a path from v_s to v_t that passes through $(i), \ldots$, where (i) represents the i-th vertex along the path and where the ellipses represent all the internal vertices between (i) and v_t. It is furthermore required that $v_s \neq (i) \neq \cdots \neq v_t$, so that the path considered is loop-less (*i.e.* no vertex appears more than once in the path). Consider, for example, the path $v_2, v_4, v_3, v_5, v_1, v_6$. In this case, v_s corresponds to v_2, v_t corresponds to v_6, and (1), (2), (3) and (4) correspond to v_4, v_3, v_5 and v_1, respectively. If $i = 2$, indicating interest in the second vertex position along the path, then $(i), \ldots, v_t$, would represent v_3, v_5, v_1, v_6. Let the k-th shortest path in G be denoted by $A^k = v_s, (2^k), (3^k), \ldots, (Q^k), v_t$, for any $k \in \{1, \ldots, K\}$. Here, (2^k), (3^k), ... and (Q^k) denote respectively the second, the third, the remaining internal vertices, and the Q-th vertex of the k shortest path.

Next, let A_i^k be a set of deviations from the $(k-1)$-th shortest path, A^{k-1}, at vertex position $i \in \{1, 2, \ldots, Q_k\}$, where the vertex itself is denoted by (i). A deviation from A^{k-1} at (i) is defined as the shortest of the paths coinciding with A^{k-1}, starting at v_s and ending at (i) along the path of A^{k-1}, and then deviating to a different vertex $(i+1)$ not included in any of the $k-1$ shortest paths already found, denoted by A^j for $j \in \{1, 2, \ldots, k-1\}$, given that the compared path should follow the same subpath from v_s to (i). The deviation should also reach v_t as the last vertex in the path, given that no loops are present and that no duplicate vertices are present in the path. The first part of the deviation path A_i^k, from vertex v_s to (i), which coincides with A^{k-1}, is called the *root* of A_i^k and is denoted by R_i^k (for example $v_s, (2^k), \ldots, (i^k)$ in A_i^k). The last part of A_i^k, from vertex (i) to v_t, is called the *spur* of A_i^k and is denoted by S_i^k (for example, $(i^k), \ldots, v_t$ in A_i^k).

The working of Yen's algorithm may be explained based on the pseudo-code description in Algorithm 2.3. The algorithm takes as input a positively weighted connected graph G of order n, an integer K representing the number of shortest paths that should be returned, a source vertex v_s and a target vertex v_t. As output, the algorithm delivers a set of K shortest paths based on the weights of the edges, all starting at v_s and ending at v_t. For the initialisation step in Line 1 of Algorithm 2.3, a shortest path is determined between v_s and v_t in G by invoking any efficient shortest path finding algorithm, such as Dijkstra's algorithm (described above). An implementation of such a shortest path finding algorithm is assumed to populate the `shortest_path` function. If multiple shortest paths are found, all are returned and placed in a list B. If B contains more than K entries at this stage, the algorithm terminates and returns K randomly chosen paths from the list B by invoking the function `random_choices`. If, however, the number of entries in B is smaller than K, the algorithm continues and initialises the first shortest path A^1 by randomly choosing a path from the list B, removing it from B and inserting it in a list A.

Algorithm 2.3: Yen's K shortest path algorithm [179]

Input : A weighted graph G of order n with only positive weights, the number K of shortest paths that should be sought, a source vertex $v_s \in V(G)$ and a target vertex $v_t \in V(G)$.

Output: A set A of K shortest paths from the source vertex v_s to the target vertex v_t in G.

```
1  B ← shortest_path(G, v_s, v_t)
2  if |B| ≥ K then
3      return random_choices(B, K)
4  else
5      A^1 ← random_choice(B)
6      B ← B \ {A^1}
7  for k = 2 to K do
8      for i ∈ {1, 2, ..., Q_{k-1}} do
9          R_i^k ← R_i^{k-1}
10         for j ∈ {1, ..., k − 1} do
11             if R_i^k = R_i^j then
12                 q ← (i + 1) ∈ A^j
13                 d_{(i)q} ← ∞
14         G_a ← Remove vertices of R_i^k except (i) from G
15         if shortest_path(G_a, (i), v_t)) is feasible then
16             S_i^k ← random_choice(shortest_path(G_a, (i), v_t))
17             A_i^k ← R_i^k + S_i^k
18             if A_i^k ∉ B then
19                 B ← B ∪ A_i^k
20         G ← Restore original graph G
21     A^k ← get_minimum_length_path(B)
22     B ← B \ {A^k}
23     return A
```

The for-loop spanning Lines 7 to 23 represents the algorithmic alterations required to determine each k-th shortest path, for $k \in \{2, 3, \ldots, K\}$. A^k can only be determined once the shortest paths

$A^1, A^2, , \ldots, A^{k-1}$ have been found. The path A^k is found by searching through each vertex (i) at position i of A^{k-1}. For each such location i, the root R_i^k is set equal to the root R_i^{k-1}. An equality comparison is then performed between the root R_i^k and the root R_i^j of each path already contained in the list A, with $j \in \{1, \ldots, k-1\}$. If two roots are found to be equal, the variable q is set to the next vertex $(i+1)$ in the sequence of path A^j. The weight corresponding to the edge $\{(i), q\}$ is set equal to ∞ so that the search will not consider that same edge for inclusion in a shortest path at a later stage of algorithmic execution, thereby ensuring the occurrence of a deviation.

In essence, a new graph G is generated by altering the weights in the for-loop spanning Lines 10–13 of Algorithm 2.3. In Line 14, the graph G is further altered by removing the vertices contained in R_i^k, except for vertex (i), so that, if a feasible shortest path is found in Line 15, a shortest path can then be determined from (i) to v_t in the altered graph G_a, yielding the spur S_i^k in Line 16. If multiple paths are retuned by the `shortest_path` function, one of them is randomly chosen by applying the `random_choice` function to the output of the `shortest_path` function. Upon having removed the vertices contained in R_i^k from the graph G, none of those vertices can be included in the spur S_i^k so that when the root and spur are concatenated in Line 17 of Algorithm 2.3, the result will be a loop-less path. The concatenation of R_i^k and S_i^k is stored in A_i^k and added to the list B if it is not yet present in B, as checked in Line 18. Thereafter, the original graph G is restored in Line 20 of Algorithm 2.3. Only the $K - k + 1$ shortest paths have to maintained in the list B throughout the search [179].

Next, when all the path deviations have been added to the list B after having performed all iterations over i, a path of the minimum length may be removed from the list B and assigned to A^k, as described in Lines 21 and 22 of Algorithm 2.3. After all iterations over k have been completed, the algorithm terminates and returns the list A which at that point contains a set of K shortest paths from vertex v_s to vertex v_t.

The working of the algorithm is elucidated by means of a worked example. Consider again the weighted graph in Figure 2.4, containing only non-negative weights. Suppose the source vertex v_s is the vertex v_1 and that the target vertex v_t is the vertex v_6. Moreover, set the number of loop-less paths K to be sought equal to 4. Table 2.2 contains a summary of the algorithmic output at each iteration until the algorithm terminates.

The algorithm is initialised by finding a shortest path between v_s and v_t by invoking Dijkstra's algorithm (see Algorithm 2.1), which returns the path v_1, v_4, v_5, v_6 of length 6. This route is assigned to A^1 and inserted into the list A. No paths are inserted into the list B as only one shortest path was returned by Dijkstra's algorithm.

The counter k is then incremented to 2, and a second shortest path is sought next. The position of the first evaluated vertex is set equal to $i = 1$, and then a comparison is performed between all the roots of the paths in A — in this case only the path A^1. In all cases where $i = 1$, the comparison would yield the value true as all roots start at the source vertex, which is always in location 1. The vertex at position $i + 1$ of the path A^1 is v_4 and therefore d_{14} is set equal to ∞. The spur S_1^2 can now be determined by finding the shortest path between the vertex at position $i = 1$ and v_t, but with the edge $\{v_1, v_4\}$ now having a weight of ∞. The spur S_1^2 is set equal to the shortest path returned, being v_1, v_2, v_5, v_6, and because it is the first position of i, the concatenation yields the same path A_1^2, of length 7. The path A_1^2 is inserted into the list B. The graph G is thereafter reset, regaining all its original weights for the next iteration.

The second vertex position, $i = 2$, is examined and the root R_2^2 is set to v_1, v_4 (the first two vertices of the $(k-1)$-th path in A, when $k = 1$). Since $i = 2$, the vertex under evaluation is v_4, and q is set equal to the vertex at position $i + 1$, which is v_5. The value of d_{45} is therefore

set equal to ∞. The spur S_2^2 is next determined from vertex v_4, at position i, to vertex v_t, and yields the path v_4, v_6. After concatenating S_2^2 with the corresponding root R_2^2, the path A_2^2 of length 8 is added to the list B, as it is not yet present in B.

The vertex in the third position of R_3^2 is evaluated next, namely v_5, when $k = 2$ and $i = 3$. The distance d_{56} is set equal to ∞, and determining the shortest path from v_5 to v_t, after having excluded the vertices v_1 and v_4 from G to form G_a, yields the spur v_5, v_2, v_3, v_6. The root and spur are concatenated to form A_3^2 with a length of 13. This path is added to the list B, as shown in Table 2.2. As the path A^{k-1} only contains four vertices, the next iteration $i = 4$, would start at the target vertex, and therefore iteration $k = 2$ is terminated. The next shortest path can now be determined from the list B. Of the three paths present in B, the path A_1^2 is the shortest (of length 7) and is therefore extracted from B and inserted into A.

With k being incremented to 3, the search for the third shortest path commences. For $i = 1$, both the roots R_1^1 and R_1^2 are the same as R_1^3, and so both d_{14} and d_{12} are set to equal ∞. Upon testing the feasibility of a shortest path from v_s, it is found that no feasible path exists. The counter i is therefore incremented with no feasible paths being added to the list B. The root R_2^3 becomes v_1, v_2, and upon evaluation of the similarity between the first two vertices of A^1, the test yields the value false, and so no weight is set to ∞. The roots are, however, the same for the path A_1^2 (at $i = 2$), and therefore the weight of the second edge, $\{v_2, v_5\}$, is set to equal ∞. After determining the spur S_2^3, the path A_2^3 is added to the list B (its length is 8). The path A_3^3 is similarly added to the list B (it has a length of 13), and at the end of iteration $k = 3$, two paths are present in B, the shortest having length 8. The path A_2^2 is randomly selected to take the position of the third shortest path. It is inserted into the list A and removed from the list B.

The last iteration ($k = 4$) commences in a similar fashion, but now there are three paths in the list of A to consider during determination of the fourth shortest path. As in iteration $i = 1$ when $k = 3$, the evaluation of the first vertex of G leads to an infeasible shortest path from v_s to v_t. When i is incremented to 2, two different edges have their weights set to equal ∞, namely d_{45} and d_{46}, and yet a feasible shortest path is still generated. This is the result of paths $j = 1$ and $j = 3$ in A having the same root R_2^4, but a different vertex q. Iteration $k = 4$ terminates at this point as the path A^{k-1} only has three vertices, and iterations over i are terminated at the second last vertex, numbered Q_{k-1}. The path A_2^4 (of length 18) is added to the list B, and finally the last shortest path is determined by extracting the other path (of length 8), namely A_2^3, from B and inserting it into A. The algorithm terminates and the set A contains the K shortest loop-less paths. This set is returned as output.

2.3 Multi-objective optimisation preliminaries

A number of basic concepts from the realm of multi-objective optimisation are reviewed in this section. A formulation of the general multi-objective optimisation problem is given in §2.3.1, while the focus shifts in §2.3.2 to the concept of dominance of solutions and how this gives rise to the notion of Pareto optimality in §2.3.3. The notion of dominance is then extended to solution sets, with various relations between these solution sets described in §2.3.4.

2.3.1 Multi-objective problem formulation

An *optimisation problem* may be defined as a problem in which solutions satisfying a set of constraints are sought which achieve extreme values of one or more objective functions [39].

TABLE 2.2: *A summary of sequential steps followed when applying Algorithm 2.3 to the weighted graph G_3 in Figure 2.4, when $v_s = v_1$, $v_t = v_6$ and $K = 4$. The variables k, i and j represent counters, while R and S represent the root and spur, respectively, associated with each counter in the sub- and super-scripts. The symbol A_i^k represents the shortest path constructed by the algorithm for each combination of values of k and i, and $c(A_i^k)$ the associated length of the path. Furthermore, A and B are lists that are maintained during the algorithm, where A represents the accepted k shortest paths, and B represents the list of potential k shortest paths, respectively.*

k	i	R_i^k	j	$R_i^k = R_i^j$	$d_{iq} \leftarrow \infty$	S_i^k	A_i^k	$c(A_i^k)$	A	B
1	—	—	—	—	—	—	v_1, v_4, v_5, v_6	6	A^1	
2	1	v_1	1	True	d_{14}	v_1, v_2, v_5, v_6	v_1, v_2, v_5, v_6	7		A_1^2
	2	v_1, v_4	1	True	d_{45}	v_4, v_6	v_1, v_4, v_6	8		A_1^2, A_2^2
	3	v_1, v_4, v_5	1	True	d_{56}	v_5, v_2, v_3, v_6	$v_1, v_4, v_5, v_2, v_3, v_6$	13		A_1^2, A_2^2, A_3^2
									A^1, A_1^2	A_2^2, A_3^2
3	1	v_1	1	True	d_{14}					
			2	True	d_{12}	infeasible				
	2	v_1, v_2	1	False						
			2	True	d_{25}	v_2, v_3, v_6	v_1, v_2, v_3, v_6	8		A_2^2, A_3^2, A_2^3
	3	v_1, v_2, v_5	1	False						
			2	True	d_{56}	v_5, v_4, v_6	v_1, v_2, v_5, v_4, v_6	13		$A_2^2, A_3^2, A_2^3, A_3^3$
									A^1, A_1^2, A_2^2	A_3^2, A_2^3, A_3^3
4	1	v_1	1	True	d_{14}					
			2	True	d_{12}					
			3	True	d_{14}	infeasible				
	2	v_1, v_4	1	True	d_{45}					
			2	False						
			3	True	d_{46}	v_4, v_7, v_3, v_6	v_1, v_4, v_7, v_3, v_6	18		$A_3^2, A_2^3, A_3^3, A_2^4$
									A^1, A_1^2, A_2^2, A_2^3	A_3^2, A_3^3, A_2^4

When only one objective function is involved, the problem is called a *single-objective optimisation problem* (SOP), while when multiple objective functions are involved, the problem is called a *multi-objective optimisation problem* (MOP). MOPs require the simultaneous optimisation of potentially conflicting objective functions in the sense that an increase in one function may cause a decrease in another function. In the case of an SOP, a single globally optimal solution is typically sought, whereas solutions that achieve desirable trade-offs between optimising the various optimisation objective functions are typically sought in MOPs [187].

Let $\mathcal{D}$ be some n-dimensional space, and suppose $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathcal{D}$ is a vector of decision variables of an MOP with M objective functions, J inequality constraints and K equality constraints. Then the MOP can be written as

$$\text{minimise} \quad z_m = f_m(\boldsymbol{x}), \qquad m = 1, \ldots, M, \tag{2.2}$$

subject to the constraints

$$g_j(\boldsymbol{x}) \geq 0, \qquad j = 1, \ldots, J, \tag{2.3}$$

$$h_k(\boldsymbol{x}) = 0, \qquad k = 1, \ldots, K, \tag{2.4}$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \qquad i = 1, \ldots, n, \tag{2.5}$$

where $x_i^{(L)}$ and $x_i^{(U)}$ denote respectively constant lower and upper bounds on the decision variable x_i [39]. If all the constraints in (2.3)–(2.5) are satisfied by the vector $\boldsymbol{x}$, the solution is called *feasible*. If, on the other hand, any one of the constraints in (2.3)–(2.5) is not satisfied by $\boldsymbol{x}$, the solution is called *infeasible*. The set of all feasible solutions to the MOP is known as the *decision space* or the *feasible region*, and is denoted here by $\mathcal{S}$. Therefore, $\mathcal{S} \subseteq \mathcal{D}$ [39].

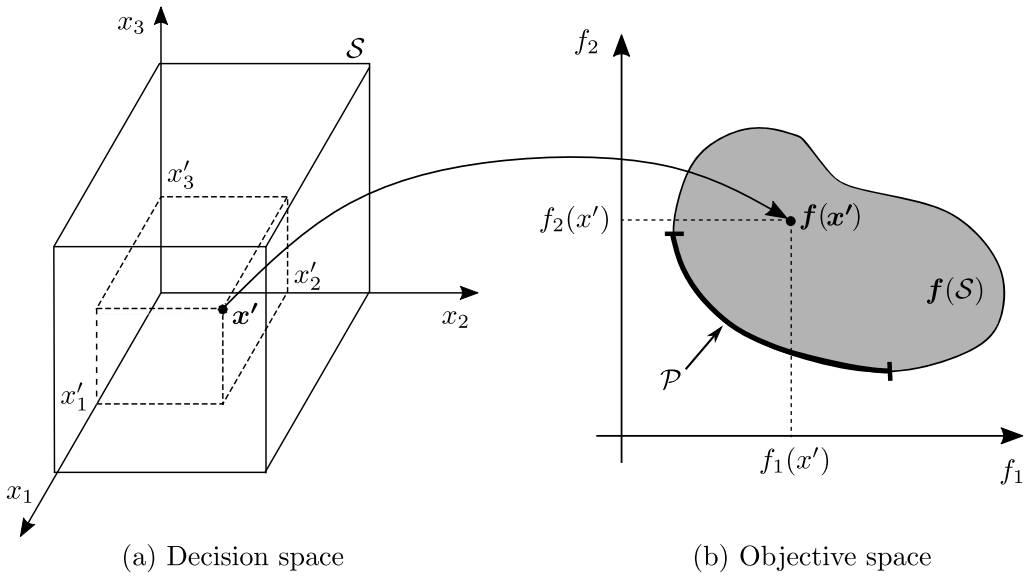
The requirement that all M objective functions in (2.2) have to be minimised is without loss of generality, for if any objective function were to be maximised, its negation could instead have been included in the formulation, which would then require minimisation [171]. The assumption of a greater-than-or-equal-to sign in all the inequality constraints of (2.3) is similarly without loss of generality.

In addition to the n-dimensional decision space $\mathcal{S}$ of the MOP (2.2)–(2.5), the problem is also equipped with an M-dimensional objective space $\mathcal{Z}$ in which its objective functions $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{z} = [f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})]$ take values. The MOP may therefore be seen as a mapping from $\mathcal{D}$ to $\mathcal{Z}$, as illustrated in Figure 2.5 for the special case where $n = 3$ and $M = 2$.

2.3.2 The notion of dominance

The notion of *dominance* is a useful tool when comparing the relative quality of candidate solutions to an MOP. A candidate solution $\boldsymbol{x}^{(1)}$ to the MOP in (2.2)–(2.5) *dominates* another candidate solution $\boldsymbol{x}^{(2)}$ if the following two conditions hold: (a) The solution $\boldsymbol{x}^{(1)}$ is no worse than the solution $\boldsymbol{x}^{(2)}$ in terms of *all* the objective functions of the MOP, and (b) the solution $\boldsymbol{x}^{(1)}$ is strictly better than the solution $\boldsymbol{x}^{(2)}$ in terms of at least one objective function of the MOP (denoted by $\boldsymbol{x}^{(1)} \prec \boldsymbol{x}^{(2)}$) [39]. The solution $\boldsymbol{x}^{(1)}$ is said to *weakly dominate* $\boldsymbol{x}^{(2)}$ (denoted by $\boldsymbol{x}^{(1)} \preceq \boldsymbol{x}^{(2)}$) if $f_i(\boldsymbol{x}^{(1)}) \leq f_i(\boldsymbol{x}^{(2)})$ for $i \in 1, \ldots, m$ [111]. The solution $\boldsymbol{x}^{(1)}$ is said to *strictly dominate* $\boldsymbol{x}^{(2)}$ (denoted by $\boldsymbol{x}^{(1)} \prec\prec \boldsymbol{x}^{(2)}$) if $f_i(\boldsymbol{x}^{(1)}) < f_i(\boldsymbol{x}^{(2)})$ for $i \in 1, \ldots, m$ [111]. Given a set $\mathcal{X}$ of candidate solutions to the MOP in (2.2)–(2.5), a member of this set is called *non-dominated in respect of* $\mathcal{X}$ if it is not dominated by any of the remaining members of the set.

The notion of solution dominance is illustrated graphically in Figure 2.6 for a set of five candidate solutions to an MOP with two objective functions f_1 and f_2. The question naturally arises as



(a) Decision space (b) Objective space

FIGURE 2.5: *Graphical representation of the decision space $\mathcal{X}$ and corresponding objective space $\mathcal{Z}$ of an MOP. Each vector $\boldsymbol{x}'$ of decision variables in $\mathcal{X}$ is mapped by the MOP to a vector $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}')$ in $\mathcal{Z}$ [39].*

to which of these solutions are superior. It is clear that solution 4 dominates solution 2, while solution 5 dominates solution 3. None of solutions 1, 4 or 5, however, dominates the other solutions. These solutions are therefore designated as non-dominated in respect of the five solutions depicted, and embody a high-quality trade-off between the objective function values, but are superior to solutions 2 and 3.

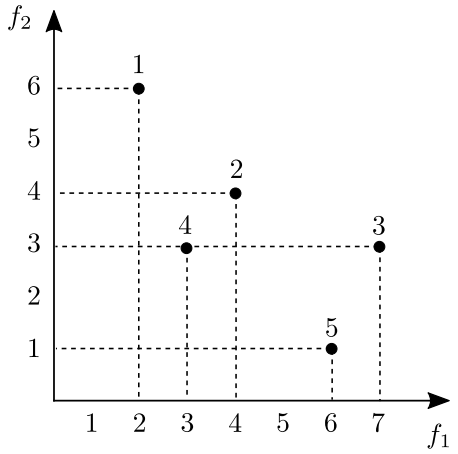


FIGURE 2.6: *The objective space $\mathcal{Z}$ of a bi-objective optimisation problem with five solutions indicated, illustrating the notion of dominance, where f_1 and f_2 denote the two objective functions [39].*

2.3.3 The notion of Pareto optimality

The *Pareto optimal* solutions to an MOP are those that are non-dominated in respect of the entire decision space $\mathcal{S}$ [39]. The Pareto optimal solutions to an MOP therefore embody the highest quality trade-offs achievable between the objective function values, and are superior to all other solutions of the MOP. No single Pareto optimal solution is, however, superior to any other Pareto optimal solution. The Pareto optimal solutions to the MOP represented in

Figure 2.5 are indicated in part (b) of the figure by the bold-faced boundary of the set $\boldsymbol{f}(\mathcal{S})$, denoted by $\mathcal{P}$.

2.3.4 Relations between solution sets

The relations of dominance described in §2.3.2 can be extended to relations between solution sets [111]. Let $\mathcal{A}$ and $\mathcal{B}$ be two non-dominated solution sets, *i.e.* all the solutions within each solution set is non-dominated in respect of all other solutions in the same set [111]. Then, the following definitions are due to Li and Yao [111]:

Strict dominance. Set $\mathcal{A}$ *strictly dominates* $\mathcal{B}$ (denoted by $\mathcal{A} \prec\prec \mathcal{B}$) if every solution $\boldsymbol{b} \in \mathcal{B}$ is strictly dominated by at least one solution $\boldsymbol{a} \in \mathcal{A}$ [193].

Dominance. Set $\mathcal{A}$ *dominates* $\mathcal{B}$ (denoted by $\mathcal{A} \prec \mathcal{B}$) if every solution $\boldsymbol{b} \in \mathcal{B}$ is dominated by at least one solution $\boldsymbol{a} \in \mathcal{A}$ [193]. This relation is also called *complete outperformance* and denoted by $\mathcal{A}\, \mathcal{O}_{\mathcal{C}}\, \mathcal{B}$ in [70].

Weak dominance. Set $\mathcal{A}$ *weakly dominates* $\mathcal{B}$ (denoted by $\mathcal{A} \preceq \mathcal{B}$) if every solution $\boldsymbol{b} \in \mathcal{B}$ is weakly dominated by at least one solution $\boldsymbol{a} \in \mathcal{A}$ [193].

Other than these three solution set relations mentioned, there are also two relations that exclusively compare the two solution sets, namely *strong outperformance* and *weak outperformance* [111].

Strong outperformance. Set $\mathcal{A}$ *strongly outperforms* $\mathcal{B}$ (denoted by $\mathcal{A}\, \mathcal{O}_{\mathcal{S}}\, \mathcal{B}$) if $\mathcal{A} \preceq \mathcal{B}$ and there exists at least one pair of solutions $\boldsymbol{a} \in \mathcal{A}$ and $\boldsymbol{b} \in \mathcal{B}$ such that $\boldsymbol{a} \prec \boldsymbol{b}$ [70].

Weak outperformance. Set $\mathcal{A}$ *weakly outperforms* $\mathcal{B}$ (denoted by $\mathcal{A}\, \mathcal{O}_{\mathcal{W}}\, \mathcal{B}$) if $\mathcal{A} \preceq \mathcal{B}$ and there also exists at least one solution $\boldsymbol{a} \in \mathcal{A}$ that is not weakly dominated by any solution in $\mathcal{B}$ [70]. Zitzler *et al.* [193] called this relation *better*, denoted by $\lhd$.

Weak outperformance ($\mathcal{A}\mathcal{O}_{\mathcal{W}}\mathcal{B}$) is the most general and weakest form of superiority between two solutions sets, *i.e.* $\mathcal{A}$ is at least as good as $\mathcal{B}$, but $\mathcal{B}$ is not as good as $\mathcal{A}$. These five relations can be ordered in terms of superiority as follows: $\mathcal{A} \prec\prec \mathcal{B} \Rightarrow \mathcal{A} \prec \mathcal{B} \Rightarrow \mathcal{A}\mathcal{O}_{\mathcal{S}}\mathcal{B} \Rightarrow \mathcal{A}\mathcal{O}_{\mathcal{W}}\mathcal{B} \Rightarrow \mathcal{A} \preceq \mathcal{B}$.

2.4 Methods for determining non-dominated sets

Establishing computationally efficient methods for determining high-quality non-dominated solution sets is critical within the realm of multi-objective optimisation. Various methods exist in the literature for this purpose, ranging in differing degrees of complexity. Three well-known methods for determining non-dominated subsets from a given set of candidate solutions to (2.2)–(2.5) are described in this section, namely a *naive and slow* method, a *continuously updating* method, and finally the efficient method of *Kung et al.* [105], in order of increasing sophistication and computational efficiency. More formally stated, the task that these methods aim to perform is, given a finite set of candidate solutions $\mathcal{C}$ to an instance of the MOP (2.2)–(2.5), to identify the non-dominated subset $\mathcal{C}'$ of solutions among those in $\mathcal{C}$.

2.4.1 Naive and slow method

The first method of interest is the naive and slow method, which exhibits the nature of its description. This method entails comparing every solution i in the given candidate solution set $\mathcal{C}$ to an instance of (2.2)–(2.5) with every other solution j in the same solution set, based on the weak dominance relation of §2.3.2. If no solution j is found to dominate solution i, then solution i is added to the subset $\mathcal{C}'$ of non-dominated solutions. Alternatively, if a solution j is found that dominates solution i, then solution i is not included in the subset $\mathcal{C}'$ of non-dominated solutions. This method is presented in a pseudo-code form in Algorithm 2.4 and is illustrated by means of an example.

Example 2.1 (Naive and slow method) *Let $\mathcal{C} = \{1, 2, 3, 4, 5\}$ be the set of candidate solutions shown in Figure 2.6 to an instance of (2.2)–(2.5) in which the goal is to minimise both objectives. Initialise $i = 1$ and $\mathcal{C}' = \emptyset$. When comparing solution 1 with the other solutions in $\mathcal{C}$, starting with solution 2, it is found that solution 2 does not dominate solution 1. Next, when solution 1 is compared with solution 3, it is found that the latter dominates the former. Similarly, solutions 4 and 5 are also compared with solution 1, and neither is found to dominate solution 1. Solution 1 is therefore a non-dominated solution, and is inserted into the non-dominated subset to obtain $\mathcal{C}' = \{1\}$. The iteration counter i is next incremented to 2.*

Comparing solution 2 with all other solutions in $\mathcal{C}$, starting with solution 1, it is found that neither solution 1 nor solution 3 dominates solution 2, but when considering solution 4, it is found that it indeed dominates solution 2. Solution 2 therefore cannot belong to $\mathcal{C}'$, the non-dominated subset. Hence, i is incremented to 3.

Comparing solution 3 with all other solutions in the set $\mathcal{C}$, it is found that solutions 1 and 2 do not dominate solution 3, but it is dominated by solution 4. Solution 3, consequently, cannot form part of the non-dominated subset $\mathcal{C}'$. The counter i is therefore incremented to 4.

Finally, incrementing the counter i to 4 and thereafter to 5, solutions 4 and 5 are both found not to be dominated when compared with all other solutions in the candidate solution subset $\mathcal{C}$. Solution 4 first, and thereafter 5, are therefore inserted into the subset of non-dominated solutions to obtain $\mathcal{C}' = \{1, 4, 5\}$.

Upon incrementing the counter i to 6, it is found to be larger than $|\mathcal{C}| = 5$, and the algorithm terminates. All solutions having been considered, the output $\mathcal{C}' = \{1, 4, 5\}$ is returned. ■

When examining the computational complexity of the aforementioned method, it is seen that $\mathcal{O}(|\mathcal{C}|)$ comparisons are required for domination testing, while M objective function comparisons are required for each domination comparison. Therefore, domination testing has a complexity of $\mathcal{O}(M|\mathcal{C}|)$, yielding an overall algorithmic complexity of $\mathcal{O}(M|\mathcal{C}|^2)$.

2.4.2 Continuously updating method

Another method for identifying the non-dominated solution subset, given a finite set of candidate solutions to an instance of (2.2)–(2.5), is the continuously updating method, which is based on the naive and slow method, but achieves a better computational complexity than its predecessor. The overarching idea of the continuous approach is that each solution in the given set of candidate solutions is compared, based on the weak dominance relation from §2.3.2, with a subset of decreasing size as the algorithm progresses. The method is initialised by randomly selecting a solution in the given set $\mathcal{C}$ of candidate solutions to the instance of (2.2)–(2.5), removing it from

Algorithm 2.4: Naive and slow method for determining non-dominated subsets [39]

Input : A set $\mathcal{C}$ of candidate solutions to an instance of (2.2)–(2.5).
Output: The non-dominated subset $\mathcal{C}' \subseteq \mathcal{C}$ of solutions.

```
1  i ← 1
2  C' ← ∅
3  while i ≤ |C| do
4      j ← 1
5      while j ≤ |C| do
6          if j ≠ i then
7              if x_j ⪯ x_i then
8                  Break
9              else
10                 j ← j + 1
11     if j = |C| then
12         C' ← C' ∪ {i}
13     i ← i + 1
14 return [C']
```

$\mathcal{C}$ and inserting it into another set $\mathcal{C}'$, which is initialised to be empty. Thereafter, each solution i in the set $\mathcal{C}$ is individually compared with each solution in $\mathcal{C}'$. If any solution in $\mathcal{C}'$ is dominated by a solution $i \in \mathcal{C}$, it is removed from $\mathcal{C}'$. If, however, any solution i in $\mathcal{C}$ is dominated by any solution in $\mathcal{C}'$, it is ignored. If a solution $i \in \mathcal{C}$ is not dominated by any solution in $\mathcal{C}'$, it is removed from $\mathcal{C}$ and inserted into $\mathcal{C}'$. Once all of the solutions in $\mathcal{C}$ have been compared with the solutions in $\mathcal{C}'$, the algorithm terminates, and $\mathcal{C}'$ contains the non-dominated solutions of the original set $\mathcal{C}$.

The continuously updating method is presented in pseudo-code form in Algorithm 2.5. An example is provided to further elucidate the working of this method. It should be noted that removal of the first selected solution from the set $\mathcal{C}$ may be achieved simply by initialising the incrementation counter i as 2 instead of 1, therefore skipping over the first solution in $\mathcal{C}$. Also, in Algorithm 2.5, x_i denotes solution i selected from the original set $\mathcal{C}$, while x_j denotes solution j being considered in the non-dominated subset $\mathcal{C}'$.

Example 2.2 (Continuously updating method) *Again, let $\mathcal{C} = \{1, 2, 3, 4, 5\}$ be the set of candidate solutions depicted in Figure 2.6 to an instance of (2.2)–(2.5) in which the goal is to minimise both objectives. Suppose that solution 1 has randomly been chosen to be inserted into the non-dominated subset, yielding $\mathcal{C}' = \{1\}$, when $i = 2$. Solution 2 is then compared with solution 1, the only solution in $\mathcal{C}'$, and it found that solution 1 does not dominate solution 2. Solution 2 is therefore included in the set $\mathcal{C}'$ to obtain $\mathcal{C}' = \{1, 2\}$. The counter i is then incremented to 3.*

Solution 3 is now compared with each solution in the subset $\mathcal{C}'$, namely solutions 1 and 2, and it is found that neither of them dominates solution 3. Therefore, solution 3 is also inserted into the non-dominated subset to obtain $\mathcal{C}' = \{1, 2, 3\}$. The next solution is then chosen for analysis, upon incrementing i to 4.

Solution 4 is compared with the three solutions in $\mathcal{C}'$, and it is found that solution 1 is not dominated by solution 4. When compared with solution 2, however, it is found that solution 4

Algorithm 2.5: Continuously updating method for determining non-dominated sets [39]

Input : A set $\mathcal{C}$ of candidate solutions to an instance of (2.2)–(2.5).
Output: The non-dominated subset $\mathcal{C}' \subseteq \mathcal{C}$ of solutions.

```
1  i ← 2
2  C' ← {1}
3  while i ≤ |C| do
4      j ← 1
5      while j ≤ |C'| do
6          if x_i ⪯ x_j then
7              C' ← C'\{j}
8          else
9              if x_j ⪯ x_i then
10                 Break
11         j ← j + 1
12     if j = |C| then
13         C' ← C' ∪ {i}
14     i ← i + 1
15 return [C']
```

dominates solution 2, and therefore solution 2 is removed from the non-dominated set $\mathcal{C}'$, now yielding $\mathcal{C}' = \{1, 3\}$. A similar result is obtained when comparing solution 4 with solution 3, and so solution 3 is also removed, leaving the singleton set $\mathcal{C}' = \{1\}$. Solution 4 is found not to be dominated by any solutions in $\mathcal{C}'$, and is therefore included in the non-dominated subset to obtain $\mathcal{C}' = \{1, 4\}$. Next, i is incremented to 5, initiating the analysis of the last solution.

Solution 5 is compared with each of the solutions in the subset $\mathcal{C}'$, and it is found that it is not dominated by either solution 1 or solution 4. It is therefore included in the subset $\mathcal{C}'$ to obtain the set $\mathcal{C}' = \{1, 4, 5\}$. Thereafter, the algorithm terminates, yielding $\mathcal{C}' = \{1, 4, 5\}$ as the non-dominated subset of the given set $\mathcal{C}$. ■

Considering the computational complexity of the continuously updating method, it is clear that after initialisation of the algorithm, the first evaluated solution is only compared with one other solution. The second solution under consideration is compared with at most two solutions in the non-dominated subset $\mathcal{C}'$, and so on. The total number of domination comparisons required is therefore at most $1 + 2 + \ldots + |\mathcal{C}| - 1$, or more compactly stated, $|\mathcal{C}|(|\mathcal{C}| - 1)/2$. A complexity of $\mathcal{O}(M|\mathcal{C}|^2)$ is therefore ascribed to this method. Although the naive and slow method has the same complexity, the total number of comparisons required for the continuously updating method is typically much smaller than that of the former method in practice. It has been estimated that approximately half the number of computations are typically required for the continuously updating method [39].

2.4.3 The efficient method of Kung *et al.* [105]

True to its name, the efficient method of Kung *et al.* [105] is computationally cheaper than both the naive and slow method and the continuously updating method. The approach taken is to sort the entire set of candidate solutions in non-improving order based on the values of the first

objective function. The sorted set is thereafter recursively halved, obtaining a top and bottom subset of solutions, denoted by $\mathcal{T}$ and $\mathcal{B}$, respectively, each time. Thus, with respect to the first objective function, the better solutions are contained within the subset $\mathcal{T}$, and the worse solutions reside in the subset $\mathcal{B}$.

Once again, incorporating the dominance relation described in §2.3.2 as comparison criterion, every solution i in the subset $\mathcal{B}$ is then compared with every solution j in the subset $\mathcal{T}$. Each solution that is found to be non-dominated in $\mathcal{B}$ when compared with solutions in $\mathcal{T}$, is then added to a merged set $\mathcal{S}$, consisting of the union of $\mathcal{T}$ and the non-dominated solutions of $\mathcal{B}$. This process is repeated in a recursive fashion, until the point where all the subsets $\mathcal{B}$ have been compared with their subset counterparts $\mathcal{T}$, with the final merged set $\mathcal{S}$ then containing the subset of non-dominated solutions. The method adopts a bottom-up approach where the domination tests and merging starts with the inner-most case, when only one member resides in either $\mathcal{T}$ or $\mathcal{B}$. A pseudo-code description of the efficient method of Kung *et al.* [105], denoted by the function `Front`, is given in Algorithm 2.6. The method is elucidated by means of the following example.

Algorithm 2.6: `Front` $(\mathcal{C})$ [105]

Input : A set $\mathcal{C}$ of candidate solutions to an instance of (2.2)–(2.5).
Output: The non-dominated subset $\mathcal{C}' \subseteq \mathcal{C}$ of solutions.

```
1  Sort(C)
2  if |C| = 1 then
3      return [C]
4  else
5      T ← Front([C_1, ..., C_⌊|C|/2⌋])
6      B ← Front([C_⌊|C|/2⌋+1, ..., C_|C|])
7      i ← 1
8      S ← T
9      while i ≤ |B| do
10         j ← 1
11         while j ≤ |T| do
12             if x_j ⊀ x_i then
13                 j ← j + 1
14             else
15                 Break
16         if j = |T| + 1 then
17             S ← S ∪ {i}
18         i ← i + 1
19     return [S]
```

Example 2.3 (Efficient method of Kung *et al.*) *Once more, let* $\mathcal{C} = \{1, 2, 3, 4, 5\}$ *be the set of candidate solutions to a bi-objective minimisation instance of* (2.2)–(2.5)*, depicted in Figure 2.6. The first objective function* f_1 *should be minimised, and therefore the solutions are sorted in increasing order based on their values in respect of this objective function. The ordered set* $\mathcal{C} = \{1, 4, 2, 5, 3\}$*, is returned after the sorting. Due to* $\mathcal{C}$ *not being a singleton set, which terminates the algorithm, it is partitioned into the top half* $\mathcal{T} =$ `Front`$(\{1, 4\})$ *and the bottom half* $\mathcal{B} =$ `Front`$(\{2, 5, 3\})$*, as depicted in the top branch of Figure 2.7.*

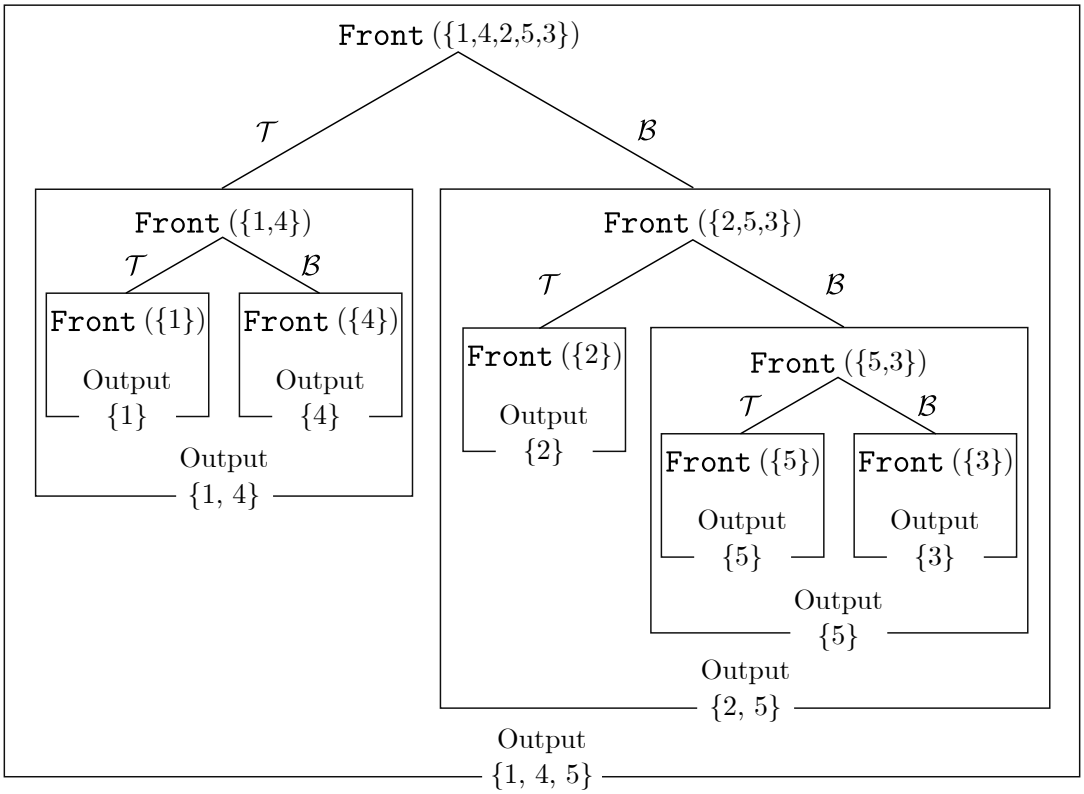


FIGURE 2.7: *Illustration of the efficient method of Kung* et al. *[105] for finding non-dominated subsets, applied to the candidate solution set* $\mathcal{C} = \{1, 2, 3, 4, 5\}$ *depicted in Figure 2.6.*

Since having a cardinality of two, the set $\{1, 4\}$ *in* $\texttt{Front}(\{1, 4\})$ *is partitioned further, yielding* $\mathcal{T} = \texttt{Front}(\{1\})$ *and* $\mathcal{B} = \texttt{Front}(\{4\})$*. Both of these inner sets are singleton sets and therefore return as output* $\{1\}$ *and* $\{4\}$*, respectively, serving as the input to* $\texttt{Front}(\{1, 4\})$*. The inner set* $\mathcal{B} = \{4\}$ *is compared with its counterpart* $\mathcal{T} = \{1\}$ *and tested for dominance. Solution 4 is not dominated by solution 1 and therefore the merged set* $\mathcal{S} = \{1, 4\}$ *is formed, being the final outcome of* $\texttt{Front}(\{1, 4\})$ *which, in turn, serves as the input for* $\texttt{Front}(\{1, 4, 2, 5, 3\})$*, as illustrated in the left leg of the recursion tree in Figure 2.7.*

Next, $\texttt{Front}(\{2, 5, 3\})$ *is evaluated. Similarly, its inner set* $\{2, 5, 3\}$ *is partitioned into* $\mathcal{T} = \texttt{Front}(\{2\})$ *and* $\mathcal{B} = \texttt{Front}(\{5, 3\})$*. The subset* $\mathcal{T} = \texttt{Front}(\{2\})$ *yields the output* $\{2\}$ *and is returned to* $\texttt{Front}(\{2, 5, 3\})$*. On the other hand, the inner set of* $\texttt{Front}(\{5, 3\})$ *is partitioned once more into* $\mathcal{T} = \texttt{Front}(\{5\})$ *and* $\mathcal{B} = \texttt{Front}(\{3\})$*, yielding the respective outputs* $\{5\}$ *and* $\{3\}$*, which are returned to* $\texttt{Front}(\{5, 3\})$*. With* $\mathcal{T} = \{5\}$ *and* $\mathcal{B} = \{3\}$*,* $\texttt{Front}(\{5, 3\})$ *can be evaluated, and when solution 3 is compared with solution 5, it is found that solution 5 dominates solution 3. Therefore, solution 3 is not included in the merged set* $\mathcal{S}$*; instead only* $\mathcal{S} = \{5\}$ *is returned to* $\texttt{Front}(\{2, 5, 3\})$*.*

The evaluation of $\texttt{Front}(\{2, 5, 3\})$ *can now be completed, by comparing* $\mathcal{B} = \{5\}$ *with* $\mathcal{T} = \{2\}$ *in terms of dominance. It is found that solution 5 in* $\mathcal{B}$ *is not dominated by the only solution in* $\mathcal{T}$*, solution 2, and therefore the merged set* $\mathcal{S} = \{2, 5\}$ *is returned to* $\texttt{Front}(\{1, 4, 2, 5, 3\})$*, as illustrated in the right-hand leg of the recursion tree in Figure 2.7.*

Finally, $\texttt{Front}(\{1, 4, 2, 5, 3\})$ *can be evaluated, pitting the subset* $\mathcal{B} = \{2, 5\}$ *against the subset* $\mathcal{T} = \{1, 4\}$*. When the domination comparison is performed, it is found that solution 2 in* $\mathcal{B}$ *is dominated by solution 4 in* $\mathcal{T}$*, and therefore is not included in the merged set* $\mathcal{S}$*. Solution 5, on the other hand, is not dominated by any solutions in* $\mathcal{T}$*, and is therefore merged with* $\mathcal{T} = \{1, 4\}$

to form $\mathcal{S} = \{1, 4, 5\}$. *Since* $\mathcal{S} = \{1, 4, 5\}$ *is the output of* `Front`$(\{1, 4, 2, 5, 3\})$, *whose inner set is the sorted set of candidate solutions, the algorithm terminates, with* $\mathcal{S} = \{1, 4, 5\}$ *being returned as the subset of non-dominated solutions.* ■

Kung *et al.* [105] proved that for $M = 2$ or 3 objectives, their method achieves a computational complexity of $\mathcal{O}(|\mathcal{C}| \log |\mathcal{C}|)$, while for $M \geq 4$ objectives, it achieves a computational complexity of $\mathcal{O}(|\mathcal{C}|(\log |\mathcal{C}|)^{M-2})$.

2.5 Non-dominated sorting of solution sets

Two sets are usually formed by search techniques designed to find high-quality solutions to MOPs, namely a non-dominated set and a remaining dominated set of solutions. Naturally, the main interest lies with the non-dominated set, but there are, however, instances where this set alone is not the only set of interest. Some algorithms require that the solution set be partitioned into various classes, based on their degree of dominance relative to each other. One such technique is the well-known *fast non-dominated sorting algorithm* (FNSA) developed by Srinivas and Deb [152], a common non-dominated sorting algorithm incorporated into solution procedures for MOPs which boasts an effective bookkeeping strategy aimed at reducing computational complexity. The FNSA is briefly discussed in this section.

The general idea of non-dominated sorting of solution sets is to partition a given set $\mathcal{C}$ of candidate solutions to an instance of (2.2)–(2.5) into different levels of non-dominated fronts. These fronts are denoted by $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots$, where $\mathcal{F}_1$ denotes the subset of non-dominated solutions in $\mathcal{C}$. The algorithm starts by identifying this subset of non-dominated solutions by employing any of the methods of the previous section and the solutions thus identified are assigned to the set $\mathcal{F}_1$. The solutions in $\mathcal{F}_1$ are then temporarily removed from $\mathcal{C}$, and the non-dominated solutions of the remaining solutions in $\mathcal{C}$ are determined. The newly identified non-dominated solutions in the smaller set $\mathcal{C}$ are then assigned to a second front, denoted by $\mathcal{F}_2$.

Next, the solutions in both $\mathcal{F}_1$ and $\mathcal{F}_2$ are removed from the candidate set $\mathcal{C}$ of solutions, and in a similar fashion a third front, denoted by $\mathcal{F}_3$, is determined. This process is repeated until all the candidate solutions in the original set $\mathcal{C}$ have been classified into one of the fronts generated. The working of this generalised approach to non-dominated sorting is illustrated in the following example.

Example 2.4 (Non-dominated sorting of solution sets) *Let* $\mathcal{C} = \{1, 2, 3, 4, 5, 6\}$ *be the set of candidate solutions to a bi-objective minimisation instance of* (2.2)–(2.5) *depicted in Figure 2.8(a). Note that this is the same set of solutions as in Figure 2.6, but with the addition of solution 6, a solution dominated by solutions 2 and 4, in order to better illustrate the working of the non-dominated sorting approach. It is clear, from Examples 2.1, 2.2 and 2.3, together with the knowledge that solution 6 is dominated, that the non-dominated subset of solutions is* $\mathcal{C}' = \{1, 4, 5\}$, *and these solutions therefore belong to* $\mathcal{F}_1$.

Solutions 1, 4 and 5 (the members of the first front) are subsequently removed from $\mathcal{C}$, *and so the reduced set* $\{2, 3, 6\}$ *is considered during the generation of the second front. These solutions are compared with one another for dominance, based on one of the methods for finding non-dominated subsets described in §2.4. When thus compared, it is clear that solutions 2 and 3 do not dominate each other, and that solution 6 is dominated by solution 2. Therefore, the non-dominated subset of* $\{2, 3, 6\}$ *is* $\{2, 3\}$, *and so the solutions 2 and 3 are assigned to the second non-dominated front,* $\mathcal{F}_2$.

The set $\{2, 3, 6\}$ *is further reduced, by removing solutions 2 and 3, to obtain the singleton set* $\{6\}$ *for consideration when generating the third front,* $\mathcal{F}_3$*. Each candidate solution in the original set* $\mathcal{C}$ *has now been partitioned into one of the fronts* $\mathcal{F}_1, \mathcal{F}_2$ *or* $\mathcal{F}_3$*, as illustrated in Figure 2.8(b).* ■

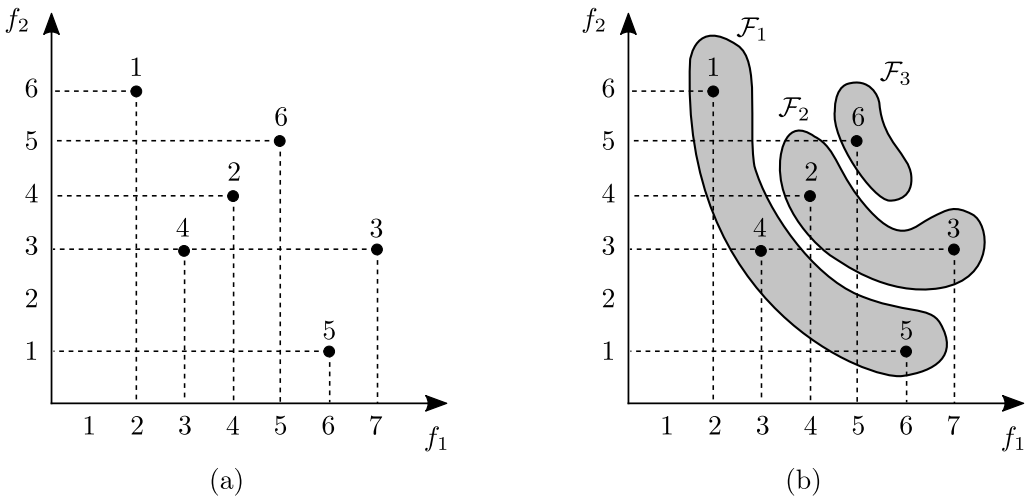


FIGURE 2.8: *An illustration of the FNSA, with (a) the input set of candidate solutions (where both the objectives* f_1 *and* f_2 *are to be minimised), and (b) the three output fronts* $\mathcal{F}_1, \mathcal{F}_2$ *and* $\mathcal{F}_3$*, based on their dominance classification.*

Certain modifications were introduced to the above general approach towards non-dominated sorting in addition to merely applying one of the methods of §2.4, and these modifications give rise to the bookkeeping strategy mentioned previously, a characteristic of the FNSA. Two parameters are determined for each solution i in $\mathcal{C}$, namely the total number of solutions that dominate solution i, denoted by the dominance count d_i, and a subset $\mathcal{S}_i$ of solutions that are dominated by i itself. The non-dominated front into which a solution is classified determines the dominance count for a solution. Calculating these parameters requires $\mathcal{O}(M|\mathcal{C}|^2)$ computations, and is described in pseudo-code form in steps 1 to 12 in Algorithm 2.7. After these calculations have been performed, the solutions with a dominance count of 0 naturally form the non-dominated subset of solutions, and are assigned to front 1, denoted $\mathcal{F}_1$.

Thereafter, for each of the solutions i, with a dominance count of $d_i = 0$, the dominance count d_j for solution j in $\mathcal{S}_i$ is reduced by 1, whereupon any solution j that has a dominance count of $d_j = 0$ is placed in a separate set $\mathcal{A}$. When all of the solutions i have been considered, the set $\mathcal{A}$ contains the second non-dominated front $\mathcal{F}_2$, with all of its members having a dominance count of 0. This procedure is repeated for all of the members of $\mathcal{A}$ in order to determine the third non-dominated front $\mathcal{F}_3$, and so on, until no solutions remain in $\mathcal{A}$ and all of the solutions in the original candidate solution set $\mathcal{C}$ have been categorised into their respective fronts [39]. A pseudo-code description of the FNSA is provided in Algorithm 2.7, with steps 13 to 23 representing the procedure just described. The working of the FNSA is illustrated in the following example.

Example 2.5 (Fast non-dominated sorting) *Consider again the set* $\mathcal{C} = \{1, 2, 3, 4, 5, 6\}$ *of candidate solutions to a bi-objective instance of* (2.2)–(2.5) *depicted in Figure 2.8(a). The first front is initialised as* $\mathcal{F}_1 = \emptyset$*.* $\mathcal{S}_i$ *is initialised as the empty set and the dominance count* d_i *is initialised as zero for each solution* i *in* $\mathcal{C}$*. Let* $i = 1$*. Starting with solution* $j = 2$*, it is tested whether solution* i *dominates solution* j*, and if this is not the case, whether solution* j *dominates solution* i*. It is found that solution 1 neither dominates any solutions, nor is dominated by any*

Algorithm 2.7: The fast non-dominated sorting algorithm [152]

Input : A set $\mathcal{C}$ of candidate solutions to an instance of (2.2)–(2.5).
Output: The set of non-dominated fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ into which $\mathcal{C}$ is partitioned.

```
1  F_1 ← ∅
2  for i ∈ C do
3      S_i ← ∅
4      d_i = 0
5      for j ∈ C and j ≠ i do
6          if i ≺ j then
7              S_i ← S_i ∪ {j}
8          else if j ≺ i then
9              d_i = d_i + 1
10     if d_i = 0 then
11         i_rank ← 1
12         F_1 ← F_1 ∪ {i}
13 k ← 1
14 while F_k ≠ ∅ do
15     A ← ∅
16     for i ∈ F_k do
17         for j ∈ S_i do
18             d_j ← d_j − 1
19             if d_j = 0 then
20                 j_rank ← k + 1
21                 A ← A ∪ {j}
22     k ← k + 1
23     F_k ← A
24 return [F_1, F_2, ..., F_k]
```

solutions, and so it is added to the level 1 non-dominated front $\mathcal{F}_1$. Next, solution $i = 2$ is considered. When compared for dominance with the remaining solutions in $\mathcal{C}$, it is found that solution 2 is dominated by one solution, solution 4, and therefore its dominance count d_2 is incremented from zero to one. Furthermore, solution 6 is found to be dominated by solution 2, and it is therefore added to the set $\mathcal{S}_2$ to obtain $\{6\}$. This procedure is continued for all the remaining solutions in $\mathcal{C}$, namely solutions 3 to 6, and the results are tabulated in Table 2.3 for brevity, corresponding to the value $k = 1$.

Examining all the results obtained for $k = 1$, it is found that solutions 1, 4 and 5 have a dominance count of $d_i = 0$, and are therefore included in the first non-dominated front $\mathcal{F}_1$, to obtain $\{1, 4, 5\}$, also known as the non-dominated subset. Although it would seem that excessive domination computations are carried out in this first portion of the algorithm only to obtain the non-dominated set, as compared with some of the methods described in §2.4, the additional bookkeeping information obtained and stored in the parameters d_i and $\mathcal{S}_i$ may be utilised in the following steps where the significant computational gains are made.

Next, for each solution i in $\mathcal{F}_1$, the dominance count d_j is decremented by one for each solution j contained in $\mathcal{S}_i$, as described in steps 13 to 23 of Algorithm 2.7. Since $\mathcal{S}_1 = \emptyset$, solution 4

TABLE 2.3: *The sequential steps followed when applying Algorithm 2.7 to the set of candidate solutions to a bi-objective instance of (2.2)–(2.5) depicted in Figure 2.8. For each front k analysed, the dominance count d_i of solution i is listed, as well as the set $\mathcal{S}_i$ containing those solutions that are dominated by solution i. Finally, the non-dominated front $\mathcal{F}_k$ is shown for $k \in \{1, 2, 3\}$.*

k	i	d_i	$\mathcal{S}_i$	$\mathcal{F}_k$
1	1	0	$\emptyset$	$\{1, 4, 5\}$
	2	1	$\{6\}$	
	3	2	$\emptyset$	
	4	0	$\{2, 3, 6\}$	
	5	0	$\{3\}$	
	6	2	$\emptyset$	
2	2	0	$\{6\}$	$\{2, 3\}$
	3	0	$\emptyset$	
	6	1	$\emptyset$	
3	6	0	$\emptyset$	$\{6\}$

is considered as the next solution in $\mathcal{F}_1$. Solutions 2, 3 and 6 are contained within $\mathcal{S}_4$, and all their dominance counts are reduced by one, yielding $d_2 = 0, d_3 = 1$ and $d_6 = 1$. With solution 2 having reached a dominance count of zero, it is included in the set $\mathcal{A}$. Considering the last solution in $\mathcal{F}_1$, solution 5, its singleton set $\mathcal{S}_5 = \{3\}$ leads to the reduction of the dominance count of solution 3, yielding $d_3 = 0$, and so solution 3 is therefore also inserted into the set $\mathcal{A}$. Next, k is incremented to 2 and the members of $\mathcal{A}$ are included in $\mathcal{F}_2$. These results are summarised in the rows of Table 2.3 corresponding to $k = 2$.

Finally, when considering the solutions contained in $\mathcal{F}_2$, solution 3 is found in $\mathcal{S}_5$. The dominance count of solution 3 is therefore reduced by one, which yields $d_3 = 0$, after which solution 3 is included in the set $\mathcal{A}$ (which has since been emptied). Next, k is incremented to 3 and the only member of $\mathcal{A}$, solution 3, is included in the third non-dominated front $\mathcal{F}_3$. Having no remaining members of $\mathcal{C}$ that are dominated by members of the set $\mathcal{F}_3$, $\mathcal{F}_4$ is therefore an empty set, leading to the termination of the algorithm. Each candidate solution in the original set $\mathcal{C}$ has now been partitioned into one of the fronts, $\mathcal{F}_1, \mathcal{F}_2$ or $\mathcal{F}_3$, as illustrated in Figure 2.8(b). ■

For the entire classification process, $\mathcal{O}(M|\mathcal{C}|^2)$ comparisons are required, where M denotes the number of objective functions and $|\mathcal{C}|$ denotes the number of candidate solutions considered.

2.6 Multi-objective optimisation solution quality evaluation

An increase in interest in MOPs has led to a variety of search techniques emerging for solving these problems [111, 137]. These search techniques range from exact to approximate methods, and from heuristics to metaheuristics (covered in detail in a later chapter). These search techniques aim to find a set of high-quality solutions to MOP instances which approximate the Pareto sets of the MOP instances. A high-quality Pareto set representation is of considerable importance when dealing with MOPs. It is indicative of the Pareto set and avoids decision maker overload during the later decision-making process, because the decision maker should choose his or her preferred solution from the Pareto set approximation [111].

Different search techniques produce different solution sets, and therefore it is important to be able to compare the different solution sets with one another as a measure of the relative performances

of the search techniques. In the context of single-objective optimisation, two solutions are easily comparable by simply defining the quality of a solution in relation of its objective function value [111]. In multi-objective optimisation problems, on the other hand, many (even infinitely many) optimal solutions may exist, and two solution sets may be incomparable. Different aspects of quality are considered when dealing with MOPs, such as the closeness to the Pareto set, coverage of the Pareto set, and the uniformity of the solutions along the approximated Pareto set [137]. These different aspects complicate the quality analysis considerably.

Visualisation is an intuitive way of comparing solution sets, but differences between solution sets cannot thus be quantified, and with more than three objectives, even visualisation becomes difficult [111]. Quantitative measures have therefore been proposed to overcome these shortcomings, and these measures are called *quality indicators* (QIs). They take the solution set as input and map it to a real number that gives an indication of one or more aspects of the solution set quality [111, 137]. QIs may be used for various purposes, such as comparing competing search techniques, monitoring the search process of search techniques and optimising their parameters, potentially improving their performance, defining the stopping criteria of search techniques (especially those of a stochastic nature) and finally, guiding the search by integrating the QIs as part of the selection criteria instead of utilising the Pareto dominance criterion directly [111].

Various fields of study have contributed research on the quality evaluation of multi-objective solution sets, such as evolutionary computation, operations research and optimisation [16, 142, 193], artificial intelligence [19, 158] and software engineering [110, 169]. QI studies can be classified into five broad categories, based on the existing literature [111]:

Design of QIs is the objective in the majority of QI studies, where the aim is to design a reliable and efficient QI for the evaluation and comparison of solution sets [111].

QI-based search was initiated in 2004 by Zitzler and Künzli [190], and the use of QIs in guiding multi-objective searches has shown growing interest [8, 160].

Theoretical studies of QIs focus on specific, but important, issues in the design of QIs, and include analyses of the computational complexity of applying QIs [17, 170], and discussions on desirable properties that QIs (or combinations thereof) should exhibit, such as Pareto compatibility and completeness [102, 193], scaling invariance [189], and minimum set construction [56].

Understanding of existing QIs includes investigations of the behaviour of specific QIs analytically [20] or empirically [87], the effectiveness of a set of QIs when compared with benchmark functions [102] or when applied to real-world problems [169], and finally, correlation analyses of different QIs [136].

Reviews of certain aspects of QIs focus on providing an overview from different perspectives, like a review for exact multi-objective optimisers [57], a critical review of various well-established QIs [99, 101, 128], a statistical review of the use of QIs in the literature [137, 169], an instructional review of QI design [70, 102] or choosing suitable QIs for particular problems [110, 169].

Li and Yao [111] discussed the main aspects related to quality evaluation, systematically reviewing a hundred QIs, analysing the strengths and weaknesses of these representative QIs, providing detailed recommendations for the design of QIs, and finally, how to select and use QIs.

2.6.1 Overview of quality indicators

QIs are judged in terms of four main criteria, namely *convergence*, *spread*, *uniformity*, and *cardinality* [111, 137]. Convergence of a solution set refers to how close the set is to the Pareto set. Spread of a solution set refers to how well the set covers the various parts of the Pareto set. This stands in contrast to the quality of *extensity*, which only considers the boundaries of the set (spread takes into account the outer, as well as the inner portions of the set). Sayın [142] also called this quality the *coverage* of the set. Uniformity of a set refers to how evenly the solutions are distributed across the set, where an equidistant spacing amongst solutions is desirable. Spread and uniformity are closely related, and together are known as the *diversity* of a set. The number of solutions contained in a solution set is referred to as the cardinality of the set, and it is desirable to have sufficient solutions to convey the set to the decision maker, but also not too many as this may be overwhelming for decision makers [111]. Nevertheless, when comparing two sets generated by expending the same amount of computational resources, the one containing the larger number of solutions is preferred.

When assumptions are made about the decision maker's preferences with regards to comparing different solution sets, the weak outperformance relations most generally form the basis for comparisons [40, 111]. When the decision maker's preferences are not known *a priori*, however, solution sets with sufficient cardinality, exhibiting good convergence, which are well spread over the range of the Pareto set, and achieve good uniformity amongst solutions are deemed indicative of high quality. Decision makers are more likely to prefer solution sets that adhere to these qualities [111]. A high-quality representation of the Pareto set can disclose the problem's underlying properties, revealing to the decision maker the shape, scale, actual dimensionality, elbow point and the relationship between the objectives.

An intuitive way of comparing MOP solution sets is by visualisation. This is, however, only practical for MOPs with two or three objectives. When the number of objectives exceeds three, direct visualisation is no longer feasible by scatter plot and other possibilities have to be sought from the field of data analysis. Furthermore, visual comparison lacks the ability to quantify the differences between solution sets, and therefore cannot be used to guide the search [111]. This problem may be overcome through the use of QIs, whereby the set is mapped to a real number, thereby providing a quantitative measure of comparison between sets. QIs have the ability to quantify precisely the quality of a solution set based on the aspects included in the QI design. Not only can a QI reveal that the one solution set outperforms another, but also by how much [111, 137]. Any function that maps a set of vectors to a scalar value can, in principle, be seen as a potential QI, but such a measure is ideally required to reflect on one or more of the aforementioned aspects of the set quality, namely convergence, spread, uniformity, and cardinality. When comparing the quality of solution sets returned by exact methods, the aspect of convergence is, of course, excluded, since the solution set is necessarily a subset of the Pareto set [111]. In the literature, QIs are classified into one of six categories, namely QIs for convergence, spread, uniformity, cardinality, both spread and uniformity, and finally, QIs for the combined quality of all four quality aspects. In the remainder of this section, QIs in all of these categories are reviewed.

Convergence is viewed as the most important aspect of a solution set's quality, and has received considerable attention in the literature over the years. There are two main classes of convergence QIs: One based on the Pareto dominance relation between solutions or sets, and one based on the distance between a solution set and a Pareto set or one/several point(s) derived from the Pareto set [111, 137]. Dominance-based QIs, however, exhibit certain weaknesses, such as that they provide little information about the extent to which one solution set outperforms another.

Furthermore, a situation in which all solutions of the sets are non-dominated may lead to incomparable solution sets — a situation that often occurs in MOPs [111]. It should also be noted that some dominance-based QIs indirectly allude to the cardinality of a solution set, since larger solution set sizes will naturally include more non-dominated solutions [111]. Distance-based QIs, on the other hand, are more common than dominance-based QIs in the context of convergence. Distance-based QIs can further be subclassified into two categories: One measuring the distance between the solution set and one or more particular points derived from the Pareto set, and measuring the distance between the solution set and a reference set that adequately represents the Pareto set [111]. The so-called $\mathcal{C}$ *indicator* [191] is a well-known dominance-based QI, whereas the *Generational Distance* [163] QI is popular for measuring the distance between solution sets and the associated Pareto set of the given problem instance [111, 137].

Spread is concerned with the area covered by a solution set, where the aim of a good spread is to cover areas of every portion of the Pareto set [111, 137]. Most QIs based on spread measure the extent of the spread, such as the range formed by the extreme solutions of the set. The aim should, however, also be to take into account the solutions between the boundary solutions, as this may yield a clearer description of a Pareto set [111]. The difference between boundary and extreme solutions is elucidated in Figure 2.9. One of the best-known QIs based on spread is *maximum spread* [188], in which the maximum extent of each objective is considered.

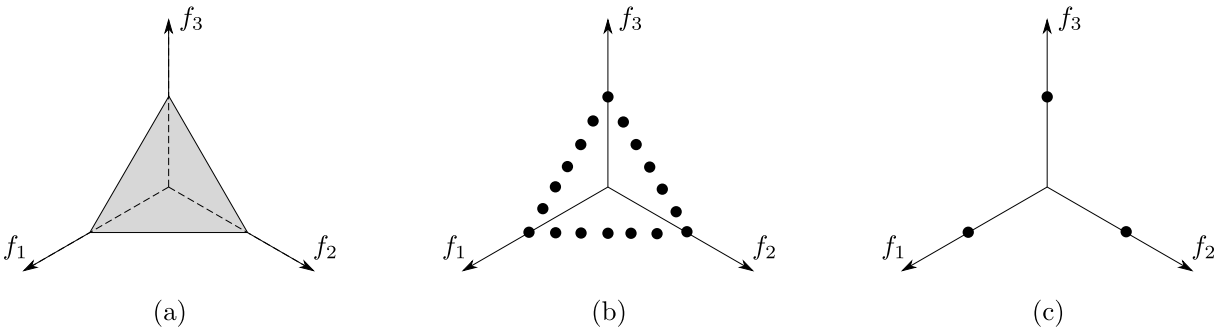


FIGURE 2.9: *A tri-objective example of boundary solutions, where the Pareto set, boundary solutions and extremal points are indicated in (a), (b) and (c) respectively [111].*

Uniformity is concerned with how uniformly a set of solutions is distributed [111, 137]. The quality of a solution set is measured by a QI based on uniformity in terms of how well it can represent the Pareto set. A solution set providing more uniformly distributed solutions is a better representation than another less uniformly distributed set — the former is therefore considered of a higher quality. A desirable QI based on uniformity should rate a solution set one of the highest possible quality if all of its solutions are equally spaced, while when one solution of this same set is disturbed, it should yield a worse measurement result [111]. The quality of uniformity can often be measured by calculating the variation of distances between the solutions in a set. It should be noted, however, that having equally spaced solutions does not necessarily imply a good diversity of solutions also, and therefore QIs based on uniformity should ideally be used in conjunction with QIs based on spread [111]. *Spacing* [144] is a popular QI for measuring uniformity in solution sets.

Cardinality may simply be described as counting the number of non-dominated solutions in a solution set [111, 137]. A necessary property of QIs based on cardinality is that when an additional non-dominated solution is added to the set, the quality measure should improve rather than degrade. In the literature, QIs based on cardinality can be partitioned into two main groups: One in which the non-dominated solutions in the set are directly considered and one in which the non-dominated solutions of the solution set are compared with the Pareto optimal set, typically

returning a ratio between the two cardinalities [111]. Since cardinality does not provide explicit information about whether a solution set accurately describes the Pareto set, it is often deemed less important than the other three aspects of quality during solution set evaluation. Cardinality measurement becomes more valuable when search techniques can actually find Pareto optimal solutions, and this is particularly the case when dealing with combinatorial MOPs in which the total number of Pareto optimal solutions is often small. In such cases, counting the number of Pareto optimal solutions uncovered can prove a reliable QI for the solution set [111]. The *error ratio* [164] is a popular QI based on cardinality, and measures the proportion of non-dominated solutions in the solution set that are also in the Pareto optimal set.

As previously mentioned, spread and uniformity are closely related and ideally have to be considered in conjunction with one another in order to reflect the diversity of solution sets. QIs measuring both spread and uniformity can be classified into two classes: Distance-based indicators and region division-based indicators [111]. One of the first measures in the class of distance-based indicators was the Δ *indicator* [42] proposed by Deb in 2002. Such QI evaluation can, however, only be performed in the context of bi-objective optimisation problems and furthermore require information about the Pareto set to be used as a reference — this is often unknown in practice [111]. Region division-based QIs, on the other hand, can be applied to more than two objectives. The main idea behind this QI is to partition a particular space into many equally sized cells, which may overlap or be disjoint, and then to count the number of cells containing solutions of the set. The underlying principle is that a more diversified solution set will result in more cells containing solutions. Most QIs based on both spread and uniformity reside in this class, where different potential shapes of the cells are taken into account. The *Chi-square-like deviation* [152] was one of the first cases of measuring diversity using a region division-based QI.

Finally, there are also QIs that take into account all the quality aspects for evaluating MOP solution sets, and these are also the most commonly used in the literature. Generally, they can be partitioned into two classes: Distance-based QIs and volume-based QIs [111, 137]. The main idea behind distance-based QIs is to measure the distance between the Pareto set and the solution set under evaluation. The smaller the distance between the Pareto set and the solution set, the more representative the solution set is of the Pareto set [111]. Only solution sets exhibiting small distances between the reference set and solution set will reflect a high quality in terms of all four aspects of convergence, spread, uniformity and cardinality. This idea is practically achieved by averaging the distances between members of the reference set and their closest solutions in the solution set, or alternatively, the maximum value of these distances may be used, referring to the worst of the solutions [111]. A popular distance-based QI measuring all four quality aspects is the *inverted generational distance* [35]. Volume-based QIs, on the other hand, measure the size of the volume that the solution set covers in objective space, based on some predefined specifications [111]. *Hypervolume* (HV), a volume-based QI [192] introduced by Zitzler and Thiele in 1998, has received considerable attention in the literature. It is based on measuring the volume of objective space enclosed between the solution set and a reference point, specified *a priori* [111]. It is worth noting that no single QI is able to perfectly capture all four quality aspects of a solution set. This is not surprising, as some of the quality aspects are conflicting, such as convergence and uniformity. Consider, for instance, the case where one may move a solution a little in objective space in order to achieve better convergence. This will lead to a set having better convergence, but worse uniformity [111]. Li and Yao [111] indicated in their comparison of the various QIs, that HV is the only QI able to reflect quality based on cardinality among the QIs designed to measure all four quality aspects, due to being fully consistent with the *weak outperformance* relation, as mentioned in §2.3.4.

2.6.2 The hypervolume quality measure

HV was originally introduced by Zitzler and Thiele [192] as the *size of space covered*, and later called the *hyperarea metric* by Van Veldhuizen [164]. Other common names for HV are the *S-metric* [191] and the *Lebesgue measure* [58]. Arguably, the HV indicator is one of the most commonly used QIs when comparing the output sets returned by MOP search techniques [111, 137, 170], due to its practical applicability, theoretical properties and the fact that it does not require a reference set representing the Pareto set [111]. The HV measure is classified as a *unary* quality indicator, mapping solution set quality to the set of positive real numbers [137]. This makes the HV suitable during analyses of real-world problems when the qualities of different candidate solution sets have to be compared. Furthermore, HV is sensitive to any improvement in a solution set with respect to Pareto dominance [111], *i.e.* when a set $\mathcal{A}$ is better than another set $\mathcal{B}$, denoted by $\mathcal{A}\mathcal{O}_{\mathcal{W}}\mathcal{B}$, then the HV measure will return a larger value for $\mathcal{A}$ than $\mathcal{B}$. Therefore, a set which achieves the maximum HV possible for a given problem instance will contain all solutions from the Pareto set of the problem [111, 137].

Let $\mathcal{S}_{\boldsymbol{x}} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}$ be a set of non-dominated solutions to an MOP instance with objective function vectors $\mathcal{S}_{\boldsymbol{z}} = \{\boldsymbol{z}^1, \ldots, \boldsymbol{z}^N\}$ in objective space. The HV of the set $\mathcal{S}_{\boldsymbol{x}}$ may be defined as the hypervolume of the corresponding objective space that is dominated by the objective function vectors in $\mathcal{S}_{\boldsymbol{z}}$ with respect to a specific reference point $\bar{\boldsymbol{z}}$, which satisfies $\boldsymbol{z}^\ell \preceq \bar{\boldsymbol{z}}$ for all $\ell \in \{1, \ldots, N\}$. This may be stated mathematically as

$$HV(\mathcal{S}_{\boldsymbol{x}}) = \lambda\left(\bigcup_{\boldsymbol{z}\in\mathcal{S}_{\boldsymbol{z}}} \{\boldsymbol{a} \mid \boldsymbol{z} \prec \boldsymbol{a} \prec \bar{\boldsymbol{z}}\}\right) \tag{2.6}$$

[111], where λ denotes the Lebesgue measure, being a *hypervolume difference* measure [58]. Simply put, the HV of a set is the union of the hypercube volumes determined by each of the solutions and the reference point, $\bar{\boldsymbol{z}}$, in objective space [111]. A visualisation is provided in Figure 2.10 of how this hypercube volume may look for a bi-objective minimisation MOP.

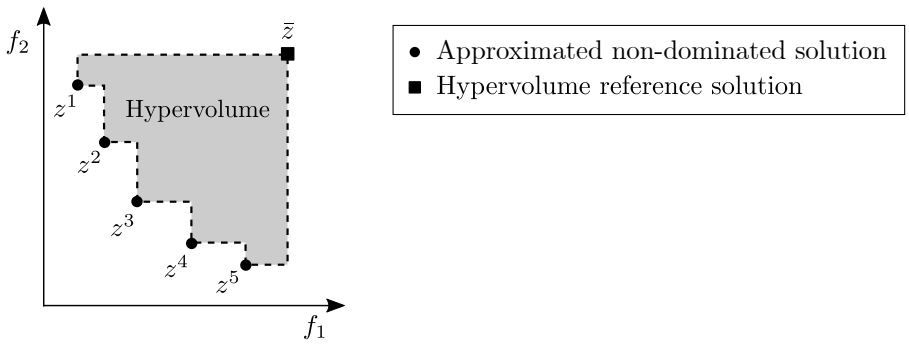


FIGURE 2.10: *An example of an objective space for a bi-objective optimisation problem in which the functions f_1 and f_2 are to be minimised, and the HV is represented by the shaded area between the non-dominated solution set $\mathcal{S}_{\boldsymbol{z}} = \{\boldsymbol{z}^1, \ldots, \boldsymbol{z}^N\}$ and the reference point $\bar{\boldsymbol{z}}$.*

One limitation of the HV QI is that its evaluation time increases exponentially as the number of objectives in the MOP increases [18], rendering it among the computationally more expensive QIs. This limits the use of the HV measure when applied to high-dimensional spaces, especially when integrated into the search mechanism itself [111]. The problem of computing the HV was shown to be a special case of Klee's measure problem [13]. For the two- and three-dimensional cases, the optimal computational time is $\mathcal{O}(N \log N)$ [14]. Li and Yao [111] have claimed that While *et al.* [170] proposed one of the most efficient methods for calculating HV. Efforts have

been made to render HV computation as efficient as possible, in order to make it practicable during solution set evaluation within a reasonable time-frame, even when having more than ten objectives, but, incorporating it as an indicator to help guide the search remains a challenge [111].

Another disadvantage of computing HV is deciding upon a suitable reference point. In the literature, no consensus has been reached on how to choose a proper reference point for a given problem instance. Different reference point choices can lead to the HV evaluation results being inconsistent, which is undesirable when comparing solution sets [100]. There are, however, certain common practices, such as taking the *ideal point*, which is the vector of best values for each of the objectives of the problem [111] (as illustrated in Figure 2.11). The true ideal point is often unavailable for a problem instance, but an estimated point may instead be used, obtained by studying the solution set under consideration, or by optimising each of the problem objectives individually [111]. It is crucial that the ideal point should not be underestimated, as this may skew the results due to certain non-dominated solutions then not contributing fully to the value of the QI [111]. Furthermore, boundary solutions may typically contribute less to the HV than inner solutions, even if an accurate ideal point is selected as reference point. It is therefore advisable to choose a reference point which is slightly better than the ideal point. An example of this is taking $z_m = z_m^* - \ell_m/(2N-2)$ in a scenario with M objectives that all have to be minimised, where z_m denotes the reference point of the m-th objective function, z_m^* denotes the ideal point of the m-th objective function, ℓ_m denotes the range of objective $m \in \{1, \ldots, M\}$ and N is the size of the solution set. This choice allows the boundary solutions to contribute approximately equally to the QI measure as do the inner solutions (when a uniformly distributed solution set is considered, spread over a linear Pareto set) [111].

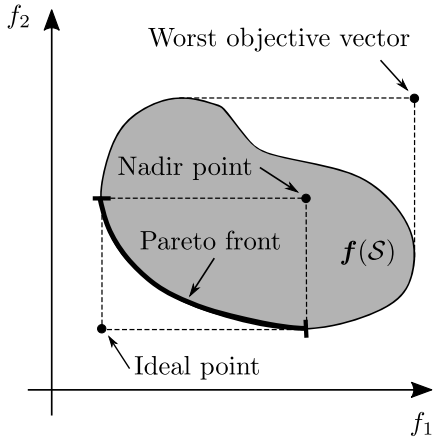


FIGURE 2.11: *The objective space of a bi-objective minimisation problem, and where the ideal point, nadir point and worst objective vector would be located for this instance, adapted from [41]. The feasible region is denoted by $\boldsymbol{f}(\mathcal{S})$, and the Pareto set is highlighted as the boldfaced portion of the boundary of $\boldsymbol{f}(\mathcal{S})$, where $\mathcal{S}$ denotes the decision space.*

Another common practice is taking the *nadir point* of the Pareto set [97] as reference point (as illustrated in Figure 2.11), or 1.1 times the nadir point of the solution set being evaluated [111]. The nadir point is the point containing the worst of the objectives contained within the Pareto optimal set [97]. The nadir point is an important element when it comes to MOPs, as it defines lower and upper bounds for the Pareto optimal set [97]. If this point is, however, chosen incorrectly, the related calculation of the HV will contain a margin of error, either giving the impression that a solution set is better or worse than it actually is [97]. For example, if the Pareto optimal set in a bi-objective minimisation problem is estimated to have a nadir point of $(10, 10)$, and somehow a model produces a value outside these bounds, such as $(15, 5)$ for

example, then the calculation of the HV will result in an error due to it not being calculated within the given bounds, or return a disproportional HV for a solution that is not considered as being of high quality. It is therefore critical to consider what the reference point should be in order to ensure that an accurate estimate is obtained of the quality of solution set produced [97].

Also, compared to the ideal point, the nadir point determination may be challenging, even for simple problem instances [70]. One main reason for this is the potential presence of *dominance resistant solutions* (DRSs) within a solution set (non-dominated solutions having an extremely weak performance in one objective, but (near) optimal performance in other objectives) [83]. If the nadir point is estimated from a collection of solution sets containing DRSs, the evaluation results could be greatly affected. Consider, for example, the situation where, during calculation of the HV measure, the reference point is set to equal to (or worse than) such DRSs, corresponding to the objective in which they are performing well. This may make the contributions of some boundary solutions significantly larger than those of the inner solutions [111].

Moreover, HV is sensitive to the scaling of a problem's objectives, as well as the presence or absence of extremal points [170]. *Normalisation*, also known as scaling, may be required for the calculation of QIs [111]. When the problem objectives are incomparable, the result is often that the larger the range of an objective, the greater the effect of that objective on the indicator value [111]. It is desirable that each different objective contribute equally to the indicator value, and therefore a proper conversion of objectives is necessary, enabling their values to lie within the same range, *e.g.* the unit interval $[0, 1]$. A common objective function transformation form is $\boldsymbol{a}'_m = (\boldsymbol{a}_m - \min_m)/(\max_m - \min_m)$, where $\min_m$ and $\max_m$ denote the minimum and maximum of objective m, respectively, ideally with respect to the Pareto set, but if not, then with respect to the combined set of non-dominated solutions of the collection of all the solutions sets being considered [111]. Once again, the presence of DRSs can skew the evaluation results.

HV has become popular as a result of its in-line use in multi-objective evolutionary algorithms [170], either promoting diversity [79], being part of the archiving mechanism [100], or playing a part in the selection process [190].

2.7 Chapter summary

This chapter was devoted to a review of a number of prerequisites for the material presented later in this dissertation. A review of relevant basic notions from the realm of graph theory was provided in §2.2.1, while two well-known procedures for finding shortest paths in weighted graphs, called Dijkstra's algorithm and Floyd's algorithm, were described and illustrated in §2.2.2, along with Yen's algorithm for finding the K shortest paths. Thereafter, the focus shifted to a review of basic concepts from the realm of multi-objective optimisation in §2.3, where a formulation of the general multi-objective optimisation problem was given in §2.3.1, while it was described in §2.3.2 how the notion of solution dominance gives rise to the concept of Pareto optimality, which was explored in §2.3.3. Thereafter, relations that may occur between different solution sets, when they are compared with one another, were considered in §2.3.4. Three methods were described in §2.4 for determining the non-dominated subset of solutions of a given set of candidate solutions to an MOP instance, namely the naive and slow method, the continuously updating method, and the efficient method of Kung *et al.* [105]. In §2.5, the notion of non-dominated sorting of solution sets was described, according to which candidate solution sets are sorted and ranked into non-dominated fronts. Consideration was finally afforded in §2.6 to the evaluation of a solution set quality and the importance thereof.

CHAPTER 3

Solving combinatorial optimisation problems

Contents

When considering a given optimisation problem, there are typically various approaches in the literature that can be applied in an attempt at solving it. A broad classification of these different approaches involves categorising them as either exact, heuristic, metaheuristic or hyperheuristic solution approaches. Within each of these classes, a further distinction may be drawn between solving SOPs or MOPs. The approach adopted also typically depends on the type of decision variables present in a problem instance, due to the considerably different qualities of problems involving differentiable functions, and those involving discrete variables.

The focus in this chapter is on solving combinatorial optimisation problems, frequently leading to a large number of possible combinations and permutations of feasible variable values as the number of input variables increases, and therefore requiring carefully tailored techniques to solve such problems within a reasonable time frame. This chapter is devoted to a discussion on the various types of solution approaches available when solving combinatorial optimisation problems, with an emphasis placed on the solution methodologies applied later in this dissertation when solving instances of combinatorial optimisation problems.

The discussion opens in §3.1 with a brief overview of exact solution methodologies for solving combinatorial optimisation problems, namely the method of total enumeration, the popular branch-and-bound method, the cutting plane method and the branch-and-cut method. These approaches have the elegant quality of being able to provide optimal solutions to optimisation problem instances in a structured manner, albeit at the cost of often large computation times. As the dimensions of optimisation problem instances increase in some cases, the computational cost associated with these approaches grow prohibitively expensive. For this reason they are typically exchanged for computationally more efficient inexact options, such as heuristics, metaheuristics or hyperheuristics, which are nevertheless often able to obtain near-optimal solutions.

Heuristic optimisation approaches are discussed briefly in §3.2, as applicable to the general context of solving combinatorial optimisation problems. Thereafter, the realm of metaheuristic optimisation is entered in §3.3, where the difference between the sub-classes of trajectory-based

and population-based metaheuristics is elucidated, and some examples are provided. Finally, one metaheuristic from each of the latter classes is examined in more detail, both in their single-objective and multi-objective contexts. These two metaheuristics are applied later in this dissertation to solve model instances. A brief overview is provided in §3.4 on hyperheuristics, along with some common classifications and a closer look at one hyperheuristic from the literature in particular, which bears a resemblance to an algorithm employed later in this dissertation. The chapter closes with a brief summary of its contents in §3.5.

3.1 Exact solution approaches

The quest for optimal solutions to combinatorial optimisation problems is a common theme in the OR literature [74]. Adopting an exact solution approach to solve such an optimisation problem instance holds the advantage of returning a guaranteed optimal solution to the problem instance. This benefit is, however, only practical when the problem dimensions allow for the computation of such a solution, because uncovering an exact solution to a combinatorial optimisation problem instance is typically computationally expensive. This section contains a brief discussion on four methods that may be adopted to solve combinatorial optimisation problems exactly, namely the method of *total enumeration*, the *branch-and-bound* method, *the cutting plane method* and the *branch-and-cut* method.

Total enumeration is one of the simplest exact solution approaches that can be applied to solve a combinatorial optimisation problem instance exactly. This method involves a complete enumeration of all possible solutions, keeping track of the solutions that achieve the best objective function values [135]. For large problem instances, however, the method is usually prohibitively expensive from a computational point of view, and is therefore typically only applied to small problem instances. The method can be classified as an explicit method under the larger umbrella of exact solution approaches.

It is recommended that for larger problem instances, alternative methods be applied which are computationally less expensive. The celebrated *branch-and-bound* method, proposed by Land and Doig [107] in 1960, is one such alternative. Application of this algorithm involves enumerating the set of candidate solutions systematically by discarding subsets of provably unsuccessful candidate solutions [135, 145]. This feature of discarding subsets of candidate solutions significantly reduces the computational resources required to enumerate the solution space in search of an optimal solution. Compared with total enumeration, the method carries a substantially less expensive computational burden. Candidate solutions are considered in subsets corresponding to sub-problems of the original combinatorial optimisation problem whose ranges are the relevant candidate solution subsets [74]. Each sub-problem, exhibiting relaxed constraints, is then solved separately and the best solution thus found is used during further analysis steps within the larger search. The method is classified as an implicit enumeration method for solving optimisation problems exactly.

Consider an SOP with objective function $f(\boldsymbol{x})$ which has to be minimised, where $\boldsymbol{x}$ is a discrete decision variable vector. Denote the set of candidate solutions to the optimisation problem by $\mathcal{S}$. This set is called the *search space* and contains all feasible decision variable vectors $\boldsymbol{x}$ [145]. The name branch-and-bound is indicative of the two main procedures employed in the algorithm, namely *branching* and *bounding*, both being user-specified procedures. The branching procedure entails identifying two or more smaller subsets of solutions, denoted by $\mathcal{S}'_1, \mathcal{S}'_2, \ldots$ and taken from a given subset $\mathcal{S}' \subseteq \mathcal{S}$, therefore each achieving a smaller cardinality than that of the entire set of candidate solutions $\mathcal{S}$. The subsets $\mathcal{S}'_1, \mathcal{S}'_2, \ldots$ are chosen such that their union is the set $\mathcal{S}'$. The

branching procedure is furthermore selected in such a way that it does not produce overlapping subsets. A *search tree* is constructed, which is a tree-like data structure in which the subsets $\mathcal{S}'_1, \mathcal{S}'_2, \ldots$ are each represented by nodes in the tree. Let $f(\boldsymbol{x}_i)$ denote the minimum objective function value attained within the subset $\mathcal{S}'_i$, at the point $\boldsymbol{x}^*_i$. The minimum value of $f(\boldsymbol{x})$ over the subset $\mathcal{S}'$ is therefore the minimum value in the set $\{f(\boldsymbol{x}^*_1), f(\boldsymbol{x}^*_2), \ldots\}$.

The bounding procedure, on the other hand, is employed to determine upper and lower bounds on the minimum value of $f(\boldsymbol{x})$ within a given subset $\mathcal{S}' \subseteq \mathcal{S}$ [145]. This may be achieved by adopting a number of methods, usually applied according to a user-specified protocol, and leads to a subset of candidate solutions in $\mathcal{S}$ being discarded. Suppose, in the context of a minimisation problem, that the lower bound on the optimal objective function value achieved by some node $\mathcal{S}'_i$ of the search tree is *larger* than the upper bound on this value of some other node $\mathcal{S}'_j$, with $i \neq j$. Node $\mathcal{S}'_i$ can then be discarded from the search, since it is clearly incapable of achieving a better solution than that already obtained in $\mathcal{S}'_j$. A global variable u_b, containing the smallest upper bound on the optimal objective function value uncovered throughout the search, is maintained as the search progresses. Any node achieving a lower bound value on the optimal objective function value which is larger than the value of u_b may be discarded from the search tree without possibly overlooking an optimal solution. Elimination of nodes from the search tree in this manner is called *pruning* [135, 145]. The branch-and-bound algorithm terminates when a user-specified stopping criterion is met. A stopping criterion typically employed involves terminating the algorithm when a candidate solution set $\mathcal{S}'$ reduces to a singleton set after repeated bounding, or when the lower bound on the optimal objective function value of a set $\mathcal{S}'$ corresponds to the upper bound of any available objective function value. In the latter case, a minimum objective function value $f(\boldsymbol{x})$ would have been achieved within $\mathcal{S}'$ at any remaining element of $\mathcal{S}'$ within the search tree.

The *branch-and-cut* algorithm [76] is, in essence, a combination of the branch-and-bound algorithm and the *cutting plane* algorithm [65]. The cutting plane algorithm was introduced in 1958 by Gomory [65], and can be viewed as an alternative to the branch-and-bound algorithm for solving integer programming problems. The difference between the two is that instead of branching on non-integer values to create subsets of the search space, as is done in the branch-and-bound method, the cutting plane method iteratively cuts away portions of the feasible region containing non-integer valued solutions. This is repeated until an optimal integer solution is obtained. These successive cuts each corresponds to an additional constraint being imposed on the relevant linear programming relaxation of the optimisation problem at hand, and are called *cutting planes*.

The notion of rewriting a non-integer valued real number y as $y = \lfloor y \rfloor + c$, where $\lfloor y \rfloor$ is the largest integer not exceeding y and c is a real number in the unit interval $[0, 1)$, is incorporated in determining cutting planes. For example, $2.45 = \lfloor 2.45 \rfloor + 0.45$, while $-1.25 = \lfloor -1.25 \rfloor + 0.75$.

A cutting plane may be created by randomly selecting a binding constraint imposed on a non-integer valued optimal solution found to the linear programming relaxation of a given integer programming problem. Next, all the terms of the selected binding constraint are rewritten with non-integer valued coefficients corresponding to the floor notation above rearranged so that all the terms having integer coefficients appear on the left-hand side of the equation, while the remainder of the terms, appearing on the right-hand side of the equation, equate to zero. This newly formed equation represents a cutting plane [171]. The method leads to more and more constraints successively being imposed on the linear programming relaxations of constrained versions of the original discrete optimisation problem, leading to better solutions containing fewer and fewer non-integer variable values [74].

The cutting plane procedure may be combined with the branch-and-bound method to accelerate finding optimal solutions in the context of integer programming [74]. This is achieved by alternating between branching and inserting new cutting planes, in the attempt to generate a smaller search tree, as opposed to branching at each level of the search tree. The branch-and-cut procedure was formalised in 1991 by Hoffman and Padberg [76].

3.2 Heuristic optimisation approaches

The word *heuristic* stems from the Greek word *heuriskein*, which means *to find*, expressing the main notion pertaining to heuristics. Heuristic solution approaches are methods employed to find near-optimal, approximate solutions to optimisation problems that are deemed too complex to be solved exactly within an acceptable time frame [74]. Typically, these approaches pursue high-quality feasible solutions, as opposed to necessarily optimal solutions, although such exact solutions could potentially be found. In the OR literature, a broad distinction is made between three classes of heuristics, namely *iterative heuristics*, *constructive heuristics* and *local search heuristics*.

As the name implies, iterative heuristics, are iterative procedures that are rigid and simple in nature [74]. A new, hopefully better, solution is considered during each iteration, and usually incorporates expert knowledge or intuitive rules of thumb for solution selection. The iterative search process is repeated until some user-specified stopping condition is met, and at termination of the search, the best solution encountered during the entire process is returned. Typically, this type of heuristic leads to solutions of inferior quality.

Constructive heuristics, on the other hand, function by iteratively choosing components by which to build up a solution incrementally, in a greedy fashion, without consideration of future consequences of these choices. Such an algorithm is typically initialised with an empty solution which is built up iteratively until a complete solution has been obtained, based on a set of predefined rules. During each step of construction, the component that yields the best improvement is typically selected for insertion into the partial solution. This type of heuristic usually also leads to solutions of inferior quality.

Local search heuristics involve the generation of an initial feasible solution to the optimisation problem at hand, and then iteratively proceed to change or perturb a single element of the solution at a time in an attempt to uncover an improved solution. This process is repeated, aiming only to accept an improving solution as the next solution to be considered. If a non-improving solution is encountered, it is simply discarded and another change or perturbation is imposed on the current solution, leading to a sequence of strictly improving solutions. When no improvement can be found to some current solution under consideration after attempting all possible changes, the algorithm terminates. Local searches, however, typically lead to locally optimal solutions instead of globally optimal solutions.

A heuristic optimisation approach is typically customised to a specific optimisation problem instance, and is therefore not suited to be applied generally to various problems exhibiting different characteristics [74]. The disadvantage of this *ad hoc* approach is that a new optimisation procedure has to be developed for each new optimisation problem considered, which makes for quite a cumbersome optimisation approach in practice.

3.3 Metaheuristic optimisation approaches

The use of *metaheuristic* solution approaches is today common practice in the multidiscipline of OR. A metaheuristic may be defined as "a general kind of solution method that orchestrates the interaction between local improvement procedures and higher-level strategies to create a process that is capable of escaping from local optima and performing a robust search of a feasible region" [74]. In other words, metaheuristics are search techniques that provide a general structure and set of rules for developing a customised solution approach that is applicable to more than one type of optimisation problem. This characteristic of a metaheuristic overcomes the somewhat limiting problem-specific nature of heuristics. Metaheuristics are commonly employed to solve combinatorial optimisation problems in light of the phenomenon of combinatorial explosion often observed whereby the exact solution of such problems quickly becomes intractable as the problem dimensions increase.

In this section, the two main classes of metaheuristics are reviewed, namely trajectory-based metaheuristics and population-based metaheuristics in §3.3.1. Thereafter, one well-known metaheuristic from each class is examined in detail in §3.3.2 and §3.3.3, respectively, in both their single- and multi-objective incarnations. These two metaheuristics, SA and the GA, are applied extensively later in this dissertation in their multi-objective incarnations.

3.3.1 Trajectory-based vs population-based metaheuristics

As alluded to above, metaheuristics may be classified into two main categories, namely *trajectory-based* metaheuristics and *population-based* metaheuristics. In trajectory-based metaheuristics, a single candidate solution is iteratively maintained and improved while seeking to uncover a globally optimal solution. Well-known trajectory-based metaheuristics include the method of SA (due to Kirkpatrick *et al.* [96]), the method of *tabu search* (due to Glover [62]), the method of *variable neighbourhood search* (due to Mladenović and Hansen [120]), and the method of *harmony search* (due to Geem *et al.* [61]). In population-based metaheuristics, on the other hand, sets of multiple candidate solutions are iteratively maintained and improved together in search of uncovering a globally optimal solution. Well-known population-based metaheuristics include the GA (due to Holland [77]), *ant colony optimisation* (due to Dorigo [46]), and *particle swarm optimisation* (due to Kennedy and Eberhart [92]).

The tabu search algorithm was originally proposed by Glover [62] in 1986. The overarching idea is to make use of information obtained during the successive search iterations, stored in memory, to render the search process more efficient. Non-improving solutions are accepted as a mechanism to help the algorithm escape from local optima. During each iteration, the entire neighbourhood of the current candidate solution is explored deterministically. What distinguishes tabu search from other metaheuristics is the element of incorporating a short-term memory for the purpose of not revisiting previously encountered solutions. The method of tabu search starts with a randomly selected initial candidate solution. A tabu list of pre-specified length (the short-term memory) is maintained, containing the recent history of previously visited solutions. Any solution found in the tabu list is not accepted as a viable solution during the search, helping the algorithm to avoid becoming trapped in cycles. A new candidate solution is generated during each iteration, and consequently, accepted as the new current solution on condition that it does not appear in the tabu list and is better than the current solution. If, however, no improvement in the objective function value is attainable in the neighbourhood of the current solution, the best acceptable neighbour is selected to replace the current solution during the next iteration [63].

The main principle applied in the variable neighbourhood search algorithm, proposed by Mladenović and Hansen [120] in 1997, is the systematic exploitation of neighbourhood change, both in finding local optima and escaping from them. Various neighbourhood structures are maintained and systematically changed for this purpose during the search. The main concept on which variable neighbourhood search is based is that one solution can be a local optimum with respect to one neighbourhood structure, yet not with respect to another neighbourhood structure. Furthermore, a local optimum with respect to all neighbourhood structures is a global optimum, while local optima with respect to one or more neighbourhoods are, for many problems, often relatively close to each other [71]. The last observation implies that a local optimum contains some information about the global optimum (for example, some variables with the same values could occur in both). It is not always clear in which instances this is true, however, and conducting an organised study of each local optimum encountered is an important feature of the method, until another local optimum is found to yield an improvement, and is further analysed [71].

Harmony search, a music-inspired metaheuristic, was originally proposed by Geem *et al.* [61] in 2001. An analogy is drawn between finding a perfect state of harmony within music and finding an optimal solution to an optimisation problem [178]. The search process is compared with the musical improvisation process usually undertaken by jazz musicians, namely playing from memory some piece of music, playing something similar to what is remembered or choosing to play something completely different. Geem *et al.* [61] formalised these concepts to be applicable to the optimisation process, where the three concepts become the usage of harmony memory, pitch adjusting, and randomisation, respectively. This alludes to incorporating some of the best known solutions in memory, generating slightly varying solutions from other solutions based on different strategies, and introducing randomisation for increased diversity [178].

The population-based method of ant colony optimisation was introduced by Dorigo [46] in 1992, drawing its inspiration from the cooperative foraging behaviour and characteristics of ants. As ants explore their environment, they deposit a pheromone trail, a substance that can be detected by other ants to help direct them to food sources, but which evaporates over time. Ants are adept at finding shortest paths, due to the pheromone concentration being higher along shorter paths taken by ants than along longer paths [63]. Ants following pheromone trails, increase the pheromone concentration along such trails even more, which eventually leads to all ants following the shortest path. This concept was adopted by Dorigo [46] when formulating the ant colony optimisation algorithm. The algorithm is initialised with artificial ants that act as agents which iteratively generate solutions stochastically. Good solutions are reinforced through an expression that determines the pheromone concentration, also incorporating an evaporation rate so that the pheromone concentration diminishes over time. As solutions improve, the pheromone concentration increases, and the notion of evaporation serves as a mechanism for avoiding becoming trapped at local optima [63].

The method of particle swarm optimisation was proposed by Kennedy and Eberhart [92] in 1995, inspired by the flocking behaviour of birds or fish. This algorithm iteratively maintains multiple individual agents, called particles, each following their own trajectories. The objective space is explored by adjusting the various trajectories of the particles, allowing for both exploration and exploitation. Each particle's velocity and position vector is composed of a deterministic element and a stochastic element, initialised with a random position and velocity [178]. The position and velocity are adjusted based on a predefined expression, frequently incorporating the best solution obtained from the swarm at that point. Diversification is introduced through the incorporation of stochastic vector components and learning parameters. Exploitation is introduced as a result of deterministic motion towards the updated, currently best solution uncovered by the particle

itself and that of the globally best solution uncovered by the swarm. As particles draw closer to local optima, their velocities and the measure of randomness are reduced [178].

The two metaheuristic methods that are applied later in this dissertation are the method of SA and the GA. In particular, their multi-objective incarnations, the *dominance based multi-objective simulated annealing* (DBMOSA) algorithm and the *non-dominated sorting genetic algorithm II* (NSGA II), are applied in this dissertation. The following two subsections are therefore devoted to more detailed discussions on these two methods.

3.3.2 The method of simulated annealing

The method of SA has its origins in an analogy between solving combinatorial optimisation problems and the physical annealing of solids in metallurgy, where metals are heated to high temperatures and then cooled slowly in stages by quenching them in water until a near-optimal, low-energy crystalline state is reached [157]. Kirkpatrick [96] proposed the method of SA in 1983 for solving combinatorial optimisation problems approximately. The method is essentially a probabilistic iterative improvement search strategy which simulates the physical process of annealing. The working of the method of SA is described in this section, first in the context of single-objective optimisation (in which it was originally conceived), and then in the more general context of multi-objective optimisation.

Single-objective incarnation

Single-objective SA is applicable to optimisation problems in which one objective function $f(\boldsymbol{x})$ has to be minimised. The process is essentially an iterative search process in which small perturbations are made to a so-called *current solution* $\boldsymbol{x}$ during each search iteration in order to obtain a neighbouring solution $\boldsymbol{x}'$ of $\boldsymbol{x}$ in decision space. The change in objective function values $\Delta f(\boldsymbol{x}, \boldsymbol{x}') = f(\boldsymbol{x}) - f(\boldsymbol{x}')$, called the *difference in energy* between the solutions $\boldsymbol{x}$ and $\boldsymbol{x}'$, is observed. If $\Delta f(\boldsymbol{x}, \boldsymbol{x}') \geq 0$, then $\boldsymbol{x}'$ is considered an improvement over $\boldsymbol{x}$, in which case $\boldsymbol{x}'$ is accepted with probability 1 as the new current solution during the following search iteration. If, however, $\Delta f(\boldsymbol{x}, \boldsymbol{x}') < 0$, then $\boldsymbol{x}'$ is considered inferior to $\boldsymbol{x}$, in which case $\boldsymbol{x}'$ is accepted stochastically according to the so-called *Metropolis rule* [96], *i.e.* with probability

$$\exp\left(-\frac{\Delta f(\boldsymbol{x}, \boldsymbol{x}')}{T}\right), \tag{3.1}$$

where T is a control parameter of the search, called the *temperature* of the system. At large temperatures, most newly generated solutions $\boldsymbol{x}'$ are accepted, whereas at small temperatures, the probability of accepting inferior solutions is small. What makes this probabilistic acceptance rule useful is the resulting ability of the search process to escape probabilistically from locally optimal solutions in order to explore other parts of the decision space [157]. The temperature therefore represents a measure of search volatility, controlling a willingness to accept worsening solutions. For this reason, the SA search process is typically initiated with a large temperature value, and the temperature is then gradually lowered over various stages according to some cooling schedule, as in the physical process of annealing. In this way, it is anticipated that the search might converge on a globally optimal solution as the temperature decreases [96, 157].

It is therefore important to determine a good starting temperature for each search, and this temperature is problem-specific. Busetti [26] claimed that a good initial temperature should allow for approximately 80% of all non-improving neighbourhood solutions to be accepted at the beginning of the search so as to encourage diversification during the early stages of the search.

A temperature adhering to this rule of thumb cannot be known *a priori*, and should therefore be estimated during a trial search during which all non-improving solutions are accepted. The initial temperature may then be calculated as

$$T_0 = \frac{\Delta_f^+}{\ln 0.8}, \tag{3.2}$$

where Δ_f^+ denotes the average of all the changes in the objective function observed due to accepting non-improving solutions. The number of trial moves should be determined by the user as deemed fit.

Typically, the temperature is kept constant for a number of consecutive iterations, together called an *epoch*. The epoch length (*i.e.* the number of iterations the search spends at a particular temperature stage) is not a fixed parameter, but rather has a maximum number of iterations associated with it. A Markov chain of length L_c is typically used to determine the length c of an epoch. Busetti [26] noted that this value should not simply be a function of c, but instead be customised based on the optimisation problem at hand. Intuitively, it makes sense to require a pre-specified minimum number of move acceptances before lowering the temperature and incrementing the epoch number, as accepted moves are necessary to achieve search convergence towards the global optimum. Let the pre-specified number of move acceptances be denoted by $A_{\min}$. As the temperature T approaches zero while the number of epochs increases, the probability of accepting non-improving solutions decreases, and therefore the number of moves being attempted before accepting $A_{\min}$ moves becomes large as the search progresses, without bound and irrespective of the value of $A_{\min}$. This situation can be countered by terminating the epoch and raising the temperature once L moves have been attempted, or by lowering the temperature once $A_{\min}$ solutions have been accepted, adhering to the prerequisite that $L > A_{\min}$. Dreo *et al.* [47] provided a good rule of thumb, by setting $L = 100$ and $A_{\min} = 12N$, where N represents some measure of the number of degrees of freedom associated with the optimisation problem at hand.

According to the scheme outlined above, multiple instances of cooling with or without reheating can occur within one SA search. Exploitation is promoted through cooling, as cooling is aimed at making it harder to accept worsening solutions, while exploration is promoted through reheating, where the aim is to make it easier to accept worsening solutions, therefore facilitating an escape from local optima. The algorithm continues in this iterative fashion, until a stopping criterion is met. At termination, the algorithm returns the incumbent solution $\hat{\boldsymbol{x}}$ representing the best solution uncovered throughout the search. A pseudo-code description of the basic working of the method of SA is presented in Algorithm 3.1.

The main goal of SA is to find an optimal solution, typically a locally optimal solution of high quality, by expending minimal computational effort. The success of achieving this is heavily dependent on the set of input parameters (*i.e.* the stopping criterion, an epoch termination criterion, the initial temperature, the heating schedule and the cooling schedule) specified by the user, and should the tailored for each optimisation problem instance. Specifying a maximum number of epochs allowed is a popular stopping criterion. Another stopping criterion that can be employed involves specifying a stopping temperature, causing the algorithm to terminate when the temperature becomes close to zero for a prolonged period of time and thus allowing for convergence towards a locally optimal solution.

Popular cooling schedules employed in SA include linear, geometric and adaptive schedules [1, 12, 78, 166]. The most common, among these, is the geometric cooling schedule. In certain contexts, however, some of the other schedules have proven to be more efficient than the geometric schedule [166]. The essence of the geometric cooling schedule is that the temperature is reduced by a

Algorithm 3.1: Simulated annealing (for a minimisation problem) [96]

Input : An initial candidate solution $\boldsymbol{x}$, a maximum allowable number of epochs $C_{\max}$, and an initial starting temperature T_0.

Output : An approximately optimal solution $\hat{\boldsymbol{x}}$ to a single-objective minimisation problem.

Initialise the cooling schedule counter $c \leftarrow 0$
Initialise the incumbent solution $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
while $c \leq C_{\max}$ **and** *all feasible moves have not been rejected* **do**
    Generate a neighbouring solution $\boldsymbol{x}'$ from the current solution $\boldsymbol{x}$
    Compute the (possibly negative) net objective function improvement $\Delta f(\boldsymbol{x}', \boldsymbol{x})$ for moving from $\boldsymbol{x}$ to $\boldsymbol{x}'$
    Generate a random number $p \in (0, 1)$
    if $p < \min\left\{1, \exp\left(-\frac{\Delta f(\boldsymbol{x}', \boldsymbol{x})}{T_c}\right)\right\}$ **then**
        $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
    if $f(\boldsymbol{x})$ *is superior to* $f(\hat{\boldsymbol{x}})$ **then**
        $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
    If the number of iterations that have passed since the last temperature change is sufficient, reduce the temperature T_c and perform the update $c \leftarrow c + 1$
return $\hat{\boldsymbol{x}}$

fixed factor α, which ranges between 0 and 1. The temperature during epoch $c+1$ is expressed as

$$T_{c+1} = \alpha T_c, \quad c = 0, 1, 2, \ldots \tag{3.3}$$

Generally, the best results are obtained when α is chosen between 0.8 and 0.99, based on Vigeh's [166] investigation on good values for α. It should be noted here that the computational effort required for the algorithm's execution will be greater for larger values of α, since there will typically be more epochs due to an increase in the total number of temperature values considered before reaching temperatures close to zero.

According to the linear cooling schedule, on the other hand, the current temperature is reduced by subtracting a constant cooling factor k at the end of an epoch. The temperature during epoch $c+1$ is expressed as

$$T_{c+1} = T_c - k, \quad c = 0, 1, 2, \ldots \tag{3.4}$$

As opposed to the geometric cooling schedule, the linear cooling schedule reduces the temperature by the same value at the end of each epoch over the entire search.

An example of an adaptive cooling schedule is that of Huang *et al.* [78], according to which the temperature during epoch $c+1$ is expressed as

$$T_{c+1} = T_c \exp\left(-\frac{\Lambda T_c}{\sigma(T_c)}\right), \quad c = 0, 1, 2, \ldots, \tag{3.5}$$

where $\Lambda \in (0, 1]$ is a parameter that should be determined empirically and $\sigma(T_c)$ denotes the standard deviation of the changes in the objective function values since the beginning of epoch c. A value typical for Λ is 0.7 [78].

Reheating is performed in a similar fashion, except that the temperature is increased by some factor instead of being decreased. The geometric reheating schedule may be used for this purpose by adopting a fixed reheating factor β being applied to the current temperature during each epoch, with β being larger than 1. In accordance with the same notation as applied to the

cooling schedules (3.3)–(3.5), the temperature during epoch $c+1$ is expressed as

$$T_{c+1} = \beta T_c, \quad c = 0, 1, 2, \ldots \tag{3.6}$$

A linear reheating schedule, on the other hand, involves increasing the current temperature by adding a positive constant reheating factor h at the end of the epoch. The temperature during epoch $c+1$ is expressed as

$$T_{c+1} = T_c + h, \quad c = 0, 1, 2, \ldots \tag{3.7}$$

A popular multi-objective incarnation of SA

This section contains a discussion on the extensions required in respect of the method of single-objective SA of the previous section in order to render the method applicable to the more general case of multi-objective optimisation, according to an algorithm called DBMOSA. The particular set of extensions described here is that proposed by Smith *et al.* [148] in 2008. The main difference between single-objective SA and DBMOSA is the introduction of the notion of archiving, which allows for the retention of more than only a single incumbent solution, as is necessary in all MOPs.

During each iteration of the generalised SA search process, a single solution is still generated during each iteration, but a set of all the non-dominated solutions uncovered during the search, called the *archive* and denoted by $\mathcal{A}$, is additionally maintained externally as the algorithm progresses. Each solution accepted during the search is a candidate for inclusion in the archive. If a newly accepted solution dominates solutions already present in the archive, those solutions are removed from the archive and the newly generated solution is inserted into the archive. If however, a newly accepted solution is dominated by some solution in the archive, the archive remains unchanged. The archive is initialised as the singleton set containing the randomly generated initial solution of the SA search process. This archive maintenance process is illustrated graphically in Figure 3.1 for a bi-objective minimisation problem.

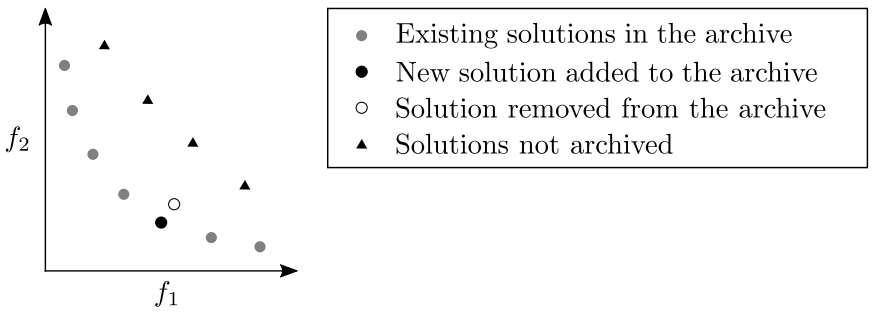


FIGURE 3.1: *The archiving process illustrated for a bi-objective minimisation problem with objective functions f_1 and f_2.*

In SOPs, the sign of the difference in energy $\Delta f(\boldsymbol{x}, \boldsymbol{x}')$ provides an indication as to whether a newly generated solution $\boldsymbol{x}'$ is superior to the current solution $\boldsymbol{x}$ during the search process [148]. If the true Pareto optimal set of solutions $\mathcal{P}$ were known *a priori*, the energy of a solution $\boldsymbol{x}$ could have been measured in the multi-objective optimisation case as that proportion of $\mathcal{P}$ which dominates $\boldsymbol{x}$. Let $\mathcal{P}_{\boldsymbol{x}}$ denote the subset of $\mathcal{P}$ that dominates $\boldsymbol{x}$. Then the energy of $\boldsymbol{x}$ may be expressed as

$$E(\boldsymbol{x}) = \mu(\mathcal{P}_{\boldsymbol{x}}),$$

where μ denotes a measure defined on $\mathcal{P}$ [148]. If $\mathcal{P}$ is a continuous set, μ may be taken as a Lebesgue measure (*i.e.* the length, area or volume associated with one, two or three objectives),

while if $\mathcal{P}$ is a discrete set, then $\mu(\mathcal{P}_{\boldsymbol{x}})$ may be taken as the cardinality of $\mathcal{P}_{\boldsymbol{x}}$ (*i.e.* the number of solutions in $\mathcal{P}$ that dominate $\boldsymbol{x}$).

Due to the fact that the true Pareto optimal set of solutions $\mathcal{P}$ is, however, unavailable during the optimisation process, Smith *et al.* [148] proposed use of an energy function defined in terms of the archive, which may be thought of as the current estimate of the Pareto optimal solutions. Define $\tilde{\mathcal{A}} = \mathcal{A} \cup \{\boldsymbol{x}\} \cup \{\boldsymbol{x}'\}$ and let $\tilde{\mathcal{A}}_{\boldsymbol{x}}$ be the subset of $\tilde{\mathcal{A}}$ that dominates $\boldsymbol{x}$. Then the energy difference between the current solution $\boldsymbol{x}$ and a newly generated neighbouring solution $\boldsymbol{x}'$ may be expressed as

$$\Delta E(\boldsymbol{x}, \boldsymbol{x}') = \frac{|\tilde{\mathcal{A}}_{\boldsymbol{x}'}| - |\tilde{\mathcal{A}}_{\boldsymbol{x}}|}{|\tilde{\mathcal{A}}|}. \tag{3.8}$$

Division by $|\tilde{\mathcal{A}}|$ in (3.8) ensures that $\Delta E(\boldsymbol{x}, \boldsymbol{x}')$ remains below unity and provides some flexibility in respect of the number of solutions in $\mathcal{A}$. Note that when $\tilde{\mathcal{A}}$ is a non-dominated set of solutions, the energy difference between any pair of its elements is zero. Inclusion of the current solution $\boldsymbol{x}$ in $\tilde{\mathcal{A}}$ ensures that $\Delta E(\boldsymbol{x}, \boldsymbol{x}') < 0$ if $\boldsymbol{x}'$ dominates $\boldsymbol{x}$, which means that newly generated solutions that move the archive closer towards the true Pareto optimal set are always accepted [148].

The nature of the energy measure in (3.8) is illustrated graphically in Figure 3.2 in the context of a bi-objective minimisation problem, where the solution $\boldsymbol{x}$ is dominated by three solutions in $\tilde{\mathcal{A}}$, and so $|\tilde{\mathcal{A}}_{\boldsymbol{x}}| = 3$, while the solution $\boldsymbol{x}'$ is dominated by a single solution in $\tilde{\mathcal{A}}$, and so $|\tilde{\mathcal{A}}_{\boldsymbol{x}'}| = 1$. This results in an energy measure of $\Delta E(\boldsymbol{x}, \boldsymbol{x}') = (1-3)/9 = -\frac{2}{9}$. Accepting $\boldsymbol{x}'$ as the new current solution during the following search iteration would therefore ensure that the search is progressing towards a more unexplored region of the non-dominated front (because $\boldsymbol{x}'$ is dominated by fewer solutions), and so $\boldsymbol{x}'$ should ideally be accepted as the new current solution. Indeed, a major benefit of the energy measure in (3.8) is that it encourages exploration of sparsely populated regions along the estimated Pareto optimal set, without taking the true Pareto optimal solutions explicitly into consideration.

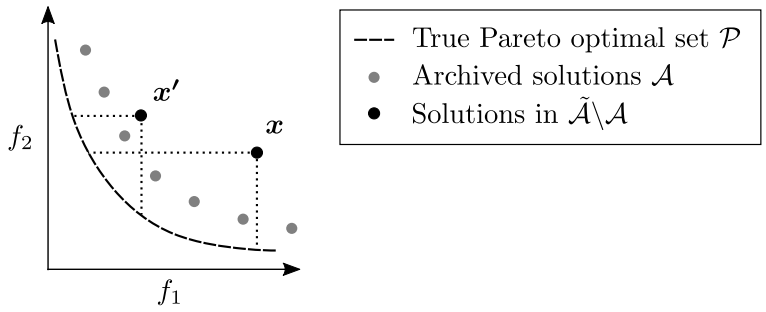


FIGURE 3.2: *The energy measures of a current solution $\boldsymbol{x}$ and a neighbouring solution $\boldsymbol{x}'$ for a bi-objective minimisation problem with objective functions f_1 and f_2. Here $|\tilde{\mathcal{A}}_{\boldsymbol{x}}| = 3$ and $|\tilde{\mathcal{A}}_{\boldsymbol{x}'}| = 1$.*

When the newly generated neighbouring solution does not move the search closer to the estimated Pareto optimal set, it is accepted stochastically as the new current solution during the following iteration according to a so-called *probability of acceptance* given by

$$P(\boldsymbol{x}') = \min\left\{1, \exp\left(-\frac{\Delta E(\boldsymbol{x}, \boldsymbol{x}')}{T}\right)\right\}, \tag{3.9}$$

where T again denotes the current temperature value [148]. Note that if $|\tilde{\mathcal{A}}_{\boldsymbol{x}'}| \leq |\tilde{\mathcal{A}}_{\boldsymbol{x}}|$, there will be a non-positive difference in energy and so $\boldsymbol{x}'$ will be accepted with probability 1 as the new current solution according to (3.9). If, however, $|\tilde{\mathcal{A}}_{\boldsymbol{x}'}| > |\tilde{\mathcal{A}}_{\boldsymbol{x}}|$, then the probability of acceptance depends on the current system temperature, with a large temperature resulting in a higher probability of acceptance of $\boldsymbol{x}'$ as the new current solution, while a small temperature results

in lower probability of acceptance. Note that the probability of acceptance does not depend on an *a priori* weighting of the objectives, and will thus remain unaffected when a rescaling of the objective functions occurs [148].

At termination of the SA search process, the archive is returned as an estimate of the true Pareto optimal set of solutions to the MOP in question. A pseudo-code description of the entire process described above is provided in Algorithm 3.2.

Algorithm 3.2: Dominance-based multi-objective SA [148]

Input : An MOP in the form (2.2)–(2.5), the maximum iterations per temperature $L_{\max}$, an initial starting temperature T_0, a minimum number of accepted moves per epoch $B_{\min}$, a maximum number of epochs $C_{\max}$ that may pass without accepting any new solutions.

Output: A set of non-dominated solutions $\mathcal{A}$ in the solution space.

Generate an initial feasible $\boldsymbol{x}$
Initialise the archive $\mathcal{A} \leftarrow \{\boldsymbol{x}\}$
Initialise the cooling schedule counter $c \leftarrow 0$
Initialise the number of epochs without accepting a solution $\epsilon \leftarrow 0$
while $\epsilon \leq C_{max}$ **do**
    Initialise the number of accepted moves per epoch $B \leftarrow 0$
    Initialise the number of iterations per epoch $t \leftarrow 1$
    while $t \leq L_{\max}$ *and* $B < B_{\min}$ **do**
        Generate a neighbouring solution $\boldsymbol{x}'$ of the current solution $\boldsymbol{x}$
        Generate a random number $p \in (0, 1)$
        if $p < \min\left\{1, \exp\left(-\frac{\Delta_E(\boldsymbol{x}', \boldsymbol{x})}{T_c}\right)\right\}$ **then**
            $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
            if $|\mathcal{A}_{\boldsymbol{x}}| = 0$ **then**
                $\mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{x}\}$
                for $\boldsymbol{y} \in \mathcal{A}$ **do**
                    if $\boldsymbol{x} \prec \boldsymbol{y}$ **then**
                        $\mathcal{A} \leftarrow \mathcal{A} \backslash \{\boldsymbol{y}\}$
                        $B \leftarrow B + 1$
        $t \leftarrow t + 1$
    $c \leftarrow c + 1$
    Update system temperature T_c
    if $B = 0$ **then**
        $\epsilon \leftarrow \epsilon + 1$
return $[\mathcal{A}]$

3.3.3 The genetic algorithm

A GA is an optimisation technique based on the principles of evolution by natural selection, as found in nature [38]. In order to construct such an artificial search algorithm which is robust and requires minimal problem information, ideas from the fundamentals of genetics are borrowed. The notion of a GA was originally proposed by Holland [77] in 1975, and has since attracted much attention. A GA takes an initial population of candidate solutions, called *chromosomes*,

and allows this population to evolve over a number of iterations within a carefully controlled environment, with the aim of uncovering approximately optimal solutions to an optimisation problem instance. This section is devoted to a discussion on the working of a GA, first in the context of single-objective optimisation, its original area of conception, and then in the more general context of multi-objective optimisation as it was later adapted.

Single-objective incarnation

Within the context of an SOP, a GA maintains a population $\boldsymbol{P}$ of candidate solutions throughout the search, and the solution with the best objective function value encountered during the search, denoted by $\hat{\boldsymbol{x}}$, is returned as an approximately optimal solution upon termination of the algorithm. A GA is an iterative search method, whereby parent solutions are selected to create offspring during each iteration. The parent solutions are chosen based on pre-determined criteria, expressed in the form of a *fitness measure*. The offspring are then incorporated, alongside the parent solutions, and a subset of this combined set forms the population during the next generation. A candidate solution's fitness is a measure of solution quality relative to those of the other solutions, based on the objective function. The objective function value of the solution itself is commonly applied as the fitness measure employed for GAs, in the case of maximisation problems.

Offspring solutions are created from the current generation's population of solutions by applying a *selection* operator to the current population, in order to choose the parent solution pairs. A *recombination* operator is then applied to the parent solution pairs, and the newly formed solutions are the offspring. This process of reproduction is repeated iteratively until either a pre-specified number of generations is reached, or until significantly fitter solutions are no longer being produced. Algorithm 3.3 contains a pseudo-code description of a GA, and the remainder of this section is devoted to a detailed explanation on the working of the algorithm.

Algorithm 3.3: Genetic algorithm

Input : An optimisation problem, a domain set for each decision variable, a population size N, a crossover probability p_c, a mutation probability p_m, the maximum number of iterations $t_{\max}$ and a tournament size n_{tour}.
Output: The best solution $\hat{\boldsymbol{x}}$ encountered during the search.

```
Generate an initial population P_0 comprising N random solutions
Determine the fitness of each solution in P_0
t ← 0
while t < t_max do
    Apply tournament selection to P_t, returning parent solutions based on their fitness values for crossover
    Apply a crossover operator to the parent solutions with probability p_c
    Apply a mutation operator to the offspring solutions with probability p_m
    Determine the fitness of each solution in P_{t+1}
    t ← t + 1
return [the best solution x̂ encountered during the search]
```

As input, the GA requires an SOP instance, a specified domain set for the decision variables, the size of the population, denoted by N, a maximum allowable number of iterations $t_{\max}$, a tournament size n_{tour} for use in the parent selection procedure, and finally crossover and mutation probabilities, denoted by p_c and p_m respectively. The algorithm is initialised by

randomly populating an initial candidate solution population $\boldsymbol{P}_0$ with N solutions. Thereafter, the fitness of each solution in the population is determined in terms of the SOP objective function.

During the next step, offspring are generated from the parent subpopulation by applying the selection and the recombination operators [135]. The two most common recombination operators are called *crossover* and *mutation*.

Crossover most often requires that two parent solutions be selected. The *tournament selection* procedure is a popular strategy for selecting parent solution pairs [47]. Tournament selection requires that small subsets of solutions be chosen stochastically, known as *tours*. The number of solutions contained within each subset is called the *tour size*. From each tour, the fittest solution is chosen deterministically to be a parent, and all the parents selected are placed within a *mating pool* of solutions, with the number of parents selected being called the *pool size*. The criterion for selecting a parent solution pair for recombination is typically based on their fitness function values, with better fitness values leading to higher probabilities of selection. *Roulette wheel selection* is a popular and simple procedure for achieving this [7]. A selection probability is assigned to each solution i within the mating pool based on its fitness function value f_i. The selection probability for solution i is taken as

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}. \tag{3.10}$$

In essence, this procedure normalises the fitness function values of the solutions in the mating pool. Next, a fictitious wheel is partitioned into arcs, each arc representing a solution in the mating pool, similar to roulette wheels found in casinos, where the spanning angles of the arcs are proportional to the selection probabilities determined according to (3.10). A reference point is fixed on the wheel, and then the wheel is spun. After the wheel has stopped, the arc coinciding with the fixed point is chosen as the first crossover parent. The solution is removed from the mating pool, the selection probabilities are recalculated and the procedure is repeated in order to select the second parent for crossover.

A crossover operator is applied to the pair of parents selected to produce offspring solutions. Crossover is a stochastic procedure, having a crossover probability p_c associated with its occurrence. In order to promote diversity, a large value for $p_c \in (0, 1)$ is usually chosen [74, 104, 135]. Various techniques for affecting crossover have been proposed in the literature, such as uniform crossover, cut-and-splice crossover, single-point crossover and two-point crossover, to name but a few [74, 103, 140]. It is the encoding of a solution that typically determines the method of crossover to be employed. The notion of single-point crossover is illustrated in Figure 3.3 in the context of binary encoded solutions. A *single* point is uniformly chosen across the two parent solution encodings and the parents are sliced at that point, yielding two segments for each parent. The parent segments beyond the crossover point are swapped around and recombined with the first segments of the other parents, thus forming the two new offspring solutions.

Two-point crossover works in a similar fashion, except that *two* crossover points are uniformly chosen, and the segments between the two crossover points are exchanged with one another, thereby forming the offspring. Uniform crossover differs from the aforementioned crossover operators in the sense that the solution encodings are not segmented at specific points. Instead, each element is considered individually to be interchanged with the other parent's counterpart during the formation of the offspring solutions, based on a pre-specified probability p_e, usually chosen as 0.5 [140].

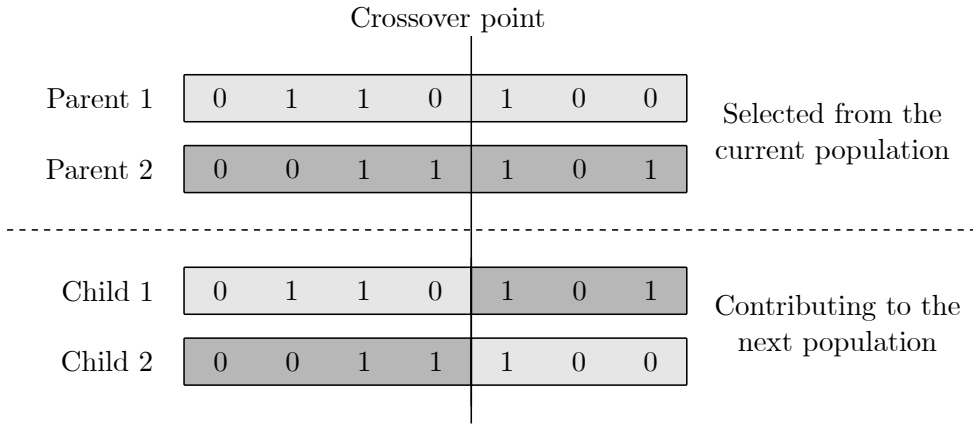


FIGURE 3.3: *A graphical depiction of single-point crossover applied to binary encoded parent solutions, Parents 1 and 2, at the crossover point, delivering Children 1 and 2 as offspring.*

The k-point and uniform crossover procedures described above are not particularly well-suited for problems in which the order or permutations of solution components are important, such as, for example, in the famous travelling salesman problem or in the vehicle routing problem [161]. Applying the aforementioned procedures often leads to infeasible solutions for these types of problems. This can be overcome by introducing problem-specific repair mechanisms for re-establishing the feasibility of permutation encodings violating constraints as a result of crossover. These repair mechanisms, in conjunction with the k-point and uniform crossover methods, will result in always creating feasible candidate solutions [140].

An alternative for ensuring feasible solutions are recombination methods designed specifically for permutation-specific problems. Such examples are uniform order-based crossover, order-based crossover, partially matched crossover and cycle crossover [140]. Uniform order-based crossover involves the use of a randomly generated binary template, in which a 0 indicates that the corresponding element positions are allowed to interchange, and a 1 indicates that the corresponding element positions are to remain fixed, as illustrated in Figure 3.4. In this illustration, each solution should contain all the elements from A–G in the alphabet, and the elements that should remain fixed have their blocks shaded grey. The remaining elements are free to be displaced, and are reordered based on the order in which they appear in the other parent. In this case, A, D, E and G from Parent 1 should be rearranged. Since they appear in the order E, D, G and A in Parent 2, they are ordered accordingly, filling the gaps, to produce Child 1. The recombination is similar for Child 2 [140].

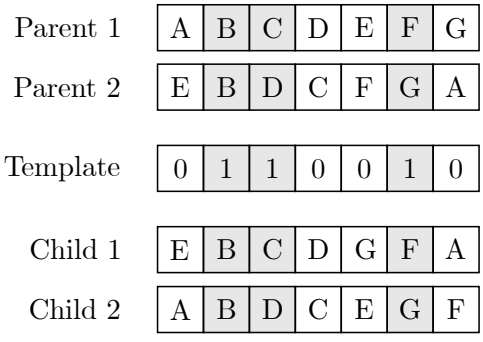


FIGURE 3.4: *A graphical depiction of uniform order-based crossover applied to alphabetically encoded parent solutions, Parents 1 and 2, in the context of a given binary template, delivering Children 1 and 2 as offspring. The shaded blocks indicate the elements that remain fixed due to the template.*

Order-based crossover [37], a variation on uniform order-based crossover, involves selecting two crossover positions. The elements between the crossover positions are copied to the children, and the remaining elements are sorted based on the order in which they appear in the other parent. This ordered list of elements is then used to populate vacant positions, but starting from the *second* crossover point, and then wrapping the remaining elements around, continuing from the start of the chromosome [140].

Partially matched crossover [64] preserves the ordering of some of the elements within the chromosome. Two random crossover positions are again generated, but the elements between these crossover sites are matched with the elements in the same location of the other parent. Within each parent, the elements between the crossover positions are then swapped with that same matched number in the parent's chromosome itself [140].

Cycle crossover [129] is based on the idea of establishing a cycle within a parent and copying this to a child. A first element is selected, and copied into the same position of the first child. The element in the same position, but in the other parent is then taken next, and copied into the child in the original position as found in the first parent. This copied element is then taken as the next starting point, and similarly, the solution in the same position of the other parent is copied into the same child in the position corresponding to the original position in the first parent. This process is repeated until the solution in the corresponding position of the other parent is the same as that of the initial solution. A cycle is thus established, and all the empty locations of the child are finally filled by mapping the elements of the other parent to the child's empty locations. The other child is created by populating it with the unutilised element of either of the parents in its corresponding position [140].

Offspring generation is only the first part of the recombination phase; the second involves application of a mutation operator. Mutation is employed for further diversification of the solutions, aiming to avoid poor local optima and premature convergence of the algorithm by altering an entry in a chromosome. This alteration occurs stochastically with a *mutation probability* of p_m, and, in contrast with the crossover probability, is usually chosen as a small value, due to large values potentially reducing the algorithm to a random search [104]. This operator helps the algorithm to explore a larger area of the search space, overcoming situations where potentially some solution elements become unalterable due to parents having the same elements in the same positions [140]. The literature contains a number of different mutation operators, such as uniform mutation, bit flip mutation and Gaussian mutation, to name but a few [77, 104, 140]. Figure 3.5 illustrates the notion of bit flip mutation, according to which one binary bit is complemented from 1 to 0, or *vice versa*. This method pertains to binary encoded solutions.

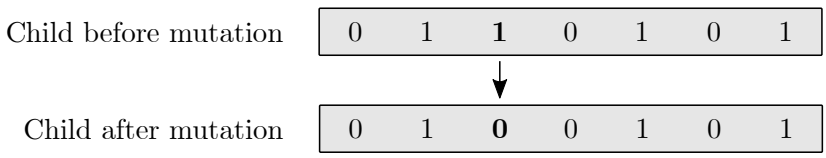


FIGURE 3.5: *A graphical depiction of bit flip mutation applied to a binary encoded offspring solution.*

Uniform mutation involves choosing an element of a solution randomly and replacing it with a random value according to a uniform distribution specified beforehand. This method is only applicable to optimisation problems in which solutions are integer or real value-encoded. Gaussian mutation, on the other hand, involves choosing a random element from a solution, and adding a random value that is Gaussian-distributed to the selected element. If the sum falls beyond the limits specified by the user, the value of the upper or lower bound encroached upon is taken as the new value instead.

After having computed the offspring population and its fitness values, a selection from the combined previous population and the newly created offspring population replaces the previous population and the process is repeated until a stopping criterion is met. Common stopping criteria are reaching a pre-specified maximum number of iterations, failing to improve significantly on the quality of the best solution encountered during the search, or achieving a satisfactory level of fitness [74, 104, 135]. Finally, at termination of the algorithm, the solution $\hat{\boldsymbol{x}}$ is returned that has achieved the best fitness during the entire search, called the *incumbent* solution.

A popular multi-objective incarnation

The general working of the GA was originally conceived within the context of single-objective optimisation, as described above. With certain extensions, it can, however, be expanded to within a multi-objective context. The well-known NSGA II is one such example, having been proposed by Deb *et al.* [42] in 2002. This multi-objective incarnation of a GA searches the solution space of an MOP of the form (2.2)–(2.5) to find approximately optimal solutions to the specific problem instance. Although sharing many similarities with a standard GA, the main difference lies in the way that the NSGA II assigns fitness values to solutions, thereby affecting the selection process of the parents destined for crossover. This section is devoted to a description of the working of the NSGA II.

The NSGA II takes as input an instance of the MOP (2.2)–(2.5), a maximum number of search generations $G_{\max}$ as the stopping criterion, and the number of solutions to be maintained per generation throughout the search, called the population size and denoted here by N. The first step involves randomly generating an initial population $\boldsymbol{P}_0$ containing N candidate solutions. Using the FNSA [152], described in §2.5, the solutions of $\boldsymbol{P}_0$ are ranked, yielding segmented non-dominated fronts. Next, a *crowding distance density measure* is computed for each solution in each non-dominated front [42]. This measure quantifies solution density within a front, in the sense that a smaller value indicates that a solution achieves closer proximity with other solutions, *i.e.* is more crowded by other solutions, while a larger value indicates that other solutions are farther removed from it, *i.e.* the solution is more isolated. A pseudo-code description of the crowding distance assignment algorithm is provided in Algorithm 3.4.

Algorithm 3.4: Crowding distance assignment algorithm [42].

Input : A non-dominated set $\boldsymbol{C}$ of solutions to an instance of the MOP (2.2)–(2.5), containing the objective function values for each solution in a vector denoted by $\boldsymbol{z}$.

Output: The crowding distance $\boldsymbol{C}[i]_{\text{dist}}$ for each solution in $\boldsymbol{C}$.

$\ell \leftarrow |\boldsymbol{C}|$
forall $i \in \boldsymbol{C}$ **do**
  $\boldsymbol{C}[i]_{\text{dist}} \leftarrow 0$
forall $m = \{1, \ldots, M\}$ *objectives* **do**
  $\boldsymbol{C} \leftarrow \texttt{sort}(\boldsymbol{C}, m)$
  $\boldsymbol{C}[1]_{\text{dist}}|m \leftarrow \infty$
  $\boldsymbol{C}[\ell]_{\text{dist}}|m \leftarrow \infty$
  forall $i = 2$ *to* $(\ell - 1)$ **do**
    $\boldsymbol{C}[i]_{\text{dist}}|m \leftarrow \boldsymbol{C}[i]_{\text{dist}}|m + (\boldsymbol{C}[i+1]|m - \boldsymbol{C}[i-1]|m)/(\boldsymbol{z}_m^{\max} - \boldsymbol{z}_m^{\min})$
return $[\boldsymbol{C}[i]_{\text{dist}}$ for each solution in $\boldsymbol{C}]$

The crowding distance assignment is applied to each non-dominated front, denoted by $\boldsymbol{C}$, successively, taking as input the solutions of the non-dominated front along with a vector $\boldsymbol{z}$ containing all the objective function values for each of the M objectives for solutions in the front. Let ℓ denote the total number of solutions present in the non-dominated front $\boldsymbol{C}$. The crowding distance of each solution i, denoted by $\boldsymbol{C}[i]_{\text{dist}}$, is initially set to zero. Thereafter, the solutions of $\boldsymbol{C}$ are sorted in ascending order according to each of the M objectives by applying the `sort` function in Algorithm 3.4. Denote the m-th objective function value of the i-th solution in the sorted list by $\boldsymbol{C}[i]|m$. Furthermore, denote the crowding distance of the i-th solution of the m-th objective function by $i_{\text{dist}}|m$. For the boundary solutions ($i = 1$ and $i = \ell$) the crowding distances for objective m are set to an infinite distance. For the intermediary solutions (between 1 and ℓ), the crowding distance is assigned by calculating the normalised distance between the solutions above $(i+1)$ and below $(i-1)$ it in rank. This is calculated as $(\boldsymbol{C}[i+1]|m - \boldsymbol{C}[i-1]|m)/(\boldsymbol{z}_m^{\max} - \boldsymbol{z}_m^{\min})$, where $\boldsymbol{z}_m^{\max}$ and $\boldsymbol{z}_m^{\min}$ denote respectively the maximum and minimum attained values of the m-th objective function. The sum of the crowding distances for every objective is computed for each solution i, and returned as the overall crowding distance $\boldsymbol{C}[i]_{\text{dist}}$ for solution i. A graphical illustration of the crowding distance measure is provided in Figure 3.6 for a bi-objective minimisation problem, where the solid points represent the non-dominated front for which crowding distances are calculated.

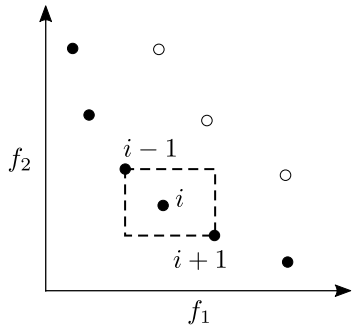


FIGURE 3.6: *A graphical illustration of the crowding distance calculation for solution i, indicating the cuboid formed around the solution within objective space, where both objective functions f_1 and f_2 should be minimised and the solid points represent solutions from the same non-dominated front [42].*

Two selection criteria are employed in the NSGA II during the selection procedure. The first criterion is the rank assigned to each solution, based on the non-dominated front in which it resides, whereby solution superiority is determined. A solution i is considered superior to a solution j if it achieves a lower rank value, *i.e.* $i_{\text{rank}} < j_{\text{rank}}$. If, however, a pair of solutions have the same rank value, the second criterion employed to break the tie of equal rank is that the solution achieving the largest crowding distance is considered superior, *i.e.* $i_{\text{rank}} = j_{\text{rank}}$ and $i_{\text{dist}}|m > j_{\text{dist}}|m$. Choosing the largest crowding distance leads to an exploration of regions within the objective space that are less crowded, therefore yielding better quality in terms of the spread and uniformity of the Pareto front approximation. Combining these two criteria, the *crowded comparison operator*, denoted by $\prec_c$, is defined as $i \prec_c j$ if $i_{\text{rank}} < j_{\text{rank}}$, or if $i_{\text{rank}} = j_{\text{rank}}$ and $i_{\text{dist}}|m > j_{\text{dist}}|m$.

Next, the initial population $\boldsymbol{P}_0$ is utilised to form the initial offspring population $\boldsymbol{Q}_0$. This is done by employing binary tournament selection, crossover and mutation (as described above) based on the crowded comparison operator $\prec_c$ as fitness measure, as opposed to only considering the objective function value as in the single-objective incarnation of the GA. The parent and offspring populations are combined to form the set $\boldsymbol{R}_0 = \{\boldsymbol{P}_0 \cup \boldsymbol{Q}_0\}$ of size $2N$. This combined set $\boldsymbol{R}_0$ is sorted and ranked according to the FNSA, yielding the various segmented non-dominated fronts. The next generation, $\boldsymbol{P}_1$, is populated with N solutions from $\boldsymbol{R}_0$, sorting the solutions based on

the crowded comparison operator. Whereas the above procedure was described for the initial generation $t = 0$, the entire process is repeated for $t = 1, 2, 3, \ldots$ until a termination condition is met, as illustrated in Figure 3.7, where the creation of $\boldsymbol{P}_{t+1}$ is described in terms of the relevant steps involved.

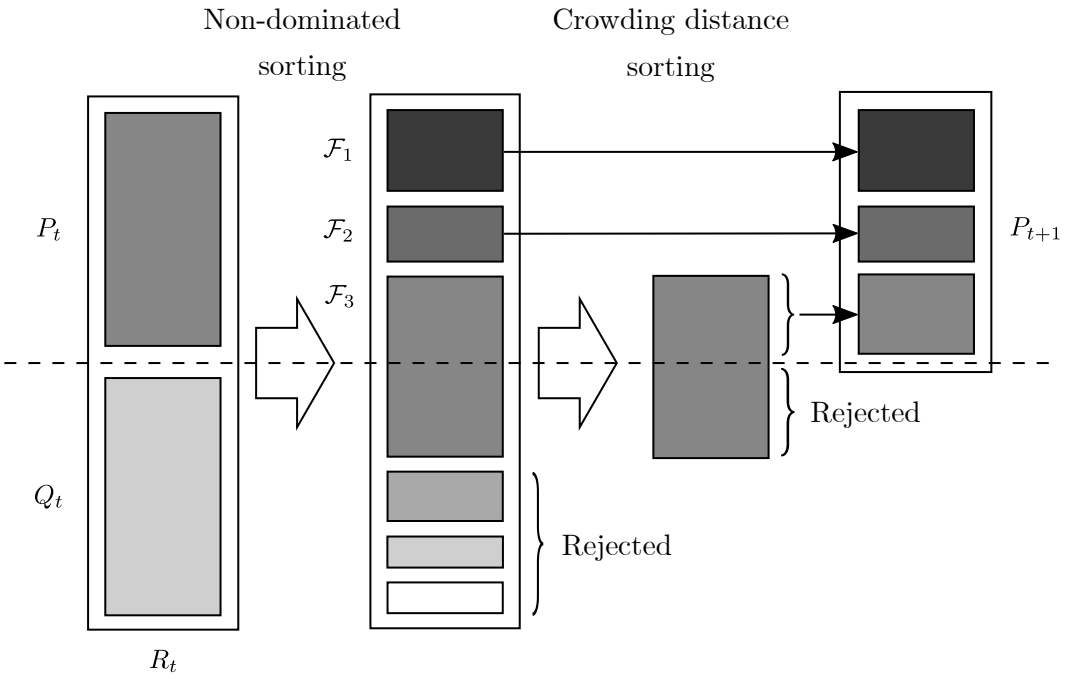


FIGURE 3.7: *A graphical representation of the formation of population $\boldsymbol{P}_{t+1}$ in the NSGA II, indicating the partitioning of the parent and offspring populations $\boldsymbol{P}_t$ and $\boldsymbol{Q}_t$. Thereafter, their combination $\boldsymbol{R}_t$ is sorted according to the FNSA, yielding the non-dominated fronts $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \ldots$ The next generation $\boldsymbol{P}_{t+1}$ is formed by first including the non-dominated fronts whose total size is less than or equal to the population size (the fronts $\mathcal{F}_1$ and $\mathcal{F}_2$ in this case). The remaining members of $\boldsymbol{P}_{t+1}$ (if any) are chosen by applying crowding distance sorting to the front following the highest indexed front included in $\boldsymbol{P}_{t+1}$ (the front $\mathcal{F}_3$ in this case) [42].*

As the non-domination rank of each solution is the weightier matter, solutions exhibiting lower ranks are prioritised for inclusion in the next generation. This is elegantly achieved by applying the crowded comparison operator. If the number of solutions of rank 1 in the first front $\mathcal{F}_1$ is less than or equal to N, the entire front $\mathcal{F}_1$ is included in $\boldsymbol{P}_{t+1}$, as illustrated in Figure 3.7. The remaining members of $\boldsymbol{P}_{t+1}$ (if any) are chosen based on the order of increasing rank, starting with 2. If the number of solutions in the second non-dominated front $\mathcal{F}_2$ is less than or equal to N minus the number of solutions currently occupying $\boldsymbol{P}_{t+1}$, all of the solutions of $\mathcal{F}_2$ are added to $\boldsymbol{P}_{t+1}$. This process is repeated. If, however, the number of solutions in the evaluated non-dominated front is more than the remaining vacant slots in $\boldsymbol{P}_{t+1}$, the particular non-dominated front is sorted and ranked based on the crowded comparison operator, in descending order. The remaining vacancies in $\boldsymbol{P}_{t+1}$ are filled with the highest ranked solutions until $\boldsymbol{P}_{t+1}$ contains exactly N solutions. The remainder of the solutions of $\boldsymbol{R}_t$ not included are simply rejected (discarded), as they are not deemed of high enough quality to be included in the next generation, alluding to the survival of the fittest concept endorsed by genetic algorithms. This procedure can be seen in Figure 3.7, where only the top performing solutions of $\mathcal{F}_3$ are included in $\boldsymbol{P}_{t+1}$, while the remaining solutions are rejected. It should be noted that while $\boldsymbol{P}_{t+1}$ is being formed, the crowding distance of each solution in the non-dominated front is calculated, and so it is important to ensure that unnecessary computations are not carried out in respect of the rejected solutions, whose ranks were not considered sufficient for inclusion in the next generation [42].

The NSGA II iteratively creates new generations $\boldsymbol{P}_{t+1}$ according to the procedure outlined above, until the stopping criterion of having performed $G_{\max}$ generations is reached, and the algorithm terminates. The non-dominated set of the last population $\boldsymbol{P}_{G_{\max}}$ is determined according to any of the methods outlined in §2.4, denoted by the function `get_non-dominated_set` in Algorithm 3.5, and is returned as output, collectively representing an approximation $\mathcal{P}^*$ of the Pareto front of the problem instance under consideration. The computational complexity of the NSGA II is $\mathcal{O}(MN^2)$ where M denotes the number of objectives of the optimisation problem instance and N denotes the population size. This complexity is dominated by the FNSA [42].

Algorithm 3.5: Non-dominated sorting genetic algorithm II [42].

Input : An MOP instance of the form (2.2)–(2.5), a maximum number of generations $G_{\max}$, and a population size N.
Output: An approximate Pareto optimal solution set $\mathcal{P}^*$ to the instance of (2.2)–(2.5).

```
Generate an initial population P_0 comprising N random solutions
Use the FNSA [Algorithm 2.7] to rank and sort P_0
Use the crowding distance assignment algorithm [Algorithm 3.4] to determine the
crowding distance of each solution in P_0
Generate a child population Q_0 of size N from P_0, by means of crossover and mutation,
using binary tournament selection, based on the ≺_c crowding distance operator
t ← 0
while t < G_max do
    R_t ← P_t ∪ Q_t
    Rank and sort R_t using the FNSA into non-dominated fronts F_1, F_2, ..., F_k
    P_{t+1} ← ∅
    k ← 1
    while |P_{t+1}| < N do
        if |F_k| + |P_{t+1}| ≤ N then
            P_{t+1} ← P_{t+1} ∪ F_k
        else
            Determine the crowding distance of each solution in F_k
            Sort the solutions in F_k in descending order of crowding distance
            P_{t+1} ← P_{t+1} ∪ [top (N − |P_{t+1}|) solutions in the sorted set F_k]
        k ← k + 1
    Determine the crowding distance of each solution in P_{t+1}
    Generate the child population Q_{t+1} from P_{t+1} through crossover and mutation, using
    binary tournament selection based on the ≺_c crowding distance operator
    t ← t + 1
return [P* = get_non-dominated_set(P_{G_max})]
```

3.4 Hyperheuristic optimisation approaches

The popularity of hyperheuristics has emerged relatively recently. According to the seminal paper of Burke *et al.* [24], the automation of heuristic-design was only officially coined as hyperheuristics in 2000, yet the origins of these methods can be traced back to the 1960s. In the context of combinatorial optimisation, a hyperheuristic may be thought of as a heuristic for choosing heuristics [24]. Hyperheuristics, therefore, focus on finding good optimisation methods,

instead of finding good solutions, in the hope that better optimisation methods will also lead to better solutions. The aim is to use limited problem-specific information and allow the algorithm to determine good search strategies automatically for the specific optimisation problem at hand. The main difference between metaheuristics and hyperheuristics is that the former methods operate on the level of the solution search space, whereas the latter methods operate on the level of the heuristic space which, in turn, operates on the solution search space [24]. A set of *low-level* heuristics is therefore managed by a hyperheuristic search space structure, and yields the advantage of potentially achieving greater flexibility and search effectiveness.

One of the main driving forces behind the design of hyperheuristics is the quest to solve computationally hard optimisation problems according to a more generalised and flexible approach, by employing automated heuristic-design. This is especially useful when attempting to solve a wide variety of optimisation problems without having to change the optimisation approach significantly from one problem to the next. Some optimisation problems are, for instance, solved more effectively by adopting a population-based search approach, and others by adopting a trajectory-based search approach. When faced with a new optimisation problem, the most suitable form of solution approach is usually not known *a priori*, and determining the best approach may be rather cumbersome. The difficulty in identifying which solution approach is best suited to certain hard optimisation problems is due to the large number of algorithmic alternatives available and the often vast parameter ranges of these algorithms, along with a general lack of literature providing guidance as how to choose from these options effectively [24].

3.4.1 Hyperheuristic classification

More formally stated, "a hyperheuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems" [25]. This definition highlights two main classes of hyperheuristics, namely *selective* hyperheuristics and *generative* hyperheuristics. Selective hyperheuristics are based on the notion of selecting heuristics to apply and the methodologies that surround the selection procedure. Generative hyperheuristics, however, focus on constructing new heuristics based on components of other heuristics so as to form new search features. Each class admits another level of sub-classification, namely *perturbative* methods and *constructive* methods. The main difference between the two is the solution type on which they operate, with perturbative methods altering one or more components of complete or feasible candidate solutions, and constructive methods altering partial or infeasible candidate solutions until they are complete or feasible.

It may further be deduced from the definition above that hyperheuristics are not necessarily search methods, but rather learning mechanisms that utilise feedback emanating from a search process. This learning can follow one of two approaches, namely *online* learning or *offline* learning. In online learning, the process of gathering feedback and adapting parameters is performed during the progression of the search, whereas offline learning takes place after the search has been completed, and the knowledge obtained from various problem instances is incorporated into the algorithmic parameters. As only selective hyperheuristics are considered in this dissertation, the discussion in this section continues only in respect of this class of hyperheuristics, with a particular focus on selective hyperheuristics of a perturbative nature.

Selective-constructive hyperheuristics take an empty or partial candidate solution as input and build upon it in a systematic and progressive manner by applying some heuristic until a complete solution is obtained, upon which they terminate. A pre-existing set of problem-specific low-level heuristics is typically available to the hyperheuristic, and the goal is to construct a complete solution by utilising a single heuristic or an appropriate sequence of heuristics. The

heuristic sequence typically has a finite length which is determined by the complexity of the underlying combinatorial optimisation problem. Applications of selective-constructive heuristics in the literature include production and workforce scheduling, educational timetabling, constraint satisfaction, strip packing, and vehicle routing [24].

In the other sub-class of hyperheuristics, selective-perturbative hyperheuristics, the goal is to improve a complete candidate solution by applying heuristics that perturb the solution in some manner. In this case, the selection of which heuristic to employ is the core of the hyperheuristic. Applications of this class of hyperheuristics in the literature include educational timetabling, personnel and sports scheduling, cutting and packing problems, space allocation and vehicle routing [24].

Selective-perturbative hyperheuristics may be applied to trajectory-based searches and to population-based searches. Trajectory-based searches, also referred to as *single-point* searches, consider one solution at a time, as described in §3.3. This sequence of solutions undergoes consecutive perturbations until termination of the search. Population-based searches, on the other hand, also referred to as *multi-point* searches, consider multiple candidate solutions simultaneously at each search step, and all of these solutions collectively undergo some perturbative change as the search progresses. Of the two, trajectory-based searches are the most popular [24]. The same selective procedure, however, underpins both approaches — selecting a single heuristic, or subset of heuristics, to be applied to the candidate solution(s) from a set of lower-level perturbative heuristics. After application of the perturbation procedure, it is decided whether or not to retain the solution(s) during continuation of the search. Selective-perturbative hyperheuristics therefore comprise two components, the first being application of a *heuristic selection method*, which is followed by application of a *move acceptance method*. For each of these components numerous strategies have been proposed in the literature [15, 24, 130].

When a learning mechanism is not employed during the selection of heuristic methods, the selection strategy is left to a random process or exhaustive process. If, however, learning is employed, online learning is adopted in the majority of the approaches, where each heuristic is accompanied by an *online score*. As part of the selection strategy, this score is processed by a recurring method or update rule. The components of such a score-based hyperheuristic framework include an initial score, a memory length adjustment, a score-based heuristic selection strategy, and a score-updating rule for both solution improvement and solution degradation [123].

According to this framework, an initial score is assigned to each available perturbative heuristic, and is typically the same for all the heuristics. The influence of the previous score performances on the heuristic selection strategy can be adjusted by the memory length. It would make sense, in the majority of cases, to allow the most recent performance scores of heuristics to carry the most weight, and those encountered earlier on during the search to carry less weight. Two common strategies include the *max strategy* and the *roulette-wheel* strategy [24]. The max strategy is based on the notion that the best-performing heuristic is selected, whereas the roulette-wheel strategy assigns probabilities to each heuristic based on its relative performance, after which a stochastic selection is performed based on these probabilities.

When considering move acceptance, a deterministic approach or a non-deterministic approach may be employed. According to a deterministic approach, the same selection will be performed every time, given the same state of the search and various scores. A non-deterministic approach, on the other hand, would lead to differing selections, even though the search state and scores might be exactly the same. Non-deterministic approaches typically require additional parameters to help mitigate the added complexity.

3.4.2 The AMALGAM method

In 2007, Vrugt and Robinson proposed a selective-perturbative hyperheuristic called AMALGAM with foundations in evolutionary optimisation, which has since proved to be very successful [167]. Vrugt and Robinson noticed that when multiple conflicting objectives are considered in search and optimisation problems, evolutionary algorithms are capable of achieving superior performance. The ability to traverse intractably large search spaces, the maintenance of a diverse set of solutions, and the incorporation of characteristics of good solutions into new solutions, make evolutionary algorithms formidable optimisation frameworks [167].

When population-based methods are adopted to solve MOPs, a *single algorithm* is typically utilised in search of high-quality solutions. Based on the no free lunch theorem, proved by Wolpert and Macready [172], the development of any single algorithm outperforming all others is impossible in the context of a diverse set of optimisation problems. This quandary is addressed by the continual design of new algorithms that may overcome the obstacles associated with no single best algorithm. These algorithms aim to increase the general applicability of solution approaches to a more diverse set of optimisation problems, while still obtaining high-quality solutions to the respective problems. In this case, the descriptor *high-quality* implies that the performance of the best lower-level heuristic or sub-algorithm utilised individually, should at least be obtained by the overarching hyperheuristic algorithm. AMALGAM is one example of such an algorithm, and has been reported to achieve solution quality improvements of orders of magnitude for complex problems in high dimensions, over and above the individual implementations of the sub-algorithms involved [167].

The research hypothesis adopted by Vrugt and Robinson [167] was that an adaptive and dynamic search procedure, which incorporates the underlying effects of the shape and peculiarities of the fitness landscape associated with the search space during the offspring-generation procedure, would lead to an evolutionary search procedure that is significantly more efficient. The basis for this hypothesis is that there are different underlying characteristics, attributable to the problem peculiarities, for each different optimisation problem. The introduction of a dynamic search mechanism may indeed hold significant benefit if it can provide a smarter approach towards searching in difficult fitness landscapes. The working of the AMALGAM method consists of two main parts, namely performing a *simultaneous multi-method search*, and applying a *self-adaptive offspring creation* procedure. These two components combine to deliver an efficient and reliable approach towards solving MOPs. The notion is to exploit the individual sub-algorithms' advantages globally, while simultaneously compensating for their respective weaknesses.

The self-adaptive offspring creation procedure works on the basis of allotting more computational resources to a sub-algorithm if that algorithm performs better relative to the other sub-algorithms based on the previous search stage's reproductive success. An example of this is the number of offspring assigned to be generated by each sub-algorithm for the next generation. Denote the contribution of solutions to the next population P_{t+1} produced by sub-algorithm $i \in \{1, \ldots, k\}$ by S^i_{t+1} during generation t. The number of solutions that should be produced by sub-algorithm i to contribute towards the formation of generation $t+1$ is

$$N^i_{t+1} = \frac{\left(\frac{S^i_{t+1}}{N^i_t}\right) N}{\sum_{i=1}^{k} \left(\frac{S^i_{t+1}}{N^i_t}\right)} \tag{3.11}$$

according to the AMALGAM method, where N represents the total population size. The ratio S^i_{t+1}/N^i_t represents the proportion of successful offspring solutions with respect to the total number of offspring solutions introduced by sub-algorithm i. The reproductive success of sub-algorithm i is therefore contained in this ratio, and it ensures that the computational resource

distribution is allotted according to each respective sub-algorithms' relative performance. In order to avoid problems associated with disproportional scaling, the denominator of (3.11) takes into account the sum of all the reproductive success ratios of all sub-algorithms, representing the combined success of all the sub-algorithms.

Vrugt and Robinson [167] pointed out that it is important to impose a constraint on the minimum value of N_t^i so as to ensure that no sub-algorithm i is deactivated completely during the execution of the hyperheuristic. Although no specific motivation was provided, Vrugt and Robinson set their minimum value for N_t^i to be 5% [167]. The concepts of non-domination rank and crowding distance, referred to in §3.3.3, are utilised in the AMALGAM method to facilitate ranking and sorting of solutions in the population.

3.5 Chapter summary

The purpose of this chapter was to provide the reader with insight in respect of a number of popular approaches towards solving combinatorial optimisation problems, starting with exact solution approaches in §3.1, which lead to optimal solutions, given that the problem dimensions under consideration are small enough. Next, the general notion of a heuristic was reviewed in §3.2 according to which *ad hoc* methods are applied to solve optimisation problems approximately. Solution approaches providing more general frameworks for solving optimisation problems of various forms were addressed in §3.3, a section devoted to the notion of metaheuristics. In this section, the two main classes of metaheuristics, namely trajectory-based metaheuristics and population-based metaheuristics, were elaborated upon, and examples were given of each in §3.3.1. One popular metaheuristic from each class was considered in more detail, both in its single- and multi-objective contexts, namely SA in §3.3.2 and the GA in §3.3.3. These two metaheuristics applied in their multi-objective optimisation contexts, namely DBMOSA and NSGA II, are incorporated into the solution approaches adopted for solving the various optimisation problem instances later in this dissertation. A short overview of hyperheuristics was finally provided in §3.4, where a distinction was made between selective and generative hyperheuristics, as well each class's further subdivision into either perturbative hyperheuristics or constructive hyperheuristics in §3.4.1. The chapter closed in §3.4.2 with a review of a hyperheuristic that is well-suited to the incorporation of evolutionary algorithms, called the AMALGAM method.

CHAPTER 4

Urban transit network design problem

Contents

The problem described in Chapter 1 is a special case of a wider, overarching class of problems encapsulated in the so-called UTNDP. The UTNDP is a complex combinatorial optimisation variation on the well-known vehicle routing problem [161] which entails finding a number of travel circuits within a given transport network connecting origin and destination pairs that results in all network vertices being served without knowing the demand levels at these vertices *a priori* [93, 115].

Passenger transport time is a key factor that has to be addressed when designing public transport systems. A reduction in this time is, of course, beneficial for a commuter using such a system. One way of improving passenger transport time involves employing better technology, such as faster transport vehicles, while another is the improvement of the transit route network. With the introduction of more transport vehicles, more routes can be served, but at a higher operating cost. In contrast, if the operating cost is fixed, the optimisation of the transit route network can improve passenger transport time without incurring additional costs [115]. Hence there is an inherent trade-off between minimising passenger transport time and minimising system operating cost in the design of any public transport system.

The nature of this trade-off may be observed by noting that if the intention is to design a network of transport vehicle routes that satisfies public transport demand in extended urban areas by the introduction of long vehicle routes visiting many possible passenger origin and destination locations, the time intervals between the arrivals of consecutive vehicles along any route is expected to increase (under the assumption that the number of vehicles utilised remains constant), thereby increasing passenger waiting time. If, on the other hand, the number of vehicle routes and their lengths are reduced in a bid to save on costs, the number of passenger

transfers (from one vehicle route to another), and therefore the expected passenger transport time, is expected to increase, as is the fraction of expected unsatisfied demand [115].

The nature of the UTNDP is examined in general in §4.1–§4.4 according to the framework proposed by Guihaire and Hao [68]. Thereafter, a focused approach is adopted in §4.5 to examine the specific variation and special case of the UTNDP concerned with route layout design, as described in Chapter 1. This special case is called the *urban transit routing problem* (UTRP) [93]. Next, the special case of bus assignment to routes, also mentioned in Chapter 1, is examined in §4.6 in the context of the UTNDP, called the *urban transit frequency setting proplem* (UTFSP) [93]. Bus assignment and frequency setting are synonymous in the context of the UTFSP. When, however, the route layout design and bus assignment are attempted simultaneously, this specific case is called the *urban transit routing and frequency setting problem* (UTRFSP), and is addressed in §4.7. Attention is finally drawn to commercial software available for solving the aforementioned problems in §4.8, after which the content of this chapter is summarised briefly in §4.9.

4.1 Transit network design problems in general

Operations researchers have been successful in solving various optimisation problems in the public transit domain for several decades [43]. Numerous commercial software systems have been developed by employing operations research techniques to aid transit agencies in their complex planning and operational decisions. Public transit problems have, in fact, attracted the attention of many operations researchers due to the size and formidable complexity of real instances of these problems. For example, the New York City Transit Authority manages more than 12 000 drivers operating approximately 4 500 buses that serve more than 240 bus routes [43]. Solving public transit problems for such large transport systems is complex, because they involve passengers as well as buses and drivers, each with their own individual preferences and constraints, and these interact with each other according to a possibly large and complex set of fixed relationships.

In the UTNDP, the goal is to find a configuration of various transit routes and associated transport vehicle frequencies that minimises or maximises a pre-selected set of objective functions subject to a set of constraints [6]. Typical examples of these objective functions and constraints are discussed in some detail in this chapter.

The UTNDP requires design choices to be made from a set of alternatives, and can often be modelled in the form of integer or mixed-integer programming models [114]. At the core of these models is the requirement that certain routes have to be considered for inclusion in the final solution, along with the frequencies at which transport vehicles have to traverse these routes. Moreover, various trade-offs typically have to be considered that mainly affect the users and operators of the transport network. From a passenger's perspective, the transport network should meet the travel demand of passengers at low costs. Other criteria from a passenger perspective include good quality of service in terms of vehicle and terminal transfer comfort, consistency of service, service coverage, direct service and the frequency level achieved [68]. From the operator's perspective, on the other hand, the goal is usually to maximise profits emanating from the transport system. Finding a suitable trade-off between these objectives poses a significant challenge [68].

In the literature, however, mathematical models of the UTNDP have primarily been concerned with minimising an overall cost function, typically in the form of a combination of passenger and operator costs [6]. Passenger costs are often measured in terms of time while operator costs are

measured in terms of the number of transport vehicles that have to be operated [5]. According to Baaj and Mahmassani [5], the most popular objective function considered in UTNDP models takes the form

$$\text{minimise} \quad z = c_1 \sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij} t_{ij} + c_2 \sum_{k \in \mathcal{R}} f_k T_k, \tag{4.1}$$

which represents the sum of passenger cost (the first term) and operator cost (the second term). In (4.1), the coefficient c_1 denotes the relative importance weight of passenger cost, while c_2 denotes the relative importance weight of operator cost. The demand between vertices i and j in a transport network containing n vertices is denoted by d_{ij}, while the total expected travel time between these vertices is denoted by t_{ij} (which includes in-vehicle travel time, the waiting time incurred before travelling and the transfer time incurred when switching between routes). Furthermore, f_k denotes the frequency with which transport vehicles travel along route k in a route set $\mathcal{R}$, while T_k denotes the round-trip travel time associated with this route.

According to Stewart [156], however, the above scalarisation of an essentially bi-objective optimisation goal is not good practice for two reasons. First, it is difficult to be sure that suitable values are selected for the weights c_1 and c_2, and, secondly, the scalarisation process itself masks the true trade-off nature of the two terms present in the objective function, especially in the case of non-convex problems (*i.e.* where the feasible domain of the optimisation problem is not a convex set). The two terms in (4.1) should therefore rather be treated as two separate objective functions.

The optimisation process in models of the UTNDP usually takes place subject to three fundamental constraint sets, namely a frequency feasibility constraint set, a load factor constraint set and a fleet size constraint. The frequency feasibility constraint set may be formulated as

$$f_k \geq f_{\min}, \qquad k \in \mathcal{R},$$

where $f_{\min}$ denotes the minimum frequency of transport vehicles travelling along any route. The load factor constraint set may, in turn, be formulated as

$$L_k = \frac{(Q_k)_{\max}}{f_k P} \leq L_{\max}, \qquad k \in \mathcal{R},$$

where L_k denotes the load factor of route $k \in \mathcal{R}$, $(Q_k)_{\max}$ denotes the maximum passenger flow occurring along any link of route k, measured in people per minute, P denotes the seating capacity of the transport vehicles operating on the network's routes, and $L_{\max}$ denotes the maximum load factor for any route. The fleet size constraint may finally be formulated as

$$\sum_{k \in R} f_k T_k < W, \qquad k \in \mathcal{R},$$

where W is the transport vehicle fleet size available.

4.2 The complexity of the UTNDP

Newell [124] noted the complexity of the UTNDP, observing that the number of possible choices in the processes of selecting routes and setting vehicle frequencies can easily reach astronomical proportions, even for relatively small transport networks. When solving instances of the UTNDP, there is also another considerable difficulty — that of translating problem requirements into traditional mathematical programming formulations for which efficient solution techniques are

available [31, 124]. The UTNDP has been established as an NP-hard, mixed combinatorial optimisation problem [5].

Moreover, Baaj and Mahmassani [5] identified five main sources of inherent complexity in the UTNDP. The first is the difficulty associated with defining decision variables in terms of which the various components of the objective function can be described succinctly and effectively. This difficulty is elucidated by the fact that the frequencies of the transport vehicles appear in typical UTNDP model formulations, while the number of routes and their nodal composition do not.

The second source of complexity is ascribed to the numerous non-convexities and non-linearities exhibited by instances of the UTNDP. An example of such a non-convexity is caused by the option that more transport vehicles can be added to the transport system by the network designer, thus increasing operator cost, yet still not improving the passenger cost associated with total travel time. Newell [124] pointed out that the non-convexity of the problem is induced by the waiting time of passengers upon entering the system or at transfer points, where this cost is not associated with the links of the network.

The third source is the combinatorial explosion that occurs due to the discrete nature of the UTNDP, making it very hard to solve from a computational perspective. The problem grows exponentially in complexity with a linear increase in the size of the transport network [5, 114, 185].

The fourth source of complexity emanates from the multi-objective nature of the UTNDP [5, 31]. Reducing user and operator cost has mainly been considered as UTNDP objectives in the literature. In practice, important trade-offs between these objectives have to be considered, possibly in the presence of other (conflicting) objectives. The total demand satisfied should be examined against the total travel time and against the transport vehicle fleet size required to operate the transit system. Total demand satisfied has the components of total demand satisfied directly, *via* one transfer, *via* two transfers, and so on. Total travel time includes the components of travel time experienced in-vehicle, waiting for vehicles and transferring between routes.

The last source of complexity is attributed to the spatial layout of routes. There is considerable difficulty in formally characterising and incorporating what constitutes a good spatial layout for a transport network route setup. This aspect has been incorporated to some extent by the implementation of certain design criteria involving, for example, route coverage, route duplication, route length and directness of service [5].

4.3 The transit planning process

The sub-activities that have to be considered when solving instances of the UTNDP have been classified in different ways in the literature, and the purpose of this section is to provide the reader with some insight into the main features of the UTNDP, based on the literature.

Hasselström [72] identified the major features to be considered during the formulation of models for the UTNDP to be passenger demand characteristics, objective functions pursued, constraints imposed, passenger behaviour catered for, solution techniques adopted, and the computational time required to solve instances of the problem. Ceder and Wilson [30] classified studies related to the UTNDP as those dealing with idealised networks and those that focus on actual routes, and suggested that the main features of the UTNDP include demand characteristics, objectives and constraints, and solution methods. According to Baaj and Mahmassani [5], however, the

UTNDP has three distinct components embedded in the solution approach, namely a route generation design algorithm, an analysis procedure and a route improvement algorithm.

The global transit planning process is based on certain inputs that are required in order to be able to solve UTNDP instances successfully. These are passenger demand, the area underlying the transport network (including its topological characteristics), the available transport vehicles and the drivers of these vehicles. The end goal is to establish a set of transport routes or lines with their associated timetables [31, 68].

According to Ceder and Wilson [30], the planning process for transit networks can be partitioned into a sequence of five distinct steps. These steps are network design, frequency setting, timetable development, bus scheduling and driver scheduling. The activities associated with these five steps require different independent inputs and yield different outputs, as summarised in Table 4.1. The output of each step (other than the last) serves as the input to the next step. The clear and distinct order, as well as the independence of each of these activities, only exists in theory as they are all intertwined with one another in reality, because decisions made earlier in the sequence will typically have an effect on the activities later in the sequence [30].

TABLE 4.1: *The overall transit network planning process as suggested by Ceder and Wilson [30].*

Level	Independent inputs	Planning activity	Output
A	Demand data Supply data Route performance indicators	Network design	Route changes New routes Operating strategies
B	Subsidy available Vehicles available Service policies Current patronage	Frequency setting	Service frequencies
C	Demand by time of day Times for first and last trips Running times	Timetable development	Trip departure times Trip arrival times
D	Deadhead times Recovery times Schedule constraints Cost structure	Vehicle scheduling	Vehicle schedules
E	Driver work rules Run cost structure	Driver scheduling	Driver schedules

The five main transit planning activities in Table 4.1 are further elaborated upon according to the literature review of Guihaire and Hao [68] in the subsections of this section. Transit network design may be thought of as a strategic planning phase, referring to long-term decision making, whereas frequency setting and transit network timetabling may be considered as tactical planning. Vehicle and driver scheduling are finally considered as the operational planning phase [43, 82]. The focus in the remainder of this section falls on the first two activities, namely network design and frequency setting, because these are particularly relevant to the topic of this dissertation [68], whereas timetabling, as well as bus and driver scheduling, are only mentioned briefly.

4.3.1 Transit network route design

The purpose of the *transit network design* facet is to establish a set of bus routes that will serve the passenger demand of a particular area of interest, each route of which is specified by a sequence of vehicle stops [68, 82]. Transit route network design therefore involves the selection of routes within an existing transport network, with each route being associated with expected travel times that collectively achieve a set of objectives specified by the user [31].

The transit network design phase takes the transport network topology and passenger travel demand as its main inputs. Passenger demand is usually presented in the form of an OD matrix in which each entry represents the number of passengers who would like to travel from a row origin location to a column destination location [68, 72]. An OD matrix assumes a pre-determined set of zones or stop points as possible origin and destination locations. The *topological* input component captures the area's topological characteristics defined by the roads, possible areas for vehicle stops and transfer zones, and sometimes the locations of depots that serve as external terminals [68].

The transit agency that governs the transport network might accommodate certain policies that may cause constraints and objectives to intermingle, and since there generally exists no rule to differentiate them, these constraints and objectives are typically considered as a list of unique features. These features may include historical background elements, area coverage, route and trip directness demand satisfaction, number of lines or total route length, as well as operator-specific objectives [68]:

Historical background elements may have an influence on the design in the sense that the new design might for some reason (sometimes for a political reason) be undesirable because it disrupts existing lines.

Area coverage measures the percentage of estimated passenger demand that can be met by the transport network. This usually depends on different characteristics such as route length, route density, as well as vehicle stop and route spacing [11]. The rules of thumb typically employed is that people who live within 400–500m from a vehicle stop are included in the coverage percentage calculation [68]. Murray *et al.* [122] claimed that the aim should be to achieve a 90% ratio for the population having access to a bus, rail or ferry stop within 400m.

Route and trip directness may be described as a measure of the degree to which users are able to travel directly from their origins to their destinations. Limits are usually imposed on the maximum expected distance over which a user can travel in the transport network with consideration of the trip demand. From the passenger's perspective, the aim of the transport network should allow him or her to travel as directly as possible from an origin location to a destination location and require the shortest walking distances in order to reach the first and final vehicle stops [68]. Directness may be measured as the degree of deviation from the linear path between the origin and destination locations, or in terms of additional mileage (kilometres for South Africa) travelled by the transport vehicle when compared with other modes of transport [11]. The number of passenger transfers is also a frequent criterion in this respect, and a passenger trip assignment process is required to compute trip directness. This process includes assigning routes and transfers to passengers with respect to some objective, such as smallest number of transfers or shortest path [43].

Demand satisfaction is a crucial issue in the UTNDP. Demand is considered unsatisfied when the distance between the origin and/or destination is too distant from bus stops or when the trip directness is insufficient. As in trip directness evaluation, demand satisfaction

evaluation also requires a trip assignment process. In general, it is assumed that when a passenger has to make more than two transfers, (s)he will rather switch to another mode of transport in order to satisfy the demand [68].

Number of lines and total route length are general objectives that have to be minimised from the operator's perspective. This entails minimising the number of vehicle and crew resources required to sustain the transport network, while routes should neither be too long nor too short [68].

Operator-specific objectives are set in place by the operators of a transit network and are requirements they impose for various reasons, such as shapes imposed on routes, for example.

4.3.2 Transit network frequency setting

Transit scheduling involves setting up schedules that govern the transit operations along each vehicle route, and these too should pursue some user-defined objective [31, 82]. During this step of solving the UTNDP, each line or bus route in the network is assigned a frequency for each time period. The number of line runs correspond to the number of scheduled services of each line and are heuristically determined during this step. The *headway* is the inverse of the frequency per period and refers to the time elapsed between consecutive line run departures [68].

Frequency setting takes the transit route network, passenger demand and transport vehicle fleet as its main inputs. The transit route network is the main input for setting the vehicle departure frequencies. Detailed OD matrices are required and they should provide demand information during uniform demand time periods. These periods typically differ according to criteria based on the time of day, the day of the week and the time of the year, because peak and off-peak periods, the individual day of the week, and annual seasons and vacations play a role in passenger demand. As demand is time-dependent and elastic, it is important that a comprehensive study be undertaken in respect of passenger demand by conducting surveys over extensive periods of time and regularly updating them. This process is necessary in order to achieve a transit network that is efficient and offers a satisfying service. The data collection associated with this step is, however, an expensive and a rather complex task for a transit agency, and such data are therefore rarely available without charge [68]. *Transport vehicle fleet* is another factor that has to be considered as line frequencies depend on the vehicle fleet size and the vehicle capacities. A description of the vehicles available is therefore required, especially if it is a heterogeneous fleet. The vehicle running times per time period associated with each route should also be provided [68].

The main constraints and objectives that have to be considered during UTNDP frequency setting are demand satisfaction, the number of line runs, headway bounds and historical background:

Demand satisfaction is characterised by the line frequencies meeting the demand requirements as closely as possible in order to avoid excessively large headways and overcrowding, thereby decreasing transfer and waiting time.

The number of line runs showcases the multi-objective nature of the UTNDP — whereas operators typically desire to minimise the number of resources, passengers typically desire the convenience of having a wide range of line runs available to them.

Headway bounds are minimum or maximum constraints imposed on some lines or areas by the governing authorities.

The historical background may induce constraints on some lines similar to those described in §4.3.1 [68].

4.3.3 Transit network timetabling

Transit network timetabling is the process whereby timetables are generated that specify the departure times from all stops that are served by each line in the network. The timetable for each line run consists of the departure time from the initial terminal, the expected departure times from each vehicle stop along the route and the expected arrival times at the final stop. The first input required is the transit route network and the running times associated with the network. Another input required is the line frequencies as determined during the frequency setting phase — this determines the time coverage of the line. Additionally, an importance level is required for each transfer so as to ensure a higher quality of service by minimising waiting times. This importance level can be deduced from detailed time period-dependant OD matrices [68, 82].

The main constraints and objectives considered during the transit network timetabling phase are passenger demand satisfaction, transfer coordination, transport vehicle fleet size and historical background:

Demand satisfaction can be measured by computing passenger travel time, where timetabling can be used to strengthen demand satisfaction *via* the network. The goal should be to minimise these travel times in order to enhance passengers' mobility. If, for some particular trip, these times are too high, the associated demand will be considered unsatisfied demand.

Transfer coordination should be smooth between lines in both time and space. This may be achieved by taking into account each transfer zone and the associated lines. Various criteria may be employed to favour transfers among lines, such as, for example, the number of passengers involved.

The transport vehicle fleet size and the associated resource usage should be considered when the vehicle schedules are created from the line runs.

Historical background can once again play a part in timetable setting [68, 82].

4.3.4 Vehicle scheduling

The purpose of the vehicle scheduling phase is to determine feasible vehicle-to-trip assignments so as to cover all the planned trips with the goal of minimising the operator's cost by minimising vehicle usage. In this way, the total number of transport vehicles required can be determined for the given time period (usually a day) [68, 82]. Bunte and Kliewer [23] elaborated extensively on this phase of the UTNDP.

4.3.5 Crew scheduling and rostering

The crew or driver scheduling and rostering problems may be considered two different aspects of the UTNDP. Driver scheduling involves determining the daily duties of drivers that cover all the scheduled trips with the goal of minimising the driver wage cost. The solution to this sub-problem should conform to various constraints, such as minimum or maximum work time, maximum working time without rest, and daily rest allowance, for all drivers.

Driver rostering, on the other hand, is an assignment of generic duties to drivers of a specific depot over a long time span (usually a month). The goal in this type of assignment is to minimise driver wages subject to labour regulations, such as regulations on minimum allowable consecutive work days [68, 82]. Wren and Rousseau [173] elaborated extensively on this phase of the UTNDP.

4.4 Classification of UTNDP sub-problems

Guihare and Hao [68] claimed that no standardised naming convention existed in 2008 in the literature on the UTNDP. For example, Fan and Machemehl [51, 52, 53] used the terminology *bus transit route network design problem* to refer to a problem in which transit routes and vehicle frequencies have to be determined, whereas Baaj and Mahmassani [6] employed the terminology *transit route network design problem* in the same context, as did Kepaptsoglou [93].

Guihaire and Hao [68] therefore proposed a classification of these different names and attempted to establish a naming convention based on the first three levels of the transit planning process model of Ceder and Wilson [30]. In their view, the UTRP[1] refers only to the selection of an appropriate set of vehicle routes (without frequency setting), while only the frequencies are set in the UTFSP[2]. These two problems were elaborated on in §4.3.1 and §4.3.2, respectively. Moreover, the *urban transit timetabling problem* (UTTP)[3] is similar to the timetabling problem mentioned in §4.3.3. If the UTRP and UTFSP are attempted simultaneously, this combined problem is called the UTRFSP[4]. The UTFSP and UTTP may, however, also be attempted simultaneously, in which case the combined problem is called the *urban transit scheduling problem* (UTSP)[5]. Finally, the overarching problem encompassing the simultaneous solution of the UTRP, the UTFSP and the UTTP is called the UTNDP[6]. Figure 4.1 contains a graphical representation of the different facets of transit network problems.

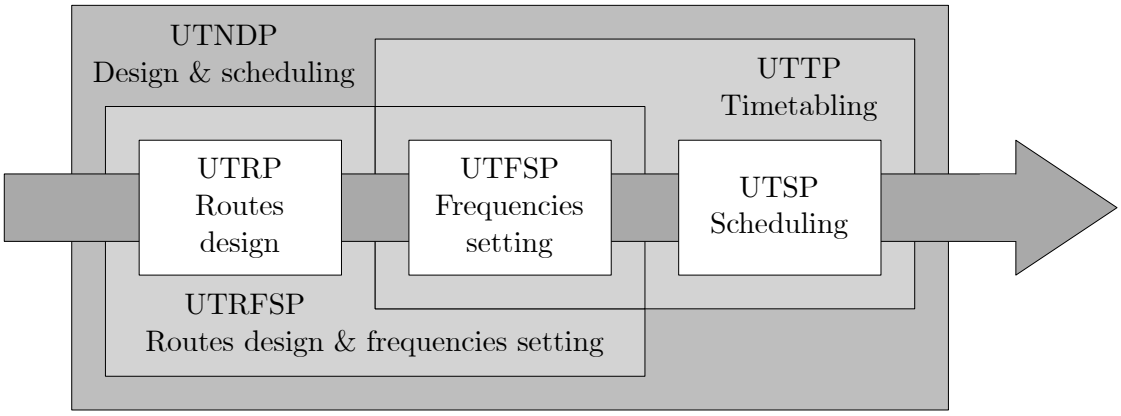


FIGURE 4.1: *The transit network problem structure proposed by Guihaire and Hao [68].*

[1]Called the transit network design problem by Guihaire and Hao [68].
[2]Called the transit network frequencies setting problem by Guihaire and Hao [68].
[3]Called the transit network timetabling problem by Guihaire and Hao [68].
[4]Called the transit network design and frequencies setting problem by Guihaire and Hao [68].
[5]Called the transit network scheduling problem by Guihaire and Hao [68].
[6]Called the transit network design and scheduling problem by Guihaire and Hao [68].

4.5 The urban transit routing problem

An instance of the UTRP can be established by specifying the objectives to be pursued, deciding on the selection of transit network items and characteristics to include in the design, and imposing constraints on the operating environment [93]. On a lower level of abstraction, a description of the transit network is required based on design variables as well as environmental and operational parameters. These parameters include the network structure, passenger demand patterns, and demand characteristics. All of these components are elaborated upon in this section.

4.5.1 Objectives

The UTRP objectives are the goals that are pursued (either minimised or maximised). Public transport systems aim to meet the needs of passengers while simultaneously aiming to minimise cost for the operators of the system [93, 116]. When deciding on the objectives to be adopted, both passenger and operator perspectives have to be considered. Most studies in the literature focus on both economic efficiency and service quality in the design of transit systems. The main objectives in the literature can be summarised as passenger benefit maximisation, operator cost minimisation, total welfare maximisation, capacity maximisation, energy conservation or individual parameter optimisation [93].

Passenger benefit maximisation may include ATT minimisation, total waiting time minimisation, access to transit system maximisation, coverage maximisation and total transfers minimisation. Operator cost minimisation, on the other hand, may include utilisation of asset maximisation, quality of service maximisation, operating cost minimisation (in terms of fuel consumption or drivers' wages), and vehicle fleet size minimisation. Total welfare maximisation includes both the minimisation of passenger and operator costs simultaneously.

The UTRP is therefore inherently multi-objective in nature, as mentioned, the aim being to achieve suitable trade-offs between different conflicting objectives [5, 31].

Criteria that researchers have generally associated with a good transport vehicle route set are when the entire transit demand is satisfied, a large percentage of transit demand is satisfied through direct routes and the ATT spent by passengers in transit is as low as possible [31].

4.5.2 Decision variables

The most common UTRP decision variables adopted in the literature define transport vehicle routes and frequencies [5, 30, 72]. Routes are typically specified in terms of lists of vertices or vehicle stops in sequence so as to provide an indication of the order in which a vehicle should traverse these vertices. Frequencies are the inverse of the desired inter-arrival times of the vehicles, and this can be increased by assigning more vehicles to routes. The higher the frequency, for example, or the lower the inter-arrival times, the less waiting time passengers will incur, but the higher the system operating cost will be. Other decision variables that have been considered relate to fares, zones, stop locations and bus types [93].

4.5.3 Network structure

Various transit network structures have been considered in the literature, such as radial networks, rectangular grids or irregular grid networks. Radial networks take the form of a hub and spokes which typically assumes the form of a central business district as the hub with roads extending

outwards to other areas or neighbourhoods that resemble the spokes of a wheel. Rectangular grid layouts are typically found in cities as a result of the physical layout of streets for city blocks around buildings. Irregular grid layouts have no specific form and since the 1980s the majority of UTRP research has been carried out in this more general context [93].

4.5.4 Demand patterns

Demand patterns describe the nature of the flow of passengers in the transit network. The two main types of demand patterns are *many-to-one* and *many-to-many* demand patterns. A many-to-one demand pattern occurs when passengers from various origins wish to travel to one destination, such as when commuters have to travel from their homes to their work in a central business district or when school children have to be transported from their homes to a school. A many-to-many demand pattern occurs, on the other hand, when people at multiple origins wish to travel to multiple destinations [93].

4.5.5 Demand characteristics

The demand can be characterised as being either *elastic* or *fixed* [93]. Elastic demand depends on the performance and service provided by the transit network and can also vary over time based on demand at certain times of the day. Since the 1980s, some researchers have included minor elastic demand aspects in their modelling of the UTRP [72]. Despite the inherent elastic nature of demand in reality, however, the complexity of the UTRP has led the majority of researchers generally to assume fixed demand when attempting to solve instances of the problem [53, 121].

4.5.6 Constraints

Constraints included in instances of the UTRP may be thought of as a mix between *performance requirements* and *resource limitations* [52, 53]. Zhao and Zeng [185] summarised the different types of performance requirement constraints in terms of route shape, route directness, maximum route length, the number of routes, the range of feasible frequencies, and minimum and maximum load factors, whereas the major resource constraints are usually operational budgetary constraints and restrictions on the size of the vehicle fleet.

Fan and Mumford [50] pointed out that the maximum and minimum length of a route can be imposed in terms of either a limited number of stops along the route or in terms of the total round trip time of the route. These are usually dictated by the transit network planner based on criteria such as driver fatigue and difficulty to maintain transport schedule adherence. Another essential constraint is ensuring that the transit route origin is connected to any destination in the network. The number of routes is also usually set beforehand by the transit network planner based on funding constraints. Other constraints may include requirements that there are no cycles or backtracks in the transit network. Such requirements form part of the shape restriction of routes [50].

4.5.7 Solution approaches for the UTRP

The sources of complexity inherent to the UTRP described in §4.2 prevent the exact solution of problem instances in almost all cases. Solution approaches applied to instances of the UTRP

in the literature can be partitioned into three main classes: Practical guidelines or *ad hoc* procedures, analytic optimisation methods for idealised situations, and metaheuristic optimisation approaches for more practical problem instances [5, 53].

During the early stages of research on the UTRP in the literature, traditional operations research optimisation models (of an analytic nature) were employed. In contrast to determining both the route network and design parameters simultaneously, these techniques were applied to determine one or several design parameters, such as stop spacing, route spacing, route length, transport vehicle size and/or frequency of service in the context of a predetermined transit route network [53]. Examples of these approaches can be found in the work of Newell [124] and in that of Leblanc [108]. These analytic optimisation models are very effective in solving small-scale network optimisation problems related to the UTRP under idealised conditions, or models with one or two variables [53].

When dealing with transit networks of realistic sizes, however, where many variable values have to be determined, the aforementioned approaches do not work well. In such cases, the inherent complexity involved in instances of the UTRP call for methods that can deal effectively with the larger sizes of realistic transit networks. For this reason, metaheuristics have often been applied to UTRP instances [53, 68, 93]. These methods yield reasonably good local optima, but are not guaranteed to find globally optimal solutions [53].

(Meta)heuristic solution methods applicable to the UTRP can be classified into four main families according to Guihaire and Hao [68]. These are specific and *ad hoc* heuristics that often follow the greedy construction principle, neighbourhood searches (such as SA and tabu search), evolutionary searches (such as GAs), and hybrid searches which combine two or more metaheuristic solution methods [68]. One of the notable advantages of the application of metaheuristics to the UTRP is that they are not designed for specific mathematical models, and can thus be applied to various different models. These solution methods are flexible and can adapt to almost any form of objective function and arbitrary constraint sets by defining general search frameworks [68]. Metaheuristic solution methodologies have also been applied successfully to the simultaneous determination of transit routes and frequency setting [53].

In the remainder of this section, the main six solution approaches prevalent in the literature are elaborated upon. These are exhaustive searches, mathematical programming, heuristics, trajectory-based searches, population-based approaches, and hybrid approaches [51, 68].

Exhaustive search approach

An exhaustive search approach involves searching through the entire solution space to find a global optimum. This approach can be followed in all cases of the UTRP, but for instances with large solution spaces it can become computationally time consuming [51, 68]. Fan and Machemehl [51] claimed that if R_{feas} feasible routes are generated by an initial candidate route set generation procedure, and at most R_{max} of these routes are sought during the transit design process, then the size of the solution space is $\sum_{R=1}^{R_{max}} C_{R_{feas}}^{R}$. For example, if $R_{feas} = 100$ and $R_{max} = 10$, the size of the solution space is approximately 1.942×10^{13}. Table 4.2 illustrates the growth of the total number of enumerations required as the dimensions of a small instance of the UTRP increases. As a result, the exhaustive search approach is usually only applied to very small UTRP instances. Because realistic networks can easily contain as many as 1 000 feasible routes, this solution approach is therefore of very limited practical value [51].

TABLE 4.2: *The estimated total number of solution sets for the UTRP when R_{max} routes may be chosen, yielding a potential of R_{feas} feasible solutions based on the given network structure.*

	$R_{feas} = 100$	$R_{feas} = 150$
$R_{max} = 5$	79 375 495	612 422 930
$R_{max} = 10$	19 415 908 147 835	1 258 067 642 133 720
$R_{max} = 15$	305 847 423 291 010 000	182 321 998 028 935 000 000
$R_{max} = 20$	707 291 297 071 285 000 000	4 277 981 965 426 130 000 000 000

Mathematical programming approaches

Although difficult, the UTRP can be cast as a mathematical programming problem. The solution approaches available to solve mathematical programming problems include various mixed-integer programming techniques [68].

In 2004, Guan *et al.* [67] proposed a linear binary integer programming model for simultaneous optimisation of routes and passenger transit assignment. A standard branch-and-bound method may be used to solve the model. Examples were given of the model applied to a few minimum spanning tree networks and a simplification of the mass transit railway network of Hong Kong. The authors acknowledged that their approach could only be applied to small networks, and that metaheuristic methods should be considered for larger networks.

Chakroborty [31], however, identified certain limitations of some of these formulations resulting in a situation where the models do not adequately represent reality, due to factors such as the need for additional discrete decision variables, the existence of logical conditions and the required incorporation of non-linearities. The benefit of mathematical programming formulations is that optimality of solutions can be proved, but these approaches are typically avoided due to the fact that they do not scale well in terms of computational complexity [31]. For this reason, heuristic and metaheuristic approaches have been applied to solve most instances of the UTRP in the literature [68, 93].

Heuristic approaches

A *heuristic* is defined by Winston [171] as a method for solving an optimisation problem by using a trial-and-error approach when an exact algorithmic approach is impractical. These methods do not, however, ensure that an optimal solution is obtained, but instead generally produce acceptable solutions when implemented correctly, and do not require the computational time of exhaustive searches or even mathematical programming approaches [93]. Mandl [115] applied such a method in 1979 in which initial candidate routes were generated by finding shortest paths between the different OD pairs. Thereafter, the paths that served the largest demand portions were incorporated into the solution iteratively. These paths were then configured to meet the requirements of service coverage and directness objectives. The routes were finally modified in an iterative fashion in pursuit of another objective, namely minimisation of the total travel time of passengers. This method was applied to a well-known 15-vertex real network in Switzerland with 15 570 demand trips per day, on average.

In 2005, Lee and Vuchic [109] minimised passenger travel time by using an iterative procedure under variable demand. An initial network is generated by applying a shortest path algorithm between all pairs of vertices, and then eliminating undesirable paths such as subsets of other paths. Demand was assigned to routes by means of a transit assignment procedure, and by concentrating the demand along certain routes, less efficient routes could be eliminated. Further

improvement of the routes was facilitated by an improvement procedure that decreased the passengers' travel time. Both fixed and variable demand were considered, and passenger travel mode choice was modelled by means of a logit formulation.

Trajectory-based metaheuristic approaches

Zhao and Gan [182] presented an aggregated metaheuristic approach in 2003, incorporating the application of an integrated SA, tabu and greedy search algorithm, as well as a greedy search algorithm and a fast hill climbing algorithm. The two objectives pursued were the minimisation of the number of transfers incurred by passengers and the maximisation of service coverage by the routes. The solution method included three phases, namely the routes' decision space representation, constraints representation, and a search algorithm for finding high-quality solutions. The authors applied their solution approach to a large-sized network of Miami-Dade County, Florida. In 2004, Zhao and Ubaka [183] published an article in respect of both the aforementioned basic greedy search algorithm and the fast hill climbing search algorithm for solving UTRP instances.

Also in 2004, Fan and Machemehl [51] wrote a report containing various detailed descriptions of the UTRP and its intricacies. The UTRP was cast as a multi-objective non-linear mixed-integer model, but weights were assigned to each objective. The sum of these weighted objectives was then minimised. The objectives represented the total passenger travel time, the total number of buses required, and the cost of unsatisfied demand. The proposed solution approach included three phases, namely an initial candidate route set generation procedure, a network analysis procedure for evaluating performance, and a metaheuristic search procedure. Five metaheuristic search procedures were evaluated, including three trajectory-based metaheuristics (local search, SA, and tabu search) and two population-based metaheuristics (a GA, and a random search). Sensitivity analyses were conducted in respect of each algorithm, and the relative performances of the algorithms were compared in the context of benchmark problem instances. Both fixed and variable transit demand were incorporated in the algorithmic implementations. The concepts of centroid vertices and distribution vertices applied to the demand of the UTRP was first introduced by the authors, yielding a more accurate distribution of passenger demand across the network.

Fan and Machemehl [53] published an article in 2006 on an SA algorithm for solving instances of the UTRP, based on an earlier 2004 report [51], claiming that the SA algorithm outperformed a GA in most example network cases considered by them. Later, Fan and Machemehl [54] proposed three tabu search implementations for solving instances of the UTRP in 2008. Their approach closely resembled the framework proposed in their previous report of 2004 [51].

The relative quality of solution representation schemes, the initialisation procedures for algorithms and neighbourhood move suitability for the SA metaheuristic in the context of the UTRP were considered in some detail by Fan and Mumford [50] in 2010. For validation purposes, a simple hill climbing algorithm was employed. Their contribution was a basic metaheuristic framework for solving instances of the UTRP, including a solution representation scheme, an initialisation procedure and a set of neighbourhood moves. Their model included a single objective function in which weights were assigned to the objectives of minimising passenger travel time and minimising the number of transfers.

Kiliç and Gök [95] introduced a new route generation procedure and tested it in the context of five benchmark instances in 2014. The procedure involved finding shortest paths in a network, and thereafter calculating the total passenger usage per edge. A discrete usage probability was determined by dividing the calculated total for each edge by the weight of each edge and

normalising the quotients returned. The route generation procedure took into account these probabilities. Thereafter, the initial solutions were improved upon by utilising a hill climbing procedure in conjunction with a tabu search procedure.

In 2018, Ahmed *et al.* [2] employed selection hyperheuristics to solve the UTRP, by exploring the space of low-level heuristics used to modify route sets. Each selection heuristic was empirically tested in respect of a number of benchmark instances made available by Mumford [121]. It was found that a sequence-based selection method in combination with a great deluge acceptance method performed the best overall, also achieving fast algorithmic run times. The same objective functions considered by John *et al.* [90] were utilised.

Ahmed *et al.* [3] continued to apply their hyperheuristic approach within the context of fixed terminal vertices for the UTRP in 2019. They required that buses be restricted to start and end their trips at specified terminal vertices. When results were compared with the performance of an NSGA II approach, it was found that their hyperheuristic approach outperformed the NSGA II.

Population-based metaheuristic approaches

Evolutionary algorithms are population-based search procedures based on the principles of natural selection in evolutionary biology. In essence, these searches are parallel searches through the solution space aimed at an iterative improvement of the mean fitness value of an evolving population of candidate solutions to an optimisation problem [51, 68]. The papers by Fan and Machemehl [52, 51], Chakroborty [31, 32] and Mumford [121] contain examples of where GAs have been applied to instances of the UTRP.

An innovative approach towards including additional routes in existing route networks was introduced by Xiong and Schneider [175] in 1992. They improved upon the standard GA by collecting non-dominated solutions during the entire search process, as opposed to only returning the last generation, and called this a cumulative GA. Furthermore, a neural network was used to determine one of the fitness functions, namely passenger travel time, by predicting the output of a passenger trip assignment algorithm. Their approach not only delivered accurate results, but also achieved a substantial time gain over other transit assignment algorithms.

Another GA-based approach was proposed by Chakroborty and Dwivedi [32] in 2002. A route set was initially constructed by means of a heuristic procedure. Five criteria were used to determine the fitness value of a candidate route set, namely passenger travel time (including in-vehicle travel time and transfer time), the percentage of demand met with either zero, one, or two transfers, and the percentage of unsatisfied demand (requiring more than two transfers). For crossover, routes or parts of routes were exchanged with those of other route sets. A string representation was adopted for route set encoding.

Fan and Machemehl [52] used a GA in 2006 to examine the underlying characteristics of the UTRP with variable transit demand, based on their earlier 2004 report [51]. The characteristics examined were redesign of existing networks, the effect of demand aggregation, and the effect of various route set sizes.

Nikolić and Teodorović [126] applied a bee colony optimisation metaheuristic for solving the UTRP in 2013, aiming to maximise the number of satisfied passengers, minimise the total travel time of passengers and minimise the total number of transfers incurred.

In 2014, Kechagiopoulos and Beligiannis [91] solved the UTRP by applying a particle swarm optimisation algorithm, considering both passenger and operator costs, and validated their ap-

proach against Mandl's benchmark instance [115]. The authors also developed a method for accommodating the discrete decision variables of the UTRP in their solution approach.

In 2013, Mumford [121] addressed the lack of UTRP benchmark data available to researchers against which they could compare their results and solution quality evaluation techniques by contributing new benchmark data sets of varying network sizes — the larger sized networks reflecting realistic transit network sizes, where previously only smaller sized networks had been considered for validation purposes. Mumford employed a so-called *simple evolutionary algorithm for multi-objective optimisation* which was able to handle the multi-objective nature of the UTRP appropriately. New problem-specific genetic operators were proposed, as was an initial population generation procedure yielding only feasible route sets. This approach was followed to establish benchmark results for the newly introduced test problem instances.

A GA was also employed by Chew *et al.* [33] in 2013 for solving instances of the UTRP. Their approach included a repair mechanism according to which vertices were inserted into routes contained in infeasible route sets, in an attempt to restore feasibility during their search progression. They also introduced a new crossover procedure for incorporation into a GA by utilising a set of feasibility criteria in an attempt to reduce the probability of producing infeasible solutions.

John *et al.* [90] proposed an NSGA II framework in 2014 for solving the UTRP cast as an MOP, and used intuitive graph-theoretic principles to model the problem. Their results led to improvements on previously published results, and incorporated the four benchmark instances earlier published by Mumford [121], as well as Mandl's Swiss network [115]. The authors also proposed a new construction heuristic for seeding an initial population of routes. John's doctoral dissertation [89] of 2016 contains more details on the UTRP and his approach towards solving it.

Furthermore, a differential evolution metaheuristic approach (a GA variant designed for continuous search spaces) was proposed by Buba and Lee [21] in 2016 for solving the UTRP. Their model incorporated only the minimisation of the average travel time of passengers. The authors proposed a new repair mechanism that may be applied to infeasible route sets. Comparisons were made between different repair mechanisms, and they identified a particular combination of multiple repair mechanisms which returned the best results.

Hybrid approaches

Zhao *et al.* [184] designed an integrated SA algorithm, a tabu search and greedy search method in 2005 for solving the UTRP. Their objectives involved minimising transfers and maximising service coverage.

In 2013, Yan *et al.* [177] introduced a hybridised SA which was used in conjunction with Monte Carlo simulation to solve instances of the UTRP, considering stochastic travel times. The objective pursued was the minimisation of operator cost subject to service level constraints.

4.5.8 Benchmark instances

Various UTRP benchmark instances have been employed in the literature for validation and verification purposes, among which Mandl's Swiss network [115] has been the most popular by far, although it is also one of the smallest test instances. As mentioned, Mumford introduced four new benchmark instances of varying sizes in 2013 with the hope that they may be used as test instances in the context of UTRP modelling by other researchers. These benchmark instances have since indeed attracted the attention of various researchers. In this section, the

aforementioned five benchmark instances are reviewed, along with the best results available in the literature for these problem instances.

Table 4.3 contains a meta-data summary related to the five benchmark instances mentioned above. These meta-data include the number of vertices contained in each instance, the number of edges present, the number of routes sought, the number of vertices allowed per route, and lower bounds on the two most common objective functions, namely minimising the ATT and the TRT.

TABLE 4.3: *UTRP benchmark instances available in the literature [121].*

Instance	Number of vertices	Number of edges	Number of routes	Vertices per route	ATT lower bound [min]	TRT lower bound [min]
Mandl	15	21	4–8	2–8	10.0058	63
Mumford0	30	90	12	2–15	13.0121	94
Mumford1	70	210	15	10–30	19.2695	345
Mumford2	110	385	56	10–22	22.1689	864
Mumford3	127	425	60	12–25	24.7453	982

The five corresponding road networks may be visualised as graphs, as shown in Figures 4.2–4.6. In these figures, the vertex coordinates have been adjusted as to yield a more pleasant viewing experience, and do not resemble their true coordinates.

Table 4.4 contains summaries of the best results obtained in the literature for the five popular benchmark instances. For Mandl's Swiss network, the number of routes was set to four, six, seven and eight in most of the studies, although the six-route instance has received the most attention. In the remainder of this dissertation, these four instances of Mandl are referred to as Mandl4, Manld6, Mandl7, and Mandl8, respectively. The passenger cost here refers to the ATT objective function, and the operator cost refers to the TRT objective function. The percentage of demand satisfied by zero, one and two transfers are denoted by d_0, d_1 and d_2, respectively, and in most instances, cases of more than two transfers are treated as unsatisfied demand, denoted by d_u. These performance metrics are some of the most common found in the literature. The results reported in eight works from the literature are represented in Table 4.4, and all of the model solution approaches adopted in these works were described in §4.5.7.

Table 4.5, on the other hand, pertains to the best results obtained in terms of operator cost found in the literature for the UTRP. Four of the eight works represented in Table 4.4 also attempted to minimise the operator objective, and are therefore represented in Table 4.5. It is interesting to note that the lower bounds have only been obtained for the Mandl instances and the Mumford0 instance.

4.6 The urban transit frequency setting problem

An instance of the UTFSP, on the other hand, can be established by specifying the objectives to be pursued, the constraints that should be adhered to, a transit network route set (returned as output from a corresponding UTRP instance) for which frequencies should be set, demand data for passengers, and a passenger transit assignment procedure for assigning passengers to routes within the transit system. Of the aforementioned characteristics, the objectives, the decision variables, the constraints, and the transit assignment for the UTFSP are elaborated upon in this section. The demand patterns and characteristics mentioned in §4.5.4 and §4.5.5,

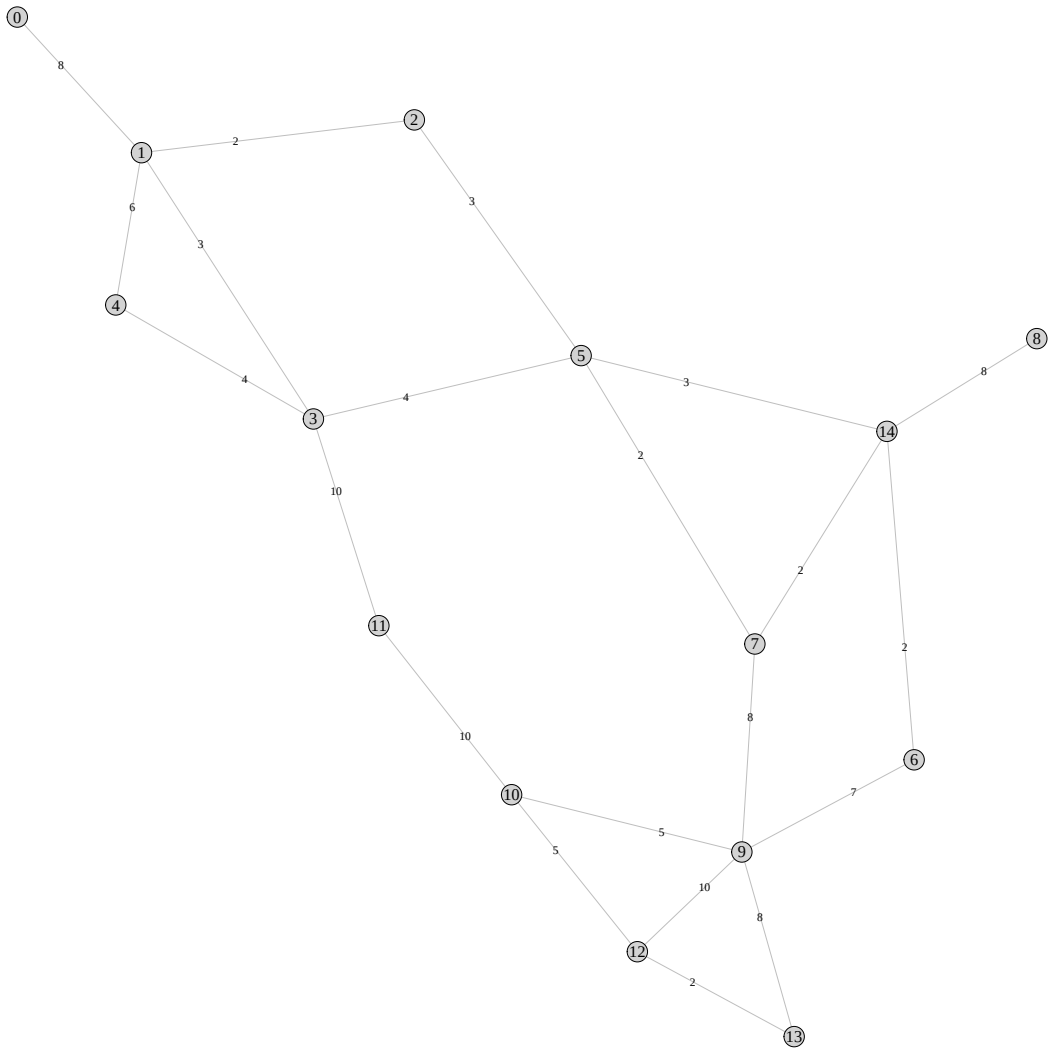


FIGURE 4.2: *A graphical representation of the road network of Mandl's Swiss UTRP benchmark instance, containing 15 stops and 21 road links. The edge weights represent travel times (in minutes).*

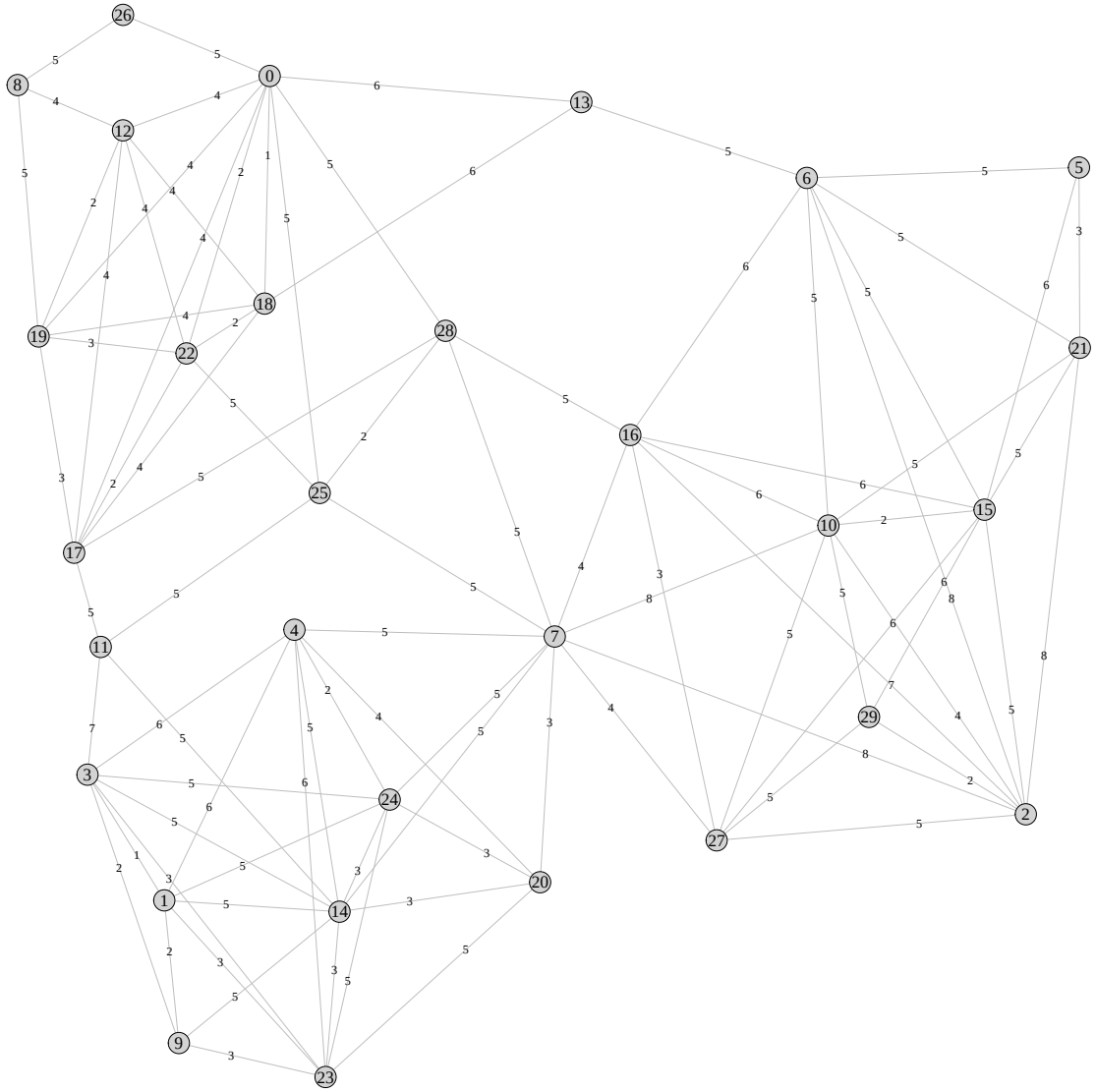


FIGURE 4.3: *A graphical representation of the road network of the Mumford0 UTRP benchmark instance, containing 30 stops and 90 road links. The edge weights represent travel times (in minutes).*

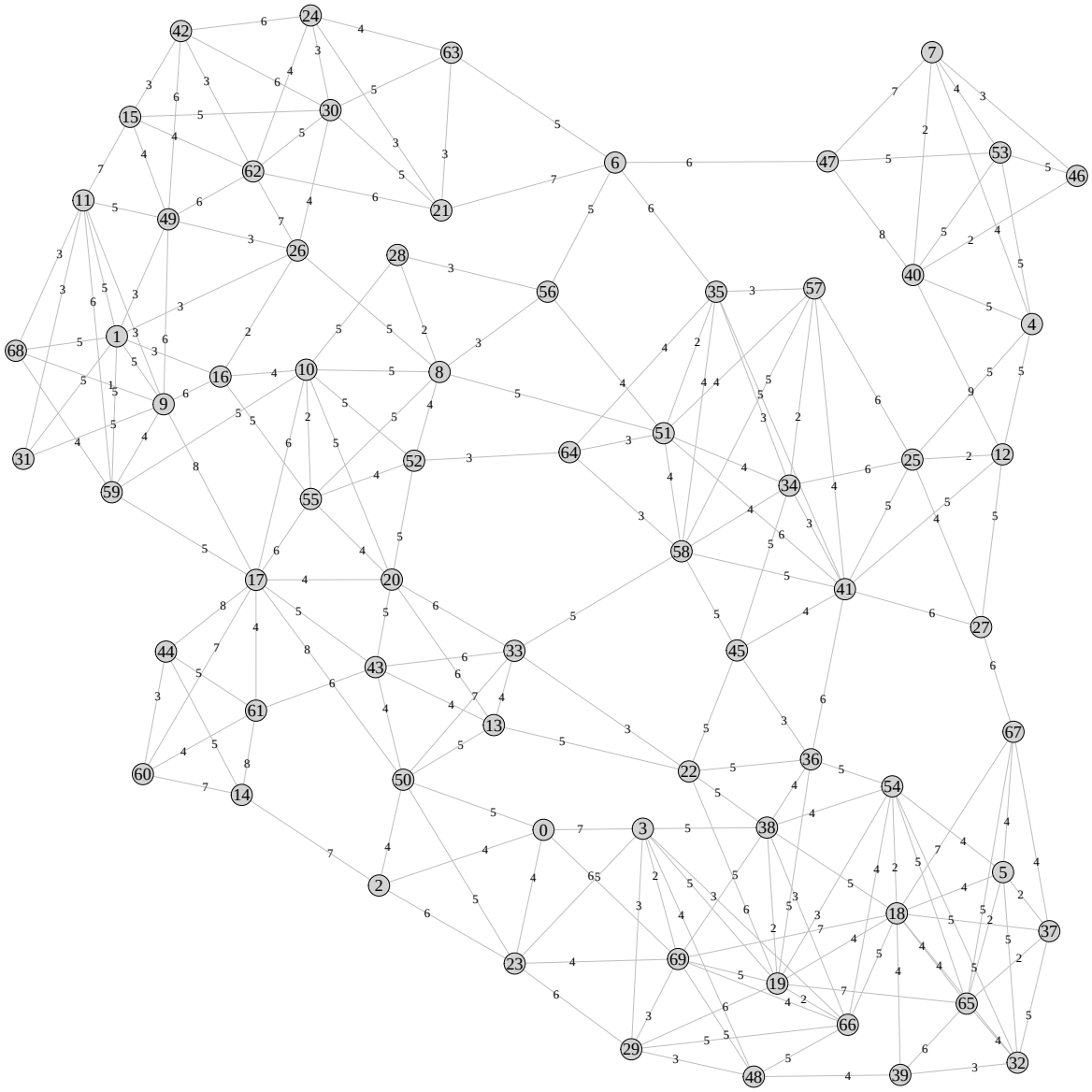


FIGURE 4.4: *A graphical representation of the road network of the Mumford1 UTRP benchmark instance, containing 70 stops and 210 road links. The edge weights represent travel times (in minutes).*

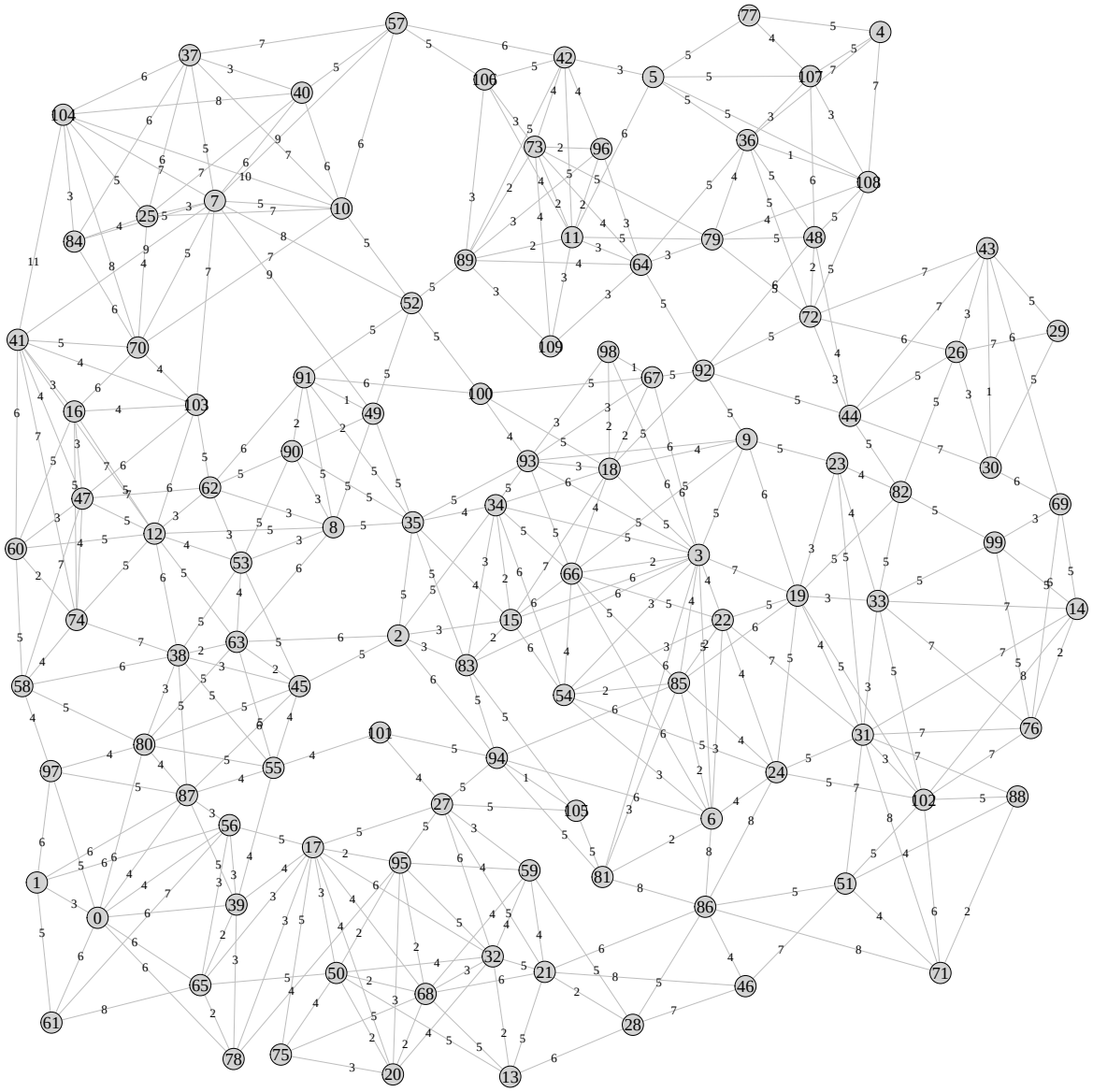


FIGURE 4.5: *A graphical representation of the road network of the Mumford2 UTRP benchmark instance, containing 110 stops and 385 road links. The edge weights represent travel times (in minutes).*

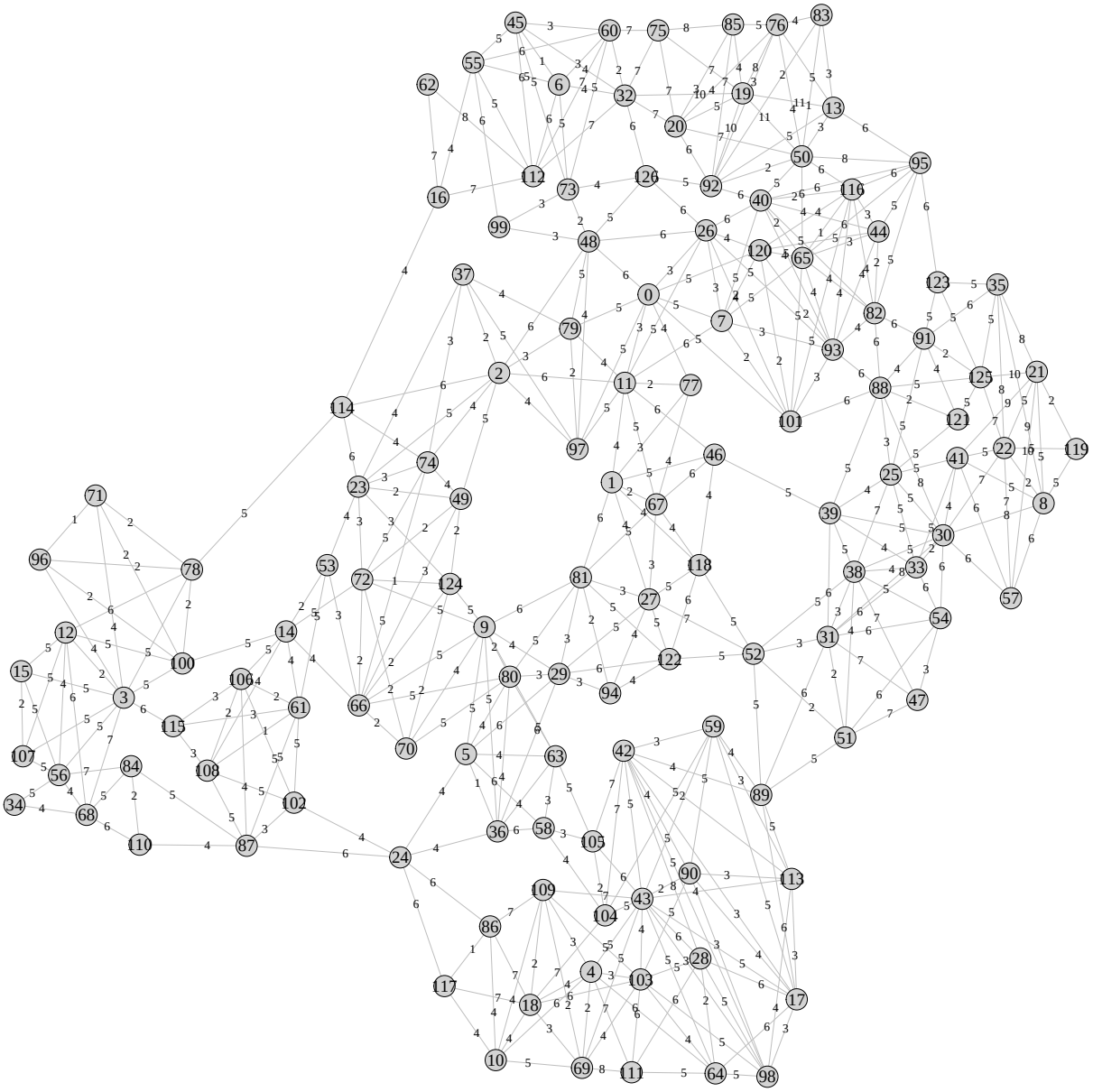


FIGURE 4.6: *A graphical representation of the road network of the Mumford3 UTRP benchmark instance, containing 127 stops and 425 road links. The edge weights represent travel times (in minutes).*

TABLE 4.4: *Best passenger cost results reported in the literature for the UTRP [2], with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mandl (1979) [115]	Chakroborty and Wivedi (2002) [32]	Fan and Mumford (2010) [50]	Mumford (2013) [121]	Chew *et al.* (2013) [33]	John *et al.* (2014) [90]	Kiliç and Gök (2014) [95]	Ahmed *et al.* (2018) [2]
Mandl4	Passenger	12.90	11.90	11.37	10.57	10.50	—	10.56	**10.48**
	Operator	—	—	147	149	150	—	137	148
	d_0	69.49	86.86	93.26	90.43	**91.84**	—	91.33	**91.84**
	d_1	29.93	12.00	6.74	9.57	**8.61**	—	8.16	8.15
	d_2	0.00	0.00	0.00	0.00	**0.00**	—	0.00	0.00
	d_u	0.00	0.00	0.00	0.00	**0.00**	—	0.00	0.00
Mandl6	Passenger	—	10.30	10.48	10.27	10.21	10.25	10.29	**10.18**
	Operator	—	—	215	221	224	212	216	212
	d_0	—	86.04	91.52	95.38	96.79	—	95.5	**97.17**
	d_1	—	13.96	8.48	4.56	3.21	—	4.5	**2.82**
	d_2	—	0.00	0.00	0.06	0.00	—	0.00	**0.00**
	d_u	—	0.00	0.00	0.00	0.00	—	0.00	**0.00**
Mandl7	Passenger	—	10.15	10.42	10.22	10.16	—	10.23	**10.10**
	Operator	—	—	231	264	239	—	274	250
	d_0	—	89.15	93.32	96.47	98.01	—	97.04	**98.84**
	d_1	—	10.85	6.36	3.34	1.99	—	2.83	**1.15**
	d_2	—	0.00	0.32	0.19	0.00	—	0.13	**0.00**
	d_u	—	0.00	0.00	0.00	0.00	—	0.00	**0.00**
Mandl8	Passenger	—	10.46	10.36	10.17	10.11	—	10.20	**10.08**
	Operator	—	—	283	291	256	—	298	272
	d_0	—	90.38	94.54	97.56	99.04	—	97.37	**99.16**
	d_1	—	9.62	5.46	2.31	0.96	—	2.63	**0.83**
	d_2	—	0.00	0.00	0.13	0.00	—	0.00	**0.00**
	d_u	—	0.00	0.00	0.00	0.00	—	0.00	**0.00**
Mumford0	Passenger	—	—	—	16.05	—	15.40	14.99	**14.09**
	Operator	—	—	—	759	—	745	707	722
	d_0	—	—	—	63.20	—	—	69.73	**88.74**
	d_1	—	—	—	35.82	—	—	30.03	**11.25**
	d_2	—	—	—	0.98	—	—	0.24	**0.00**
	d_u	—	—	—	0.00	—	—	0.00	**0.00**
Mumford1	Passenger	—	—	—	24.79	—	23.91	23.25	**21.69**
	Operator	—	—	—	2 038	—	1 861	1 956	1 956
	d_0	—	—	—	36.60	—	—	45.10	**65.75**
	d_1	—	—	—	52.42	—	—	49.08	**34.18**
	d_2	—	—	—	10.71	—	—	5.76	**0.07**
	d_u	—	—	—	0.26	—	—	0.06	**0.00**
Mumford2	Passenger	—	—	—	28.65	—	27.02	26.82	**25.19**
	Operator	—	—	—	5 632	—	5 461	5 027	5 257
	d_0	—	—	—	30.92	—	—	33.88	**56.68**
	d_1	—	—	—	51.29	—	—	57.18	**43.26**
	d_2	—	—	—	16.36	—	—	8.77	**0.05**
	d_u	—	—	—	1.44	—	—	0.17	**0.00**
Mumford3	Passenger	—	—	—	31.44	—	29.50	30.41	**28.05**
	Operator	—	—	—	6 665	—	6 320	5 834	6 119
	d_0	—	—	—	27.46	—	—	27.56	**50.41**
	d_1	—	—	—	50.97	—	—	53.25	**48.81**
	d_2	—	—	—	18.79	—	—	17.51	**0.77**
	d_u	—	—	—	2.81	—	—	1.68	**0.00**

TABLE 4.5: *Best operator cost results reported in the literature for the UTRP [2], with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mumford (2013) [121]	Chew *et al.* (2013) [33]	John *et al.* (2014) [90]	Ahmed *et al.* (2018) [2]
Mandl6	Operator	**63**	**63**	**63**	**63**
	Passenger	**13.48**	**13.48**	**13.48**	14.28
	d_0	70.91	70.91	—	62.23
	d_1	25.5	25.5	—	27.16
	d_2	2.95	2.95	—	9.57
	d_u	0.64	0.64	—	1.028
Mumford0	Operator	111	—	95	**94**
	Passenger	32.4	—	32.78	26.32
	d_0	18.42	—	—	14.61
	d_1	23.4	—	—	31.59
	d_2	20.78	—	—	36.41
	d_u	37.4	—	—	17.37
Mumford1	Operator	568	—	462	**408**
	Passenger	34.69	—	39.98	39.45
	d_0	16.53	—	—	18.02
	d_1	29.06	—	—	29.88
	d_2	29.93	—	—	31.9
	d_u	24.66	—	—	20.19
Mumford2	Operator	2 244	—	1 875	**1 330**
	Passenger	36.54	—	32.33	46.86
	d_0	13.76	—	—	13.63
	d_1	27.69	—	—	23.58
	d_2	29.53	—	—	23.94
	d_u	29.02	—	—	38.82
Mumford3	Operator	2 830	—	2 301	**1 746**
	Passenger	36.92	—	36.12	46.05
	d_0	16.71	—	—	16.28
	d_1	33.69	—	—	24.87
	d_2	33.69	—	—	26.34
	d_u	20.42	—	—	32.44

respectively, are also applicable to the UTFSP. The section finally closes with a brief overview of the approaches that have been adopted in the literature to solve instances of the UTFSP.

4.6.1 Objectives

When considering suitable frequencies for a transit network, provision should be made to service passengers sufficiently on a regular basis, while simultaneously achieving a sparse enough service so that the fleet size can be reduced appropriately, and the operator cost thus lowered [68]. Both passenger and operator perspectives therefore have to be considered when selecting objectives for the UTFSP, as it is inherently also multi-objective in nature. Some of the most popular objectives incorporated into models of the UTFSP are passenger travel time minimi-

sation, passenger waiting time minimisation, passenger utilisation maximisation, and fleet size minimisation [68, 82].

4.6.2 Decision variables

As the name suggests, frequencies at which transit vehicles should operate along transit routes are the main decision variables for the UTFSP [68]. Some authors have also utilised headways as their decision variables. Headway is the inverse of frequency [68]. Another approach taken by certain authors involves the direct assignment of buses to routes, whereby the frequency can be determined. A simple relationship exists between the number of buses operating along a route and the frequency with which buses operate along these routes, and can be stated as frequency (measured in buses per minute) multiplied by round trip time (measured in minutes) [5]. The round trip time of a route is defined as twice the duration of the route.

4.6.3 Constraints

Bus fleet size is one of the popular constraints in the UTFSP, as this is typically the most limiting factor for operators when choosing how to assign their buses to routes. The constraint is usually prescribed by requiring that the sum of the number of buses operating along all the routes be less than or equal to the maximum fleet size available [5].

Other constraints might include a minimum or maximum frequency that should be maintained along a route, typically imposed by the operators [68]. A constraint on the required service level may further be imposed so as to ensure that the demand of the passengers is sufficiently met. Headways that are excessively large, as well as overcrowding, should be avoided, thereby reducing transfer and waiting time [68].

4.6.4 Transit assignment

Desaulniers and Hickman [43] defined the *transit assignment problem* (TAP) as, given OD flows of passengers, to determine the passenger flows incurred along the paths (or transit routes) through the transit network. It is assumed that a passenger's objective is to minimise his or her generalised cost or travel time. Travel time is affected by various components, namely waiting time at a stop, access time to a stop, egress time, in-vehicle travel time, transfer time, and the total number of transfers. Monetary cost may also be included in the objective function. From the passenger's perspective, (s)he then has to make a routing decision based on the minimum cost of reaching his or her destination from a specified origin [43]. In essence, the TAP is concerned with modelling passenger route choice, and various different approaches have been applied in the literature to solve this problem [112, 151].

The TAP is indispensable when conducting transit modelling and assigning demand to the respective paths of a transit system in order to analyse the system's performance [112]. Transit assignment has commonly been considered a part of the UTNDP, but has also been studied as a separate problem in the literature [43, 151]. Liu *et al.* [112] conducted a literature review on the TAP in 2010, distinguishing between static transit assignment, within-day dynamic transit assignment, and emerging approaches as three different perspectives for passenger route-choice behaviour.

Static transit assignment methodologies include shortest-path-based all-or-nothing assignment, user equilibrium paradigm-based assignment, and random utility maximisation choice models.

The fact that these approaches do not consider time dimensions in the route choice of passengers distinguishes this category of decision models from the rest. Within-day dynamic transit assignment, on the other hand, is based on time-dependent OD demand matrices and involves consideration of the departure times of transit schedules. Emerging approaches are relatively theoretical and comprise miscellaneous methodologies. These approaches have been employed in studies on the complexities involved in passengers' behavioural mechanisms related to information integration and decision making, modelled in the context of a day-to-day and/or real-time dynamic framework [112]. Examples of each of these categories may be found in the review of Liu *et al.* [112].

All-or-nothing assignment and user equilibrium-based assignment approaches have been adopted when solving the strategic sub-problems of the UTNDP, namely the UTRP, the UTFSP, and the UTRFSP [68, 82]. It is important to note that the computational complexity of these models may affect the overall complexity of a frequency optimisation model, due to it involving numerous non-linearities. The inversely proportional relationship between waiting time and frequencies gives rise to one of these non-linearities, as do the various interactions among different routes [117].

Finally, Shih and Mahmassani claimed that a multi-path assignment model is preferred above a single-path assignment model [146], a statement with which Dial [44] also agreed. Dial commented that single-path assignment contradicts actual trip behaviour, and proposed a multi-path assignment model that could more accurately reflect true passenger behaviour. The popular *optimal strategies* assignment model of Spiess and Florian [151] is, however, widely accepted in the literature on the UTFSP [117].

4.6.5 Solution approaches for the UTFSP

Various solution approaches exist for solving instances of the UTFSP. Guihaire and Hao [68] identified mathematical programming approaches, heuristic approaches, and population-based metaheuristic approaches as the main solution methodologies that have been applied in the literature to solve UTFSP instances. These methodologies are discussed in this section, together with trajectory-based metaheuristic solution approaches.

Mathematical programming approaches

A non-linear model was proposed in 1980 by Schéele [143] in which the goal was to determine travel patterns and frequencies for a given transit network simultaneously. Minimisation of the total passenger travel time was the objective pursued in this study with a constrained fleet size being imposed on the problem.

LeBlanc [108] formulated a UTFSP mathematical programming model in 1987, with the objective of minimising multimodal user-optimal flow, subject to penalties associated with transit lines. A mode-split assignment model was required to capture the effects of modal split when frequencies were altered along each transit line, and he introduced a refinement to the conventional models considered to solve UTFSP instances at the time. His model considered access time, ride time and transfer delays when attempting to find the fastest transit paths for passengers.

In 1995, the UTFSP was formulated as a non-linear, non-convex mixed-integer programming problem by Constantin and Florian [36]. It was then reformulated as a bi-level minimisation problem for finding the optimum frequencies for buses, adopting a projected sub-gradient ap-

proach to solve the problem subject to fleet size constraints. The objective pursued was the minimisation of the total travel time (with waiting time included).

Heuristic approaches

Han and Wilson [69] proposed setting route frequencies in 1982 with a view to minimise the maximum occupancy level for each route at the maximum load point. The constraints that they imposed were the total fleet size available and the flow capacity that each route could handle. Their solution approach included two phases. During the first, frequencies were set to a minimum in order to satisfy all demand, and during the second, frequencies were uniformly increased along routes so that all available vehicles are utilised.

Ceder [28] presented an approach towards solving the UTFSP in 1984 by utilising passenger count data. His research focused on describing and analysing various data collection approaches that may be applied by operators to help efficiently set the frequencies of buses along routes. Four methods for setting bus frequencies were presented — two based on maximum load (point check) data and two based on load profile (ride check) data. Although more complete data are provided by a ride check than by a point check, the former are more costly than the latter, and so Ceder's work provides guidance to operators to help choose which method will be the most appropriate. The objective pursued was that of minimising the required bus trips and the number of buses required for each route when considering alternative bus schedules. He provided a method for analysing headways while simultaneously determining the effect on the required fleet size. The subsequent textbook written by Ceder [29] provides further details on frequency setting, his methods and further background.

In 2003, Gao *et al.* [60] dealt with the UTFSP by adopting a bi-level programming technique. For the upper-level model, the objectives considered by the authors were the minimisation of the total deterrence in the form of in-vehicle travel time, waiting time and the cost incurred by the system due to the assigned frequencies. A transit equilibrium assignment model was employed as the lower-level model for the passenger assignments to paths. The model was solved by means of a heuristic based on a sensitivity analysis approach. Operators may use this approach to adapt to various changes in demand patterns.

Trajectory-based metaheuristic approaches

In 2014, Martínez *et al.* [117] reformulated the aforementioned non-linear bi-level UTFSP model as a mixed-integer linear programming model, and were able to solve the model to optimality for small instances using standard mixed-integer linear programming techniques. A metaheuristic was, however, proposed to solve the model for larger instances, and the authors implemented a tabu search for setting route frequencies in particular. The performance of the tabu search was measured against exact solutions, where possible, and was found to perform satisfactory. The optimal strategies transit assignment model of Spiess and Florian [151] was again used in this case.

Population-based metaheuristics

Park [131] utilised GAs and simulation to solve an instance of the UTFSP in 2005. Two cases were considered, namely the arrival of buses following a deterministic process, or following a stochastic process. When considering a deterministic arrival process, a simple GA was used

to obtain high-quality headways, with problem-specific genetic operators incorporated into the approach. When considering a stochastic arrival process, however, a simulation-based GA was used to determine headways and slack times. Crossover, mutation and a coordinated headway generator constituted the problem-specific operators used in the GA.

John [89] solved the UTFSP in a multi-objective context in 2016, simultaneously minimising the mean expected travel time of passengers, based on the expected travel times returned by the optimal strategies transit assignment model proposed by Spiess and Florian [151], and minimising the total number of vehicles required to operate the network. In order to reduce the search space, John discretised the possible frequencies, as was also done by Martínez *et al.* [117]. He implemented the NSGA II, a multi-objective first decent, and a multi-objective tabu search to solve instances of the UTFSP, and compared the performance of these metaheuristics. It was found that the NSGA II consistently outperformed the other two metaheuristics.

4.7 The urban transit routing and frequency setting problem

The UTRFSP deals with designing transit networks, with routes and their associated frequencies being the decision variables, and is aimed at optimising some objective function related to passenger and/or operator cost, subject to certain constraints [68]. The main challenge, when solving the UTRP and UTFSP simultaneously, is that the considerable computational complexity of the UTRP is worsened by the requirement that appropriate frequencies should be set for each route as well, and additionally requires application of a computationally expensive passenger transit assignment for sufficiently accurate evaluations of the network. For this reason, the aforementioned two problems have commonly been solved separately in succession (first the UTRP and then the UTFSP), rather than simultaneously [82].

In terms of the objectives of, and constraints in, the UTRFSP, the same objectives are applicable as those mentioned in §4.5.1 and §4.6.1 for the UTRP and UTFSP, respectively, and so are the constraints of §4.5.6 and §4.6.3, respectively [68]. It is, however, more common to solve instances of the UTRP or the UTRFSP, in which routes are applicable, instead of only setting the frequencies, as in the UTFSP [85]. Many authors have solved the UTRFSP in stages, by first determining route sets, and then setting the frequencies of buses for these route sets appropriately, or by adopting heuristic approaches to determine the frequencies along the routes [68, 85]. In this section, various solution approaches adopted in the literature for solving instances of the UTRFSP are reviewed, namely mathematical programming approaches, heuristic approaches, trajectory-based metaheuristic approaches, population-based metaheuristic approaches, and hybrid approaches.

4.7.1 Mathematical programming solution approaches

The UTRFSP was modelled as a radial transit system using polar coordinates by Byrne in 1975 [27]. His approach involved determining both routes and frequencies under the assumption of either a constrained or an unconstrained fleet size, with the objective of minimising both passenger and operator costs.

Hasselström [72] presented a complex two-level mathematical programming model in 1981 which sought to generate routes by trip assignment and simultaneously determine the frequencies with which buses should operate along each route. The objective pursued was the maximisation of consumer surplus subject to variable demand formulations. The disadvantage of this ap-

proach was that, although routes and frequencies were determined simultaneously, it required the formulation of two optimisation problems.

In 2003, a mixed-integer formulation for solving the UTRFSP was presented by Wan and Lo [168] in which both routes and frequencies were determined. The objective pursued was the minimisation of the sum of operating costs by modification of a pre-existing network. In order to be solved by standard commercial software, the mixed-integer formulation was recast as a mixed-integer linearised formulation. Only small instances could be solved by their approach.

4.7.2 Heuristic solution approaches

In the 1991 seminal paper of Baaj and Mahmassani [5], one of the first artificial intelligence approaches towards solving the UTRFSP was presented. Their hybrid solution approach incorporated the knowledge and expertise of transit network planners, along with efficient search techniques. The three major components of the solution approach were a route generation procedure, a network analysis procedure, and a route improvement procedure. The route generation procedure employs a demand matrix to guide the route generation procedure by ordering the demand matrix OD pairs in decreasing order of trips. A user-specified number of routes are generated, and the initial set of routes is established by taking the user-specified number of OD trip pairs with the highest demand from the ordered list. A shortest-path finding algorithm is employed to find shortest routes between the origin and destination vertices of the selected OD trip pairs, and these shortest routes are taken as the initial bus route set. Vertices were later added to the established routes to connect the network, while other subsets of routes were removed, and the process was repeated until the user-specified number of routes had been established.

4.7.3 Trajectory-based metaheuristic solution approaches

Mauttone and Urquhart [119] proposed a multi-objective model formulation of, and solution approach for, the UTRFSP in 2009, commenting on the fact that previous works had employed a weighted sum objective function, whereas the UTRP is inherently multi-objective in nature. Two objectives were pursued, namely the minimisation of total passenger transport time (including in-vehicle travel time, waiting time, and transfer time) and minimisation of the fleet size required to maintain the network. They implemented a greedy randomised adaptive search procedure and proved that, with the same computational effort, more non-dominated solutions can be uncovered than when adopting the weighted sum approach. They considered Mandl's Swiss network [115] as one of their benchmark instances.

Xu *et al.* [176] considered the characteristics of bus-lines in 2014, and introduced the concept of bus-line segments to help solve the UTRFSP efficiently. They employed an SA algorithm to solve a bi-level programming model for the UTRFSP. Their SA implementation included seven bus-line adjustment methods for altering the route set, and two frequency adjustment methods for altering the frequencies of buses operating along the routes. The transit assignment for passengers problem was solved by employing a strategy equilibrium transit assignment model. The objective was to minimise a weighted sum of various passenger costs, various operator costs, and a bus-line management cost.

4.7.4 Population-based metaheuristic solution approaches

One of the first GA implementations for solving the UTRFSP was proposed by Pattnaik *et al.* [132] in 1998, and involved simultaneously determining routes and setting bus frequencies. The objective was to minimise a weighted sum of passenger and operator cost, subject to headway constraints. Their approach followed a two-part design in which a population of candidate route sets was generated, and then iteratively improved by a GA. Furthermore, a benchmark problem was introduced in their work.

In 2003, Ngamchai and Lovell [125] used a GA and proposed a method for determining routes and frequencies with the objectives of minimising total fleet size cost and minimising passenger travel time (including in-vehicle travel time and waiting time). Three phases were included in their solution approach. First, a route generation procedure for randomly generating connected route sets was applied. Secondly, a route evaluation procedure followed along with frequency setting applied to each route, with headways only being coordinated at one single transfer point along each route, for computational reasons. Lastly, a route improvement procedure was applied to the network incorporating a number of problem-specific genetic operators. A benchmark instance from Pattnaik *et al.* [132] was used for validation.

Yu *et al.* [180] proposed an ant colony optimisation metaheuristic for solving the UTRFSP in 2012. They built on the work of a direct traveler density model by extending it to the UTRFSP context upon incorporating demand density related to direct demand and transfers, as well as route lengths. Their approach was aimed at maximising the demand density of routes subject to resource constraints.

In 2014, Nikolić and Teodorović [127] adopted a bee colony optimisation metaheuristic for determining transit routes, and then setting appropriate frequencies for the routes. The objective function pursued was the minimisation of a weighted sum of the total number of buses required, the total number of rejected passengers, and the total travel time of passengers.

In 2015, Zhao *et al.* [186] employed a memetic algorithm to solve the UTRFSP. The objective functions associated with their model included minimising passenger cost and minimising unsatisfied demand. Embedded within the memetic algorithm was a local search operator based on the classical GA for improved computational performance. Furthermore, four types of local search operators were introduced for identifying high-quality chromosomes in the GA.

Arbex and da Cunha [4] proposed an alternating objective genetic algorithm in 2015 for solving a multi-objective UTRFSP instance, where the objectives pursued were cyclically alternated over generations of the GA. The two objectives considered were minimising passenger cost, taking into consideration the total number of transfers, waiting times, and in-vehicle travel times, and minimising the operator's cost, which was based on the total fleet size required to maintain the route set.

In 2018, Buba and Lee [22] put forward a differential evolution metaheuristic for simultaneously determining transit routes and bus frequencies. The objective function pursued was the minimisation of a weighted sum of passenger travel time (including transfer penalties) and the cost associated with unmet demand. Mandl's Swiss network [115] was considered as the benchmark instance for validation purposes.

4.7.5 Hybrid solution approaches

In 2006, Zhao and Zeng [185] combined a GA and an SA algorithm to solve the UTRFSP, taking routes and headways as the decision variables. The total passenger travel time and route

directness (corresponding to the minimum number of transfers) were the objectives minimised in their solution approach.

Bagloee and Ceder [9] combined two population-based metaheuristics in 2011, namely a GA and an ant colony optimisation framework. Route configurations and appropriate vehicle assignments for large-scale networks were returned by their approach, which featured a categorisation of stops, multiple classes of transit vehicles, hierarchy planning, system capacity (previous studies had largely ignored this), and route construction and frequency setting integration.

Recently, Jha *et al.* [88] used a GA to determine transit route sets and a particle swarm optimisation algorithm for setting the route frequencies. Their approach involved two phases. The first involved formulating a UTRP model and solving it by means of a GA, and secondly, formulating a UTFSP model and solving it in a multi-objective context by applying both the NSGA II and a multi-objective particle swarm optimisation algorithm, with the latter returning more favourable results.

4.8 Commercial software

The world's leading traffic planning software, Visum [134], was designed by the PTV Group to aid transport planners in their decision making. The software adopts a demand-driven and service-oriented approach towards transport planning. Various modes of public and private transport are supported, such as bus, rail and private vehicles. A user-friendly interface is provided for decision makers, and bus operators may use the software to design a transit network system simply by clicking on two terminals to establish a transit line. Visum then generates a route between these two terminals, and performance measures are available for transit network evaluation, both from the passengers' perspective and from the operator's perspective [134].

Emme [86] is a transportation forecasting software suite developed by INRO for planning the movement of people on an urban, regional and national scale, based on demand. An application programming interface is provided for the planner to extend the Emme framework, allowing for customised user interfaces to be constructed, along with planning and modelling facilities also being provided. Access to the underlying demand patterns is possible. Scenario comparisons may be performed and performance measures are displayed through a variety of graphical representations and maps.

SATURN [141] is a congested highway assignment software suite developed by Dr Dirck van Vliet at the Institute for Transport Studies at the University of Leeds, which also incorporates the practical knowledge of Atkins Limited. It facilitates fast and reliable highway assignments, one-of-a-kind junction representation functionality and detailed diagnostic information. This flexible highway assignment package is able to create both strategic, as well as local, traffic models. SATURN provides users with easy access to various forms of data that may be applied across a variety of platforms, including multimodal transportation software.

Cube Voyager [34], developed by Citilabs, is a transportation modelling software suite fit for urban, regional and long distance demand forecasting and assignment. The modular and script-based structure of the software allows for flexibility and custom applications to be produced through the Cube Voyager framework. Easy-to-use templates are available for simple trip-based modelling, including trip generation, trip distribution, mode choice and assignment. Advanced trip-based models may also be constructed by developers, with the software providing sufficient tools to utilise during this process.

The aforementioned software alternatives are popular in industry, and mainly serve the purpose of visualisation, simulation, and decision support [89]. None, however, is able to inherently design a new transit bus network automatically. Recently, Soares *et al.* [150] attempted to bridge this gap in optimisation algorithms and commercial software by creating an interface between the UTRP and the PTV Visum software. The interface manages the different data requirements of the two platforms and allows for optimisation capabilities of public transport lines in the context of PTV Visum network models. Their solution approach followed that of Ahmed *et al.* [2], while incorporating the same two objective functions utilised by Mumford [121], and John *et al.* [90], namely ATT and TRT.

A complete redesign of bus routes will cause a significant disruption to the system and to the passengers, and for this reason complete redesigns are typically of questionable practical value. Incremental changes or improvements to current transit systems would be more fitting in most cases, potentially benefiting industry significantly [89]. Ceder [29], however, argued that when only incremental changes are performed to individual routes, it may result in an overall confusing and inefficient transit system. He emphasised the difficulty associated with predicting the potential benefits that a complete redesign would hold, but that it is reasonable to expect the benefits to be significant, and that efforts should be aimed towards solving this problem instead of focusing on problematic scheduling activities.

4.9 Chapter summary

A thorough review of the literature pertaining to the UTNDP was carried out in this chapter. This review included a discussion in §4.1 on the UTNDP in general, focusing on its purpose, considerations required to comprehend the nature of the problem and to formulate a general mathematical model formulation for the UTNDP. Thereafter, various causes of the complexity of the UTNDP were reviewed in §4.2, contributing to the NP-hard status of the problem. The focus then shifted in §4.3 to the overall transit planning process and the different stages of planning that are required to establish a transit network. The main components of the UTNDP were identified and elaborated upon, namely transit network design, transit network frequency setting, transit network timetabling, vehicle scheduling, and crew scheduling and rostering. The discussion proceeded to the different classifications of subproblems of the UTNDP in §4.4.

Three of the subproblems were then considered in greater detail as they pertain specifically to this dissertation. These are the UTRP, the UTFSP, and the UTRFSP, discussed in §4.5, §4.6 and §4.7, respectively. The key characteristics of the UTRP were considered, namely its objectives, decision variables, network structure, demand patterns, demand characteristics and constraints. Appropriate solution methodologies for the UTRP were finally discussed. The main solution methodologies prevalent in the literature are exhaustive searches, mathematical programming techniques, heuristics, neighbourhood searches and evolutionary algorithmic approaches. The best results for five well-known UTRP benchmark instances found in the literature were also summarised. For the UTFSP, the objectives, decision variables, constraints, and the notion of transit assignment were discussed. Finally, solution approaches towards solving instances of the UTFSP were again considered. An emphasis was placed on solution approaches for solving the problem, and not on its characteristics, as many of the characteristics of the UTRP and the UTFSP are applicable to the UTRFSP as well.

The chapter closed in §4.8 with a short discussion on the available commercial software for solving transit-related design problems in the same vein as the UTNDP. None of the software suites reviewed, however, exhibits the capability of automatically designing a bus transit network

anew. Figure 4.7 contains a summary of all the aspects considered in this chapter and how each relates to the UTNDP.

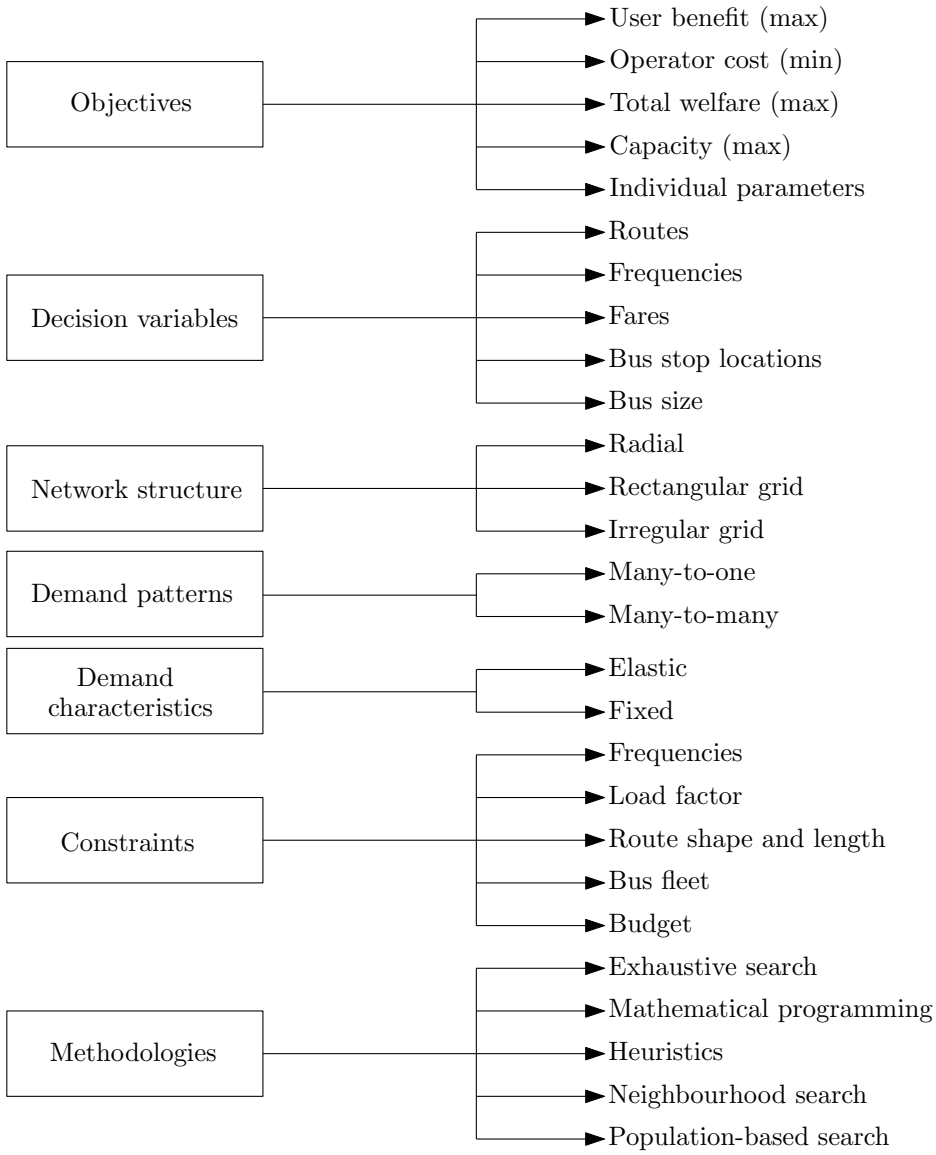


FIGURE 4.7: *Key characteristics of the UTNDP discussed in this chapter.*

Part II

Mathematical modelling

CHAPTER 5

Mathematical models

Contents

The purpose of this chapter is to document the two mathematical models adopted in this dissertation for solving the UTRP and the UTFSP, respectively. The models are inspired by the literature review of Chapter 4. The UTRP model is utilised during the design of the bus routes in a transit network, whereas the UTFSP model is utilised for the purpose of assigning frequencies to bus routes, in other words determining the number of buses that should operate along each individual bus route. The chapter opens with a mathematical formulation of the standard UTRP model in §5.1. It is proposed in §5.2 that one of the objectives of the UTRP be exchanged with a novel objective function which may be employed when designing transit route sets in such a manner that a degree of similarity with a reference route set is pursued. This is followed by a mathematical formulation of the standard UTFSP model in §5.3. The chapter closes in §5.4 with a brief summary of its contents.

5.1 The UTRP model

This section is devoted to a derivation of one of the mathematical models for the UTRP adopted in this dissertation. The main aspects incorporated into the model are illustrated in Figure 5.1. The figure contains a selection of the model aspects depicted in the more general Figure 4.7. The various aspects in Figure 5.1 are described in detail in this section, with the exception of the methodologies for solving the model, which are described in the next chapter. The model objectives are to minimise passenger cost and minimise operator cost simultaneously (both measured in units of time), with the decision variables representing routes selected for inclusion in the transit network. The assumed road network structure is an irregular grid, and the demand is assumed to be fixed over time and to exhibit a many-to-many pattern. The applicable constraints involve route shape, route length and vehicle stop specification, as well as a requirement that the resulting transit network should be connected. The section opens with a discussion on the assumptions underlying the model, after which the mathematical model derivation follows in terms of basic concepts from the realm of graph theory, as reviewed in §2.2.

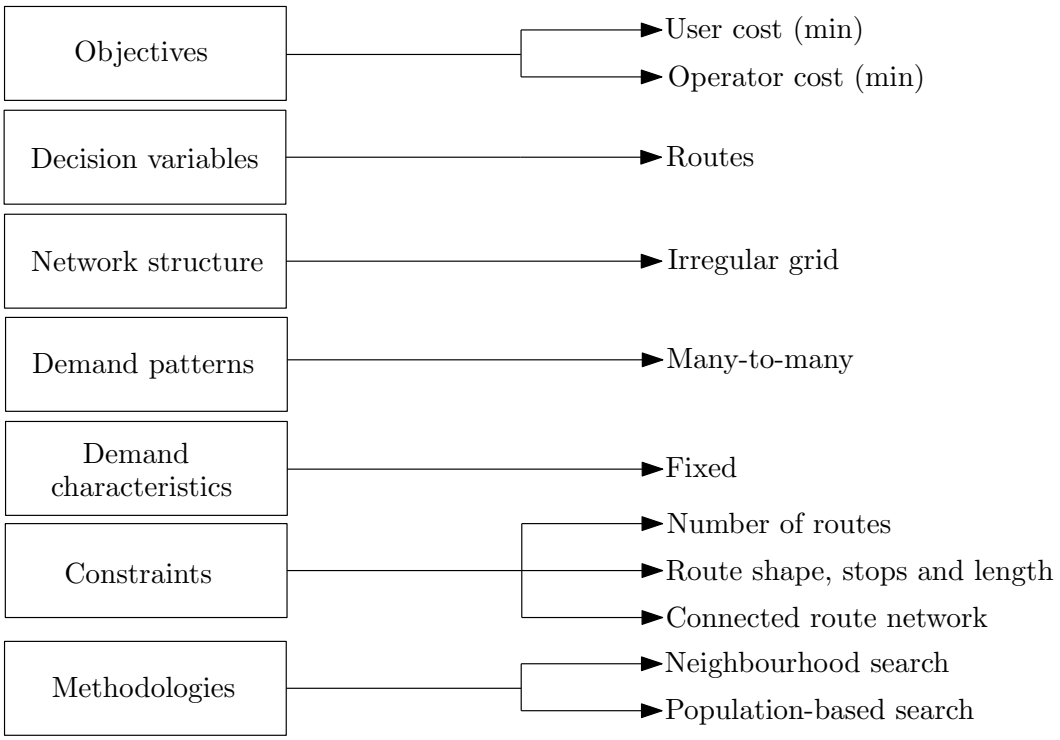


FIGURE 5.1: *UTRP modelling approach adopted in this dissertation.*

The following assumptions are made in order to construct a model for the UTRP:

1. There are sufficient buses for deployment along each route so that the OD demand between all pairs of stops can be satisfied if appropriate transit routes are available. The setting of vehicle frequencies along routes is therefore not considered in the model for the UTRP due to the additional complexities induced by this assignment task, as was also the assumption in [90, 121].

2. Each bus is assumed to traverse back and forth along its assigned route, reversing its direction each time a terminal vertex is reached [90].

3. The inconvenience caused to passengers when having to transfer from one bus (route) to another is represented by a fixed constant transfer penalty of 5 min for each transfer incurred. This value is in line with previous studies [32, 146], and corresponds to a fixed frequency of $\frac{1}{10}$ for all of the routes, in other words, one bus every ten minutes [89].

4. The passenger demand, expected travel time and distance matrices associated with the UTRP are assumed to be symmetric, as was also the assumption in [121].

5. The passenger demand exhibits a many-to-many pattern, but is fixed over time, as was assumed by Mumford [121].

6. The total travel time of a passenger comprises only in-vehicle travel time and transfer penalties, as assumed in [121].

7. Each passenger's route choice is assumed to be based on the shortest expected travel time, regardless of the number of transfers required. This assumption is in line with those adopted by Fan and Mumford [50] and by Mumford [121].

8. The bus stops have already been determined, as is the case in most UTRP instances, and the aim is only to design a bus route set servicing these stops. An OD demand matrix is specified in terms of these bus stops.

John *et al.* [90] proposed a mathematical model formulation for the UTRP based on graph theoretic concepts that is much more intuitive than earlier integer programming model formulations for the UTRP. His general modelling paradigm is adopted in this section.

Suppose the road network on which the UTRP must be solved is modelled by an edge-weighted graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of vertices representing the required bus stops and $\mathcal{E} = \{e_1, \ldots, e_m\}$ is a set of edges, each of which is incident with two vertices in $\mathcal{V}$. These edges represent shortest-time direct road links between pairs of vertices in G (*i.e.* not containing any intermediate bus stops). The weights of the edges of G are specified in the form of an $n \times n$ *weight matrix* $\boldsymbol{W}$ whose entry in row i and column j denotes the expected travel time along the direct road link represented by an edge joining v_i and v_j if such an edge is present in G, or ∞ otherwise [73]. An $n \times n$ *demand matrix* $\boldsymbol{D}$ is also associated with G, which contains as entry in row i and column j the passenger demand from vertex v_i to vertex v_j.

A *transit route* is defined as a simple path in G, represented by an ordered sequence of distinct vertices of G, each successive pair of which are adjacent in G. A transit route therefore contains no loops or repeating vertices. Let $G_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{E}_{\mathcal{R}})$ be the subgraph induced in G by the vertices in a set of transit routes $\mathcal{R}$. Such a set $\mathcal{R}$ of transit routes is considered an infeasible solution to the UTRP unless each vehicle stop is contained in at least one route $R \in \mathcal{R}$ [50, 90]. Stated mathematically, the transit route set $\mathcal{R}$ is considered infeasible unless

$$\bigcup_{R \in \mathcal{R}} \mathcal{V}_R = \mathcal{V}, \tag{5.1}$$

where $\mathcal{V}_R$ is defined as the vertices contained in R. A minimum and maximum number of vehicle stops is also specified per route in the transit route set $\mathcal{R}$ [50, 90]. That is, it is required that

$$m_{\min} \leq |\mathcal{V}_R| \leq m_{\max}, \qquad R \in \mathcal{R}. \tag{5.2}$$

These limiting values are based on various considerations such as driver fatigue and schedule adherence [182]. It is, of course, also required that the subgraph of G induced by the vertices of all the routes in a feasible set of transit routes be a connected graph [90]. That is,

$$G_{\mathcal{R}} = \left(\bigcup_{R \in \mathcal{R}} \mathcal{V}_R, \bigcup_{R \in \mathcal{R}} \mathcal{E}_R \right) \text{ must be connected.} \tag{5.3}$$

The cardinality of the required set of transit routes is finally also specified. That is, a requirement of the form

$$|\mathcal{R}| = r \tag{5.4}$$

is specified for some natural number r.

In general, passengers prefer to travel to their destinations in the shortest possible time while avoiding the inconvenience of having to make transfers if this is possible. The length of a shortest path for an OD pair $v_i, v_j \in \mathcal{V}$ along a transit route set $\mathcal{R}$, denoted by $\alpha_{v_i,v_j}(\mathcal{R})$, is measured in units of time and can be computed by applying Dijkstra's algorithm (described in §2.2.2) to the subgraph $G_{\mathcal{R}}$ of G. This shortest path duration includes both transport time and transfer time (if transfers between different routes in $\mathcal{R}$ are required). Mumford [121] measured the passenger

(time) cost associated with a route set $\mathcal{R}$ as the ATT for all passengers, measured in units of time and expressed as

$$\min F_1(\mathcal{R}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{D}_{v_i,v_j}\alpha_{v_i,v_j}(\mathcal{R})}{\sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{D}_{v_i,v_j}}, \tag{5.5}$$

where the numerator represents the total travel time of all passengers and the denominator represents the total number of passengers in the system.

Operator cost may include many facets, as discussed in §4.5. A simple proxy was, however, proposed by Mumford [121] for operator cost. This is the sum of the total travel time for a single transport vehicle along each route (in one direction). This *total route time* (TRT) may be expressed as

$$\min F_2(\mathcal{R}) = \sum_{R\in\mathcal{R}} \sum_{(v_i,v_j)\in R} \boldsymbol{W}_{v_i,v_j}. \tag{5.6}$$

When solving the above model for the UTRP, both objective functions (5.5)–(5.6) are minimised simultaneously in order to establish a Pareto front representing effective trade-offs between minimising passenger cost and minimising operator cost.

5.2 The UTRP model with an incremental redesign extension

As John [89] pointed out in his doctoral dissertation, a complete redesign of a bus transit system is likely to cause significant disruption for both passengers and the system itself. The practical value of a complete redesign is therefore questionable, and incremental changes or improvements to a current system may, in fact, prove more beneficial. A new objective function is proposed in order to bridge this gap in the literature, based on the similarity measure proposed by John [89] for measuring route set population diversity. His approach included measuring similarity between different route sets by utilising the Sørensen-Dice index, a common similarity measure in the literature, typically applied to discrete sets. Each route set is decomposed into a multi-set of all its edges, and the number of identical matches between the two sets is counted and multiplied by two, and then divided by the sum of the cardinality of both sets. This yields a similarity percentage between two route sets.

Consider, for example [89], the two route sets $\mathcal{R}_1 = \{\langle 1,2,3,4\rangle, \langle 5,2,6\rangle\}$ and $\mathcal{R}_2 = \{\langle 1,2,3,4\rangle, \langle 5,2,3,6\rangle\}$. For each route set, a multi-set is defined containing all the edges of each route present. These multi-sets are $E_{\mathcal{R}_1} = \{\{1,2\},\{2,3\},\{3,4\},\{5,2\},\{2,6\}\}$ and $E_{\mathcal{R}_2} = \{\{1,2\}, \{2,3\},\{3,4\},\{5,2\},\{2,3\},\{3,6\}\}$. An important multi-set feature is that the edge $\{2,3\}$ appears twice in $E_{\mathcal{R}_2}$. The Sørensen-Dice index, denoted here by S, for multi-sets $E_{\mathcal{R}_1}$ and $E_{\mathcal{R}_2}$ may be expressed as

$$S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2}) = \frac{2|E_{\mathcal{R}_1} \cap E_{\mathcal{R}_2}|}{|E_{\mathcal{R}_1}| + |E_{\mathcal{R}_2}|}.$$

Here, the intersection $E_{\mathcal{R}_1} \cap E_{\mathcal{R}_2}$ evaluates to $\{\{1,2\},\{2,3\},\{3,4\},\{5,2\}\}$, which has cardinality 4. The similarity measure $S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2})$ therefore evaluates to $\frac{8}{11}$. If, however, $E_{\mathcal{R}_1}$ were to include another instance of the edge $\{2,3\}$, the intersection of $E_{\mathcal{R}_1} \cap E_{\mathcal{R}_2}$ would be $\{\{1,2\},\{2,3\},\{3,4\},\{5,2\},\{2,3\}\}$, with cardinality 5, as two identical matches can be made with both instances of the edge $\{2,3\}$.

The difference between two route sets may then be quantified as $1 - S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2})$. A new objective function is proposed for the UTRP which incorporates this concept of route set difference, where a reference route set specified *a priori*, denoted by $\mathcal{R}_0$, is taken as the first argument of

the difference measure calculation, and another route set $\mathcal{R}$ is taken as the second argument. The objective may be expressed as

$$\min F_3(\mathcal{R}_0, \mathcal{R}) = 1 - \frac{2|E_{\mathcal{R}_0} \cap E_{\mathcal{R}}|}{|E_{\mathcal{R}_0}| + |E_{\mathcal{R}}|}. \tag{5.7}$$

This objective function, called the *disruption proportion* (DP), may be employed in the UTRP as one of the objectives that should be minimised, where the reference route set $\mathcal{R}_0$ represents a route set that is currently being maintained by an operator, and the (variable) route set $\mathcal{R}$ is a newly proposed route set. By minimising (5.5) and (5.7) simultaneously, and establishing a set on non-dominated trade-off solutions, where $\mathcal{R}$ represents the decision variables, a new UTRP model is established. These trade-off solutions would be useful to a decision maker as (s)he would be able to choose a new transit route set to be implemented according to the DP that (s)he is willing to tolerate, along with the potential gain in the ATT objective for passengers. This modelling approach may be adopted to improve transit networks incrementally without the need for a complete redesign. These improvements may be implemented in stages, with the reference route set being replaced by the previously proposed transit route set in the sequence of improvements, thus exhibiting limited disruption during each improvement stage.

This alternative model would therefore involve minimisation of the two objective functions (5.5) and (5.7) simultaneously, subject to the constraints in (5.1)–(5.4), and adhering to all the assumptions of §5.1.

5.3 The UTFSP model

This section is devoted to a derivation of the mathematical model for the UTFSP adopted in this dissertation. The main aspects incorporated into the model are illustrated in Figure 5.2, and are discussed in some detail in this section. As in the figure in §5.1, Figure 5.2 contains a selection of the model aspects depicted in the more general Figure 4.7. The model objectives are to minimise passenger cost (measured in units of time) and to minimise operator cost (measured in number of buses required) simultaneously, with the decision variables being the operating frequencies assigned to each of the respective routes in the given transit network. As in the model of the previous section, the assumed road network structure is an irregular grid, and the demand is assumed to be fixed over time and exhibit a many-to-many pattern. The constraints applicable to this model require that frequencies should be between pre-specified minimum and maximum frequency values.

The following assumptions are made in order to derive the UTFSP model:

1. There are sufficient buses for deployment along each route so that the OD demand between all pairs of stops can be satisfied if appropriate transport routes are available, as was also the assumption by John [89].

2. Each bus is assumed to traverse back and forth along its assigned route, as is the case in most UTFSP instances, reversing its direction each time a terminal vertex is reached [117].

3. The passenger demand and distance matrices associated with the UTFSP are assumed to be symmetric, as has also been the assumption in many UTFSP instances in the literature [89, 117].

4. The passenger demand is generated for the bus stops, instead of using centroid vertices for demand zones with connections to the bus stops, for the sake of simplicity [117].

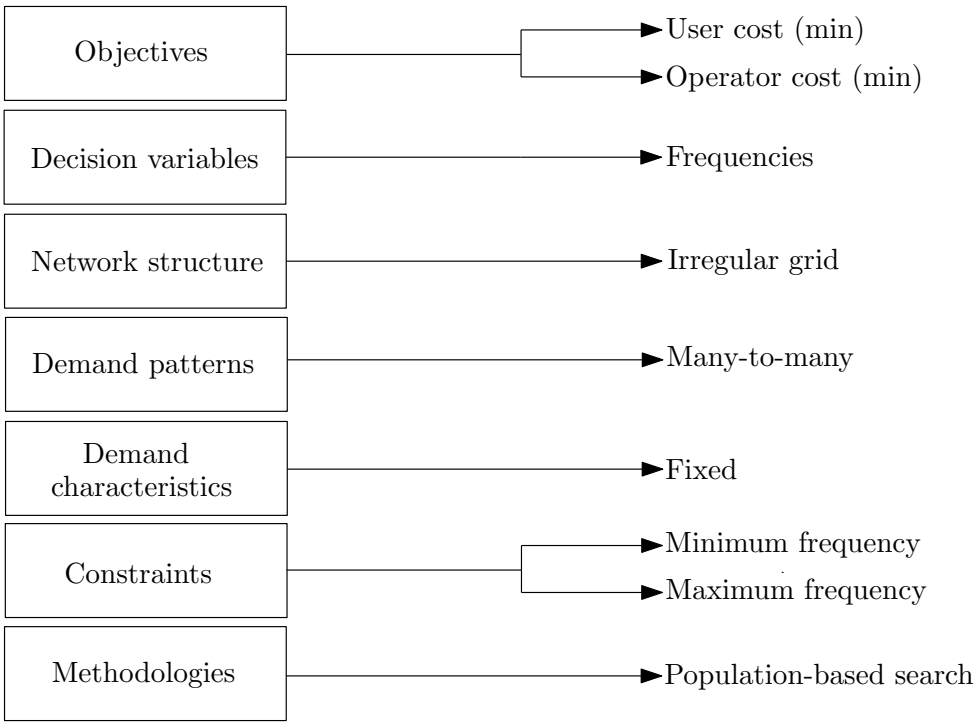


FIGURE 5.2: *UTFSP modelling approach adopted in this dissertation.*

5. The passenger demand exhibits a many-to-many pattern, but is fixed over time, as is usually assumed in UTFSP instances in the literature [82, 112].

6. The total travel time of a passenger comprises in-vehicle travel time, initial waiting time, and transfer waiting time [151], as assumed by John [89].

7. Each passenger's route choice is assumed to be based on the widely accepted optimal strategies transit assignment model proposed by Spiess and Florian [151], as was also assumed by Martinéz *et al.* [117]. This includes the assumption that a passenger takes the first bus that arrives which serves one of the strategies to reach his or her destination.

8. The option of walking (or utilising an alternative mode of transport) is included in the model, but at a cost factor of three times the cost that a bus would have incurred if it were to make a direct trip from the origin to the destination. The cost factor can be adjusted, but for small network instances, a value of three is assumed.

Frequency setting entails assigning the number of buses to operate along each of the bus routes in a given transit network. The number of buses required to operate along each route is determined by the headway. The frequency of a route is the inverse of headway, as discussed in §4.3.2. Furthermore, the frequencies that are set defines, to a large extent, the schedule of each bus, constituting the third phase of transit network design, as addressed in the UTTP. The aim of frequency setting is therefore to minimise the total waiting time of passengers, while simultaneously minimising the cost incurred by the operators to maintain the set frequencies.

The UTFSP is inherently multi-objective in nature, involving trade-offs between costs incurred by passengers and operators. For passengers, routes with high frequencies are beneficial, as this leads to more buses serving a route and hence shorter waiting times until the next bus is expected to arrive, therefore reducing the total times that passengers are expected to spend travelling

from their origins to their respective destinations. Another factor taken into consideration is the carrying capacity of buses. Having too few buses may lead to all passengers waiting for a bus not being able to board it when it arrives due to a large load factor on the route, resulting in customer dissatisfaction and inconvenience [117]. The operator cost, on the other hand, is incurred by the number of buses required to maintain the respective service frequencies, where more buses are indicative of a larger operational cost in terms of fuel, driver remuneration and maintenance costs [82].

Setting bus frequencies to appropriate levels is a complex task due to the real-world intricacies involved and the stochastic nature of passenger arrivals [89]. Modelling the performance of such a system therefore requires an appropriate modelling scheme for the behaviour of passengers, called a *transit assignment model*, as mentioned in Chapter 4. Transit assignment models usually involve complex formulations and accompanying solution methodologies, especially when considering the influence of bus capacity on the system [117]. Moreover, the complexity of modelling passenger behaviour largely determines the complexity associated with UTFSP models. Non-linearities arise due to the inversely proportional relationship between the waiting times of passengers and the frequencies applicable along bus routes, and also in modelling the various interactions between different routes.

In order to derive a model for the UTFSP, a distinction is made between the supply and demand components of the transit network. The supply component constitutes the bus routes, or *lines*, for which the itineraries are defined by the collection of bus stops and road segments, and each route's frequency as determined by the model. The demand component, on the other hand, constitutes the passengers having to undertake trips from their origin bus stops to their respective destination bus stops. Given a set of bus routes, the passengers therefore have to decide how they will perform those trips using the available routes [117].

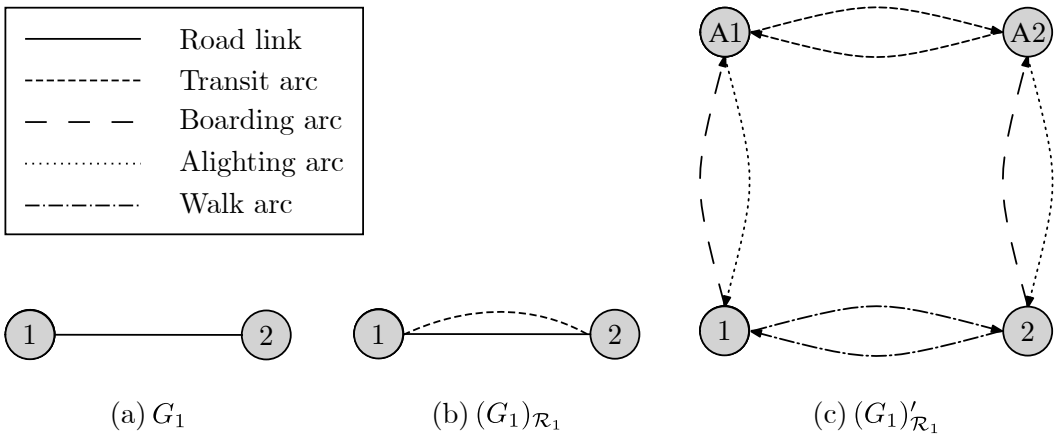
John [89] proposed an intuitive mathematical model formulation for the UTFSP, based on graph theoretic concepts, and his general modelling paradigm is adopted in this section.

Suppose again that the road network on which the UTFSP must be solved is modelled by an edge-weighted graph $G = (\mathcal{V}, \mathcal{E})$, as described in §5.1. Let $G_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{E}_{\mathcal{R}})$ be the subgraph induced in G by the vertices in a set of transit routes $\mathcal{R}$ produced as output by solving the UTRP. Such a set of transit routes, which serves as input to the UTFSP, is assumed to be feasible in terms of containing all the bus stops present in the road network, and induce a connected graph, corresponding to constraints (5.1) and (5.3), respectively. It is important to reiterate the relationship between the UTRP and the UTFSP: A route set $\mathcal{R}$ delivered as output from the UTRP serves as the input to the UTFSP, and this route set is typically not altered during the latter problem solution.

The *supply* side of the transit network is expressed by an expanded directed graph of the transit network, denoted by $G'_{\mathcal{R}} = (\mathcal{V}'_{\mathcal{R}}, \mathcal{E}'_{\mathcal{R}})$, where $\mathcal{V}'_{\mathcal{R}}$ includes all the vertices (bus stops) in the original transit network, as well as additional vertices corresponding to the transit route at a given bus stop. Whereas the set of vertices representing the bus stops is denoted by $\mathcal{V}$, the set of transit vertices is denoted by $\mathcal{V}_T$, with $\mathcal{V}'_{\mathcal{R}} = \mathcal{V} \cup \mathcal{V}_T$. The arcs between vertices in $\mathcal{V}_T$ are called *transit arcs*, and are denoted by $\mathcal{E}_T$. Transit arcs facilitate the movement of buses between bus stops (naturally requiring that a bus route should be available to service the line), and the cost (measured in minutes) incurred for this traversal from bus stop $i \in \mathcal{V}$ to bus stop $j \in \mathcal{V}$ is captured by the weight matrix entry $w_{ij} \in \boldsymbol{W}$. Furthermore, arcs directed from vertices in $\mathcal{V}$ to vertices in $\mathcal{V}_T$ are called *boarding arcs*, representing passengers boarding buses, and are denoted by $\mathcal{E}_B$. On the other hand, arcs directed from vertices in $\mathcal{V}_T$ to vertices in $\mathcal{V}$ are called *alighting arcs*, representing passengers alighting from buses, and are denoted by $\mathcal{E}_A$ [117]. *Walk arcs*, denoted by $\mathcal{E}_W$, are also included in the model, due to it possibly being a viable option

for passengers to walk in smaller transit networks, where distances between bus stops are short and frequencies on bus routes are low. The set of transit, boarding, alighting, and walk arcs are therefore subsets of $\mathcal{E}'_{\mathcal{R}} = \mathcal{E}_T \cup \mathcal{E}_B \cup \mathcal{E}_A \cup \mathcal{E}_W$.

Figure 5.3 illustrates the different types of aforementioned arcs in the context of a small example graph, G_1, consisting of only two vertices, 1 and 2. A bus route connecting vertices 1 and 2, denoted by $\mathcal{R}_1 = \{\langle 1, 2\rangle\}$, is applicable in this hypothetical case. Figure 5.3(a) contains the road network, corresponding to G_1, whereas Figure 5.3(b) contains the subgraph induced by the route $\mathcal{R}_1$, namely $(G_1)_{\mathcal{R}_1}$, in which the bus route is represented by a transit arc. Figure 5.3(c) contains the expanded transit network $(G_1)'_{\mathcal{R}_1}$, in which the route $\mathcal{R}_1$ is represented by vertices A1 and A2. The name of each vertex in a route can therefore be concatenated by a letter and the bus stop number to distinguish the routes from one another.



(a) G_1 (b) $(G_1)_{\mathcal{R}_1}$ (c) $(G_1)'_{\mathcal{R}_1}$

FIGURE 5.3: *An illustration of the different graph types applicable in transit networks, when, for example, considering the graph G_1 in (a). The road network is illustrated in (a), the transit network together with its bus route is illustrated in (b), and the expanded transit network is illustrated in (c).*

In essence, the entire transit network is modelled as a graph containing arcs, each characterised by a non-negative travel cost associated with it, measured in minutes, and also a service frequency applicable to it [151]. Boarding, alighting, walking and being in transit are therefore no longer components considered separately, but find their relationships with one another expressed within the expanded transit network graph $G'_{\mathcal{R}}$. Each arc in $\mathcal{E}'_{\mathcal{R}}$ has a cost-frequency pair associated with it. The cost associated with the transit arcs $\mathcal{E}_T$ are the on-board travel times between bus stops, as specified in the weight matrix $\boldsymbol{W}$ of §5.1, and the frequencies of these arcs are set to ∞, indicating that once a passenger is present on one of these arcs (in transit on a bus), there is immediate service.

The boarding arcs in $\mathcal{E}_B$ are assumed to exhibit a cost of six seconds, representing the time a passenger takes to board the bus, and the frequencies of these arcs are set to the route frequency f_i associated with route R_i, where i is the index of the route, indicating that a passenger has to wait until the boarding arc is served prior to boarding the bus. The frequencies $f_i \in \mathcal{F}$ assigned to each route $R_i \in \mathcal{R}$ are the decision variables of the model, and each frequency f_i can take as value a member in a predefined set θ.

Similar to the boarding arcs, the alighting arcs $\mathcal{E}_A$, are assumed to exhibit an alighting time of six seconds, representing the time a passenger takes to alight from a bus. The frequencies of these arcs are, however, set to ∞, representing the fact that once a bus arrives at a bus stop, a passenger can immediately alight and does not have to wait for service, as opposed to the case of boarding. The walking arcs $\mathcal{E}_W$ are not commonly present in UTFSP models, and therefore the assumption is made that a penalty factor of $3w_{ij}$ is incurred when a passenger chooses this

alternative as opposed to travelling by bus, when transitioning from bus stop i to bus stop j. This assumption is, however, only applicable to small transit networks, and should be adjusted when considering transit networks in which bus stops are located further away from one another. This feature could alternatively have been eradicated by assigning a very large number as a penalty factor, making the option of walking unattractive to passengers. Figure 5.4 illustrates how these cost-frequency pairs are assigned to each of the respective arc types mentioned thus far.

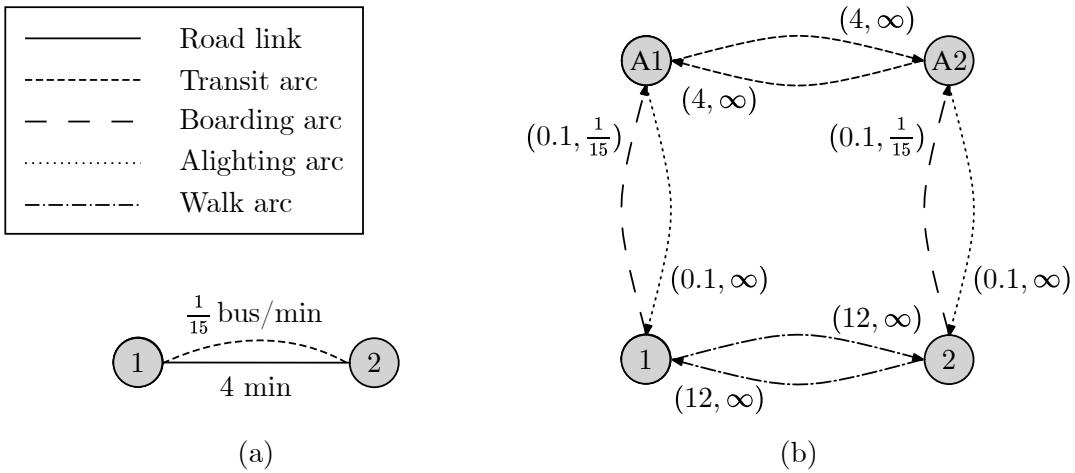


FIGURE 5.4: *An illustration of the cost-frequency pairs applicable to the expanded transit network when, for example, considering the graph G_1 of Figure 5.3, with the road link $(1, 2)$ having a travel cost of 4 min, and the corresponding bus route $\langle 1, 2\rangle$ having a frequency of one bus every 15 min. A transit network on this bus route is illustrated in (a), and the corresponding expanded transit network is illustrated in (b) with each arc having a cost-frequency pair associated with it, the first entry being the cost, and the second entry the frequency operating on that arc.*

The *demand* side represents the passenger's need to travel from his or her origin to the corresponding destination. An $n \times n$ *demand matrix* $\boldsymbol{D}$ is again associated with G for this purpose, which contains as entry in row i and column j the passenger demand between vertices v_i and v_j, measured in persons per minute, for a given time horizon. The demand matrix is defined only for vertices in the set $\mathcal{V}$, representing the bus stops, due to no demand existing to or from the set of transit vertices $\mathcal{V}_T$, which are instead utilised to satisfy the demand between bus stops.

In order to model the behaviour of passengers moving through the transit system, a transit assignment model is required [82]. The end result of such a model is the respective passenger flows through each arc, used during further calculations when solving the UTFSP. The *optimal strategies* transit assignment model proposed by Spiess and Florian [151] in 1989 is still widely used in the literature today [82, 117], and was accordingly chosen as an appropriate model for passenger behaviour in the UTFSP. This transit assignment model functions under the assumption that each passenger has a set of attractive bus lines that he or she could follow, depending on which line is served first by a bus, in order to arrive at the desired destination, thus minimising the customer's expected arrival time, and this set is called the *optimal strategy* for a passenger.

The core assumptions associated with transit assignment models are those that affect waiting time computation and passenger line selection [43]. For the sake of simplicity, congestion is not taken into account when modelling the UTFSP. Suppose a passenger is waiting for a bus along a set of routes $\mathcal{R}_i = \{R_1, \ldots, R_m\}$ that serves bus stop i and which also serves the destination of the passenger, or a transfer vertex *en route* to the destination, characterised by the respective bus frequencies $\mathcal{F}_i = \{f_1, \ldots, f_m\}$ for the corresponding routes in $\mathcal{R}_i$. The cost associated

with waiting time can then be estimated for each passenger as $E[W] = \phi / \sum_{R_j \in \mathcal{R}_i} f_j$, where ϕ is a parameter that depends on service regularity assumptions, with $\phi = 1$ corresponding to an exponential distribution of bus inter-arrival times, whereas $\phi \approx 0.5$ corresponds to an approximation of constant inter-arrival times [151].

When more than one route serves a bus stop i and also leads to a passenger's destination based on his or her optimal strategy, however, the volume assignment of passengers to routes are made on a so-called *frequency share* basis [43]. This entails determining a fraction of passengers desiring to travel from bus stop i to bus stop j served by route R', denoted by $P_{R'}$, being expressed as $P_{R'} = f_{R'} / \sum_{R_j \in \mathcal{R}_i} f_j$. The assumption that passengers take the first bus to arrive at their origin bus stop, serving the strategy of arriving at the corresponding destination in the shortest possible time, is a prerequisite for the aforementioned formulation [43, 151].

Once the transit assignment model has been used to assign passenger volumes to each of the arcs of the expanded transit graph, based on the optimal strategies of passengers, the total time that passengers are expected to spend travelling in the transit system can be calculated by multiplying the number of passengers moving along each arc by that arc's corresponding cost (measured in minutes), and summing all the products thus obtained over all the individual arcs. Adding the total travel time and the total waiting time together yields the total time spent by passengers in the transit system. Another way of expressing the expected time that a passenger is expected to spend travelling from origin i to destination j involves taking the proportions of passenger flows over each arc, and calculating the expected time a passenger would take to travel through the system for each OD pair, and then multiplying that expected travel time by the corresponding demand for each OD pair. The expected travel time constitutes the in-vehicle travel time, as well as the waiting time incurred for each passenger.

The objective functions adopted in this dissertation for the UTFSP model are based on the graph-theoretic formulation proposed by John [89], as mentioned. The first objective function, measuring the *average expected travel time* (AETT) for passengers in minutes, may be expressed as

$$\min F_4(\mathcal{R}, \mathcal{F}) = \frac{\sum_{i,j=1}^{n} \boldsymbol{D}_{v_i,v_j} u_{v_i,v_j}(\mathcal{R}, \mathcal{F})}{\sum_{i,j=1}^{n} \boldsymbol{D}_{v_i,v_j}}, \tag{5.8}$$

where $u_{v_i,v_j}(\mathcal{R}, \mathcal{F})$ denotes the expected travel time (measured in minutes) for a passenger travelling from origin i to destination j, calculated according to the optimal strategies transit assignment model described above [151]. It should be noted that $v_i, v_j \in \mathcal{V}$, and if all constraints are satisfied, the requirement $\mathcal{V} = \mathcal{V}_{\mathcal{R}}$ should hold. This objective should be minimised, representing the passengers' interest in minimising their costs, measured in time, when travelling from their origins to their destinations.

The operator's cost, on the other hand, is the total cost incurred by maintaining the service frequencies along each route. This is measured in terms of the *total buses required* (TBR) to maintain the frequencies along each route in the route set $\mathcal{R}$. The number of buses required for a route can be calculated by multiplying the desired frequency of the route by the total round trip time of the route. The round trip time is calculated by doubling the route duration (measured in minutes) [5]. The goal of minimising the TBR may therefore be expressed as

$$\min F_5(\mathcal{R}, \mathcal{F}) = 2 \sum_{\forall f_i \in \mathcal{F}} f_i \sum_{\forall e_j \in R_i} \boldsymbol{W}_{e_j}. \tag{5.9}$$

The only constraint applicable to this model occurs when the operator requires that the bus frequency along each route should be between minimum and maximum frequencies, denoted by

$f_{\min}$ and $f_{\max}$, respectively, expressed as

$$f_{\min} \leq f_i \leq f_{\max}, \qquad f_i \in \mathcal{F}. \tag{5.10}$$

When solving the above model for the UTFSP, both objective functions (5.8)–(5.9) are minimised simultaneously in order to establish a Pareto front representing effective trade-offs between minimising passenger costs and minimising operator cost.

It should be noted that, for the UTFSP, the route set $\mathcal{R}$ is kept constant, and that another instance of the UTFSP is generated by altering the route set $\mathcal{R}$. Another important remark is that the transit assignment model has to be resolved each time any of the frequencies change along any of the routes, due to the inter-connectedness of the various modelling objectives in the entire transit system.

5.4 Chapter summary

In this chapter, mathematical models were derived for solving two aspects of the UTNDP, one being for the UTRP, and the other the UTFSP. In particular, a bi-objective minimisation model was derived for solving the UTRP. The two objectives pursued in the model are simultaneous minimisation of the ATT and minimisation of the TRT, while the constraints involve a requirement that the route set should span all vehicle stops, that the graph induced by the route set should be connected, and that each route should contain no fewer than a minimum allowable number of stops and no more than a maximum allowable number of stops. This model was formulated mathematically from a graph theoretic perspective in §5.1.

A new approach aimed at facilitating incremental transit route set redesign was presented in §5.2, where the TRT objective function of the UTRP model of §5.1 was replaced by a DP objective function that would lead to trade-off solutions that exhibit varying degrees of similarity with respect to a reference route set.

A bi-objective minimisation model was also derived for solving the UTFSP. The two objectives pursued in this model are the simultaneous minimisation of the AETT and the TBR, while each frequency should be at least as large as a minimum required frequency, but no more than a maximum allowable frequency. Section 5.3 contained a formulation of this model, also derived from a graph theoretic perspective.

The solution methodologies adopted to solve each of the aforementioned models are elaborated upon in the next chapter.

CHAPTER 6

Approaches towards solving the models

Contents

This chapter is devoted to a description of the solution methodologies adopted in this dissertation to solve the models derived in the previous chapter. The UTRP model solution methodology is described in §6.1 and consists of an *initial candidate route set generation procedure* (ICRSGP), a *network analysis procedure* (NAP) and a metaheuristic search procedure. Two such search procedures are employed, one being a trajectory-based approach (the method of SA), and the other a population-based approach (a GA). Section 6.1 also contains some of the main components that form part of the search procedures for the UTRP, namely a number of *lower-level heuristics* (LLHs) and a hyperheuristic approach for managing these LLHs. Next, the UTFSP model solution methodology is described in §6.2 and consists of generating initial feasible solutions, applying an NAP, and executing a metaheuristic search procedure (a GA). The chapter closes in §6.3 with a brief summary of its contents.

6.1 UTRP model solution implementations

The UTRP solution methodology proposed by Fan and Machemehl [51, 53] consists of three main components, namely an ICRSGP, an NAP, and a metaheuristic search that combines the ICRSGP and NAP. This is also the solution methodology adopted in this dissertation. The two main classes of metaheuristics, (population-based) evolutionary algorithms and (trajectory-based) neighbourhood searches, were considered for implementation in this dissertation due to the complexity of the UTRP. Zhao and Gan [182], Zhao and Zeng [185], and Fan and Machemehl [53] have claimed that SA is the best-suited solution approach for the UTRP. They found that SA outperformed a genetic algorithm with respect to the quality of route sets achieved. SA was consequently selected as the solution method within the class of trajectory-based metaheuristics implemented in this dissertation due to its simplicity and the high-quality results that it reportedly achieves, providing near-optimal results within very reasonable time frames [51].

Population-based approaches are, however, more commonly adopted for solving instances of the UTRP than trajectory-based approaches, with GAs most frequently being applied as solution methodology for the UTRP, often leading to high-quality solutions [82, 89, 121]. Therefore,

a GA was chosen as solution approach for the UTRP within the class of population-based metaheuristics in this dissertation, thus paving the way for a comparison of the results returned by the method of SA and a GA. Furthermore, the combination of such a trajectory-based approach and population-based approach may potentially lead to higher-quality solutions than when only applying one of these metaheuristics individually [55, 185].

The specific variant of the method of SA adopted in this dissertation is the DBMOSA (described in general in §3.3.2), and for the GA, the NSGA II (described in general in §3.3.3). Each methodology is described in detail in this section in the context of the UTRP. The section opens in §6.1.1 with a brief discussion on a number of model implementation prerequisites, after which a detailed description follows in §6.1.2 of how an initial candidate route set is generated. The method of network analysis is then described briefly in §6.1.4, after which the section closes with an in-depth discussion in §6.1.7 and §6.1.8 on how the entire model solution methodology was implemented by the author in the computer language Python [162] for both the cases of DBMOSA and the NSGA II, respectively.

6.1.1 Model variable representation

A suitable computer representation of the decision variables of the UTRP model (5.1)–(5.6), namely transit route sets considered for implementation between a set of pre-specified bus stops, is required in order to facilitate the effective implementation in Python of the model and its associated solution methodology. The *list* data type in Python was employed for route representation. This data type has the ability to store a variety of Python objects, such as scalars, vectors or even other lists, making it an ideal dynamic data type for the purposes of this dissertation. The main implementation challenge arose from the fact that routes are not scalar values or vectors of fixed length, but rather lists that vary in length, based on the number of vertices they contain. This challenge is easily overcome by using lists as data types for decision variable representation. Figure 6.1 contains a graphical representation of how the transit route set $\mathcal{R} = \{R_1, R_2, R_3\}$ would, for example, be represented in this manner, where $R_1 = \langle 1, 4, 5, 7, 9, 8\rangle$, $R_2 = \langle 4, 2, 3\rangle$ and $R_3 = \langle 9, 7, 5, 6, 1\rangle$.

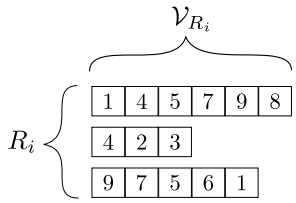


FIGURE 6.1: *Computer representation of a route set containing three routes.*

Furthermore, the convention is adopted whereby a route set is printed to a *csv* file in one line for the sake of brevity and for visualisation purposes. For example, the route set depicted in Figure 6.1 in the list format would be represented in the *csv* file in string format as 1-4-5-7-9-8*4-2-3*9-7-5-6-1*, where the dashes represent links between vertices and the asterisks represent the ends of routes. A function was created in Python that can convert route sets between these two formats. This makes for convenient representations of the route sets in computer memory and in file storage for further analysis.

6.1.2 The initial candidate route set generation procedure

The purpose of the ICRSGP is to generate an initial transit route set $\mathcal{R}$ that satisfies the constraints in (5.1)–(5.4). This feasible solution to the UTRP model (5.1)–(5.6) is then taken as the starting point for the NAP described in the next section. Four different methods for generating initial feasible solutions were considered for implementation and testing, of which three are novel. Thereafter, a novel method is proposed that enhances the quality of the routes generated by the ICRSGP, in terms of ATT or TRT.

Shortest paths ICRSGP

The first of the four ICRSGPs, called the *shortest paths ICRSGP* (SP-ICRSGP), generates a set of candidate routes $\mathcal{C}$ according to Algorithm 6.1, populated with shortest-time paths between pairs of vertices in G, each containing at least $m_{\min}$ vertices and at most $m_{\max}$ vertices. Dijkstra's algorithm, described in §2.2.2, is iteratively invoked for this purpose, taking each vertex of G as starting vertex in turn, so as to produce a complete set $\mathcal{S}$ of shortest paths in G. Since $\mathcal{S}$ may, however, contain paths with fewer than $m_{\min}$ vertices or more than $m_{\max}$ vertices, the entire route set generated by Dijkstra's algorithm is filtered by removing routes that are too short or too long, so that each resulting set indeed satisfies constraint (5.2). This process is described in pseudo-code form in Algorithm 6.1.

Algorithm 6.1: Generate all candidate shortest paths

Input : An $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, the minimum allowed number of vertices in a route $m_{\min}$, and the maximum allowed number of vertices in a route $m_{\max}$.

Output: A shortest paths candidate route set $\mathcal{C}$.

1 $\mathcal{S} \leftarrow$ Generate a list of shortest paths between all pairs of vertices in G using Dijkstra's algorithm
2 **for** $i \leftarrow |\mathcal{S}|$ **to** 1 **do**
3 **if** $|V_{S_i}| < m_{\min}$ *or* $|V_{S_i}| > m_{\max}$ **then**
4 $S_i \leftarrow \emptyset$
5 $\mathcal{C} \leftarrow \mathcal{S}$
6 **return** $[\mathcal{C}]$

Algorithm 6.2 is then used to generate a feasible transit route set $\mathcal{R}$ satisfying all the constraints (5.1)–(5.4). The latter algorithm takes as input the output produced by Algorithm 6.1. Algorithm 6.2 generates a random sample of r routes from the candidate route set $\mathcal{C}$ [51]. If these r routes together satisfy the graph spanning requirement in (5.1) and the connectivity requirement in (5.3), then they form the initial transit route set $\mathcal{R}$. If not, another r routes are randomly sampled form $\mathcal{C}$, and the process is repeated until a feasible initial solution $\mathcal{R}$ has been obtained for the UTRP model (5.1)–(5.6).

It is sometimes the case that a feasible solution $\mathcal{R}$ cannot be obtained during the random sampling process described above due to an overly stringent specification of a combination of values $m_{\min}$, $m_{\max}$ and r that is incompatible with feasible solutions. For this reason, if no feasible solution has been found after i iterations of the while-loop spanning Steps 4–6 of Algorithm 6.2, constraint (5.4) is relaxed by increasing r by one and repeating the process until a feasible solution is found.

Algorithm 6.2: Generate feasible route set

Input : A candidate route set $\mathcal{C}$, the number of allowed routes r, and the number of allowed iterations i.
Output: A feasible set of routes $\mathcal{R}$.

```
1  R ← r random routes from C
2  while R is not feasible do
3      j ← 1
4      while j < i and R is not feasible do
5          R ← r random routes from C
6          j ← j + 1
7      if R is feasible then
8          return [R]
9      else
10         r ← r + 1
```

K shortest paths ICRSGP

The second method for generating initial feasible candidate route sets entails the use of K shortest paths, instead of only the shortest paths as in the SP-ICRSGP. This alternative method is called the *K shortest paths ICRSGP* (KSP-ICRSGP). The use of K shortest paths has widely been adopted in the literature in various ways in the context of generating initial feasible solutions for the UTRP [53, 89, 146]. One of the interesting features of the UTRP is that shortest paths alone are not always sufficient, as there is an underlying demand component which is not always adequately accounted for.

The KSP-ICRSGP is designed to take both demand covered and short travel times into account. Algorithm 6.3 is a pseudo-code description of generating all candidate K shortest paths, and then selecting only the best ℓ of these K shortest paths based on first filtering on the demand covered, and secondly filtering on the route length, representing travel time for passengers. This approach aims to minimise the average travel time for passengers not just by simply taking the shortest path, but by determining other paths which are also short, but potentially cover more demand.

Algorithm 6.3 takes the same input as that of Algorithm 6.1, with the addition of the specified number K of shortest paths to be generated, as well as the best ℓ of these routes to be inserted into the candidate solution set $\mathcal{C}$. The algorithm is initialised by invoking Yen's K shortest path finding algorithm (described in §2.2.2). In this dissertation, the value of $K = 50$ was selected in order to ensure a large enough pool of routes from which to choose in terms of satisfying the most demand. The output of Line 1 is assigned to the set $\mathcal{S}$. Next, in Lines 2–4, the routes in $\mathcal{S}$ which violate the minimum or maximum vertex length constraint are removed. A filter is thereafter applied to the set $\mathcal{S}$ for each source-target pair i–j, so as to reduce the initial 50 shortest paths to ℓ shortest paths. Each source-target pair of K shortest paths are filtered first in descending order of the total demand satisfied by utilising that route alone in the UTRP context, and filtered secondly in ascending order of total route length. The top ℓ routes are then selected for inclusion in the candidate K shortest paths set $\mathcal{C}$. In the literature, Shih and Mahmassani [146] argued that 10 shortest paths are sufficient, as any more than this is expected to violate in-vehicle travel time constraints. This may, however, only be true for smaller networks since large networks typically have many shortest paths achieving very similar TRT values.

Algorithm 6.3: Generate all candidate K shortest paths

Input : An $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, the minimum allowed number of vertices in a route $m_{\min}$, the maximum allowed number of vertices in a route $m_{\max}$, the number of shortest paths K to generate and the number of source-target pairs to retain ℓ.

Output: A K shortest paths candidate route set $\mathcal{C}$.

```
S ← Generate a list of K shortest paths between all pairs of vertices in G using Yen's K shortest path algorithm
for i ← |S| to 1 do
    if |V_{S_i}| < m_min or |V_{S_i}| > m_max then
        S_i ← ∅
foreach source-target pair i–j do
    Filter the demand serviced in descending order first, and filter the route length in ascending order second
    Select the top ℓ number of i–j source-target pairs and insert into C
return [C]
```

Thereafter, Algorithm 6.2 is invoked by providing as input the candidate K shortest paths set $\mathcal{C}$, and returning a feasible route set $\mathcal{R}$. For larger problem instances, the generation of 50 shortest paths for each source-target pair becomes quite time-consuming, and therefore the K shortest paths are saved in a *csv* file which may be loaded when solving a problem instance.

Shortest paths maximising missing vertices ICRSGP

During some preliminary testing of the SP-ICRSGP and the KSP-ICRSGP, it was found that for larger instances of the UTRP it is difficult to find initial feasible route sets when these methods are adopted. This is due to no certainty of including all the vertices in the network by randomly selecting routes from a candidate routes list $\mathcal{C}$, hence violating constraint (5.1). This led to a novel approach towards generating initial feasible solutions that leads to consistently finding feasible solutions for instances of various sizes rather quickly. This approach was inspired by Mumford's [121] method of crossover of 2013. The approach takes as input two parent feasible route sets to the UTRP and returns a feasible offspring solution. Approximately half of the routes of each selected parent are used to produce one offspring solution $\mathcal{R}'$. In particular, one route is selected from each parent in an alternating fashion and included in the offspring route set, until $|\mathcal{R}'| = r$. Routes are favoured if their inclusion in the offspring lead to a larger proportion of missing vertices being included. Measures are put in place to ensure that the routes included remain connected.

More specifically, the offspring $\mathcal{R}'$ is seeded by randomly selecting one route from a parent solution [121]. Next, a route from the second parent is selected. Connectivity is ensured by only considering routes from the second parent which have at least one vertex in common with the current offspring solution $\mathcal{R}'$. This requirement can be expressed as $\mathcal{V}_{R_i} \cap \mathcal{V}_{\mathcal{R}'} \neq \emptyset$ where $\mathcal{V}_{\mathcal{R}'}$ is defined as

$$\mathcal{V}_{\mathcal{R}'} = \bigcup_{R_j \in \mathcal{R}'} \mathcal{V}_{R_j} \tag{6.1}$$

and R_j is the j-th route from the second parent that is eligible for inclusion [89]. Set union operations can be performed succinctly for the *set* data type present in the Python programming language, where the vertices of routes are taken as the elements of sets.

The proportion of missing vertices from $\mathcal{R}'$ contained in R_i is determined for each eligible route R_i, and can be expressed as

$$\frac{|\mathcal{V}_{R_i} \setminus \mathcal{V}_{\mathcal{R}'}|}{|R_i|}, \tag{6.2}$$

where $\setminus$ refers to the set-theoretic difference operation, which can also be succinctly performed in Python for sets. Consider, for example, a current offspring solution $\mathcal{R}' = \{\langle 1, 2, 4, 5, 6\rangle\}$ and a route $R_i = \langle 3, 4, 7, 8, 9, 11\rangle$ being considered for subsequent inclusion. The evaluation of $\mathcal{V}_{R_i} \setminus \mathcal{V}_{\mathcal{R}'}$ then yields $\{3, 7, 8, 9, 11\}$, and so the expression in (6.2) renders the value $\frac{5}{6}$ [89]. An eligible route R_i that achieves the maximum proportion of missing vertices is chosen to be inserted into the offspring solution. If more than one route achieves this maximum value, a uniform random choice is made between them for insertion [121]. Attention is then reverted to the first parent, and a similar process is followed to augment the routes in $\mathcal{R}'$, until $|\mathcal{R}'| = r$. A pseudo-code description of the aforementioned crossover procedure is presented in Algorithm 6.4.

Algorithm 6.4: Route generation by maximising missing vertices procedure

Input : Two sets of routes, $\mathcal{R}_1$ and $\mathcal{R}_2$, and the number of routes allowed in a route set r.
Output: A feasible set of routes $\mathcal{R}'$.

```
1  P ← list(R1,R2)
2  b ← random_boolean()
3  R' ← random route from P[b] and remove route from P[b]
4  b ← switch(b)
5  while |R'| ≤ r do
6      forall R_i in P[b] if V_{R_i} ∩ V_{R'} ≠ ∅ do
7          p_i ← |V_{R_i} \ V_{R'}| / |R_i|
8      R_t ← R_i with maximum p_i from P[b]
9      R' ← R' ∩ {R_t} and remove R_t from P[b]
10     b ← switch(b)
11 if R' is feasible then
12     return R'
13 else
14     return false
```

The algorithm takes as input two sets of routes, $\mathcal{R}_1$ and $\mathcal{R}_2$, as well as the number of routes that are required in a route set r. In Line 1 of the algorithm, the two route sets presented as input are inserted into a parent list $\mathcal{P}$. This is done so that the alternating sequence may be performed more conveniently. Next, a random boolean value b is generated for use as an indexing variable to access the parent list $\mathcal{P}$, where 0 represents the first position of the list's entries, and 1 the second. The feasible route set may then be constructed by choosing a random route from the route set list found at the b-th location of $\mathcal{P}$ for first inclusion in the feasible route set placeholder $\mathcal{R}'$. The route is then removed from the associated parent list. The boolean value b is next toggled by the `switch` function, thereby changing a zero to a one, or a one to a zero.

Lines 5–10 span the procedure of selecting routes from alternative parents that maximises the proportion of unseen vertices included as described above. The while condition governs the

inclusion of more routes into the route set $\mathcal{R}'$ and terminates once the number of routes in $\mathcal{R}'$ reaches r. Line 6 is a conditional statement ensuring that each route R_i considered within $\mathcal{P}[b]$ should have one at least one common vertex within $\mathcal{R}'$. For all the eligible routes R_i, the proportion of missing vertices is calculated as p_i. The route R_i with the maximum p_i-value is selected and stored in the temporary route placeholder R_t. When similar proportions of missing vertex inclusions are present among the routes in the list $\mathcal{P}[b]$, one of them is randomly selected to be R_t. R_t is then included in the route set $\mathcal{R}'$ and removed from the list $\mathcal{P}[b]$. Thereafter, the boolean value b is switched in Line 10. If r routes are present in $\mathcal{R}'$, $\mathcal{R}'$ adheres to all feasibility tests, it is returned as output; otherwise, the value of `false` is returned.

It is possible that not all route sets constructed according to this procedure may be feasible, and in that case, a *repair strategy* is required to rectify the route set $\mathcal{R}$. The repair strategy proposed by Mumford [121] is adopted for this purpose, during which attempts are made to include the missing vertices from $\mathcal{V}_{\mathcal{R}}$ with respect to $\mathcal{V}$. The missing vertices can be joined to an adjacent terminal vertex, if such a relationship exists in the transit network [121]. More specifically, this can be achieved by first identifying all the missing vertices, and then selecting each one in a random order for evaluation and possible insertion. Thereupon, all the terminal vertices are identified and it has been determined whether any adjacency relationship exists between a missing vertex and a terminal vertex. If such a relationship exists, the missing vertex is added to the corresponding terminal vertex in the relevant route. If more than one such relationship exists, a terminal vertex is randomly selected and joined to the missing vertex.

This crossover method proposed by Mumford [121] was used to create feasible route sets where, instead of taking two feasible parent route sets as input, two identical lists of all the shortest paths for the network are presented as input to Algorithm 6.4. This method is called the *shortest paths maximising missing vertices ICRSGP* (SP-MMV-ICRSGP).

K shortest paths maximising missing vertices ICRSGP

The SP-MMV-ICRSGP succeeded in generating feasible solutions for larger instances with ease, but quality improvements may still be achieved by considering the demand component of the UTRP during the generation of initial candidate solutions, following the same reasoning as for the KSP-ICRSGP. The SP-MMV-ICRSGP can easily be extended to use the K shortest paths candidate routes $\mathcal{C}$ yielded by Algorithm 6.3 for both route sets $\mathcal{R}_1$ and $\mathcal{R}_2$ and provide this as input to Algorithm 6.4. This method is called the *K shortest paths maximising missing vertices ICRSGP* (KSP-MMV-ICRSGP). Its added benefit over the SP-MMV-ICRSGP is that larger demand proportions are catered for by utilising those candidate routes returned by Algorithm 6.3 filtered according to the identification of shortest paths achieving the highest demand.

6.1.3 Initial candidate route set enhancement procedure

When comparing the quality of the route sets returned by the aforementioned ICRSGP with the benchmark solutions, it seemed that the objective function values were relatively more concentrated in certain regions in objective space. It is, however, beneficial to obtain a good spread of solutions in objective space in terms of enhancing the ensuing effectiveness of population-based search procedures. This may be achieved by enhancing the route sets returned by the ICRSGP according to a novel approach described in some detail in this section. The basic underlying principle is that when vertices are added to routes, the ATT decreases while increasing the TRT. Similarly, when vertices are removed, the ATT increases and the TRT decreases. Randomly adding vertices to or removing vertices from routes may, however, lead to sub-optimal

choices. Smarter ways of altering routes are therefore required to help guide an enhancement procedure towards finding higher quality solutions which also achieve a good spread in objective space.

Two low-level modular operations are defined for this purpose, called the *cost-based grow* and the *cost-based trim* procedures. These procedures operate on the level of only one vertex by taking into account the demand per route length. The procedures may be applied a random number of times to route sets in order to improve their quality in either the ATT objective or the TRT objective. Finally, these procedures may be applied to route sets until a model constraint is violated, giving more preference to a particular objective at the cost of the other.

Cost-based trim

The term *cost-based trim* alludes to the fact that for every vertex removed from a route, a certain demand per cost trade-off occurs. The reduction in TRT due to an edge no longer being present in a route, for instance, comes at the cost that certain direct routes no longer exist for passengers if no other alternatives are available, which increases the ATT. This cost trade-off is quantified by measuring the total direct demand satisfied by a route, and establishing a ratio in terms of the travel time cost, either by dividing demand by cost, or cost by demand. When a route has to be reduced in length, it would make sense to remove a vertex that exhibits the largest cost per unit of demand. When demand is measured in the case of adding a new vertex to a route, however, every combination of the current route's vertices and the new vertex should be considered as this additional vertex provides a direct route to all other vertices contained in the current route under consideration. An assumption is made that only direct routes are considered as they carry the most weight in the ATT calculation by avoiding transfer penalties, and in the end may be considered a proxy for identifying superior routes, relative to one another.

Figure 6.2 illustrates this concept, where the straight lines represent the road network or edges, and the curved lines represent the demand relationships exhibited by passengers. Figure 6.2(a) contains three vertices connected by two edges, with six demand combinations, taking into account that a demand relationship occurs in both directions. Figure 6.2(b) contains an illustration of the situation where the red vertex is added to the route, and the dashed lines represent the demand relationship lines added. Three additional demand relationship lines are added, therefore bringing the total of demand combinations to 12. If only one route is considered, the calculation of the total direct demand satisfied by adding one additional vertex may be performed by Algorithm 6.5. The algorithm simply steps through each combination of vertices in a route R, denoted by $\mathcal{V}(R)$, and cumulatively adds the demand matrix entries $\boldsymbol{D}$ corresponding to the vertex pair combinations to the value of d, finally returning the value of d as output.

(a) (b)

FIGURE 6.2: *Graphical visualisation of the effect of meeting direct demand by adding a vertex to a route. The straight lines represent the road network and the curved lines represent the demand. In (a), a demand link is made to each other vertex in the route, while in (b), the effect of the inclusion of the red vertex is illustrated with respect to the direct demand met by all the other vertices, represented by the dashed lines.*

Routes, however, function in conjunction with one another, forming interrelationships and connections. The effect of adding one vertex to a route set should therefore be measured accordingly.

Algorithm 6.5: Calculate cumulative direct demand for a route

Input : A single route R, and an $n \times n$ demand matrix $\boldsymbol{D}$.
Output: The direct demand satisfied d by the route R.

```
1 d ← 0
2 for i in V(R) do
3     for j in V(R) do
4         if i < j then
5             d ← d + D[i, j] + D[j, i]
6 return d
```

This may be achieved by considering the demand of all distinct combinations of vertex pairs that have direct route connections between them. Such an approach is proposed in pseudo-code form in Algorithm 6.6. The algorithm takes as input a route set $\mathcal{R}$ and an $n \times n$ demand matrix $\boldsymbol{D}$. It is initialised by setting the cumulative direct demand satisfied to zero, denoted by d. A list c is also initialised to contain all combinations of distinct vertex pairs having direct connections. Next, the vertex pair combinations are extracted for all the vertices of a route R, denoted by $\mathcal{V}(R)$. This is done for each route R in the route set $\mathcal{R}$ and stored in the list c. In Line 5, all duplicate combinations are removed from the list c, if any are present. The cumulative demand for each vertex pair combination is then summed up, captured in the value d, and returned as output.

Algorithm 6.6: Calculate cumulative direct demand for a route set

Input : A route set $\mathcal{R}$, and an $n \times n$ demand matrix $\boldsymbol{D}$.
Output: The direct demand satisfied d by the route set $\mathcal{R}$.

```
1 d ← 0
2 c ← list()
3 for R in R do
4     Insert all (i, j) vertex pair combinations of V(R) into c
5 Remove any duplicate combinations from c
6 foreach (i, j)-pair in c do
7     d ← d + D[i, j]
8 return d
```

Algorithm 6.6 facilitates evaluation of the direct demand met by a route set $\mathcal{R}$. This is a required tool if comparisons of relative qualities between two different route sets are to be made, especially when measuring the effect of removing vertices from different terminal positions in order to find smart deletions. The cost-based trim operator is presented in Algorithm 6.7. The working of the algorithm is based on the basic notion that a terminal vertex is stochastically removed based on a probability that is proportional to the cost per unit of demand. This probability is calculated by dividing the edge weight of the removed edge by the demand no longer being satisfied because of removal of the edge.

Algorithm 6.7 takes as input a route set $\mathcal{R}$, an index i used to select the route within the route set $\mathcal{R}$ that should be trimmed, and an $n \times n$ weight matrix $\boldsymbol{W}$, as well as an $n \times n$ demand matrix $\boldsymbol{D}$. The algorithm is initialised by calculating the cumulative demand met by direct routes. This is achieved by invoking Algorithm 6.6, which is called by the command `calc_cum_demand_route_set`. This value is used as a reference to compare with the cumulative

Algorithm 6.7: Cost-based trim

Input : A route set $\mathcal{R}$, the index i of a route in $\mathcal{R}$ to be evaluated, an $n \times n$ weight matrix $\boldsymbol{W}$, and an $n \times n$ demand matrix $\boldsymbol{D}$.
Output: A trimmed route set $\mathcal{R}'$.

```
1  d_0 ← calc_cum_demand_route_set(R, D) [Algorithm 6.6]
2  j ← 1
3  R_t ← R after removing the first vertex from route R[i]
4  if test_feasibility(R_t) then
5  |  d_j ← d_0− calc_cum_demand_route_set(R_t, D)
6  |  c_j ← W[R[i][0], R_t[i][0]]
7  |  if d_j = 0 then
8  |  |  d_j ← 0.001
9  |  b_j ← c_j/d_j
10 |  a_j ← true
11 |  j ← j + 1
12 R_t ← R after removing the last vertex from route R[i]
13 if test_feasibility(R_t) then
14 |  d_j ← d_0− calc_cum_demand_route_set(R_t, D)
15 |  c_j ← W[R[i][−1], R_t[i][−1]]
16 |  if d_j = 0 then
17 |  |  d_j ← 0.001
18 |  b_j ← c_j/d_j
19 |  a_j ← false
20 |  j ← j + 1
21 ℓ ← j − 1
22 if ℓ = 0 then
23 |  return R
24 for j = 1 to ℓ do
25 |  p_j = b_j / Σ_{k=1}^{ℓ} b_k
26 Randomly choose a trim candidate j based on the the probabilities p_j
27 if a_j then
28 |  R' ← R after removing the first vertex from route R[i]
29 else
30 |  R' ← R after removing the last vertex from route R[i]
31 return R'
```

demand met when terminal vertices are removed from the route set $\mathcal{R}$. Furthermore, a counter j is initialised to 0 for tracking the number of candidates that have been included, from which one trim is selected to be performed in Line 26. Lines 3–11 span the evaluation of removing the first vertex of route $\mathcal{R}[i]$, and Lines 12–20 represent the evaluation of removing the last vertex from the route $\mathcal{R}[i]$. In Line 3, the first vertex is removed from route i of the route set $\mathcal{R}$, and this new route set is stored in the temporary route set placeholder $\mathcal{R}_t$. Next, a test for feasibility is performed on the route set $\mathcal{R}_t$ in Line 4 to ensure that none of the constraints (5.1)–(5.4) is violated when forming the route set.

The demand no longer satisfied upon removal of the vertex is evaluated in Line 5, where the cumulative demand calculation for route set $\mathcal{R}_t$ is subtracted from the initial demand d_0, and stored in d_j. The edge weight, or travel time cost, between vertex $\mathcal{R}[i][0]$ and vertex $\mathcal{R}_t[i][0]$ is stored in c_j. The zero here indicates the first position of the list representing the i-th route in the route set. There are, however, certain cases where the deletion of a vertex will not lead to any change in the total direct demand met, therefore potentially leading by division by zero. Line 7 avoids this case by assigning a small value of 0.001 to d_j. This is done because in Line 18, the edge cost c_j is divided by the demand contribution d_j, and the result is assigned to b_j. Later in the algorithm larger b_j-values are preferred for deletion as they contribute the least to high-quality solutions. Next, the boolean value `true` is assigned to a_j, indicating that the vertex is removed from the front of route $\mathcal{R}[i]$, upon which the counter j is incremented by 1.

After evaluation of the first vertex in route $\mathcal{R}[i]$, the last vertex is evaluated in a similar fashion in Lines 12–20, with the exception that instead of removing a vertex from the front of the route, a vertex is removed from the rear. Also, a_j is assigned the boolean value `false`, indicating that the vertex has not been removed from the front. The -1 index in Line 15 denotes the first position from the back of the list representing the route. In Line 21, ℓ is assigned as value the total number of candidate trims identified. If $\ell = 0$, the original route set $\mathcal{R}$ is returned as output.

A probability is associated with applying candidate trim j, denoted by p_j in Line 25, and is computed as the cost per demand value b_j divided by the sum of all the b_j-values for all values of j. Trim j is then randomly selected based on the probability p_j, and applied by considering the corresponding a_j-value. If a_j is `true`, the first vertex is removed from $\mathcal{R}[i]$, else its last vertex is removed. The algorithm terminates, and the trimmed route set $\mathcal{R}'$ is returned as output.

Cost-based grow

Whereas cost-based trim focused on removing vertices, cost-based grow focuses on adding vertices, and may be seen as the former operation's counterpart. Its implementation is, however, slightly more complex as not only the potential deletion of each terminal vertex is evaluated. Instead, the insertion of each possible feasible adjacent vertex is evaluated at both terminals of the route. A description of this procedure is presented in Algorithm 6.8. The algorithm takes as input all the same inputs as Algorithm 6.7, with the addition of the maximum number of vertices that may be included in a route, denoted by $m_{\max}$. As in the case of cost-based trim, the algorithm is initialised by calculating the cumulative demand for the route set $\mathcal{R}$ by invoking Algorithm 6.6, to be taken as the reference demand. Furthermore, the candidate counter j is assigned the value of 1, and the route set $\mathcal{R}$ is assigned to the temporary placeholder $\mathcal{R}_t$.

A feasibility test is performed in Line 4 to ensure that there is capacity in the route $\mathcal{R}_t[i]$ for more vertices to be inserted, by ensuring the number of vertices contained in $\mathcal{R}_t[i]$ is less than the maximum number of vertices $m_{\max}$ that may be contained in a route. If there is capacity, the analysis continues in Line 5, where each vertex h adjacent to the first vertex in route $\mathcal{R}_t[i]$ not already contained in the route itself is identified. In Line 6, the adjacent vertex h is inserted into the front of route $\mathcal{R}_t[i]$, and the demand contribution is subsequently evaluated by determining the cumulative demand of the route set $\mathcal{R}_t$ and subtracting from it the initial route set's cumulative demand met. This difference is assigned to d_j. Next, the cost of the added edge is accounted for by assigning this cost to c_j.

Various value updates are conducted in Line 9. The cost-based grow operator focuses on improving the ATT, at the cost of increasing the TRT, and therefore the larger the demand served to cost ratio, the better the route set improvement anticipated. This demand to cost ratio b_j can

Algorithm 6.8: Cost-based grow

Input : A route set $\mathcal{R}$, the index i of a route in $\mathcal{R}$ to be evaluated, an $n \times n$ weight matrix $\boldsymbol{W}$, an $n \times n$ demand matrix $\boldsymbol{D}$, and the maximum number of allowed vertices $m_{\max}$.

Output: A grown route set $\mathcal{R}'$.

$d_0 \leftarrow$ `calc_cum_demand_route_set`($\mathcal{R}$, $\boldsymbol{D}$) [Algorithm 6.6]
$j \leftarrow 1$
$\mathcal{R}_t \leftarrow \mathcal{R}$
if $|\mathcal{R}_t[i]| < m_{\max}$ **then**
    foreach vertex h adjacent to the first vertex in route $\mathcal{R}_t[i]$ not in route $\mathcal{R}_t[i]$ **do**
        $\mathcal{R}_t \leftarrow \mathcal{R}_t$ with vertex h inserted into the front of route $\mathcal{R}_t[i]$
        $d_j \leftarrow$ `calc_cum_demand_route_set`($\mathcal{R}_t$, $\boldsymbol{D}$) $-d_0$
        $c_j \leftarrow \boldsymbol{W}[\mathcal{R}[i][0], \mathcal{R}_t[i][0]]$
        $b_j, a_j, v_j, j \leftarrow d_j/c_j$, `true`, h, $j+1$
$\mathcal{R}_t \leftarrow \mathcal{R}$
if $|\mathcal{R}_t[i]| < m_{\max}$ **then**
    foreach vertex h adjacent to the last vertex in route $\mathcal{R}_t[i]$ not in route $\mathcal{R}_t[i]$ **do**
        $\mathcal{R}_t \leftarrow \mathcal{R}_t$ with vertex h inserted into the back of route $\mathcal{R}_t[i]$
        $d_j \leftarrow$ `calc_cum_demand_route_set`($\mathcal{R}_t$, $\boldsymbol{D}$) $-d_0$
        $c_j \leftarrow \boldsymbol{W}[\mathcal{R}[i][-1], \mathcal{R}_t[i][-1]]$
        $b_j, a_j, v_j, j \leftarrow d_j/c_j$, `false`, h, $j+1$
$\ell \leftarrow j - 1$
if $\ell = 0$ **then**
    return $\mathcal{R}$
if $\sum_{k=1}^{\ell} b_k \neq 0$ **then**
    for $j = 1$ **to** ℓ **do**
        $p_j = b_j / \sum_{k=1}^{\ell} b_k$
else
    for $j = 1$ **to** ℓ **do**
        $p_j = 1/\ell$
Randomly choose a grow candidate j based on the the probabilities p_j
if a_j **then**
    $\mathcal{R}' \leftarrow \mathcal{R}$ after inserting vertex v_j into the first vertex position of route $\mathcal{R}[i]$
else
    $\mathcal{R}' \leftarrow \mathcal{R}$ after inserting vertex v_j into the last vertex position of route $\mathcal{R}[i]$
return $\mathcal{R}'$

be expressed as the demand contribution d_j divided by the cost c_j of adding the specific edge. Note that in the case of cost-based trim, the ratio is the inverse of that for cost-based grow, because for each operator the goal is improving a different objective function. Moreover, the front vertex boolean value of a_j is assigned the boolean value `true`, the adjacent vertex h under evaluation is assigned to v_j, and the candidate counter is incremented by 1. This concludes the evaluation of the adjacent vertices for the front of route $\mathcal{R}_t[i]$.

Next, the adjacent vertices at the rear of route $\mathcal{R}_t[i]$ are evaluated. Line 10 resets the route set $\mathcal{R}_t$ to the original route set $\mathcal{R}$. The same feasibility test is conducted in Line 11 to ensure sufficient capacity to add additional vertices to the route $\mathcal{R}_t[i]$. A similar evaluation to that in Lines 5–9 is conducted in Lines 12–16, with the exception that now vertices h adjacent to the last vertex in the route $\mathcal{R}_t[i]$, which are not already present in that route, are considered. Once again, when assigning the value c_j, negative indexing is utilised, and the values b_j, a_j, v_j and j updated accordingly, while the front vertex boolean value of a_j is not assigned the value `false`.

In Line 17, the total number of candidate grows is assigned to the value ℓ. If $\ell = 0$ in Line 18, the original route set $\mathcal{R}$ is returned as output and the algorithm terminates. If, however, $\ell > 0$, the algorithm continues to Line 20 where the potential case of all b_j-values being zero is handled. This occurs when the addition of one additional vertex at any terminal vertex does not result in any increase in demand met by direct routes. If the sum of all the b_j-values is not zero, probabilities p_j are calculated for each candidate grow j being applied according to the expression in Line 22. If, however, the sum of all the b_j-values is zero, equal values are assigned to all the probabilities p_j according to the expression in Line 25.

After having assigned probabilities for all candidate grows, one grow j is randomly selected, based on the probabilities p_j, and if the corresponding a_j-value is `true`, the corresponding vertex v_j is inserted into the first position of route $\mathcal{R}[i]$. Otherwise, vertex v_j is inserted into the last position of route $\mathcal{R}[i]$. The algorithm terminates by returning the grown route $\mathcal{R}'$ as output.

Random all cost-based trim

The two operators mentioned above, namely cost-based trim and cost-based grow, each operates on only a single vertex at a time. In order to make significant improvements to route sets, various of these operators should ideally be performed subsequently on a given route set $\mathcal{R}$. This reasoning led to the creation of the *random all cost-based trim* and *random all cost-based grow* operators. Random all cost-based trim randomly steps through each route R_i in a given route set $\mathcal{R}$, and applies Algorithm 6.7 a random number of times. This random number is determined by selecting a random integer uniformly between 1 and $|R_i| - m_{\min}$, inclusive, where $|R_i|$ is the number of vertices already contained in route R_i, and $m_{\min}$ is the minimum number of vertices allowed in a route of the UTRP instance under consideration.

Random all cost-based grow

Random all cost-based grow, on the other hand, randomly steps through each route in a given route set $\mathcal{R}$, and applies Algorithm 6.8 a random number of times to each route. This random number of applications for route R_i is determined by selecting a random integer uniformly between 1 and $m_{\max} - |R_i|$, inclusive. Here, $m_{\max}$ denotes the maximum number of vertices allowed in a route of the UTRP instance under consideration.

Full cost-based trim

Preliminary testing revealed that the extremal values of each objective function associated with the non-dominated set returned by generating initial feasible solutions were not representative of the range of attainable values, and this led to the creation of the *full cost-based trim* and *full cost-based grow* operators. Here, Algorithm 6.7 is applied to all the routes in a given route

set $\mathcal{R}$, in a random order. The cost-based trim algorithm is applied $|R_i| - m_{\min}$ times to route R_i, or until a feasibility constraint is violated, upon which the algorithm moves on to the next route in the random sequence of routes in $\mathcal{R}$. The reasoning behind the working of this operator is that the network graph is trimmed to a point where each route in a route set contains the minimum number of vertices required to maintain feasibility. Moreover, the highest cost per demand vertices are removed first, retaining the routes that are most beneficial in terms of both objectives. The aim of this operator is to find high-quality solutions to an UTRP instance that gives precedence to the TRT objective over the ATT objective.

This approach may be improved upon by evaluating all the potential deletions of edges from a network, based on their cost per demand ratios, instead of only taking into account one route at a time. This modification may be achieved by not taking as input the route index i, but rather inserting a for-loop starting between Lines 2–3, and ending between Lines 20–21 of Algorithm 6.7, looping over all the route indices of the route set $\mathcal{R}$. This is expected to lead to improved solution quality as a global view of the route set is taken, instead of focusing locally on one route at a time.

Full cost-based grow

In order to achieve solutions that give precedence to the ATT objective over the TRT in the ATT-extreme point of the non-dominated set returned, a counterpart to the full cost-based trim operator is proposed. The operator involves applying Algorithm 6.8 a maximum allowed number of times to a network until a model constraint is violated. This is achieved by randomly stepping through each route R_i in a given route set $\mathcal{R}$, and applying the cost-based grow algorithm $m_{\max} - |R_i|$ times to R_i, or until a model constraint is violated, upon which the next route is taken as input to Algorithm 6.8. The network graph will thus be densely packed with routes, giving precedence to adding vertices to terminals that would result in more demand being met by direct routes, and hence lowering the ATT at the cost of the TRT.

Algorithm 6.8 may also be modified in the sense that instead of considering only route $\mathcal{R}[i]$, all the routes in $\mathcal{R}$ are considered to find the best insertions for the overall route set, stochastically choosing among all the possible insertions. This modification is achieved by not taking as input the index of a route i, but rather inserting a for-loop from between Lines 3 and 4 of Algorithm 6.8 to between Lines 16 and 17, looping through all the route indices i of a given route set $\mathcal{R}$. This modification is the final full cost-based grow operation implemented. Although it is computationally more expensive, the quality of solutions returned is higher. The algorithm may be executed once, generating various initial solutions for each UTRP test instance, and saving them into a *csv* file for later retrieval.

The enhancement procedure

The initial candidate route set enhancement procedure for generating initial feasible solutions is described in this section. When generating initial solutions, especially for population-based searches, it is important to achieve a good spread and diversity of solutions. The enhancement procedure adopted in this dissertation is aimed at improving the spread of the initial solutions in objective space, as they typically tend to bunch in the mid portion of the obtainable non-dominated set for larger UTRP instances. The procedure creates a pre-specified number of initial candidate solutions by invoking one of the ICRSGPs described in §6.1.2. For each solution thus generated, the full cost-based grow, random all cost-based grow, random all cost-based trim, and full cost-based trim operators are performed, creating four additional initial feasible solutions,

but containing properties of solutions typically residing in other portions of the Pareto set of a UTRP instance.

An illustration of this procedure and the solution quality and spread one might expect from this approach is provided in Figure 6.3. The full cost-based grow operator typically yields solutions with the smallest ATT objective function values, and the largest TRT objective function values, whereas the full cost-based trim operator typically yields solutions with the smallest TRT objective function values, and the largest ATT objective function values. Moreover, the solutions returned by the random all cost-based grow operator are typically between those returned by the full cost-based grow operator and those returned by the ICRSGP in objective space. Similarly, the solutions returned by the random all cost-based trim operator are between those returned by the ICRSGP and those returned by the full cost-based trim operator in objective space. For all numerical results reported later in this dissertation, the number of initial feasible candidate route sets to be produced by the ICRSGP was set to 2 000, and saved in a separate *csv* file. An additional 8 000 candidate solutions were therefore generated by the enhancement procedure described in this section (also saved in the *csv* file), yielding a total of 10 000 solutions for each UTRP instance considered. A non-dominated set containing a pre-specified number of solutions could then be extracted from these 10 000 initial feasible solutions based on the FNSA (Algorithm 2.7) and by applying the crowding distance operator (Algorithm 3.4).

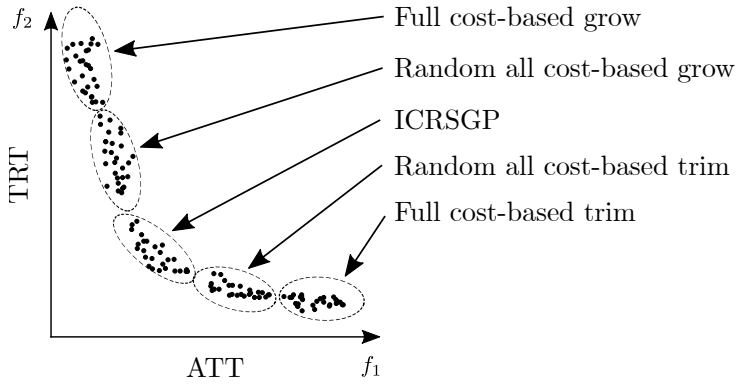


FIGURE 6.3: *An illustration of the initial candidate route set enhancement procedure, where the initial candidate solutions are generated by invoking an ICRSGP, and then enhanced by applying the full cost-based grow, random all cost-based grow, random all cost-based trim, and full cost-based trim operators to each solution generated by the ICRSGP. The horizontal axis measures the ATT objective function, and the vertical axis the TRT objective function. The figure illustrates the areas of the Pareto front that each procedure aims to explore.*

6.1.4 The network analysis procedure

The NAP is responsible for assessing the objective function values $F_1(\mathcal{R})$ and $F_2(\mathcal{R})$ associated with a transit route set $\mathcal{R}$, namely the passenger (time) cost in (5.5) and the operator (time) cost in (5.6) induced by $\mathcal{R}$ [51, 53]. For this purpose, the graph $G_{\mathcal{R}}$ has to be analysed so that values for $\alpha_{v_i,v_j}(\mathcal{R})$ may be determined for all OD pairs $v_i, v_j \in \mathcal{V}$ in terms of the weight matrix $\boldsymbol{W}$ specified.

The values of $\alpha_{v_i,v_j}(\mathcal{R})$ are computed by applying Floyd's algorithm (described in §2.2.2) so as to form the expanded transit network induced by the subgraph $G_{\mathcal{R}}$. An example of how this expanded transit network is formed from a given route set is illustrated in Figure 6.4, where each route is denoted by a separate set of vertices with a corresponding route label appended to each vertex number. Additionally, transfer links are inserted between the different route vertices

at points where transfers may occur. The results returned by Floyd's algorithm are saved in a list $\mathcal{T}$, containing the transport times spent by passengers when travelling from any vertex v_i to another vertex v_j in $G_{\mathcal{R}}$. The shortest-time path entries of $\mathcal{T}$ constitute in-vehicle travel times, as well as the transfer penalties incurred when passengers utilise a vehicle transfer in respect of the transit route set $\mathcal{R}$. An example of the calculation of $\alpha_{v_i,v_j}(\mathcal{R})$, when applied to a small transit network, is provided in Figure 6.5, where the effects of direct routes and transfers are also elucidated.

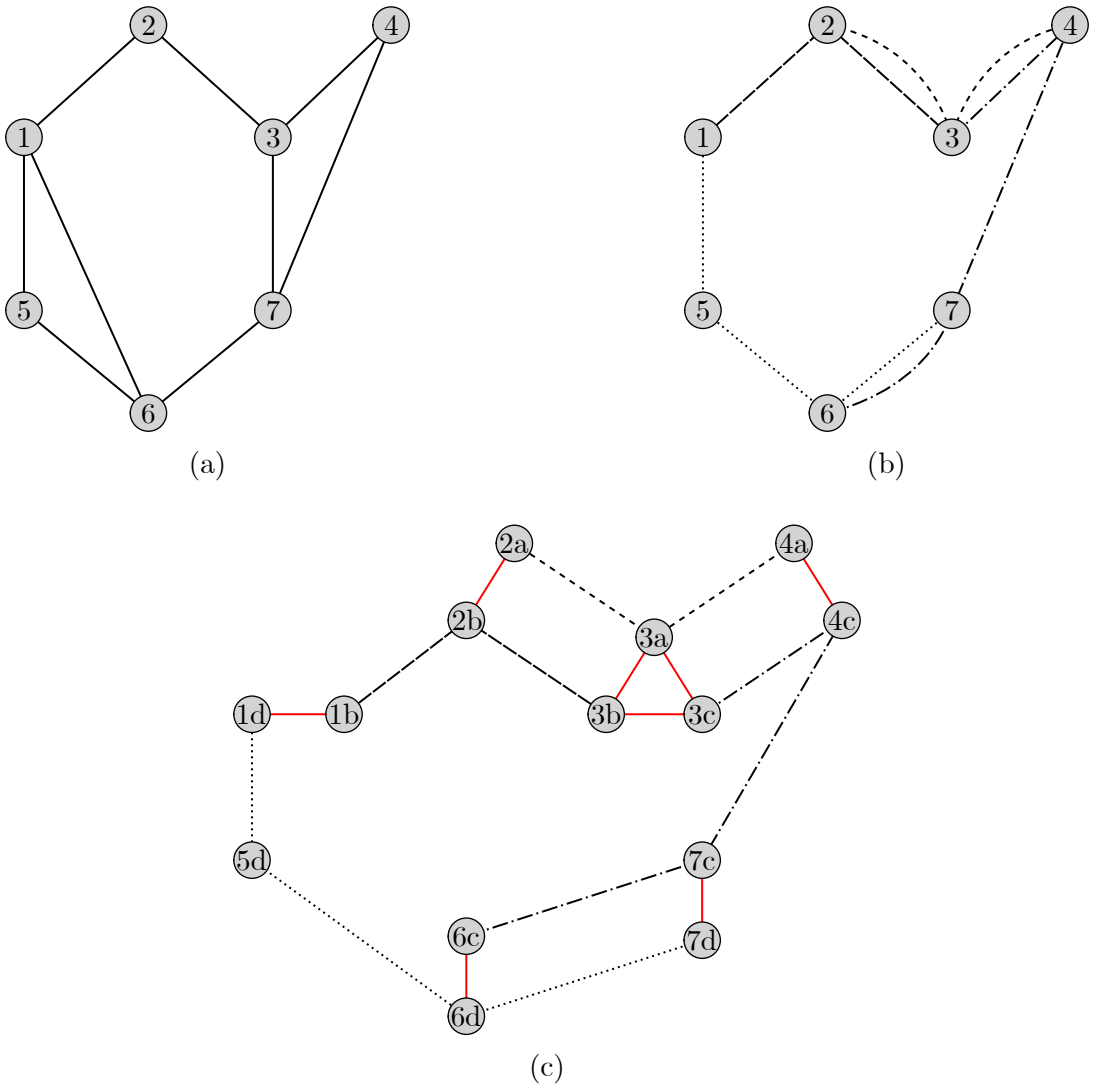


FIGURE 6.4: *An example of how a transit network can be constructed from the road network graphical representation (a), adopted from [89]. The route network in (b) is induced by a route set* $\mathcal{R} = \{\langle 2,3,4\rangle, \langle 1,2,3\rangle, \langle 3,4,7,6\rangle, \langle 1,5,6,7\rangle\}$. *Furthermore, the transit network in (c) is formed by expanding the route network as a result of splitting each transfer vertex into route-specific transfer vertices for each route adjacent to the respective transfer vertex. Thereafter, transfer links are inserted between the subcomponents of each transfer vertex, indicated by the red edges. Each route in* $\mathcal{R}$ *is assigned a letter a to d as a label, respectively, so that it can be distinguished from other vertices. Transfer vertex subcomponents are labelled with the vertex number concatenated to the route label.*

As the objective function requires the execution of Floyd's algorithm, a time complexity of $\mathcal{O}(|\mathcal{V}|^3)$ is observed, where $\mathcal{V}$ is the number of vertices present in the network. When, however, the transit network is expanded, as in the example of Figure 6.4, the total number of vertices in-

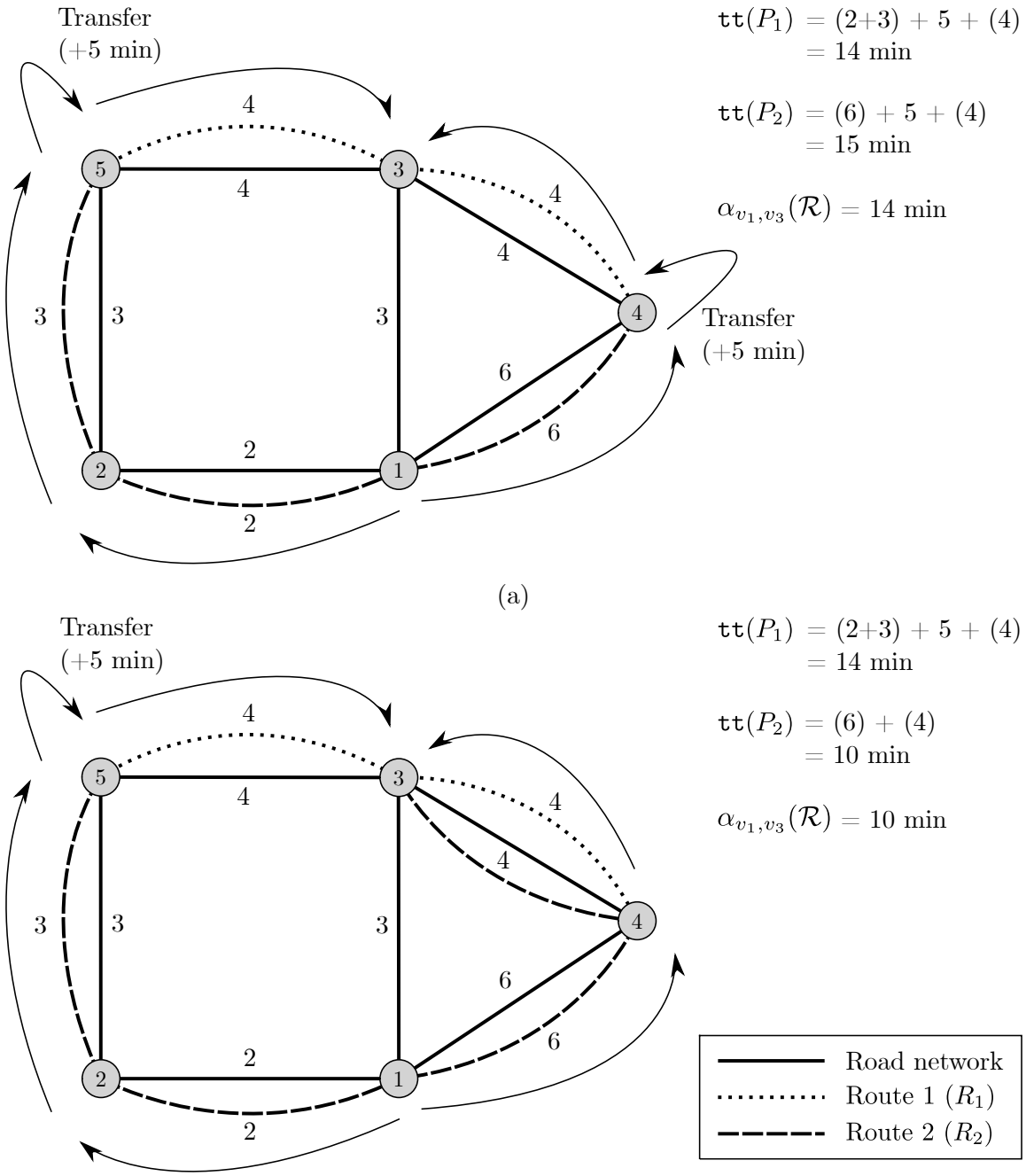


FIGURE 6.5: *An illustration of the evaluation of $\alpha_{v_i,v_j}(\mathcal{R})$ for a small example network, where $v_i = 1$ and $v_j = 3$, and $\mathcal{R} = \{R_1, R_2\}$. The arrows represent passenger movement through the system, along the bus routes present in the system. Where transfers occur at changes between routes, a penalty of 5 min is incurred each time. In both cases (a) and (b), two possible route choices exist for the passenger, namely $P_1 = \langle 1, 2, 5, 3\rangle$ and $P_2 = \langle 1, 4, 3\rangle$, given the origin vertex being v_1 and the destination vertex being v_3. The applicable calculations for each path is shown on the right-hand side next to each case, with the* `tt` *function denoting the calculation of the travel time of each path. In (a), one transfer is required for both paths, with the shortest path being P_1. In (b), an extra route segment $\{3, 4\}$ was added to Route 2, and therefore path P_2 now requires no transfers, making path P_2 shorter than P_1. The calculation of $\alpha_{v_1,v_3}(\mathcal{R})$ is therefore made clear by returning the travel time of the shortest path.*

creases. John [89] provides an analysis on the number of vertices present in an expanded network for the five popular benchmark instances mentioned in §4.5.8, as summarised in Table 6.1.

TABLE 6.1: *UTRP expanded network benchmark analysis presented by John [89]. The mean, minimum, and maximum vertices of an expanded network for the solutions found in a non-dominated set $\mathcal{P}$ are presented, along with the number of solutions $|\mathcal{P}|$ in the non-dominated set and the number of vertices contained in the original transport network for each instance.*

Instance	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{V}\rvert$	Mean vertices	Minimum vertices	Maximum vertices
Mandl	97	15	29	20	41
Mumford0	99	30	86	41	179
Mumford1	93	70	231	150	450
Mumford2	101	110	715	560	1 199
Mumford3	103	127	930	720	1 500

John [89] also tested the runtime of Floyd's algorithm on graphs of various orders using the computer language C [94]. He utilised the *graphical processing unit* (GPU) of a computer in order to reduce the runtime of Floyd's algorithm significantly, as compared with the use of a *central processing unit* (CPU), which is common to most implementations of Floyd's algorithm. In this dissertation, the runtime of the Floyd's algorithm in Python proved the most significant challenge, as Python is a high-level, dynamic programming language and therefore conducts various checks in the background that a user is often unaware of and which may slow down runtime unnecessarily.

One example is that variable types are not always defined in Python, but are rather left to the programming language interpreter to determine. For instance, x may be a list in one place, and then assigned to be an integer in another place, without explicitly defining the variable type. Static, low-level programming languages, like C, require a definition of the type of a variable beforehand and this type does not change throughout the execution of the algorithm. As a result, and because of employing low-level code that is readily interpretable by a computer, these low-level languages generally achieve faster runtimes than high-level programming languages.

In order to overcome this obstacle, the Cython programming language [10], embedded within Python and useful for writing C extensions, was used to write and compile various bottleneck functions in the model solution implementation. Floyd's algorithm is by far the largest bottleneck of all the functions implemented, and Cython helped reduce the total runtime by a factor of about 30, as compared with a single thread Python implementation. This was achieved by enabling parallel multi-core processing and by neglecting to enable bounds checking in terms of negative indices. The multi-processing capability allowed the for-loop in Floyd's algorithm to be calculated in parallel, thereby utilising 100% of the computer's CPU when the model objective function was being evaluated. This was implemented on a computer running in the Windows operating system, equipped with an i7-4770 CPU performing at 3.40GHz, and with 16 GB of RAM. This speed-up is a necessary requirement when considering larger UTRP instances.

Table 6.2 contains a summary of some of the runtimes computed by John [89], both for his GPU and CPU implementations, along with the Cython implementation of Floyd's algorithm, where the wall time refers to the total time elapsed, and the CPU time is the total time the computer spent processing on the CPU, and where all the separate computational times spent on each core are summed together. No details were unfortunately available for the computer used by John. A speed-up comparison of John's CPU time and the Cython implementation is presented, along with the speed-up effect achieved by using multiprocessing. It is interesting to note that the Cython implementation achieved the best results for graphs of orders at most

300, while the GPU performed better for larger graphs. In terms of the CPU comparison, the Cython implementation performed better than the CPU implementation of John [89] in all the instances, achieving a speed-up of 2.22 times or more. The multiprocessing effect yielded a speed-up of between 5.04 and 6.79 times relative to single thread processing.

TABLE 6.2: *Floyd's algorithmic runtime comparisons for various implementations over various instance sizes. The first two approaches are due to John [89], and the CPU Cython implementation of this dissertation in terms of wall time, and the computational time spent on the CPU, a CPU speed-up comparison between the two CPU approaches, expressed as the Cython implementation speed-up over John's CPU implementation. Lastly, the multiprocessing effect is expressed as the CPU time divided by the wall time of the Cython implementation. The average runtimes of 20 instances were used in the values for the Cython implementation of this dissertation. The overall best runtime is shown in boldface for each instance.*

Vertices	GPU [89]	CPU [89]	Wall time Cython	CPU Time Cython	CPU speed up comparison	Multiprocessing effect
100	0.005	0.002	**0.001**	0.005	2.22	5.22
200	0.008	0.014	**0.004**	0.030	3.50	7.41
300	**0.013**	0.049	**0.013**	0.067	3.68	5.04
400	**0.023**	0.115	0.028	0.159	4.05	5.61
500	**0.041**	0.226	0.053	0.362	4.24	6.79
1 000	**0.243**	1.919	0.416	2.795	4.62	6.73
2 000	**1.766**	19.741	4.212	28.082	4.69	6.67
3 000	**5.935**	54.411	14.146	91.277	3.85	6.45
4 000	**14.077**	125.663	32.95	218.889	3.81	6.64
5 000	**26.902**	244.407	63.554	420.894	3.85	6.62

For the benchmark instances, the mean vertices for the largest instance is 930, according to the analysis John conducted, and the maximum observed vertices in the expanded network is 1 500. The performance of the Cython implementation of Floyd's algorithm is therefore on par and on the same order as that of the GPU for the smaller instances, as it is about 2–3 times slower than using a GPU for graphs of about 1 000–2 000 vertices.

A common network analysis metric involves measuring the number of transfers that passengers undergo while travelling from their origins to their respective destinations. This is expressed in the form of the percentage demand satisfied without transfers, with one transfer, with two transfers, while any number of transfers more than two is considered as unsatisfied demand. This analysis is achieved by counting the number of transfers required for each OD pair, grouping the demand by each of the aforementioned categories, and dividing those values by the total demand.

6.1.5 Lower-level heuristics as move operators

In this dissertation, 21 *low-level heuristics* (LLHs) were considered for incorporation into the DBMOSA algorithm and the NSGA II solution implementation when solving instances of the UTRP. Thirteen of these LLHs were taken from the literature, and were chosen based on their reported success and their ease of implementation. The other eight LLHs are novel contributions of this dissertation. This section is devoted to a description of the various LLHs applied in this dissertation, and Table 6.3 provides a summary of each of these operators, along with each operator's name, source and a type convention introduced to elucidate the nature of the operator. Figures 6.6 and 6.7 contain illustrations of each of the operators taken from the

literature, whereas Figure 6.8 contains illustrations of the novel operators introduced in this dissertation. Descriptions of each of these operators in the context of the UTRP follow in the remainder of this section. The term LLH is used interchangeably with the terms perturbation and mutation, depending on the context. In order to help the reader navigate this section, the letters (a)–(m) were assigned to the operators in Table 6.3. This numbering convention is also adopted in the text, and in Figures 6.6–6.8.

TABLE 6.3: *The set of LLHs considered in this dissertation. Letters are assigned to help navigate this section, and correspond to the letters included in the illustrations of the operators in Figures 6.6–6.8.*

Letter	LLH name	Source	Type
a	Add vertex	[50]	Add
b	Delete vertex	[50]	Delete
c	Add inside vertex	[2]	Add
d	Delete inside vertex	[2]	Delete
e	Invert vertices	[2]	Rearrange
f	Relocate vertex	[2]	Rearrange
g	Replace vertex	[2]	Rearrange
h	Donate vertex	[2]	Inter route rearrange
i	Swap between routes	[2]	Inter route rearrange
j	Exchange routes	[115]	Rearrange
k	Merge terminals	[89]	Route addition
l	Replace low demand route	[89]	Route replacement
m	Replace subsets	[89]	Route replacement
n	Remove low demand terminal	This dissertation	Delete
o	Remove large cost terminal	This dissertation	Delete
p	Cost-based grow	This dissertation	Add
q	Cost-based trim	This dissertation	Delete
r	Random cost-based grow	This dissertation	Add
s	Random cost-based trim	This dissertation	Delete
t	Random all cost-based grow	This dissertation	Add
u	Random all cost-based trim	This dissertation	Delete

The first four operators represent the simplest form of adding or deleting operator types. The *add vertex* (a) and *delete vertex* (b) operators were originally introduced by Fan and Mumford [50]. They entail randomly adding an adjacent vertex to a terminal vertex of a randomly selected route in a route set, and randomly deleting a terminal vertex from a randomly selected route, respectively. The *add inside vertex* (c) and *delete inside vertex* (d) operators, proposed by Ahmed *et al.* [2], involve randomly adding or deleting a non-terminal vertex from a randomly selected route in a route set.

The next six operators involve rearranging vertices within routes in some way. The first five of these operators were proposed by Ahmed *et al.* [2], and the last was introduced by Mandl [115]. The *invert vertices* (e) operator inverts the vertices of a randomly selected route portion in a chosen route set. The *relocate vertex* (f) operator selects one random vertex in a randomly selected route from a route set, and relocates it to another position in that same route. The *replace vertex* (g) operator selects one random vertex from a randomly selected route of a route set, and replaces it with another vertex. The *donate vertex* (h) operator randomly selects a vertex from a randomly selected route, removes it from its current route, and then inserts it into a random location in another randomly selected route. The *swap between routes* (i) operator

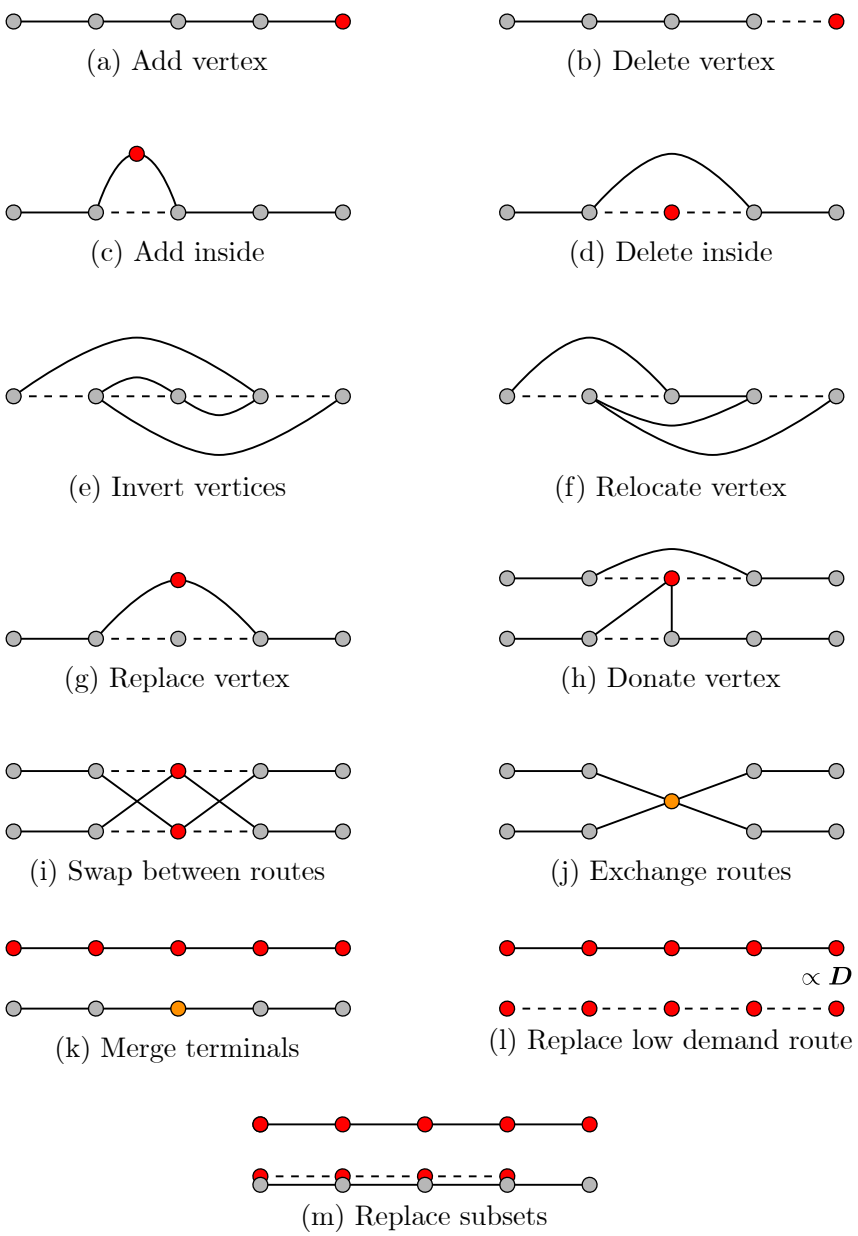


FIGURE 6.6: *LLH operators proposed in the literature for solving instances of the UTRP. The solid lines represent the resulting edges in a route, while the dashed lines represent edges that have been deleted after having applied the LLH to the route set. The grey vertices represent stops in a route that have not been altered, red vertices represent either stops that have been included or deleted, based on the context of the adjacent edges, and the orange vertices represent special operations performed, such as exchanging or merging route segments. The curves represent changes in edges that have occurred after having applied the LLH operator. The* $\propto \boldsymbol{D}$ *represents an operator that is proportional to demand.*

randomly selects a vertex from each of two randomly selected routes, and swaps the two vertices with each other.

The *exchange operator* (j), introduced by Mandl [115], identifies a random common vertex in two randomly chosen routes, and then splits each of these routes at the common vertex, recombining a segment of the first route with a segment of the second route. This recombination is performed randomly, as illustrated in Figure 6.7. There are two possible outcomes for this recombination procedure. If, however, both changes lead to infeasible routes, an attempt is made to repair

the route set, and if unsuccessful, the exchange procedure is re-applied to the same offspring until all route combinations have been exhausted and repairs attempted. If no feasible change is found, the operation is abandoned and no change is performed on the original offspring.

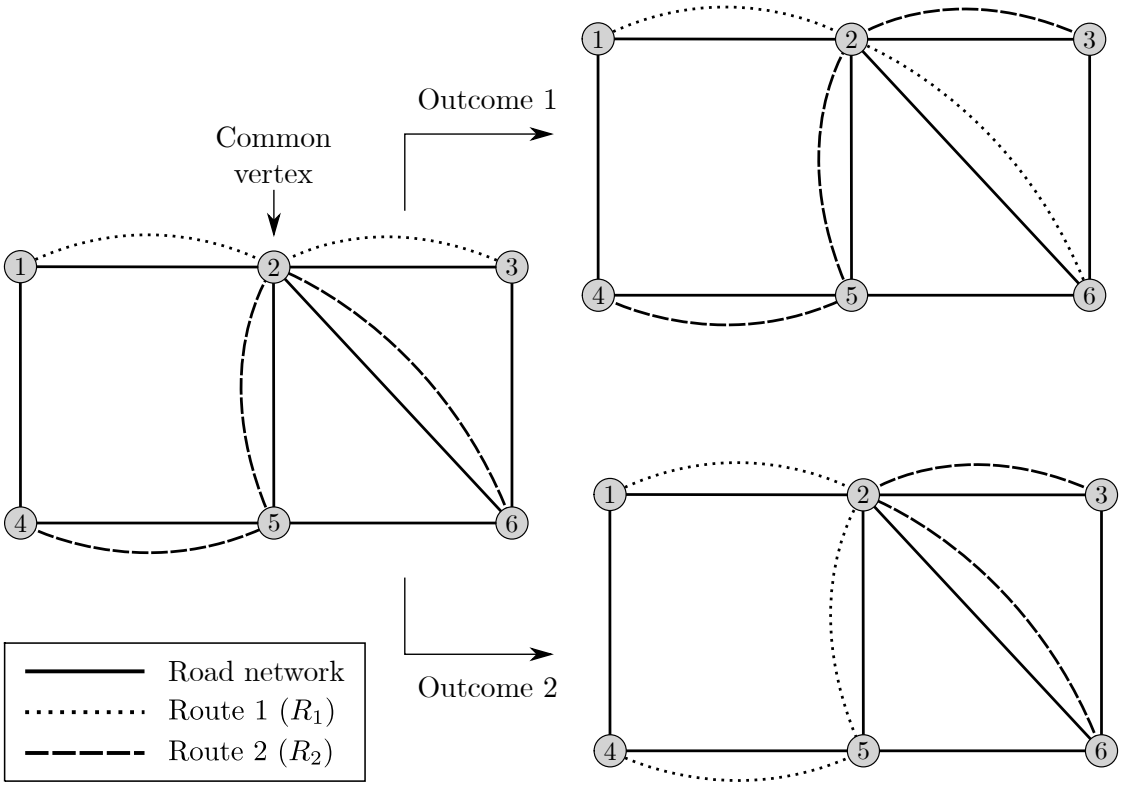


FIGURE 6.7: *An illustration of the mutation operator applied to two routes R_1 and R_2 in the NSGA II when solving the UTRP model. The mutation occurs at a common transfer vertex between two routes — in this case vertex v_2 — and involves swapping one of the two route segments on either side of the common vertex of route R_1 with either of the two route segments formed by the common vertex of route R_2. The two possible outcomes are illustrated on the right-hand side.*

The last three operators borrowed from the literature were introduced by John [89], and these operators involve the introduction of new routes to a route set by creating space for one additional route without violating any model constraints. The *merge terminals* (k) operator searches through all the routes in a route set to ascertain whether any two terminals are the same vertex, and then randomly selects one of these merge points, and merges the two routes together, subject to all the model constraints. This search for a feasible merge is continued until all possibilities for the route set have been exhausted. If a feasible merge is found, another route is generated for inclusion in the route set based on the route generation procedure of Shih and Mahmassani [146].

The *replace low demand route* (l) operator replaces the route that satisfies the lowest demand directly with a route based on the aforementioned route generation procedure of Shih and Mahmassani [146]. The *replace subsets* (m) operator determines whether there is any route in the route set which is a subset of another route, and replaces the smaller of these routes with another route generated according to Shih and Mahmassani's route generation procedure [146]. Routes that are subsets of other routes contribute nothing to the ATT objective function, due to there already being a direct route for passengers to utilise. Such subroutes only increase the TRT objective function value unnecessarily.

For all of the above operators, additional checks were implemented so that each operator takes as input a route set, and efficiently searches through all possibilities of changing the route set so that only feasible routes are returned. Ahmed *et al.* [2] allowed the operators to return infeasible solutions and penalised those solutions, but the approach adopted here is aimed at maintaining feasible solutions only.

The next eight operators are new operators proposed for use within the context of the UTRP, and are illustrated graphically in Figure 6.8. The *remove low demand terminal* (n) operator searches through all the terminal vertices and removes the terminal vertex satisfying the least demand, subject to the route set feasibility constraints. The *remove large cost terminal* (o) operator, functions similarly, but instead deletes the terminal vertex incident with the largest edge weight.

The remaining six operators work on the basis of the cost-based growth and cost-based trim paradigm described in §6.1.3. The *cost-based grow* (p) and *cost-based trim* (q) operators were described in detail in §6.1.3. The *random cost-based grow/trim* (r)/(s) operator is the same as applying the random all cost-based grow/trim operator described in §6.1.3 to one route only, instead of all the routes in a route set. Finally, the *random all cost-based grow/trim* (t)/(u) operator was described fully in §6.1.3.

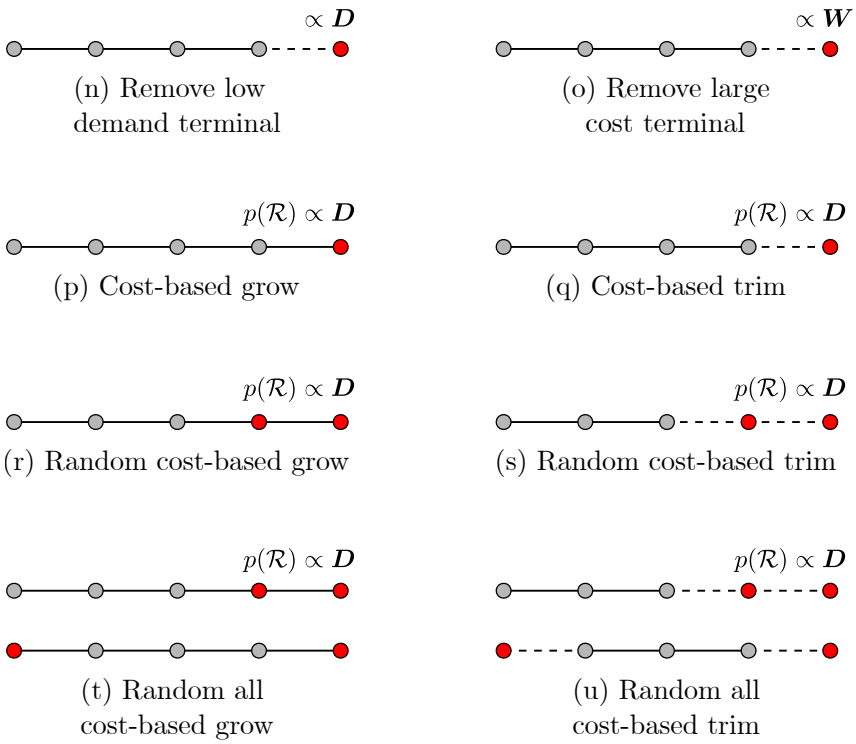


FIGURE 6.8: *LLH operators introduced in this dissertation for solving the UTRP. The solid lines represent the resulting edges in a route, while the dashed lines represent edges that have been deleted after having applied the LLH to the route set. The grey vertices represent stops in a route that have not been altered, red vertices represent either stops that have been included or deleted, based on the context of the adjacent edges, and the orange vertices represent special operations performed, such as exchanging or merging route segments. The curves represent changes in edges that have occurred after having applied the LLH operator. The* $\propto \boldsymbol{D}/\boldsymbol{W}$ *represents an operator that is proportional to demand/edge weights. The* $\propto p(\mathcal{R})$ *represents an operator that is proportional to a specific probability associated with a route set.*

6.1.6 Hyperheuristic approaches towards managing LLHs

When many LLHs are available for use, it is often difficult to determine which LLHs are anticipated to be most beneficial and contribute most significantly towards finding high-quality solutions, as well as the extent to which these LLHs aid in quality improvement. The need therefore arose to create methods for managing these LLHs effectively during the UTRP solution procedure. These methods were inspired by a hyperheuristic proposed by Vrugt and Robinson [167]. To the best of the author's knowledge, only one other hyperheuristic approach has been adopted in the context of the UTRP in the literature by Ahmed *et al.* [2], as also confirmed by Soares [149]. This section is devoted to discussions on the methods incorporated in this dissertation for managing LLHs, first for trajectory-based searches, and secondly for population-based searches.

Perturbation management for trajectory-based searches

Recall that trajectory-based searches are single-point searches during which a single solution is perturbed iteratively. Three approaches were explored for managing the probabilities of applying any of the LLH perturbations to the current candidate solution of such a search, the first being a stochastic roulette-wheel strategy, the second an AMALGAM-based strategy, and the third an every n-th iteration AMALGAM-based strategy. The working of the AMALGAM method was described in §3.4.2. The idea is that a set of LLHs may be provided as perturbation mechanisms during a trajectory-based search, and that an overarching hyperheuristic dynamically updates the probabilities according to which LLHs are applied with a view to guide the search towards finding high-quality solutions. The selection probabilities are updated based on the success ratios of the LLHs during the previous iteration and a temporal record is kept of each LLH's performance. This equips the trajectory-based search with a robust search mechanism, prioritising perturbations that have previously led to the inclusion of solutions in the non-dominated set. For all three of the aforementioned approaches, each LLH starts off with an equal probability of being chosen.

As mentioned by Burke *et al.* [24], all score-based heuristic selection techniques comprise five main component implementations, namely initial scoring, memory length adjustment, a strategy for heuristic selection based on the scores, a score update rule accounting for improvements, and a score update rule for worsening solutions. A roulette-wheel selection strategy associates a probability of selection with each LLH, whereas a max strategy involves deterministically selecting the LLH that performs the best.

For the first implementation, the stochastic roulette-wheel search strategy, the initial LLH scores were set to $\lceil (1/I) \times 100 \rceil$, where I is the total number of LLHs considered during the search, and the memory length was set to encompass the entire search history. The roulette-wheel selection strategy was chosen for selecting an LLH to be applied as the perturbation methodology during a trajectory-based search. The score update rule for improvement entailed increasing the score of the relevant LLH by one if the LLH applied, in fact, led to the solution's inclusion in the non-dominated set (and if the total score before the perturbation was less than 100). The score update rule for worsening solution quality involved decreasing the score of the relevant LLH by one if the solution was rejected (and if the score before the perturbation was larger than 1). The probabilities for the roulette-wheel selection were calculated by dividing the score of LLH i by the total score of all the LLHs in the set.

Algorithm 6.9 contains a pseudo-code description of the stochastic roulette-wheel perturbation strategy for trajectory-based searches. The algorithm takes as input the selection probabilities

p_i and the LLH scores q_i at the current stage of the search for LLH $i \in \{1, \ldots, I\}$. Furthermore, it takes as input the LLH j applied to the previous solution, as well as the acceptance state of the previous solution $\gamma \in \{-1, 0, 1\}$. Here, an acceptance state of -1 means that the solution was rejected as the current solution, 0 means that the solution was accepted as the current solution, but not included in the non-dominated set, and 1 means that the solution was accepted as the current solution and inserted into the non-dominated set during the current iteration of the search.

Algorithm 6.9: Roulette-wheel perturbation strategy for trajectory-based searches

Input : The selection probabilities p_i and the LLH scores q_i for each LLH $i \in \{1, \ldots, I\}$, the LLH j applied to the previous solution, and the acceptance state $\gamma \in \{-1, 0, 1\}$ of the previous solution.
Output: Updated selection probabilities p_i and LLH scores q_i for each LLH $i \in \{1, \ldots, I\}$.

1 **if** $\gamma = -1$ **and** $q_j > 1$ **then**
2 $\quad q_j \leftarrow q_j - 1$
3 **if** $\gamma = 1$ **and** $q_j < 100$ **then**
4 $\quad q_j \leftarrow q_j + 1$
5 $\quad$ **foreach** $i \in \{1, \ldots I\}$ **do**
6 $\quad\quad p_i = q_i \Big/ \sum_{i=1}^{I} q_i$
7 **return** $p_1, \ldots, p_I, q_1, \ldots, q_I$

If $\gamma = -1$ and $q_i > 1$, then q_j is reduced by 1, as may be seen in Lines 1 and 2. If $\gamma = 1$ and $q_j < 100$, then q_j is increased by one, and the selection probabilities are updated according to Line 6. The updated selection probabilities p_i and scores q_i are returned as output. Algorithm 6.9 is invoked after each iteration of the trajectory-based search. The LLH to be applied during the next iteration is selected stochastically based on the selection probabilities p_i.

The second trajectory-based LLH management strategy is the AMALGAM perturbation strategy for trajectory-based searches. The initial scoring involves running the algorithm for M memory length iterations, chosen as 200 in this dissertation, and the selection probabilities are kept constant at $1/I$ during the first M iterations. A successful LLH application occurs when the resulting solution is included in the non-dominated set, and the total number of successes of LLH i are tallied and assigned to q_i for the last M iterations. The total number of applications of each LLH is captured by a parameter t_i for the previous M iterations.

Algorithm 6.10 contains a pseudo-code description of the AMALGAM perturbation strategy for trajectory-based searches. The algorithm takes as input the memory length M and the selection probabilities p_i for each LLH $i \in \{1, \ldots, I\}$ during the current iteration of the search, as well as a selection minimum probability threshold p_t. This threshold is required so that none of the LLHs obtains a selection probability of 0, and thereby being eliminated from the search. Furthermore, the success scores q_i and the total LLH applications t_i for each LLH $i \in \{1, \ldots, I\}$ over the previous M iterations are also presented as input. The first step of the algorithm involves determining whether the sum of all the success scores is zero. If this is not the case, success proportions s_i are calculated for each LLH $i \in \{1, \ldots, I\}$ according to the expression in Line 3, and the selection probabilities are updated in Line 4. The first term on the right-hand side of the expression in Line 3 resembles the AMALGAM update strategy, while the second term on the right-hand side of the expression, $(1 - p_t)$, takes into account the selection probability threshold, and in Line 4 adds this as a minimum selection probability for each LLH $i \in \{1, \ldots, I\}$. If all the

success scores q_i sum up to zero in Line 1, no adjustments are made to the selection probabilities (*i.e.* they are kept the same as during the previous iteration). The algorithm finally terminates upon having returned the selection probabilities $p_1, \ldots, p_I$ as output.

Algorithm 6.10: AMALGAM strategy for trajectory-based searches

Input : The memory length M, selection probabilities p_i, a selection probability threshold p_t, and the LLH success scores q_i and total LLH applications t_i for each LLH $i \in \{1, \ldots I\}$ for the last M iterations.

Output: Updated selection probabilities p_i.

1 **if** $\sum_{i=1}^{I} q_i \neq 0$ **then**
2 **foreach** $i \in \{1, \ldots I\}$ **do**
3 $s_i \leftarrow \left(\frac{q_i}{t_i} \Big/ \sum_{j=1}^{I} \frac{q_j}{t_j} \right) \times (1 - p_t)$
4 $p_i \leftarrow s_i + p_t$
5 **return** $p_1, \ldots, p_I$

The third method, called the every n-th AMALGAM perturbation strategy for trajectory-based searches, entails adopting the same approach as in the second method, but with the exception that Algorithm 6.10 is invoked every n search iterations, and the selection probabilities $p_1, \ldots, p_I$ are only updated every n iterations.

Mutation management for population-based searches

Multiple solutions are maintained simultaneously during population-based searches over a number of generations. Excessive random application of a variety of LLH mutations during a population-based search may lead to a random search during which little exploitation is achieved. Population-based algorithms are intrinsically dynamic and adaptive processes. Adopting search parameters that are static therefore stand in stark contrast to such dynamic and adaptive processes. It has, in fact, been demonstrated both empirically and theoretically that, at different stages of an evolutionary process, different parameter values might be optimal [49].

As a consequence, a mutation management strategy for population-based searches is proposed for use in the context of the UTRP based on the AMALGAM method, as described in §3.4.2. The approach follows closely that of the AMALGAM strategy for trajectory-based searches, with the exception that now an entire population's LLH success scores and LLH application totals are maintained, for each LLH $i \in \{1, \ldots, I\}$, one generation at a time. In terms of initial scoring, a selection probability of $1/I$ is assigned to each LLH. The memory length is taken as one generation, and the strategy for LLH selection is based on selecting an LLH randomly according to updated selection probabilities p_i for each LLH $i \in \{1, \ldots, I\}$. The update rule is taken as the expression in (3.11).

The AMALGAM mutation strategy for population-based searches is described in pseudo-code form in Algorithm 6.11. The selection probabilities p_i for each LLH $i \in \{1, \ldots, I\}$, along with a selection minimum probability threshold is provided as input. Moreover, the success scores q_i achieved and the total number of applications t_i implemented during the previous generation are also provided as input to the algorithm for each LLH $i \in \{1, \ldots, I\}$. The success score q_i is the total number of solutions perturbed according to LLH i that were included into the non-dominated set. The algorithm starts off by testing whether the sum of all the success scores is not equal to zero. If this is the case, a success proportion s_i is calculated and the selection

probability p_i updated accordingly for each LLH $i \in \{1, \ldots, I\}$ in Lines 2–4. The AMALGAM expression is formulated in terms of different notation to that used in (3.11) in the first term on the right-hand side of the expression in Line 3. The second term on the right-hand side of the expression in Line 3 takes into account the selection minimum probability threshold. The selection probabilities are updated in Line 4, where the success proportion s_i is added to the selection probability threshold. If Line 1 evaluated as `false`, no updates are applied and the selection probabilities remain constant. The updated selection probability is returned as output when the algorithm terminates.

Algorithm 6.11: AMALGAM mutation strategy for population-based searches

Input : The selection probabilities p_i, a selection probability threshold p_t, and the LLH success scores q_i and total LLH applications t_i for each LLH $i \in \{1, \ldots, I\}$ during the previous generation.

Output: Updated selection probabilities p_i.

1 **if** $\sum_{i=1}^{I} q_i \neq 0$ **then**
2 **foreach** $i \in \{1, \ldots I\}$ **do**
3 $s_i \leftarrow \left(\frac{q_i}{t_i} \Big/ \sum_{j=1}^{I} \frac{q_j}{t_j} \right) \times (1 - p_t)$
4 $p_i \leftarrow s_i + p_t$
5 **return** p_i

6.1.7 Simulated annealing implementation

The SA algorithm employed in this dissertation to solve instances of the UTRP incorporates the ICRSGP and the NAP described above, and guides the candidate solution generation procedure within an iterated search framework [51, 53]. The particular version of SA implemented in this dissertation is the DBMOSA algorithm proposed by Smith [148]. A pseudo-code description of this procedure within the context of the UTRP is provided in Algorithm 6.12. This procedure was adapted from [50, 148].

Although the literature contains various suggestions for values of the SA parameters, Dreo *et al.* [47] noted that in the absence of general theoretical results, empirical adjustments are typically required by the analyst in respect of these parameter values. These adjustments become all the more time consuming as the problem complexity increases, and may further be complicated by increased sensitivity of the results. Therefore, the choice of parameter values was first based on suggestions from the literature, and thereafter adjusted empirically for improved results. The method of determining these values is described in more detail in the following chapter.

Algorithm 6.12 takes as inputs the UTRP instance parameters $\boldsymbol{W}$, $\boldsymbol{D}$, $m_{\min}$, $m_{\max}$ and r described in §5.1. In addition, the required number $L_{\max}$ of iterations per epoch is an input parameter to the algorithm, initially taken as $L_{\max} = 100$, as suggested by Fan and Mumford [50], but later adjusted to 250 upon empirical experimentation for implementation in this dissertation. Throughout the SA search process, the temperature is progressively lowered according to a pre-specified cooling schedule. Reheating is also incorporated so to avoid search entrapment at local optima, as mentioned in §3.3.2. In this dissertation, the well-known geometric schedule is employed for both cooling and reheating.

The initial temperature T_0 is determined by carrying out a short trial search, consisting of M iterations, during which all candidate solutions are accepted and for which the average positive

energy

$$\Delta_{ave} = \frac{\sum_{i=1}^{i=M-1} |\Delta_{E_i}(\mathcal{R}'_i, \mathcal{R}_i)|}{M} \tag{6.3}$$

is computed, where

$$\Delta_{E_i}(\mathcal{R}'_i, \mathcal{R}_i) = \frac{|\tilde{\mathcal{A}}_{\mathcal{R}'_i}| - |\tilde{\mathcal{A}}_{\mathcal{R}_i}|}{|\tilde{\mathcal{A}}|} \tag{6.4}$$

denotes the difference in energy between the current solution $\mathcal{R}_i$ and that of its successor $\mathcal{R}'_i$. In (6.4), $\tilde{\mathcal{A}}$ denotes the union $\mathcal{A} \cup \{\mathcal{R}_i\} \cup \{\mathcal{R}'_i\}$, and $\tilde{\mathcal{A}}_{\mathcal{R}}$ denotes the set of solutions in $\tilde{\mathcal{A}}$ that dominate $\mathcal{R}$. The value $M = 1\,000$ is adopted in this dissertation, as suggested by Fan and Mumford [50]. The initial temperature is then computed as

$$T_0 = -\frac{\Delta_{ave}}{\log P_0} \tag{6.5}$$

for an initial acceptance probability of worsening solutions of $P_0 = 0.999$, as proposed by Fan [50]. The final temperature for $N = 1\,000$ SA search iterations and for a final acceptance probability of $P_N = 0.001$ is similarly computed as

$$T_N = -\frac{\Delta_{ave}}{\log P_N}. \tag{6.6}$$

The cooling parameter β of the geometric cooling schedule, specifying that the temperature during search epoch c should be

$$T_c = \beta^c T_0, \tag{6.7}$$

is computed as

$$\beta = \exp\left(\frac{\log T_N - \log T_0}{N}\right), \tag{6.8}$$

as suggested by Fan and Machemehl [53]. After experimentation, the reheating parameter ζ of the geometric reheating schedule was chosen as 1.05.

Furthermore, a number of parameters which control the SA search are taken as input. These include $B_{\min}$, which denotes the minimum number of accepted moves allowed per epoch, $H_{\max}$ which denotes the maximum number of reheatings allowed, $A_{\max}$ which denotes the maximum number of attempts allowed per epoch, $C_{\max}$ which denotes the maximum number of epochs that may pass without inserting any new solutions into the archive, and $K_{\max}$ which denotes the maximum number of epochs allowed during the entire search.

A number of initialisation steps are performed in Lines 1–4 of Algorithm 6.12. More specifically, an initial solution is generated (as described in §6.1.2), the archive $\mathcal{A}$ to be maintained throughout the search is initialised to contain this initial solution, the epoch counter c is initialised to zero, and so is the number ϵ of epochs that have elapsed without having accepted a candidate solution. Further initialisations comprise setting values for the selection probability p_i and the corresponding success scores q_i for each LLH $i \in \{1, \ldots, I\}$. In Line 6, counters are initialised which are used to keep track of certain parameters during the search per epoch. For the number of attempts, the number of acceptances, and the number of iterations per epoch, the counters a, b and t are utilised, respectively, and each is initialised with a value of 0 at the start of every epoch. Furthermore, a boolean flag is also included in Line 7, called *poor_epoch_flag*, and is set to 1 in order to indicate that no solutions have been inserted into the archive during the current epoch, and is only lowered to zero once a solution is inserted into the archive. Later, in Lines 32–33 of Algorithm 6.12, this mechanism is used to determine whether an epoch was a poor epoch, in which case the poor epoch counter ϵ is incremented accordingly.

Algorithm 6.12: DBMOSA for the UTRP

Input : An $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, an $n \times n$ demand matrix $\boldsymbol{D}$, the minimum allowed number of vertices in a route $m_{\min}$, the maximum allowed number of vertices in a route $m_{\max}$, the maximum allowable number of iterations per temperature $L_{\max}$, the initial temperature T_0, the minimum accepted number of moves per epoch $B_{\min}$, the maximum number of reheatings allowed $H_{\max}$, the maximum number of attempts allowed $A_{\max}$ per epoch, the maximum number of epochs that may pass without inserting any new solutions into the archive $C_{\max}$, the maximum number of epochs $K_{\max}$, and a set of LLHs.

Output: A set of non-dominated solutions $\mathcal{P}_S$ in solution space and each solution's objective function values as a vector P_O in objective space.

Generate an initial feasible route set $\mathcal{R}$ containing r routes
Initialise the archive $\mathcal{A} \leftarrow \{\mathcal{R}\}$
Initialise the counters $c \leftarrow 0$ and $\epsilon \leftarrow 0$
Initialise the p_i and q_i according to the LLH management strategy for each LLH i
while $\epsilon \leq C_{\max}$ *and* $c \leq K_{\max}$ **do**
    $a, b, t \leftarrow 0$
    $poor\_epoch\_flag \leftarrow 1$
    while $b < B_{\min}$ *and* $t < L_{\max}$ **do**
        Generate a neighbour $\mathcal{R}'$ from the current solution $\mathcal{R}$ using LLH i based on p_i
        Generate a random number $p \in (0, 1)$
        if $p < \min\{1,\ \exp(-\Delta_E(\mathcal{R}', \mathcal{R})/T_c)\}$ **then**
            $\mathcal{R} \leftarrow \mathcal{R}'$
            $b \leftarrow b + 1$
            if $|\mathcal{A}_{\mathcal{R}}| = 0$ **then**
                $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{R}\}$
                $b \leftarrow b - 1$
                $poor\_epoch\_flag \leftarrow 0$
                $\epsilon \leftarrow 0$
                for $\mathcal{Q} \in \mathcal{A}$ **do**
                    if $\mathcal{R} \prec \mathcal{Q}$ **then**
                        $\mathcal{A} \leftarrow \mathcal{A} \backslash \{\mathcal{Q}\}$
        else if $h < H_{\max}$ **then**
            $a \leftarrow a + 1$
            if $a > A_{\max}$ **then**
                Update system temperature $T_c \leftarrow \zeta T_c$ by reheating
                $h \leftarrow h + 1$
                `Break`
        Update p_i and q_i values based on LLH management strategy update rules
        $t \leftarrow t + 1$
    $c \leftarrow c + 1$
    Update system temperature $T_c \leftarrow \beta T_{c-1}$ by cooling
    if $poor\_epoch\_flag = 1$ **then**
        $\epsilon \leftarrow \epsilon + 1$
$\mathcal{P}_S \leftarrow \mathcal{A}$
$P_O \leftarrow$ evaluate the objective function for each solution $\mathcal{R}$ in $\mathcal{P}_S$
return $[\mathcal{P}_S, P_O]$

The generation of a neighbouring solution takes place in Line 9 of Algorithm 6.12. The method of generating this solution is governed by applying one of the LLHs provided as input to the algorithm, randomly chosen based on the selection probabilities $p_1, \ldots, p_I$. Checks are put in place to ensure that the solutions generated remain feasible, and when an infeasible solution is encountered, a repair attempt is made by adding vertices to terminal vertices in order to repair feasibility, if possible.

The probability of acceptance of a worsening neighbouring solution $\mathcal{R}$ is governed by Line 11 of Algorithm 6.12 according to the Metropolis rule

$$P(\mathcal{R}') = \min\left\{1, \exp\left(-\frac{\Delta_E(\mathcal{R}', \mathcal{R})}{T_c}\right)\right\}, \tag{6.9}$$

where $\Delta_E(\mathcal{R}', \mathcal{R})$ denotes the energy function which evaluates the energy difference between the current and neighbouring solutions $\mathcal{R}$ and $\mathcal{R}'$, and T_c denotes the temperature of the search during epoch c [96]. The random number generated in Line 10 is compared with the probability of acceptance function value, and if the random number is smaller than the probability of acceptance, the neighbouring solution is accepted as the new solution. A neighbouring solution achieves a lower energy value than the current solution when it is dominated by fewer elements of the estimated Pareto front than the current solution. In this case, the neighbouring solution is accepted with probability 1 as the new current solution since it is interpreted as representing an improving move during the search process. When, however, the energy difference between the two solutions is large and positive, and the temperature is low, the probability of accepting the neighbouring solution is small. Whenever a move is accepted, the neighbouring solution becomes the new current solution during the next iteration of the search [148], and the number b of accepted moves per epoch is incremented by 1.

Lines 14–21 of Algorithm 6.12 represent the handling of a non-dominated solution, where Line 14 specifically represents the process of counting the number of solutions dominating the current solution. If $|\mathcal{A}_{\mathcal{R}}| = 0$, then the current solution is non-dominated and should be included in the archive $\mathcal{A}$. When a new solution is inserted into the archive $\mathcal{A}$, the number of accepted moves b is decremented by 1 in order to compensate for the fact that the current move is not accepted as a non-improving move, but instead as an improving move. Furthermore, the $poor\_epoch\_flag$ is lowered by setting it equal to zero and the poor epoch counter ϵ is also reset to zero. Lines 19–21 represent the removal of those solutions from the archive that are dominated by the current solution $\mathcal{R}$.

If, however, the current solution is not accepted in Line 11, the else if statement is invoked in Line 22, testing whether the maximum number of reheatings $H_{\max}$ has been reached. If it has not been reached, the attempts per epoch counter a is incremented, and if a is larger than $A_{\max}$, then the system temperature is updated by reheating the temperature by a factor ζ, and the total number of reheats h is incremented. Moreover, the `Break` statement breaks out of the current while loop in Line 27, thereby entering a new epoch. The selection probabilities $p_1, \ldots, p_I$ and success scores $q_1, \ldots, q_I$ are updated during each iteration according to the LLH management strategy update rules. Any one of the three LLH management strategies defined in §6.1.6 may be used in this regard.

The stopping criteria for the inner while-loop are found in Line 8 of Algorithm 6.12. This loop controls the temperature decrease per epoch and the criteria state that as long as the number of accepted moves per epoch b is smaller than the minimum allowed number of accepted moves per epoch $B_{\min}$ and the iteration counter t is smaller than the allowable number $L_{\max}$ of iterations per epoch, new neighbouring solutions should continue to be generated and tested

for inclusion in the archive. The value of $B_{\min}$ is chosen as 25 in this dissertation, based on empirical experimentation.

The overall stopping criteria are found in Line 5 of Algorithm 6.12 and are met when the maximum number of epochs $C_{\max}$ that may elapse without inserting any solution into the archive is reached, or when the maximum number of epochs $K_{\max}$ is reached. When the algorithm terminates, ϵ denotes the last number of epochs that have elapsed without accepting a solution. A value of $C_{\max} = 3$, as proposed by Dréo *et al.* [47], was initially considered, and then later adjusted to the value 400 upon empirical experimentation. The latter value is subsequently adopted in this dissertation. The maximum number of epochs is introduced to avoid a situation where the solutions are non-improving and computational power is wasted, and the value of $K_{\max}$ is taken as 1 000 in this dissertation, as suggested by Fan and Mumford [50].

When the set of non-dominated solutions $\mathcal{P}_S$ is returned by Algorithm 6.12, further exploration can be performed of the search space by restarting the SA algorithm, but instead of generating an initial feasible solution as described above, taking each solution in turn from the current non-dominated set $\mathcal{P}_S$ as the initial solution for the trajectory-based search [148].

6.1.8 NSGA II implementation

The GA employed in this dissertation to solve instances of the UTRP also incorporates the ICRSGP and NAP described above, and guides the population of solutions to converge towards a set of high-quality non-dominated solutions. The particular GA variant implemented in this dissertation is the NSGA II proposed by Deb *et al.* [42]. A pseudo-code description of this procedure within the context of the UTRP is given in Algorithm 6.13.

Algorithm 6.13 takes as inputs the UTRP instance parameters $\boldsymbol{W}$, $\boldsymbol{D}$, $m_{\min}$, $m_{\max}$ and r described in §5.1. In addition, the required number $L_{\max}$ of generations is an input parameter to the algorithm, taken here as $L_{\max} = 200$, as suggested by John *et al.* [90] and Mumford [121]. An initial population size N is also required as an input parameter to the NSGA II, chosen here as $N = 200$ [90, 121].

The algorithm is initialised by an initial population $\boldsymbol{P}_0$ of $N = 200$ random feasible route sets $\mathcal{R}$, generated according to the ICRSGP described in §6.1.2. Next, members of the initial population $\boldsymbol{P}_0$ are ranked and sorted using the FNSA (refer to Algorithm 2.7), described in §2.5. The crowding distance of each solution is then determined by applying the crowding distance assignment algorithm (refer to Algorithm 3.4) to $\boldsymbol{P}_0$. A child population $\boldsymbol{Q}_0$ is generated based on the crowding distance operator, incorporating the aforementioned crowding distance values of $\boldsymbol{P}_0$, for selection of parents during binary tournament selection. Child solutions are formed by applying crossover and mutation operators specific to the UTRP [90, 121].

Crossover operators typically combine information about parent solutions in order to produce offspring solutions [89]. The crossover operator adopted in this dissertation for the UTRP is due to Mumford [121]. This crossover operator was explained fully in §6.1.2. The approach involves maximising the proportion of missing vertices included when crossover occurs between two parent routes. Crossover can then be applied by presenting two parent solutions, $\mathcal{R}_1$ and $\mathcal{R}_2$, as input to Algorithm 6.4.

In this dissertation, two extensions are proposed for inclusion in Mumford's crossover procedure. The first is that instead of deterministically selecting the route that maximises the proportion of missing vertices added, a route is selected stochastically with a probability proportional to the proportion of missing vertices included. The other extension is applying the remove subsets LLH (operator (m) in §6.1.5) after having applied each crossover, as the crossover operation

Algorithm 6.13: NSGA II for the UTRP.

Input : An $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, an $n \times n$ demand matrix $\boldsymbol{D}$, the minimum allowed number of vertices in a route $m_{\min}$, the maximum allowed number of vertices in a route $m_{\max}$, the number of required routes r, the maximum number of generations $L_{\max}$, the population size N, and a set of LLHs.

Output: A set of non-dominated solutions $\mathcal{P}_S$ in the solution space together with each solution's objective function values captured in a vector P_O in objective space.

Generate an initial population $\boldsymbol{P}_0$ comprising N random route sets $\mathcal{R}$, each containing r routes, by means of the enhanced ICRSGP of §6.1.3
Initialise the selection probabilities $p_1, \ldots, p_I$ for the LLHs
Use the FNSA [Algorithm 2.7] to rank and sort $\boldsymbol{P}_0$
Use the crowding distance assignment algorithm [Algorithm 3.4] to determine the crowding distance of each solution in $\boldsymbol{P}_0$
Generate a child population $\boldsymbol{Q}_0$ of size N from $\boldsymbol{P}_0$, by crossover and mutation, using binary tournament selection, based on the $\prec_c$ crowding distance operator
Repair any infeasible offspring; if irreparable, generate further offspring from the parents until $|\boldsymbol{Q}_0| = N$
$t \leftarrow 0$
while $t < L_{\max}$ **do**
    $\boldsymbol{R}_t \leftarrow \boldsymbol{P}_t \cup \boldsymbol{Q}_t$
    Rank and sort $\boldsymbol{R}_t$ by means of the FNSA into non-dominated fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$
    $\boldsymbol{P}_{t+1} \leftarrow \emptyset$
    $k \leftarrow 1$
    while $|\boldsymbol{P}_{t+1}| < N$ **do**
        if $|\mathcal{F}_k| + |\boldsymbol{P}_{t+1}| \leq N$ **then**
            $\boldsymbol{P}_{t+1} \leftarrow \boldsymbol{P}_{t+1} \cup \mathcal{F}_k$
        else
            Determine the crowding distance for each solution in $\mathcal{F}_k$
            Sort the solutions in $\mathcal{F}_k$ in descending order of crowding distance
            $\boldsymbol{P}_{t+1} \leftarrow \boldsymbol{P}_{t+1} \cup$ [top $(N - |\boldsymbol{P}_{t+1}|)$ solutions in the sorted set $\mathcal{F}_k$]
        $k \leftarrow k + 1$
    Determine the crowding distance for each solution in $\boldsymbol{P}_{t+1}$
    Generate a child population $\boldsymbol{Q}_{t+1}$ from $\boldsymbol{P}_{t+1}$ by crossover and mutation, using binary tournament selection based on the $\prec_c$ crowding distance operator
    Repair any infeasible offspring; if irreparable, generate further offspring from the parents until $|\boldsymbol{Q}_{t+1}| = N$
    Update the selection probabilities $p_1, \ldots, p_I$ according to the AMALGAM LLH management strategy for each LLH
    $t \leftarrow t + 1$
$\mathcal{P}_S \leftarrow$ get non-dominated set obtained over all populations $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_{L_{\max}}$
$P_O \leftarrow$ return objective function values for each solution $\mathcal{R}$ in $\mathcal{P}_S$
return [$\mathcal{P}_S$, P_O]

may, at times, lead to subsets of routes being included in the offspring solution which are never beneficial in the context of the UTRP, as explained in §6.1.5. When subsets are replaced, two approaches may be followed: A subset of a route is either replaced by another route in one of the parents, maintaining the qualities of the parents, or else a different pool of routes may be introduced from which the replacement route may be drawn. The set of K shortest paths generated for the UTRP instance may, for example, be utilised for this selection instead. A total of six combinations may be obtained if these three components are considered. The relative performances of these combinations are tested in the next chapter.

The repair strategy mentioned in §6.1.2 is repeated for all missing vertices. If a route set cannot be repaired, the offspring is abandoned and the next two parents are selected for crossover. Offspring are created and added to $\boldsymbol{Q}_0$ in this fashion until $|\boldsymbol{Q}_0| = N$.

The LLHs described in §6.1.5 are employed as mutation operators in the solution approach adopted in this dissertation. The probability that an offspring solution is mutated is specified by the decision maker, and remains constant throughout the search. For each offspring solution, a random number is generated in the unit interval, and if this number is smaller than the pre-specified mutation probability, a mutation operator is selected and applied to the respective offspring solution. This is achieved by generating another random number in the unit interval, upon which LLH i is selected if the random number falls within the cumulative selection probability interval of the respective LLH, based on the selection probability p_i, as described in §6.1.6. These selection probabilities are updated after each generation according to the LLH management strategy update rule described in §6.1.6, as may be seen in Line 24 of Algorithm 6.13.

Next, the initial population $\boldsymbol{P}_0$ and the offspring population $\boldsymbol{Q}_0$ are merged, and subsequently ranked and sorted into non-dominated fronts using the FNSA (refer to Algorithm 2.7), pertaining to the objective functions (5.5) and (5.6). The next population $\boldsymbol{P}_1$ is then selected, based first on the non-dominated rank, and then on the crowding distance to break ties. This procedure is repeated iteratively, until $L_{\max}$ generations have elapsed, as described in Lines 8–25 of Algorithm 6.13. Finally, the non-dominated set $\mathcal{P}_S$ is extracted from all the populations $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_{L_{\max}}$, along with the objective function values P_O of this set, and returned as output at termination of the algorithm.

6.2 UTFSP model solution implementation

Due to the complexity of the UTFSP, metaheuristics were also considered for solving instances of this problem, as is common in the literature [82, 117]. Wu *et al.* [174] and Yu *et al.* [181], for example, employed GAs to solve variants of the UTFSP. Few works in the literature have used trajectory-based searches to solve instances of this problem [82], and given the nature of the decision variables and constraints, GAs indeed seem a natural approach towards solving instances of the UTFSP approximately.

The UTRP solution methodology proposed by Fan and Machemehl [51, 53] was adapted for the UTFSP. This methodology consists of three main components, namely an *initial candidate frequency set generation procedure* (ICFSGP), an NAP, and a metaheuristic search that combines the ICFSGP and NAP. The mechanisms applied by John [89] are adopted for the UTFSP in this dissertation.

This section opens in §6.2.1 with a description of the ICFSGP incorporated into the solution approach, and thereafter contains a short discussion on the NAP adopted, in §6.2.2. A population-based metaheuristic search technique, in combination with the ICFSGP and the

NAP, guides the search for high-quality solutions to the UTFSP model formulation of §5.3. The NSGA II [42] is chosen for this purpose due to its flexibility and success in previous attempts [89] at solving instances of the UTFSP, leading to high-quality solutions within reasonable time frames. The author's implementation of the NSGA II in the computer language Python [162] is finally described in detail in §6.2.3.

6.2.1 Initial candidate frequency set generation procedure

The initialisation of the NSGA II requires that a population of feasible solutions be produced. For the UTFSP, an initial route set $\mathcal{R}$ is provided as input to the model, and for each route in this route set a frequency has to be assigned from a pre-specified frequency set, denoted by θ. The frequency set $\theta = \left\{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\right\}$, employed by Martínez *et al.* [117], is adopted in this dissertation. In order to ensure that the initial population of frequencies is feasible, frequencies that are either lower than the pre-specified minimum allowable frequency $f_{\min}$ or higher than the pre-specified maximum allowable frequency $f_{\max}$ can be removed. An initial frequency set $\mathcal{F}$ is then constructed by random uniform sampling from the set θ, with replacement, until $|\mathcal{R}|$ frequencies have been inserted into the set $\mathcal{F}$.

6.2.2 Network analysis procedure

The NAP is responsible for evaluating the network in terms of its performance. As described in §5.3, the UTFSP model adopted in this dissertation has two objective functions that have to be evaluated, the AETT and the TBR. Of the two, the AETT is the most complex to evaluate, and requires that the underlying transit assignment model be solved, so that the AETT can be determined after passengers have been assigned to their respective paths. For brevity, the arcs of the graph $G'_{\mathcal{R}}$ is denoted by $\mathcal{A}$ instead of $\mathcal{E}'_{\mathcal{R}}$, and this transit graph is taken as input to solve the transit assignment model, as described in Algorithm 6.14. The remaining input required for the optimal strategies [151] demand assignment model are the cost and frequency associated with each arc in $\mathcal{A}$, and a demand matrix $\boldsymbol{D}$, containing all the passenger demand from vertex i to vertex j in $\mathcal{V}$.

The optimal strategies assignment model works on the basis that during a first pass, in Lines 1–12, one destination vertex r is chosen and then in a backward fashion with respect to all the other vertices in $\mathcal{V}$ (excluding r itself), the optimal strategy $\bar{\mathcal{A}}^*$ is determined along with the expected total travel time ${u_i}^*$ from each vertex $i \in \mathcal{V}$ to the destination node r. During a second, forward pass, the demand is assigned, according to the optimal strategy from all origins to the destination r [151].

The algorithm is initialised in Lines 1–5, where a vertex label u_i is assigned to each vertex $i \in \mathcal{V}$, denoting the expected travel time from vertex i to reach the destination vertex r. The initial values for these labels are set to ∞, with the exception that u_r is set to 0. These labels are updated throughout the execution of the algorithm. Auxiliary variables are constructed, denoted by f_i for each $i \in \mathcal{V}$, and contain the combined frequencies of all the currently selected arcs adjacent to vertex i, initially set to zero [151]. Furthermore, the set $\mathcal{S}$ is used to identify these arcs that have not been examined within the network, whereas a set $\bar{\mathcal{A}}$ contains the optimal passenger strategy, consisting of arcs and taking the frequencies of each arc into account.

Lines 6–12 in Algorithm 6.14 contain the method for selecting those arcs that should be included in the optimal strategy set $\bar{\mathcal{A}}$. This is achieved by selecting the arc a which yields the shortest travel time to the destination vertex r from vertex i, among the arcs not yet selected, if it were

Algorithm 6.14: Optimal strategies demand assignment algorithm [151].

Input : A graph $G'_{\mathcal{R}} = (\mathcal{V}'_{\mathcal{R}}, \mathcal{A})$ representing the transit network as defined in §5.3, with each arc $a \in \mathcal{A}$ having a corresponding cost c_a and frequency f_a associated with it, an $n \times n$ demand matrix $\boldsymbol{D}$, and a destination vertex r.

Output: The demand assignment for each arc $a \in \mathcal{A}$.

1 $u_i \leftarrow \infty$ for all $i \in \mathcal{V} \setminus \{r\}$
2 $u_r \leftarrow 0$
3 $f_i \leftarrow 0$ for all $i \in \mathcal{V}$
4 $\mathcal{S} \leftarrow \mathcal{A}$
5 $\bar{\mathcal{A}} \leftarrow \emptyset$
6 **while** $\mathcal{S} \neq \emptyset$ **do**
7 find $a = (i, j) \in \mathcal{S}$ which satisfies $u_j + c_a \leq u_{j'} + c_{a'}$ for any $a' = (i', j') \in \mathcal{S}$
8 $\mathcal{S} \leftarrow \mathcal{S} \setminus \{a\}$
9 **if** $u_i \geq u_j + c_a$ **then**
10 $u_i \leftarrow \dfrac{f_i u_i + f_a(u_j + c_a)}{f_i + f_a}$
11 $f_i \leftarrow f_i + f_a$
12 $\bar{\mathcal{A}} \leftarrow \bar{\mathcal{A}} \cup \{a\}$
13 $H_i \leftarrow d_{ir} \in \boldsymbol{D}$ for each $i \in \mathcal{V}$
14 Sort all the arcs $a \in \mathcal{A}$ in decreasing order of $(u_j + c_a)$
15 **foreach** $a \in \mathcal{A}$ **do**
16 **if** $a \in \bar{\mathcal{A}}$ **then**
17 $h_a \leftarrow \dfrac{f_a}{f_i} H_i$
18 $H_j \leftarrow H_j + h_a$
19 **else**
20 $h_a \leftarrow 0$
21 **return** $[H, h, u]$

to be added to the optimal strategy. The sum $u_j + c_a$, denoting the time required to progress from vertex i in arc $a = (i, j)$ to the destination vertex r without accounting for waiting time at vertex i, is considered when choosing an arc. If this time $u_j + c_a$ is smaller than the current time u_i associated with vertex i, the arc a is included in the optimal strategy $\bar{\mathcal{A}}$, and u_i and f_i are updated according to Lines 10 and 11, respectively. It should be noted that when the label u_i of a vertex i is initially improved, the product of $f_i u_i = 0 \times \infty$ occurs. For compactness of the algorithm, Spiess and Florian [151] adopted the convention that whenever $f_i u_i = 0 \times \infty$, it is simply replaced by $f_i u_i = \phi$, where ϕ is taken as 0.5 in this model. The algorithm proceeds when all of the arcs in $\mathcal{S}$ have been examined.

In the next part of the algorithm, Lines 13–20, demand is assigned across the transit network based on the optimal strategy identified in Lines 6–12. The demand flowing from vertex i to destination r is denoted by H_i. The initial value is found in the demand matrix $\boldsymbol{D}$ as the entry d_{ir}, and is updated as the algorithm progresses. The proportion of demand flowing from each vertex is assigned across the adjacent arcs corresponding to their frequencies. The list of all the arcs and their characteristics, f_a and $u_j + c_a$, along with a binary variable indicating whether they form part of an optimal strategy, is then sorted in decreasing order based on the values of $u_j + c_a$.

This ordered list is traversed, and if an arc is found in the optimal strategy set $\bar{\mathcal{A}}$, the labels h_a and H_i are updated according to Lines 17 and 18, respectively, where h_a denotes the volume of passengers flowing through that arc. If the arc is not included in an optimal strategy, the demand flowing through the arc is set to zero instead [151]. Due to the arcs being processed in reverse topological order, the vertex volumes may be assigned in parallel, thus making it possible for each arc to be examined only once. After every arc has been considered in the demand assignment part, the volume of passengers flowing through each vertex i, denoted by H_i, and through each arc a, denoted by h_a, are returned as output by the algorithm, along with the corresponding expected travel time u_i from vertex i to vertex r.

Due to only taking vertex r as the destination, only that portion of the demand flowing from the various origins to that specific destination is taken into account. The algorithm can therefore be adapted by implementing it for each of the destination nodes sequentially, and then summing together all the demand values for each vertex and each arc [151]. The complexity of this algorithm is determined by the computation of the optimal strategy set in Lines 6–12, and is therefore $\mathcal{O}(|\mathcal{V}^2|)$ for each destination vertex. Spiess and Florian [151], however, pointed out that this complexity can be reduced to $\mathcal{O}(\log|\mathcal{V}|)$ if heap structures are incorporated. When computing the demand assignment for each destination vertex in the network, the complexity is multiplied by another factor of $|\mathcal{V}|$.

Evaluating the AETT objective function (5.8) requires the output of Algorithm 6.14, and then involves multiplication of the expected travel time u_{ij} by the demand d_{ij} for each OD pair, divided by the total demand. Evaluating the TBR objective function in (5.9), on the other hand, is a much simpler task, and requires that twice the route length, measured in minutes, for each route be multiplied by the corresponding frequency of vehicles operating along that route, and then taking the sum of these products.

6.2.3 NSGA II implementation

The GA employed in this dissertation to solve instances of the UTFSP incorporates the ICFSGP and NAP described above, and iteratively guides the population of solutions towards successive improvements of the non-dominated front. The NSGA II proposed by Deb *et al.* [42] is the particular GA variant chosen for this purpose. Algorithm 6.15 contains a pseudo-code description of this procedure within the context of the UTFSP.

The algorithm takes as input an $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, and a route set $\mathcal{R}$ which, when superimposed on G, forms the transit network graph $G'_{\mathcal{R}}$, as mentioned in §5.3. An $n \times n$ demand matrix $\boldsymbol{D}$ is also provided as input to the algorithm, along with the minimum allowed frequency for a route $f_{\min}$ and the maximum allowed frequency for a route $f_{\max}$, chosen as $\frac{1}{30}$ and $\frac{1}{5}$, respectively, following John [89]. Furthermore, the GA parameters (the maximum number of generations $L_{\max}$ and the population size N) are also provided as input, both chosen as 200 (the same value selected by John [89]).

The set $\theta = \left\{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\right\}$ of allowed frequencies for the network is also provided and so the decision variables of the UTFSP model can take any value in this set. This discretisation of frequencies is employed to reduce the size of the search space. The discretised frequency set θ corresponds to values in the range twelve buses per hour to two buses per hour. It is acknowledged that this discretisation may potentially lead to some unexplored areas in the non-dominated front, due to the actual frequency of a route being determined by division of the number of buses assigned to the route by the round trip time of that route. When considering the schedules of buses, however, it makes sense to have well-defined intervals for bus arrivals (such as those induced by the set θ), and this may be achieved by assigning

Algorithm 6.15: NSGA II for the UTFSP.

Input : An $n \times n$ weight matrix $\boldsymbol{W}$ representing the graph G, a route set $\mathcal{R}$, an $n \times n$ demand matrix $\boldsymbol{D}$, the minimum allowed frequency of a route $f_{\min}$, the maximum allowed frequency of a route $f_{\max}$, a set θ containing the allowed frequencies, the maximum number of generations $L_{\max}$, and the population size N.

Output: A set of non-dominated solutions $\mathcal{P}_S$ in the solution space and each solution's objective function values as a vector P_O in objective space.

Generate an initial population $\boldsymbol{P}_0$ comprising N random frequency sets $\mathcal{F} \in \theta$, specifying a frequency for each route in $\mathcal{R}$
Use the FNSA [Algorithm 2.7] to rank and sort $\boldsymbol{P}_0$
Use the crowding distance assignment algorithm [Algorithm 3.4] to determine the crowding distance of each solution in $\boldsymbol{P}_0$
Generate child population $\boldsymbol{Q}_0$ of size N from $\boldsymbol{P}_0$, by applying crossover and mutation, using binary tournament selection, based on the $\prec_c$ crowding distance operator
$t \leftarrow 0$
while $t < L_{\max}$ **do**
    $\boldsymbol{R}_t \leftarrow \boldsymbol{P}_t \cup \boldsymbol{Q}_t$
    Rank and sort $\boldsymbol{R}_t$ using FNSA into non-dominated fronts $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$
    $\boldsymbol{P}_{t+1} \leftarrow \emptyset$
    $k \leftarrow 1$
    while $|\boldsymbol{P}_{t+1}| < N$ **do**
        if $|\mathcal{F}_k| + |\boldsymbol{P}_{t+1}| \leq N$ **then**
            $\boldsymbol{P}_{t+1} \leftarrow \boldsymbol{P}_{t+1} \cup \mathcal{F}_k$
        else
            Determine the crowding distance of each solution in $\mathcal{F}_k$
            Sort the solutions in $\mathcal{F}_k$ in descending order of crowding distance
            $\boldsymbol{P}_{t+1} \leftarrow \boldsymbol{P}_{t+1} \cup$ [top $N - |\boldsymbol{P}_{t+1}|$ solutions in the sorted set $\mathcal{F}_k$]
        $k \leftarrow k + 1$
    Determine the crowding distance of each solution in $\boldsymbol{P}_{t+1}$
    Generate a child population $\boldsymbol{Q}_{t+1}$ from $\boldsymbol{P}_{t+1}$ by applying crossover and mutation, using binary tournament selection based on the $\prec_c$ crowding distance operator
    $t \leftarrow t + 1$
$\mathcal{P}_S \leftarrow$ `get_non-dominated_set`$(\boldsymbol{P}_{L_{\max}})$
$P_O \leftarrow$ return objective function values for each solution $\mathcal{F}$ in $\mathcal{P}_S$
return $[\mathcal{P}_S, P_O]$

the number of buses that will at least yield the required frequency, also making provision for unforeseen events that could occur along routes like traffic congestion and vehicle breakdowns. The decision maker may, of course, impose his or her own set of desired frequencies upon θ, as desired.

Algorithm 6.15 is initialised in Line 1, where an initial population of frequencies of size N is generated by means of the ICFSGP described in §6.2.1. Next, the initial population is ranked and sorted by applying the FNSA (see Algorithm 2.7), after which the crowding distance assignment algorithm (see Algorithm 3.4) is applied to the sorted population. A child population of size N is then generated by the application of crossover and mutation, after pairs of parents have been selected by binary tournament selection. The crossover technique adopted is that of single-point crossover, as described in §3.3.3, where a crossover point is selected uniformly. The crossover probability is chosen as 0.9 (the same value as that selected by John [89]).

The mutation technique employed is similar to bit-flip mutation, as described in §3.3.3, but instead of having only binary variables, the possibilities are the entries of θ. Therefore, when a specific frequency is selected within the frequency set $\mathcal{F}$, an equal probability is applied to either choosing the value in θ that is one position above or below the current value of the frequency in question. When, however, the frequency is selected that is either the first or last entry of θ, a wrap-around concept is applied. For example, if the last frequency is the current value under consideration, the two options are the previous frequency in θ, or the first entry of θ, and *vice versa* for the first entry. The mutation probability is set to $\frac{1}{|\mathcal{R}|}$ for each entry in $\mathcal{F}$.

Thereafter, the parent population $\boldsymbol{P}_t$ and the child population $\boldsymbol{Q}_t$ are merged to form the combined set $\boldsymbol{R}_t$, where t denotes the generation counter, which is set to zero initially. The set $\boldsymbol{R}_t$ is ranked and sorted into its various non-dominated fronts using the FNSA. A new population $\boldsymbol{P}_{t+1}$ is then formed and populated by choosing the best solutions from $\boldsymbol{R}_t$, first based on rank, and thereafter based on crowding distance in order to resolve ties, as described in Lines 9–18. More specifically, Lines 12–13 indicate that if the sum of the number of solutions in front $\mathcal{F}_k$ and the number of solutions currently in $\boldsymbol{P}_{t+1}$ is less than N, then all the solutions of the non-dominated front $\mathcal{F}_k$ are inserted into $\boldsymbol{P}_{t+1}$. This process is repeated until $|\mathcal{F}_k| + |\boldsymbol{P}_{t+1}| > N$, and then the solutions in the currently observed front with the largest crowding distances are selected.

The new population $\boldsymbol{P}_{t+1}$ is subsequently used to generate a new child population $\boldsymbol{Q}_{t+1}$, and Lines 7–21 are repeated until $L_{\max}$ generations have been reached, at which point the non-dominated set $\mathcal{P}_S$ of the solutions of population $\boldsymbol{P}_{L_{\max}}$ is returned, along with the accompanying objective function values P_O.

6.3 Chapter summary

This chapter was devoted to descriptions of the various solution methodologies implemented by the author for solving instances of the UTRP and the UTFSP. The discussion in §6.1 focussed on the solution method implemented for solving instances of the UTRP. The section opened in §6.1.1 with a description of the data structure employed to represent candidate solutions. It was further described how the solution method adopted consists of three phases, namely an ICRSGP, an NAP, and a metaheuristic search procedure. The first phase was discussed in §6.1.2, containing some novel contributions, and an enhancement procedure was proposed in §6.1.3 for improving the solution quality of solutions returned by adopting the approaches of §6.1.2. The second phase was discussed in §6.1.4, where several computational obstacles were mentioned, as well as how these were overcome. Sections 6.1.5 and 6.1.6 contained discussions on various

prerequisite components for the search procedures. The LLHs employed in this dissertation were discussed in §6.1.5, and a hyperheuristic approach towards managing these LLHs for trajectory-based and population-based searches was proposed in §6.1.6.

The last phase involves the application of either of two different metaheuristic search types, namely a trajectory-based metaheuristic and a population-based metaheuristic. The trajectory-based approach is the DBMOSA algorithm of Smith *et al.* [148], as discussed in §6.1.7, while the population-based approach is the NSGA II of Deb *et al.* [42], as discussed in §6.1.8. The algorithmic implementations and their various components were explained systematically. Thereafter, the solution method adopted in this dissertation for solving instances of the UTFSP was discussed in §6.2. The discussion followed a similar structure as that for the UTRP, and covered an ICFSGP, an NAP, and a metaheuristic search procedure. The first two of these components were discussed in §6.2.1 and §6.2.2, respectively. The NSGA II [42] was chosen as the metaheuristic search technique, and the algorithmic implementation was discussed in §6.2.3.

The verification and validation of these solution approaches, as well as the results returned by these methods, are the topic of the next chapter.

CHAPTER 7

Validation of algorithms

Contents

This chapter is devoted to the presentation of numerical validation results obtained when applying the algorithms of the previous chapter to benchmark instances of the UTRP and the UTFSP. A brief discussion on the novel ICRSGP of §6.1.2 and §6.1.3 is presented and the results returned by the procedure are touched upon in §7.1. Thereafter, the three algorithms of the previous chapter for solving instances of the UTRP and the UTFSP are validated in respect of well-known benchmark instances. In particular, the DBMOSA algorithm and the NSGA II for solving instances of the UTRP are validated separately in §7.2 and §7.3, and are then combined in §7.4 to test whether benefit might thus be gained in terms of solution quality. Next, instances of the novel UTRP with incremental redesign are presented and solved in §7.5. Thereafter, the NSGA II for solving instances of the UTFSP is validated in §7.6. The chapter finally closes in §7.7 with a short summary of its contents. All results reported in this chapter are available online[1].

Throughout this chapter, the approach followed when testing and validating the UTRP solution implementations involves three distinct phases, namely minor tests, parameter testing, and performance runs. The minor tests involve fine-tuning certain facets of the algorithmic implementation, experimenting with new techniques and approaches, or determining which sub-algorithms work best. The parameter testing phase constitutes the setting of the various metaheuristic parameters relevant to each algorithmic implementation after having incorporated changes based on the outcomes of the minor tests. Finally, performance runs are executed based on the best parameter values uncovered during parameter testing, and involve allowing the searches to run for longer intervals than in the previous two testing phases in order to observe the performance of the algorithmic implementations. Only the three smallest benchmark instances described in §4.5.8 are considered during the minor tests, due to time practicalities, but all five benchmark

[1] `https://github.com/GHuss7/PhD_Results`

instances described in §4.5.8 are considered during parameter testing and when executing performance runs. A table is provided for each of these phases to help guide the reader through the validation process. This elaborate approach is not followed for the UTFSP solution implementation due to its solution implementation being much simpler than that of the UTRP.

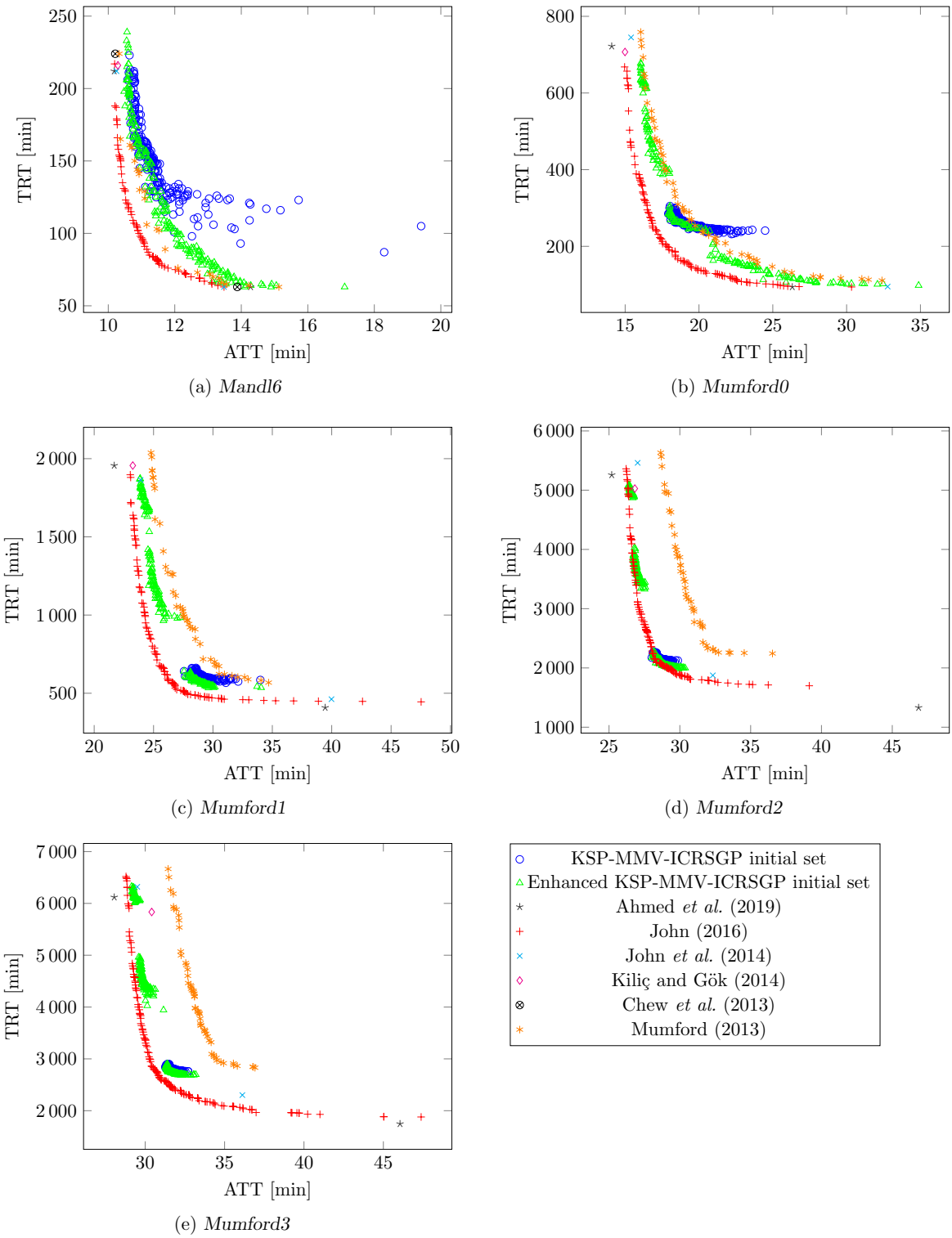
7.1 UTRP ICRSGP results

Choosing an adequate ICRSGP when solving instances for the UTRP is crucial, as starting out from poor initial solutions may lead to premature convergence for population-based searches, and also hamper exploitation by such searches. Initially, the establishment of high-quality initial solutions was one of the main obstacles that had to be overcome. It was observed for the Mandl6 instance that the quality of the initial solutions does not influence the final solutions significantly, as good final solutions are eventually found due to the small size of the benchmark instance. The quality of the initial solution set does, however, become increasingly important when considering larger problem instances. During prior testing, it was established that among the ICRSGPs mentioned in §6.1.2, the KSP-MMV-ICRSGP performed the best. The KSP-MMV-ICRSGP was therefore applied to all five UTRP benchmark instances mentioned in §4.5.8, generating 2 000 initial solutions and saving them in *csv* files. The ICRSGP enhancement procedure of §6.1.3 was then applied to the 2 000 solutions returned by the KSP-MMV-ICRSGP for each benchmark instance, and the 8 000 new solutions thus obtained were also saved in a different set of *csv* files.

When an initial solution set had to be generated, the FNSA (Algorithm 2.7) was applied to the solutions retrieved from the saved *csv* files, the crowding distance operator (Algorithm 3.4) applied, and the best required number of solutions selected based on the crowding distance. For visualisation purposes of the quality of these initial solutions, 200 were selected. These initial solutions, along with the results reported by other researchers, are presented in Figure 7.1 for the five benchmark instances mentioned in §4.5.8. The blue circles and the green triangles represent the initial solutions returned by the KSP-MMV-ICRSGP and the enhanced KSP-MMV-ICRSGP, respectively, in objective space. It is interesting to note how the initial solutions returned by the KSP-MMV-ICRSGP become all the more concentrated in a smaller area in the objective space as the problem instance size increases. The subfigures of Figure 7.1 are conveniently presented in order of increasing instance size. It is also evident how the enhanced KSP-MMV-ICRSGP spreads out these concentrated initial solutions returned by the KSP-MMV-ICRSGP over wider regions of the attainment fronts reported by other researchers, indicating that a good solution spread has been achieved. The high quality of the initial solutions returned by the enhanced KSP-MMV-ICRSGP is also clear as the attainment fronts of the initial solutions are in close proximity to the attainment fronts reported by John [89], which were, at the time of writing, the best reported multi-objective non-dominated attainment fronts reported in the literature for the five benchmark instances.

7.2 UTRP results returned by the DBMOSA algorithm

In order to validate the implementation of the DBMOSA solution process for the UTRP described in the previous chapter, the methodology was applied to the five benchmark problems mentioned in §4.5.8. Recall that the names of these instances are Mandl6, Mumford0, Mumford1, Mumford2, and Mumford3. This section is partitioned into subsections devoted to minor tests, parameter tests, and performance runs.



(a) *Mandl6*

(b) *Mumford0*

(c) *Mumford1*

(d) *Mumford2*

(e) *Mumford3*

FIGURE 7.1: *Initial solution sets returned by the KSP-MMV-ICRSGP and the enhanced KSP-MMV-ICRSGP of the previous chapter, together with the final results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8.*

7.2.1 Minor tests

When considering candidate solution sets to a multi-objective optimisation problem, the principles discussed in §2.6 should be considered when evaluating the relative quality of these solution sets. The performance metric chosen to evaluate the quality of solution sets in this chapter is HV, due to its incorporation of the four main Pareto front quality aspects, as well as additional benefits, as discussed in §2.6.2. When adopting the HV performance indicator, it is important to choose the reference point appropriately. The reference points chosen for minimum and maximum ATT, as well as the minimum and maximum TRT, adopted in this chapter for each benchmark UTRP instance are shown in Table 7.1.

TABLE 7.1: *UTRP benchmark instance HV reference points adopted in this chapter, all measured in minutes.*

Instance	Minimum ATT	Maximum ATT	Minimum TRT	Maximum TRT
Mandl6	10	15	63	224
Mumford0	13	32	94	700
Mumford1	19	50	345	2 000
Mumford2	22	45	864	6 000
Mumford3	24	50	982	6 600

The values in Table 7.1 were used to measure the normalised HV for a given set of non-dominated solutions, as described in §2.6.2. The lower bounds on the TRT in Table 7.1 were calculated by determining minimum spanning trees for the networks, since the weights of these trees represent theoretical lower bounds on the TRT that operators would be able to achieve. These lower bounds on the TRT were calculated by Mumford [121], and were tabulated in Table 4.3. The lower bounds on the ATT in the table may be determined in the ideal case where all passengers have direct routes towards their destinations [121]. Mumford [121] also calculated these values, which were also tabulated in Table 4.3. The minimum reference points were therefore taken as the respective lower bounds on the ATT and TRT. Furthermore, a comparison of the published results of Mumford [121] and John [89] was performed, observing the bounds achieved by their results. These bounds led to the upper bounds in Table 7.1 (as well as a confirmation of the lower bounds).

The minor tests performed in respect of the DBMOSA implementation are summarised in Table 7.2. These tests were partitioned into sets, numbered 1 to 4, each of which involved three test instances. The minor tests were only performed on the three smallest benchmark instances, namely Mandl6, Mumford0 and Mumford1, as mentioned. The parameter values for each of the minor tests were kept constant, except for a single variable being tested, so that its effect could be measured. The baseline parameter values utilised for the tests in Table 7.2 may be found in Table 7.3. These values were chosen in such a way that the most basic components of a DBMOSA are at least represented. The values of the test parameters were incrementally changed, and the difference in HV obtained by averaging over the 10 runs was measured. If an improvement in HV was observed for the three instances within the minor test, the component tested was subsequently included in the DBMOSA final implementation, and used henceforth.

TABLE 7.2: *Minor test cases for the UTRP DBMOSA.*

Test set	Test number	Instance	Test parameter	Parameter values
1	1	Mandl6	Initial solutions	0, 1, 2
1	2	Mumford0	Initial solutions	0, 1, 2
1	3	Mumford1	Initial solutions	0, 1, 2
2	1	Mandl6	Perturbation management strategy	0, 1, 2, 3
2	2	Mumford0	Perturbation management strategy	0, 1, 2, 3
2	3	Mumford1	Perturbation management strategy	0, 1, 2, 3
3	1	Mandl6	Perturbations	All LLHs
3	2	Mumford0	Perturbations	All LLHs
3	3	Mumford1	Perturbations	All LLHs
4	1	Mandl6	Perturbation combinations	LLH subsets
4	2	Mumford0	Perturbation combinations	LLH subsets
4	3	Mumford1	Perturbation combinations	LLH subsets

TABLE 7.3: *The parameters considered during the minor tests when designing the DBMOSA algorithm for solving instances of the UTRP, along with the baseline value of each parameter.*

Parameter	Baseline value
Initial solutions	$\{0\}$
Perturbation management strategy	$\{3\}$
Perturbation LLHs	$\{a, b\}$
Number of runs	10
Termination criterion	First to breach
Maximum total iterations	40 000
Iterations for HV improvement comparison	4 000
HV improvement threshold	0.005%
Initial temperature	0.1
Maximum reheatings	5
Cooling rate	0.95
Reheating rate	1.1
Maximum iterations per epoch	100
Maximum poor epochs	400
Minimum accepts	25
Maximum attempts	50

Stepping through Table 7.3, a short reasoning is provided for each of these choices. In terms of initial solution generation, the procedure chosen entails generating a random feasible solution by means of the KSP-MMV-ICRSGP for each starting solution for the DBMOSA. A static approach was adopted in terms of the perturbation management strategy, thereby not updating any of the perturbation selection probabilities, but adopting a uniform selection probability for each LLH employed. In terms of the LLHs incorporated, the add vertex and delete vertex operators were chosen, as these two in combination have the ability to add and remove vertices to the transit network, and so sufficiently explore the search space. Preliminary testing showcased the success of these two operators in combination. They are denoted by a and b, respectively, in Table 7.3. The number of runs was set to 10.

One of three main stopping criteria were adopted in this chapter. The first is stopping by iterations, which requires termination of the algorithm after having performed a predefined number of iterations. The second is stopping by non-improving HV, which requires termination of the algorithm as soon as the HV improvement over some specified number of consecutive iterations drops below a pre-specified HV improvement threshold. The first to breach termination criterion, which is the last stopping criterion adopted, requires termination of the algorithm as soon as either of the first two stopping criteria are met. The latter criterion was utilised during the minor tests, where the maximum number of iterations was set to 40 000, the iterations for the HV improvement comparison was set to 4 000, and the HV improvement threshold was taken as 0.005%. These values were selected so as to allow the DBMOSA enough search iterations to find high-quality solutions. The remainder of the parameter values are specific to the DBMOSA algorithm, and good values for these SA parameters were established upon extensive testing in the context of the three benchmark problems at hand, so as to ensure a good performance generally for all three benchmark problems. A more in-depth discussion on these parameter values follows in the next section.

The first minor test considered in the context of the DBMOSA implementation was aimed at the initial solutions that should be adopted when solving the UTRP. Three approaches were considered. The first was the baseline approach mentioned above. The second approach entailed selecting the required number of equispaced starting solutions for the DBMOSA based on the crowding distance from the *csv* file containing the 2 000 initial solutions generated by the KSP-MMV-ICRSGP. The number of starting solutions was taken equal to the number of runs that had to be performed. The third and final approach was similar to the second, but with the exception that the initial solutions returned by the enhanced KSP-MMV-ICRSGP were adopted instead. These approaches are indicated in Table 7.2 by means of the parameter values as 0, 1 and 2, respectively.

Figure 7.2 contains box plots of the normalised HV obtained over 10 runs of the DBMOSA algorithm when solving the three benchmark instances mentioned in the figure. The solid diamonds represent the mean HV obtained over the runs, asterisks denote outliers, and the solid circles indicate the HV obtained by the non-dominated set formed by merging together the non-dominated sets of all the runs. It is clear that the mean HV obtained by the enhanced KSP-MMV-ICRSGP was the largest for the Mandl6 and Mumford1 instances, and the second largest for Mumford0. It is also clear that the random KSP-MMV-ICRSGP performed the worst among the three procedures considered. The enhanced KSP-MMV-ICRSGP was therefore the choice of ICRSGP going forward during the DBMOSA solution implementation for solving the UTRP.

The perturbation management strategy was alternated in the second minor test, where the 0, 1, 2 and 3 denote respectively the AMALGAM, every n-th AMALGAM, stochastic roulette-wheel, and static perturbation strategy. Based on empirical testing, the perturbation selection probability threshold was set to 3%, and the value of n was taken as 200, thereby updating the n-th AMALGAM perturbation strategy every 200 iterations. The results of Test set 2 may be found in Figure A.1 in Appendix A. It is evident that the stochastic roulette-wheel perturbation management strategy outperformed the other three strategies, and was therefore selected as the perturbation management strategy for the DBMOSA implementation. One of the main takeaways of this test was that there is value in updating the perturbation selection probabilities, and that further tests involving multiple LLHs may lead to deeper insight.

Test sets 3 and 4 were concerned with the LLHs that should be included in the perturbation management strategy. The DBMOSA was updated with the outcomes of Test sets 1 and 2, and therefore the initial solutions of the enhanced KSP-MMV-ICRSGP were utilised, and the

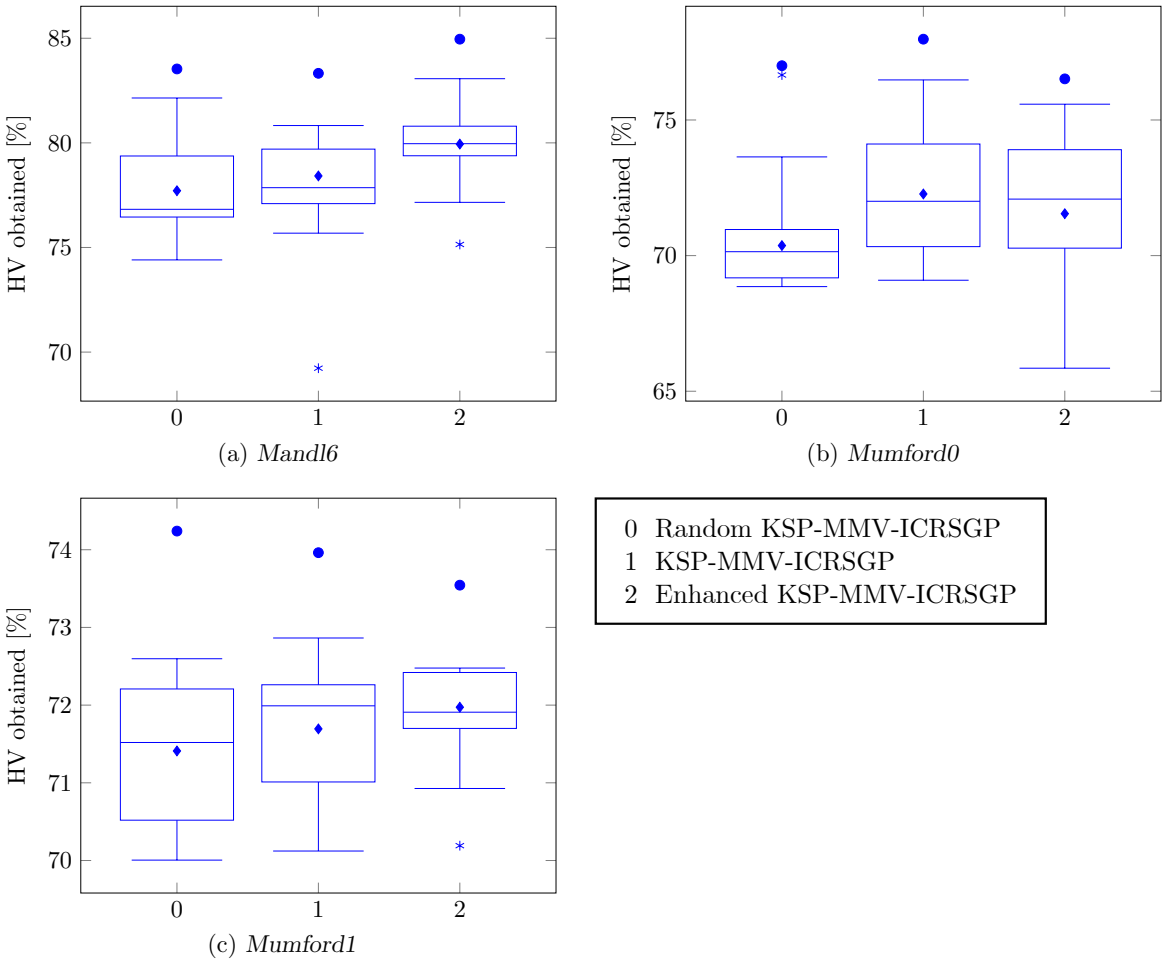


FIGURE 7.2: *ICRSGP analysis (Test set 1) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the ICRSGP, as indicated on the horizontal axes.*

stochastic roulette-wheel perturbation management strategy was incorporated. Test set 3 took various LLH combinations as input, as may be seen in the legend of Figure A.2 in Appendix A, where the letters correspond to the LLHs of Table 6.3. These combinations were chosen in such a way that there is a balance between the add and delete types of LLHs, so that the search can effectively traverse both extremal ends of the approximate Pareto set in objective space related to the ATT and TRT objective functions. Furthermore, counterpart LLHs were usually selected together, such as a and b, c and d, and p and q. The aim of this test set was to determine which LLH achieves the best performance, and also roughly how many LLHs should ideally be included in the set. It is interesting that the LLH set containing the most LLHs performed the best, as may be seen in Figure A.2 of Appendix A.

A further set of tests, Test set 4, was then conducted on five combinations of LLHs, but this time including at least six LLHs in each set, choosing the LLHs that are anticipated to make the most sense in a trajectory-based search context. These combinations of LLHs may be seen in the legend of Figure A.3 in Appendix A. The number of iterations for HV improvement comparison was increased to 8 000 for this test to ensure that no premature termination occurs, instead allowing enough search time as a result of the increase in the number LLHs considered. It is interesting to note that over the varying test problem sizes, different mutation combinations performed better. As the problem sizes increased, additional LLHs proved more beneficial. Also,

in the smaller two instances, containing 15 and 30 vertices, respectively, the random add and delete vertex perturbations performed better than the random cost-based grow and cost-based trim operators. This may be due to the smaller instances accommodating more randomness when searching through the smaller search space, whereas the larger instances have such large search spaces that providing additional information to help make smart choices may be beneficial so as not to waste too much search time on performing seemingly poor perturbations.

Based on performance, the various perturbation combinations that performed the best in each instance were chosen for that specific instance, and used throughout. For Mandl6, Mumford0, and Mumford1, combinations 0, 2 and 3 were selected, respectively, from Figure A.3 in Appendix A. Combination 0 was the LLH set $\{a, b, c, d, e, j\}$, 2 was the LLH set $\{a, b, c, d, e, g, h, j\}$, and 3 was the LLH set $\{c, d, e, g, h, j, p, q\}$, where the letters refer to the LLHs in Table 6.3. The parameters selected for Mumford1 are also subsequently used in the context of Mumford2 and Mumford3, as Mumford1 reflects the nature of the largest two problem instances better than the smaller two problem instances.

7.2.2 Parameter tests

The DBMOSA algorithm presented in §6.1.7 requires specification of various input parameters that affect the algorithmic performance. These parameter values were identified when designing the algorithm for best performance. The literature recommends various values for these parameters which were adopted as a starting point. Table 7.4 contains a list of parameter values that were considered, as well as the baseline value to which each parameter was set during experimentation. These values produced acceptable performance. When subsequently testing the suitability of a parameter value, the test parameter values in Table 7.4 were varied one at a time, while all other parameters were kept at their baseline values. A summary of all the parameter tests sets is presented in Table 7.5. The four values, other than the baseline value, were chosen in such a way that they reflect acceptable minimum and maximum parameter values, upon which two parameter values were selected midway between the extremes and the baseline value. It should be noted that although all five parameters are present in the parameter values column in Table 7.5, one set of runs may be performed for the baseline case and used for comparison throughout, along with all the other parameters test. It is included here as the middle value only for visualisation purposes.

The initial temperature, the maximum number of reheatings that may occur, the cooling rate and the reheating rate control the temperature of the DBMOSA. Figures A.4–A.7 contain graphical representations of the normalised HV obtained in the form of box plots when performing 10 runs of the DBMOSA in respect of each parameter test (Test sets 5–8), each test utilising 10 identical starting solutions generated by the enhanced KSP-MMV-ICRSGP to ensure consistent testing.

The initial temperature was determined according to the expression in (6.5), and found to be approximately 100. After extensive empirical testing, it became evident that a value of 100 is too large for the benchmark instances, and that temperatures in the range of 0.1 are much more appropriate. This value was therefore selected as the initial temperature baseline instead. The results thus obtained when varying the initial temperatures are depicted in Figure A.4 of Appendix A in the form of box plots measuring the HV obtained. It may be observed in the figure that when taking a starting temperature as 0.1, 0.001, and 0.0001 for the Mandl6, Mumford0, and Mumford1 benchmark instances, respectively, the results obtained are, on average, of higher quality than for the other initial temperature values, and that these temperature values also deliver HV results with relatively lower variability. This is due to the overall temperature values

TABLE 7.4: *Parameters considered during the parameter tests when designing the DBMOSA algorithm for solving instances of the UTRP, along with the baseline value of each parameter and the test values adopted during algorithmic performance measurement. The values in brackets denote a set choice previously mentioned in this chapter, whereas the forward slash within a set of brackets denotes different choices for each of three benchmark instances, Mandl6, Mumford0, and Mumford1, respectively.*

Parameter	Baseline value	Test values
Initial solutions	{2}	—
Perturbation management strategy	{2}	—
Perturbation LLHs	{0/2/3}	—
Number of runs	10	—
Termination criterion	First to breach	—
Maximum total iterations	40 000	—
Iterations for HV improvement comparison	8 000	—
HV improvement threshold	0.005%	—
Initial temperature	0.1	0.0001, 0.001, 1, 10
Maximum reheatings	5	1, 3, 10, 25
Cooling rate	0.97	0.95, 0.96, 0.98, 0.99
Reheating rate	1.5	1.05, 1.1, 2, 3
Maximum iterations per epoch	100	10, 50, 250, 500
Maximum poor epochs	400	100, 200, 600, 800
Minimum accepts	25	5, 10, 50, 100
Maximum attempts	50	10, 25, 100, 250

typically being lower than those corresponding to the other initial temperature values, therefore leading to the acceptance of worsening solutions with a lower probability.

Next, the maximum number of reheatings was tested. After preliminary experimentation, a value of 5 was chosen as the baseline value and compared with the test values listed in Table 7.4. The results are presented in Figure A.5 of Appendix A, and indicate that a value of 10 performed the best, on average, for the Mandl6 instance. For the Mumford0 instance, values of 1 and 5 performed similarly, but it was decided to opt for the value of 5 as this would afford the algorithm a greater chance of escaping from local optima. The test conducted on the Mumford1 instance indicated that a value of 3 reheatings performed the best, and with low variability also.

The cooling rate was calculated according to the expression in (6.8) and found to be 0.996 168. After some experimentation, it was, however, found that this value performed poorly, and that a value of 0.97 performed better, which was subsequently chosen as the baseline instead. The HV results for the cooling rate tests are presented in Figure A.6 of Appendix A, from which it is clear that the value of 0.97 yielded the best results in terms of both average and variability for the Mandl6 and Mumford1 instances, but that a value of 0.95 performed best for the Mumford0 instance.

Next, the reheating rate was considered, and after some preliminary experimentation the value of 1.5 was chosen as the baseline value. Figure A.7 in Appendix A contains the HV results obtained when varying the reheating rate. A value of 1.5 was found to perform the best for the Mandl6 instance, a value of 1.1 for the Mumford0 instance, and a value of 1.05 for the Mumford1 instance.

The next sets (Test sets 9–12) of parameters collectively achieve epoch control in the DBMOSA algorithm, namely the maximum number of iterations allowed per epoch, the maximum number

TABLE 7.5: *Parameter test cases for the UTRP algorithm.*

Test set	Test number	Instance	Test parameter	Parameter values
5	1	Mandl6	Initial temperature	0.0001, 0.001, 0.1, 1, 10
5	2	Mumford0	Initial temperature	0.0001, 0.001, 0.1, 1, 10
5	3	Mumford1	Initial temperature	0.0001, 0.001, 0.1, 1, 10
6	1	Mandl6	Maximum reheatings	1, 3, 5, 10, 25
6	2	Mumford0	Maximum reheatings	1, 3, 5, 10, 25
6	3	Mumford1	Maximum reheatings	1, 3, 5, 10, 25
7	1	Mandl6	Cooling rate	0.95, 0.96, 0.97, 0.98, 0.99
7	2	Mumford0	Cooling rate	0.95, 0.96, 0.97, 0.98, 0.99
7	3	Mumford1	Cooling rate	0.95, 0.96, 0.97, 0.98, 0.99
8	1	Mandl6	Reheating rate	1.05, 1.1, 1.5, 2, 3
8	2	Mumford0	Reheating rate	1.05, 1.1, 1.5, 2, 3
8	3	Mumford1	Reheating rate	1.05, 1.1, 1.5, 2, 3
9	1	Mandl6	Maximum iterations	10, 50, 100, 250, 500
9	2	Mumford0	Maximum iterations	10, 50, 100, 250, 500
9	3	Mumford1	Maximum iterations	10, 50, 100, 250, 500
10	1	Mandl6	Minimum accepts	5, 10, 25, 50, 100
10	2	Mumford0	Minimum accepts	5, 10, 25, 50, 100
10	3	Mumford1	Minimum accepts	5, 10, 25, 50, 100
11	1	Mandl6	Maximum attempts	10, 25, 50, 100, 250
11	2	Mumford0	Maximum attempts	10, 25, 50, 100, 250
11	3	Mumford1	Maximum attempts	10, 25, 50, 100, 250
12	1	Mandl6	Maximum poor epochs	100, 200, 400, 600, 800
12	2	Mumford0	Maximum poor epochs	100, 200, 400, 600, 800
12	3	Mumford1	Maximum poor epochs	100, 200, 400, 600, 800

of poor epochs allowed, the minimum number of acceptances required per epoch and the maximum number of attempts allowed per epoch. The results of the analysis conducted in order to identify suitable values for each of these parameters are presented in Figures A.8–A.11 of Appendix A. When considering the baseline value for the maximum number of iterations per epoch, the suggested value of 100 [50] was chosen, and after some experimentation, was confirmed to deliver favourable results. Figure A.8 in Appendix A indicates that values of 500, 50, and 50, however, yield even higher-quality results for the Mandl6, Mumford0 and Mumford1 instances, respectively. The value of 500 was chosen for Mandl6 instead of the slightly better performing value of 250, on average, due to the value of 250 value having returned two outliers, the one outlier being extremely large, thereby rendering the results less reliable.

When considering the maximum number of epochs that may elapse without inserting a non-dominated solution into the archive, it was concluded after preliminary experimentation that larger values perform better, due to smaller values typically leading to premature search termination. The baseline value was therefore chosen as 400. Figure A.9 in Appendix A, however, indicates that the values of 800 for Mumford0 and 600 for the other two instances performed the best overall. The HV results distribution in the figure is due to the smallest value of 100 terminating the search prematurely, while from 200 onwards sufficient time was allowed for the search to make meaningful progress. Another factor to keep in mind, when interpreting the re-

sults of Figure A.9, is that search convergence only occurs when the temperature is sufficiently low — a state never attained when adopting the poorer-performing parameter values.

The minimum number of solutions that should be accepted per epoch refers to those solutions not inserted into the archive, but which are nevertheless accepted according to (6.9). This number of acceptances should be controlled. The value of 3 suggested in the literature [47] was considered, but after experimentation, a value of 25 was found to produce higher-quality results. The results of the design analysis, however, showed that values of 10, 5 and 3 yielded even more favourable results for Mandl6, Mumford0, and Mumford1, as shown in Figure A.10 of Appendix A, while values of 50 and above resulted in less favourable results. It may be reasoned that accepting too many solutions will result in the DBMOSA resembling a random search which does not converge towards the Pareto front. Attempts, on the other hand, are defined as worsening solutions not accepted according to (6.9), upon which the attempts counter is incremented. Initial experimentation indicated that 50 may be a good value for this parameter, and Figure A.11 of Appendix A indicates that a value of 50 indeed yields superior results, in terms of both the mean and variability.

After having conducted the aforementioned design analysis tests, a sensitivity analysis was performed to determine which parameters have the largest impact on the HV results. Special attention should be afforded to the choice of such parameter values. Figure 7.3 contains the results of the sensitivity analysis, with the parameter names on the vertical axis, descending in increasing order of magnitude of the impact they have on the HV performance measure of the DBMOSA algorithm. It was found that the minimum number of acceptances per epoch, and the reheating rate have the largest impact on the HV results. The cooling rate, the initial temperature, the maximum number of iterations per epoch, and the maximum number of reheatings have a moderate impact, while the maximum number of attempts per epoch and the maximum number of poor epochs have the smallest impact on the performance of the algorithm.

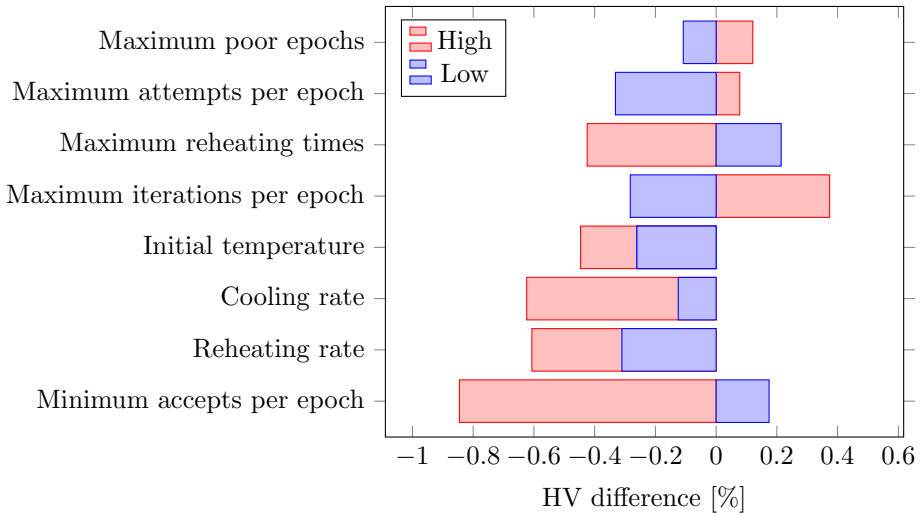


FIGURE 7.3: *Results of a sensitivity analysis conducted for the DBMOSA algorithm with the various algorithmic parameters ordered on the vertical axis in increasing magnitude of impact on algorithmic performance, measured in terms of the difference in HV on the horizontal axis achieved when adopting the low and high values, respectively.*

7.2.3 Performance runs

The performance runs conducted by means of the DBMOSA algorithm are numbered in Table A.1 of Appendix A for reference throughout this chapter. After having conducted the parameter tests and having evaluated their outcomes, the parameter values may be set in preparation for long runs (Test set 13) of the DBMOSA algorithm so that its performance may be measured against the results reported by other researchers. The same SA parameters identified to be best suited for Mumford1 is also adopted for Mumford2 and Mumford3, as mentioned. A summary of all the final parameter values adopted in the DBMOSA algorithm are presented in Table 7.6. For the long runs, 30 individual DBMOSA algorithmic runs are initialised starting with 30 different initial solutions returned by the enhanced KSP-MMV-ICRSGP. The main changes during the long runs are that the maximum number of total iterations is set to 400 000, and the number of iterations for HV improvement comparison is set to 10 000. This means that the algorithm will take longer to terminate, thereby allotting more computational time to the algorithm. The average run times are presented for each instance in Table A.2 of Appendix A, with the average run time of the Mumford3 instance being 7 hours, 52 minutes and 56 seconds.

TABLE 7.6: *The parameter values adopted during the performance tests of the DBMOSA algorithm when solving the five UTRP benchmark instances of §4.5.8.*

Parameter	Mandl6	Mumford0	Mumford1/2/3
Initial solutions	Enhanced KSP-MMV-ICRSGP		
Perturbation management strategy	Stochastic roulette-wheel strategy		
Perturbation LLHs	$\{a, b, c, d, e, j\}$	$\{a, b, c, d, e, g, h, j\}$	$\{c, d, e, g, h, j, p, q\}$
Number of runs	30	30	30
Termination criterion	First to breach	First to breach	First to breach
Maximum total iterations	400 000	400 000	400 000
Iterations for HV improvement comparison	10 000	10 000	10 000
HV improvement threshold	0.005%	0.005%	0.005%
Initial temperature	0.1	0.001	0.0001
Maximum reheatings	10	5	3
Cooling rate	0.97	0.95	0.97
Reheating rate	1.5	1.1	1.05
Maximum iterations per epoch	500	50	50
Maximum poor epochs	600	800	600
Minimum accepts	10	10	10
Maximum attempts	50	50	50

Mumford [121] published results for the five benchmark instances of §4.5.8 in 2013 obtained by an evolutionary algorithm, and later John [89] also published results for the same benchmark instances in 2016, demonstrating that his NSGA II algorithm outperformed the approach of Mumford. The approximate Pareto sets of Mumford and John are presented graphically in Figure 7.4 together with the attainment fronts obtained by 30 different initialisations of the DBMOSA search process of §6.1.7 with the best parameter values as identified during the aforementioned design analysis. As may be seen in the subfigures of Figure 7.4, the DBMOSA algorithm outperformed the evolutionary algorithm of Mumford, and outperformed almost all of the results of John's population-based search, dominating most of John's results in terms of the ATT objective function and uncovering solutions in areas of the Pareto front not represented in the results of John.

The DBMOSA algorithm could, however, not outperform the results reported by John for the Mandl6 benchmark instance, which is the smallest of the five test instances. It is interesting to

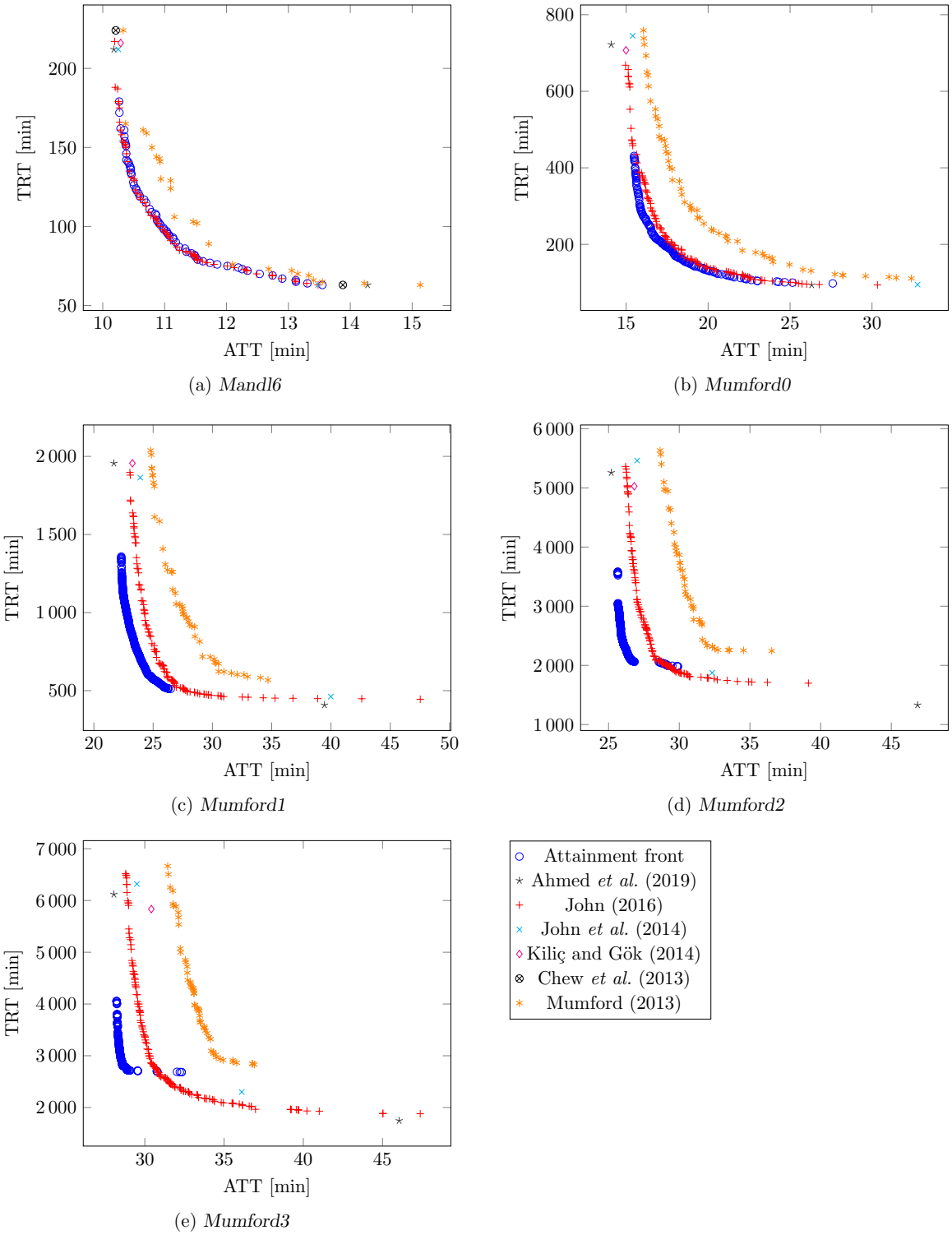


FIGURE 7.4: *Attainment fronts of solutions produced by the DBMOSA algorithm of the previous chapter during the long runs of Test set 13, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8.*

note the difference in the quality of the non-dominated sets returned by the DBMOSA algorithm in Figure 7.4. It would seem that the DBMOSA algorithm performs better in respect of the ATT objective function, but less so for the TRT objective function. This may be because the DBMOSA algorithm is not equipped with mechanisms whereby feasibility may be lost and then regained later — a required mechanism when extremely low TRT objective function values are to be obtained, like in the case of Ahmed *et al.* [2].

The HV values for the solutions returned by the DBMOSA algorithm of §6.1.7, compared with the HV values obtained by John [89], are summarised in Table 7.7 for each of the five benchmark instances. Some interesting insights may be gained when stacked area charts are plotted for the perturbation selection probabilities over the total number of iterations for each of the long runs of the DBMOSA algorithmic implementation of §6.1.7. Such plots may be seen in Figure 7.5 for each of the benchmark instances. The data for these plots were generated by keeping track of the selection probabilities of each LLH during each run of the DBMOSA, and were updated by means of the stochastic roulette-wheel selection probability management strategy. The averages of all the runs were taken, up to the points of the minimum termination iterations over all the runs, so that insight might be gained into trends that arise during the progression of each search. Furthermore, exponential smoothing was applied on the data points so that a more pleasurable viewing experience may be achieved.

The selection probabilities for the Mandl6 instance in Figure 7.5(a) exhibit rapid variations in the selection probabilities closer to the start of the search, but then even out towards the end of the search. It would seem that the add vertex operator was the most favoured among the LLHs, while the add inside vertex was the least favoured. When considering the selection probabilities for the Mumford0 instance, shown in Figure 7.5(b), it is clear that even more rapid variations are present at the start of the search, where much computational resources were afforded to the add and delete vertex operators, and less so for the add inside vertex, replace vertex, and donate vertex operators. As the search progressed, however, the selection probabilities tended to even out once again.

The observations for the larger instances are more interesting. Here, the novel cost-based trim and cost-based grow operators were employed, and Figures 7.5(c)–(e) indicate that these two operators were greatly favoured throughout the searches for the Mumford1, Mumford2, and Mumford3 instances. The exchange routes and invert vertices operators also enjoyed some favour, but not to the degree of the cost-based trim and cost-based grow operators. These changes in selection probabilities are intuitive in the sense that throughout a DBMOSA search run, different operators may be required in different regions of the search space in order to exploit or escape from local optima.

Additional figures for the DBMOSA long runs (Test set 13) are provided in Appendix A, where the average HV over the total number of iterations is shown in Figure A.12, and the average objective function values over the total number of iterations is shown in Figure A.13.

Apart from the two objective functions, the TRT and the ATT, represented by the two axes of Figure 7.4, another measure of the quality of a route set is the percentage of demand satisfied without any transfer (denoted by d_0), with one transfer (denoted by d_1), with two transfers (denoted by d_2), and with more than two transfers (denoted by d_u). Recall that Chakroborty [31] claimed that a good route set is one that serves the entire transit demand, with a large percentage of the demand being satisfied by direct connections and with the ATT per passenger as small as possible.

A brief discussion follows on the nature of the two extremal solutions in Figure 7.4(a). Only the solutions of the Mandl6 instance were visualised in solution space, as it is the smallest

TABLE 7.7: *The HV values obtained during the performance tests of the DBMOSA algorithm when solving the five UTRP benchmark instances of §4.5.8, along with the HV values obtained by John [89]. The HV values typeset in boldface are the best from among the two approaches.*

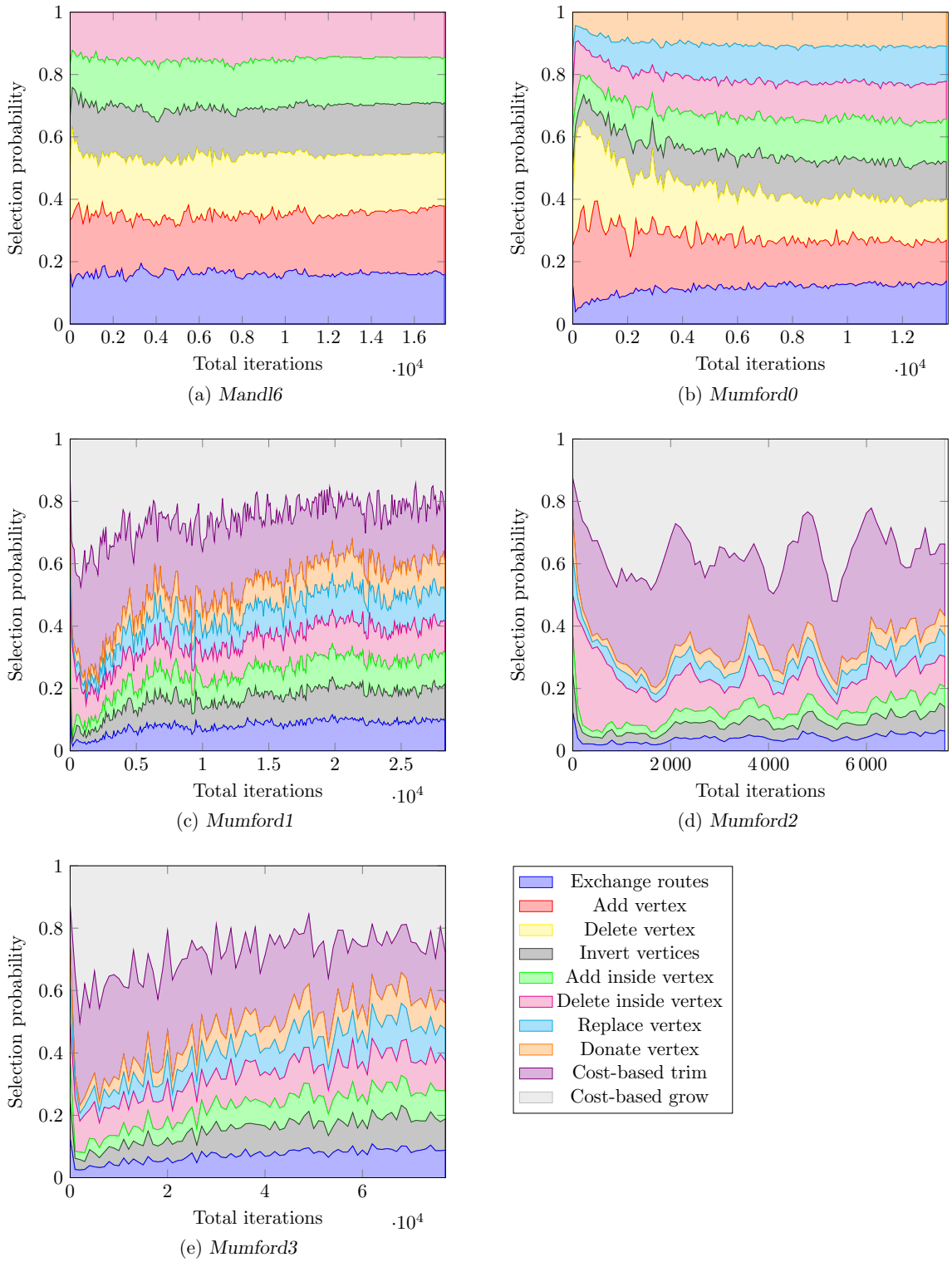
Solution approach	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
The DBMOSA of §6.1.7	85.3846	**81.6712**	**78.7682**	**65.3447**	58.2235
John 2016 [89]	**85.9769**	80.8722	77.5761	64.9003	**63.8547**

TABLE 7.8: *Objective function and transfer percentage evaluations for the ATT extremal solutions in Figure 7.4 uncovered by the DBMOSA algorithm in respect of the five benchmark instances of §4.5.8, with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mumford (2013) [121]	John *et al.* (2014) [90]	Kiliç and Gök (2014) [95]	Ahmed *et al.* (2018) [2]	DBMOSA of §6.1.7
Mandl6	Passenger	10.27	10.25	10.29	**10.18**	10.27
	Operator	221	212	216	212	179
	d_0	95.38	—	95.50	**97.17**	95.94
	d_1	4.56	—	4.50	**2.82**	3.93
	d_2	0.06	—	0.00	**0.00**	0.13
	d_u	0.00	—	0.00	**0.00**	0.00
Mumford0	Passenger	16.05	15.40	14.99	**14.09**	15.48
	Operator	759	745	707	722	431
	d_0	63.20	—	69.73	**88.74**	65.50
	d_1	35.82	—	30.03	**11.25**	34.50
	d_2	0.98	—	0.24	**0.00**	0.00
	d_u	0.00	—	0.00	**0.00**	0.00
Mumford1	Passenger	24.79	23.91	23.25	**21.69**	22.31
	Operator	2 038	1 861	1 956	1 956	1 359
	d_0	36.60	—	45.10	**65.75**	57.14
	d_1	52.42	—	49.08	**34.18**	42.63
	d_2	10.71	—	5.76	**0.07**	0.23
	d_u	0.26	—	0.06	**0.00**	0.00
Mumford2	Passenger	28.65	27.02	26.82	**25.19**	25.65
	Operator	5 632	5 461	5 027	5 257	3 583
	d_0	30.92	—	33.88	**56.68**	48.07
	d_1	51.29	—	57.18	**43.26**	51.29
	d_2	16.36	—	8.77	**0.05**	0.64
	d_u	1.44	—	0.17	**0.00**	0.00
Mumford3	Passenger	31.44	29.50	30.41	**28.05**	28.22
	Operator	6 665	6 320	5 834	6 119	4 060
	d_0	27.46	—	27.56	**50.41**	45.07
	d_1	50.97	—	53.25	**48.81**	54.37
	d_2	18.79	—	17.51	**0.77**	0.56
	d_u	2.81	—	1.68	**0.00**	0.00

TABLE 7.9: *Objective function and transfer percentage evaluations for the TRT extremal solutions in Figure 7.4 uncovered by the DBMOSA algorithm in respect of the five benchmark instances of §4.5.8, with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mumford (2013) [121]	John *et al.* (2014) [90]	Ahmed *et al.* (2018) [2]	DBMOSA of §6.1.7
Mandl6	Operator	**63**	**63**	**63**	**63**
	Passenger	**13.48**	**13.48**	14.28	13.55
	d_0	70.91	—	62.23	70.99
	d_1	25.5	—	27.16	24.44
	d_2	2.95	—	9.57	4.00
	d_u	0.64	—	1.028	0.58
Mumford0	Operator	111	95	**94**	98
	Passenger	32.4	32.78	26.32	27.61
	d_0	18.42	—	14.61	22.39
	d_1	23.4	—	31.59	31.27
	d_2	20.78	—	36.41	18.82
	d_u	37.4	—	17.37	27.51
Mumford1	Operator	568	462	**408**	511
	Passenger	34.69	39.98	39.45	26.48
	d_0	16.53	—	18.02	25.17
	d_1	29.06	—	29.88	59.33
	d_2	29.93	—	31.9	14.54
	d_u	24.66	—	20.19	0.96
Mumford2	Operator	2 244	1 875	**1 330**	1 979
	Passenger	36.54	32.33	46.86	29.91
	d_0	13.76	—	13.63	22.77
	d_1	27.69	—	23.58	58.65
	d_2	29.53	—	23.94	18.01
	d_u	29.02	—	38.82	0.57
Mumford3	Operator	2 830	2 301	**1 746**	2 682
	Passenger	36.92	36.12	46.05	32.33
	d_0	16.71	—	16.28	23.55
	d_1	33.69	—	24.87	58.05
	d_2	33.69	—	26.34	17.18
	d_u	20.42	—	32.44	1.23



(a) *Mandl6* (b) *Mumford0*

(c) *Mumford1* (d) *Mumford2*

(e) *Mumford3*

FIGURE 7.5: *Average perturbation selection probability (Test set 13) results over 30 separate long runs of the DBMOSA algorithm. Stacked area charts illustrate the changes in LLH selection probabilities on the vertical axes over the total number of iterations on the horizontal axes.*

ATT = 10.267 min
TRT = 179 min

Routes:
0-1-2-5-7-9-10-12
0-1-2-5-14-6
0-1-3-5-7-14-6
0-1-4-3-5-7-9-13
5-2-1-3-11-10-12-13
8-14-6-9-10-11

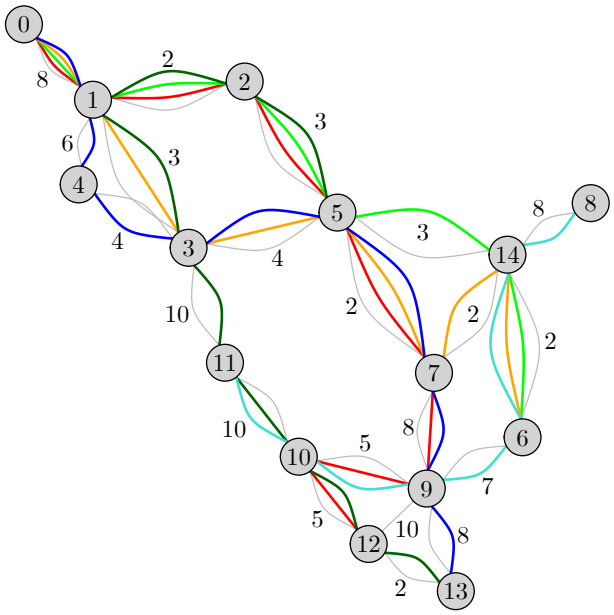


FIGURE 7.6: *Route set corresponding to the best ATT objective for the Mandl6 instance in Figure 7.4(a) returned by the DBMOSA algorithm.*

ATT = 13.549 min
TRT = 63 min

Routes:
9-6-14-7-5-2-1-3
3-4
11-10
1-0
13-12-10-9
8-14

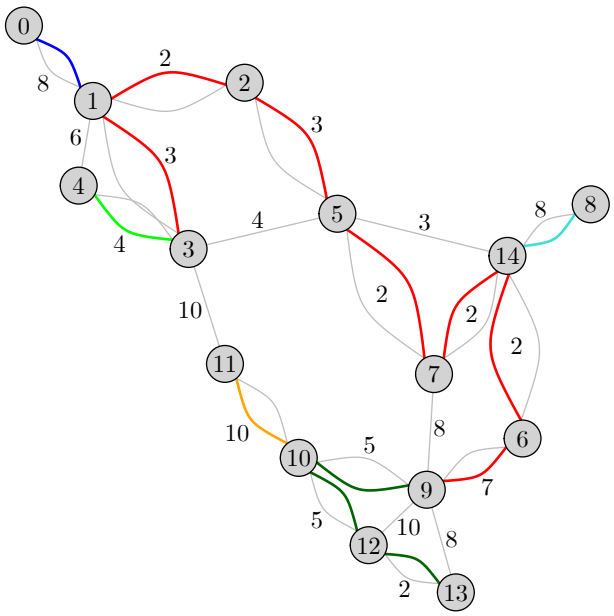


FIGURE 7.7: *Route set corresponding to the best TRT objective for the Mandl6 instance in Figure 7.4(a) returned by the DBMOSA algorithm.*

instance (the larger networks become increasingly difficult to present in an aesthetically pleasing manner on an A4-sized sheet). The route set corresponding to the best ATT objective function value in Figure 7.4(a) is presented in Figure 7.6, while that corresponding to the best TRT objective function value in Figure 7.4(a) is presented in Figure 7.7. Note that while both these route sets have the required $r = 6$ routes, each containing at least $m_{\min} = 2$ vertices and at most $m_{\min} = 8$ vertices, the routes in Figure 7.6 are, on average, much longer than those in Figure 7.7. These longer routes offer more flexibility to passengers and result in shorter transfer times, as may be seen in the transfer percentage differences between the best solutions from the passenger's perspective in Table 7.8, and that of the best solutions from the operator's perspective in Table 7.9.

Tables 7.8 and 7.9 contain the results of the best extremal solutions uncovered by the DBMOSA over all the runs conducted, compared with the best extremal solutions found in the literature. The results obtained by the DBMOSA algorithm of this dissertation do not outperform any of the best extremal solutions found in the literature, but its power lies in the high-quality trade-off routes along the non-dominated front returned. In a practical sense, these solutions would be preferred over those found at the extremes. It is clear that all the sets of results in the table meet the aforementioned criteria of Chakroborty [31] and that the DBMOSA algorithm returned very favourable results overall. Finally, a last set of runs was performed in which a static perturbation management strategy was adopted (*i.e.* adopting the same selection probabilities for all LLH perturbations). The results, captured in Figure A.14 of Appendix A, indicate that the stochastic roulette-wheel perturbation management strategy of §6.1.6 performed significantly better than when a static strategy was adopted for all the benchmark instances.

7.3 UTRP results returned by the NSGA II

This section is devoted to a validation of the NSGA II presented in §6.1.8 for solving instances of the UTRP. As in the previous section, the same five benchmark instances of §4.5.8 are considered. The results obtained for these instances are compared with the non-dominated sets published by Mumford [121] and by John [89], as well as with the single-point results reported by various other researchers. The same reference points as in the previous section were chosen for calculating the HV performance measure, listed in Table 7.1. These reference points were also used to normalise the objective function values for each instance during execution of the NSGA II, as is appropriate for GAs. This section is also partitioned into the same three subsections as the previous section, namely minor tests, parameter tests, and performance runs.

7.3.1 Minor tests

The test sets for the minor tests of the NSGA II are assigned a numbering scheme from 21 upwards, so that a clear distinction can be made between the aforementioned minor tests associated with the DBMOSA algorithm and those associated with the NSGA II. The minor tests encompass Test sets 21–26, and a summary of these test sets is provided in Table 7.10. The tests are further partitioned on each instance considered, and here it was decided only to conduct the minor tests in respect of the three smallest UTRP instances, as before. The values and functions adopted in the minor tests for the NSGA II solution implementation of §6.1.8 were chosen in such a way that a basic NSGA II is used as the baseline, upon which one component is incrementally altered based on the normalised HV obtained for the different approaches. Details on the baseline NSGA II configuration may be found in Table 7.11.

The decisions behind the values chosen in Table 7.11 are motivated briefly by stepping through the entries of the table. The population of initial solutions was generated by invoking the KSP-MMV-ICRSGP of §6.1.2, as this ICRSGP delivers high-quality initial solutions, as mentioned in §7.1. Next, the successful crossover strategy proposed by Mumford [121] was adopted, as was also the case in the analysis by John [89], denoted by the 0 in the first three rows of Table 7.11. A static mutation management strategy was adopted that assigns an equal probability to selecting one of the mutation LLHs to be applied to the offspring. The combination of LLHs considered was the add vertex, delete vertex, and the exchange routes operators, as initial testing indicated that these operators achieved good results. Mumford [121] also used the add vertex and delete vertex operators, but allowed a random number of vertices to be added or deleted, subject to route set feasibility, whereas in this case, only one vertex may be added or deleted at a time. The exchange routes operator of Mandl [115] is a good operator to help exchange subroute combinations between routes in a route set without adding or deleting vertices. The combination of these three operators equips the NSGA II to traverse the search space effectively. These LLHs are indicated by the letters a, b and j, in correspondence with the numbering in Table 6.3.

TABLE 7.10: *Minor test sets for the UTRP NSGA II.*

Test set	Test number	Instance	Test parameter	Parameter values
21	1	Mandl6	ICRSGP	$0, 1, 2$
21	2	Mumford0	ICRSGP	$0, 1, 2$
21	3	Mumford1	ICRSGP	$0, 1, 2$
22	1	Mandl6	Crossovers	$0, 1, 2, 3, 4, 5$
22	2	Mumford0	Crossovers	$0, 1, 2, 3, 4, 5$
22	3	Mumford1	Crossovers	$0, 1, 2, 3, 4, 5$
23	1	Mandl6	Mutations	All LLHs
23	2	Mumford0	Mutations	All LLHs
23	3	Mumford1	Mutations	All LLHs
24	1	Mandl6	Update mutation ratio	$0, 1$
24	2	Mumford0	Update mutation ratio	$0, 1$
24	3	Mumford1	Update mutation ratio	$0, 1$
25	1	Mandl6	Repairs	$0, 1$
25	2	Mumford0	Repairs	$0, 1$
25	3	Mumford1	Repairs	$0, 1$
26	1	Mandl6	Mutation threshold	$0.07, 0.05, 0.03, 0.02, 0.01, 0.005$
26	2	Mumford0	Mutation threshold	$0.07, 0.05, 0.03, 0.02, 0.01, 0.005$
26	3	Mumford1	Mutation threshold	$0.07, 0.05, 0.03, 0.02, 0.01, 0.005$

The number of runs was set to 20 in order to allow for sufficient testing, while the termination criterion was set to stopping when a specified number of iterations, related to the number of offspring solutions evaluated, had been completed per run. The maximum total number of iterations was set to 40 000. This value was arrived at by multiplying the number of generations by the population size, which were both set to 200. The crossover probability and mutation probability were determined empirically after numerous initial tests, and was determined to be appropriate values for the Mandl6, Mumford0, and Mumford1 instances.

The first minor test conducted for the NSGA II solution implementation when solving instances of the UTRP, numbered Test set 21, was aimed at establishing which initial solutions should be used. The question was which of the initial solutions generated by the KSP-MMV-ICRSGP or

TABLE 7.11: *The parameters considered during the minor tests when designing the NSGA II algorithm for solving instances of the UTRP, along with the baseline value of each parameter.*

Parameter	Baseline value
Initial solutions	{0}
Crossover strategy	{0}
Mutation management strategy	{0}
Mutation LLHs	$\{a, b, j\}$
Number of runs	20
Termination criterion	Stopping by iterations
Maximum total iterations	40 000
Generations	200
Population size	200
Crossover probability	0.6
Mutation probability	1

the enhanced KSP-MMV-ICRSGP provide better long-term results. Figure 7.8 indicates that the enhanced KSP-MMV-ICRSGP resulted in the largest normalised HV measurements, and this procedure was therefore selected as the ICRSGP choice for the NSGA II.

The second minor test, Test set 22, was aimed at deciding which of the six different crossover strategies of §6.1.8 should be adopted. The first is the standard crossover operator of Mumford [121], the second is *stochastic maximising missing vertices* (SMMV) operator, the third is the Mumford crossover operator in which subset routes are replaced by a K-shortest path, called *replace subsets deterministic K-shortest path* (RSDKSP) operator, the fourth is the *replace subsets stochastic K-shortest path* (RSSKSP) operator, the fifth is the *replace subsets deterministic* (RSD) operator, and sixth and last is the *replace subsets stochastic* (RSS) operator. In the latter two a route within one of the parents is used to replace a subset route. Figure A.15 in Appendix A indicates that very similar performances resulted from all six crossover operators, and based on the wide spread of HV values that was achieved by the RSD crossover for the Mumford3 instance and the number of runs that performed higher than average, this operator was selected to perform the crossover procedure in the NSGA II.

Test set 23 constituted the third minor test in which the various LLHs of Table 6.3 were considered as single mutations being applied in conjunction with the other NSGA II operators of §6.1.8. The HV obtained for each of these runs may be seen in Figure A.16. It was decided that ten LLHs should be available as mutation operators for the NSGA II so that a sufficient diversity of perturbations could be affected on the solutions throughout the search. Only one mutation operator was applied in conjunction with the baseline crossover procedure. The idea was to ascertain the performance associated with only applying one LLH. The mutations were then ranked according to their mean HV performance. Thereafter, a diverse variety of LLH operators were selected for implementation. The best combination of operators was found to be add vertex, delete vertex, add inside vertex, delete inside vertex, invert vertices, replace vertex, donate vertex, exchange routes, cost-based grow, and cost-based trim. The letters that correspond to these operators are $a, b, c, d, e, g, h, j, p$, and q in Table 6.3. These ten LLHs were selected as the mutation operators for the NSGA II.

The effects of changing the mutation management strategy were considered in Test set 24. Here, the baseline strategy involved adopting a static approach in which the selection probabilities are the same for all the LLHs. The alternative is the AMALGAM mutation management strategy

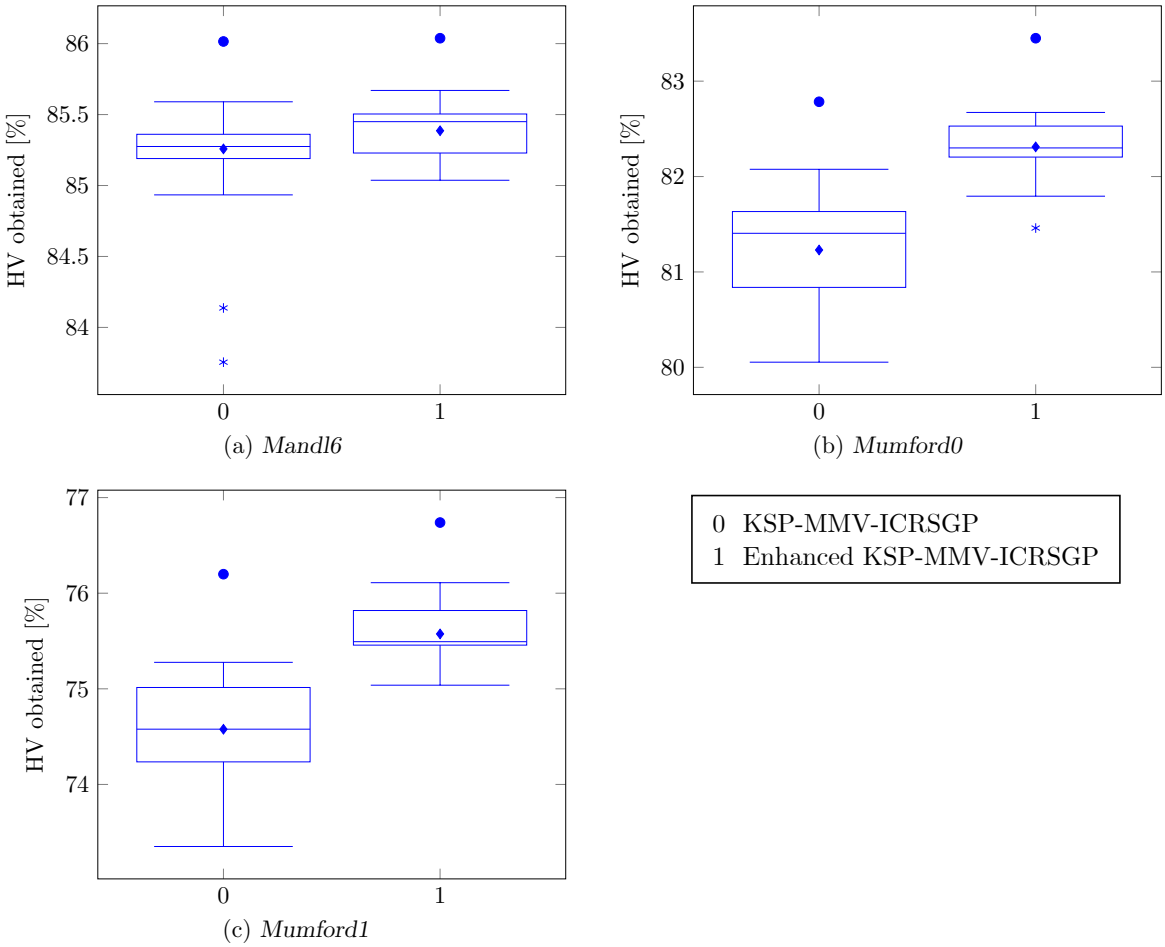
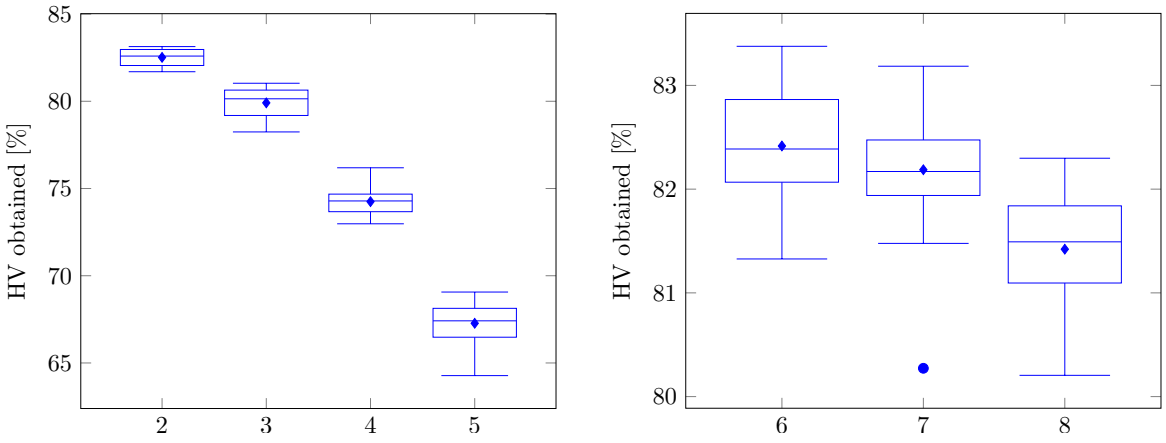


FIGURE 7.8: *ICRSGP analysis (Test set 21) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when the KSP-MMV-ICRSGP of §6.1.2 was utilised and when the enhanced KSP-MMV-ICRSGP of §6.1.3 was utilised, indicated on the horizontal axes as* 0 *and* 1*, respectively.*

of §6.1.6. The three baseline mutation operators were adopted in this minor test. Figure A.17 indicates that the performances of the two strategies were very similar, but because the aim is to include more LLHs later, the AMALGAM mutation management strategy was chosen as it can manage the selection probabilities automatically based on the LLH performances of previous generations.

The repair strategy used was considered in Test set 25. The repair multiple vertices strategy of §6.1.2 was adopted as the baseline repair strategy, with the alternative repair strategy being where only one vertex may be added according to the repair strategy, called the repair single vertex strategy. For this test, the ten mutation operators suggested by Soares *et al.* [150] were incorporated so that some infeasible solutions could potentially be generated during the search. The HV results for this test may be seen in Figure A.18. The results did not indicate a significant difference, but the single repair strategy was selected due to multiple vertex repairs potentially being capable of leading to a gradual build-up of vertices in a transit network, as Mumford mentioned in 2013 [121].

The last minor test involved setting an appropriate mutation probability selection threshold intrinsic to the AMALGAM approach [167]. For this test, all the NSGA II component changes suggested by the outcomes of Test sets 21–25 were incorporated so that all ten mutation operators



(a) *Minimum allowed vertices* (b) *Number of allowed routes*

FIGURE 7.9: *The design analysis results for 20 separate runs of the NSGA II, in which two of the input parameters to the Mandl6 UTRP benchmark instance were considered, namely the minimum number of vertices allowed per route, and the number of routes considered. The box plots of the HV obtained when varying each parameter are depicted in (a) and (b) respectively, and the horizontal axis indicates the values to which the parameters were set.*

to be used in the NSGA II are in place for an appropriate mutation selection threshold to be set. Figure A.19 indicated that a mutation selection threshold of 3% performed the best for the largest instance of the three tested, namely the Mumford1 instance, and this threshold value performed well for the Mandl6 and Mumford0 instances as well. This value was therefore chosen as the mutation probability selection threshold.

Two further tests were conducted to ascertain how the input parameters related to the constraints of the Mandl6 UTRP benchmark instance affect the performance of the search. These parameters are the minimum allowed number of vertices per route, and the number of routes allowed per route set. Figure 7.9(a) indicates that as the number of allowed vertices, or bus stops, per route decreases, the potential for improved network performance increases, as indicated by the larger HV values obtained when allowing fewer vertices in a route. When considering the number of allowed routes in a network, Figure 7.9(b) indicates that allowing fewer routes is better for network performance, and would benefit the operator's perspective more than the passenger's perspective.

7.3.2 Parameter tests

A similar approach was followed in order to select parameter values for the NSGA II as that described in §7.2. For the NSGA II, the relevant parameters are the crossover probability, the mutation probability, the population size, and the number of generations. Initially the values of these parameters were set to the baseline values presented in Table 7.12. Both Mumford [121] and John [89] adopted a crossover probability of 1 and employed a crossover operator proposed by Mumford [121]. After extensive empirical experimentation, it was determined that this is not an appropriate crossover probability for the NSGA II of §6.1.8, but that a value of 0.6 provides better performance. Moreover, John allowed r mutation attempts for each offspring solution produced, with a mutation probability of $\frac{1}{r}$ for each trial. He also incorporated eight different mutation operators, while the NSGA II of §6.1.8 incorporates one of ten mutation operators for each trial, as mentioned in Table 7.12. The mutation probability for the NSGA II of §6.1.8 was

set to 0.9 after having determined this value to be a good baseline value after initial testing over the benchmark instances of varying sizes. The other parameter test values were chosen based on the same reasoning as for the DBMOSA parameter test values. The two extreme high and low values were chosen above and below the baseline value, and then two parameter test values were selected between the extreme values and the baseline value.

The parameter test sets, numbered Test tests 27–29, are summarised in Table 7.13. It was decided to conduct these parameter tests on all five benchmark instances of §4.5.8 so that appropriate GA parameters may be set for each of the instances. Although the baseline parameters are also included in the parameter values in Table 7.13, this is purely for visualisation purposes, and one baseline value test may again be conducted and used for comparison in all the test sets of Table 7.13.

TABLE 7.12: *The parameters considered during the parameter tests when designing the NSGA II algorithm for solving instances of the UTRP, along with the baseline value of each parameter and the test values adopted during algorithmic performance measurement.*

Parameter	Baseline value	Test values
Initial solutions	{1}	—
Crossover strategy	{1}	—
Mutation management strategy	{1}	—
Mutation LLHs	$\{a, b, c, d, e, g, h, j, p, q\}$	—
Number of runs	10	—
Mutation selection threshold	3%	—
Termination criterion	Stopping by iterations	—
Maximum total iterations	80 000	—
Generations	200	—
Population size	400	$200, 300, 500, 600$
Crossover probability	0.6	$0.4, 0.5, 0.7, 0.8$
Mutation probability	0.9	$0.8, 0.85, 0.95, 1$

The parameter tuning analysis involved performing ten algorithmic runs for each parameter test value combination, (the test values were varied for each parameter in turn, while keeping all the other parameters at their baseline values). The set of the test values for each parameter can be found in Table 7.12. Thereafter, box-plots were constructed to evaluate the algorithmic performance in terms of the HV values achieved by the ten runs.

When considering the crossover probability, the parameter tuning analysis delivered interesting results, shown in Figure A.20, indicating that a smaller crossover probability yields significantly better results than larger probability values, and that a crossover probability of 0.8, which was considered as the highest crossover probability, resulted in the poorest performance. This value was therefore adjusted to 0.5, 0.4, 0.4, 0.5, and 0.6, for the Mandl6, Mumford0, Mumford1, Mumford2, and Mumford3 instances, respectively, as these values yielded the best results in each of the tests.

The results of the mutation probability tuning process also delivered interesting results, as shown in Figure A.21, indicating that a larger mutation probability yields better results. John's approach [89] of allowing r trials, each with a probability of $\frac{1}{r}$, is approximately equivalent to performing one mutation per attempt, on average. The mutation probability of the NSGA II was therefore adjusted to 1. It should be noted that the crossover procedure in essence disrupts the composition of parent solutions in quite a significant manner, whereas mutation only produces relatively small changes to a solution locally, yet introduces some diversity into the population.

TABLE 7.13: *Parameter test cases for the UTRP NSGA II.*

Test set	Test number	Instance	Test parameter	Parameter values
27	1	Mandl6	Crossover probability	$0.4, 0.5, 0.6, 0.7, 0.8$
27	2	Mumford0	Crossover probability	$0.4, 0.5, 0.6, 0.7, 0.8$
27	3	Mumford1	Crossover probability	$0.4, 0.5, 0.6, 0.7, 0.8$
27	4	Mumford2	Crossover probability	$0.4, 0.5, 0.6, 0.7, 0.8$
27	5	Mumford3	Crossover probability	$0.4, 0.5, 0.6, 0.7, 0.8$
28	1	Mandl6	Mutation probability	$0.8, 0.85, 0.9, 0.95, 1$
28	2	Mumford0	Mutation probability	$0.8, 0.85, 0.9, 0.95, 1$
28	3	Mumford1	Mutation probability	$0.8, 0.85, 0.9, 0.95, 1$
28	4	Mumford2	Mutation probability	$0.8, 0.85, 0.9, 0.95, 1$
28	5	Mumford3	Mutation probability	$0.8, 0.85, 0.9, 0.95, 1$
29	1	Mandl6	Population size	$200, 300, 400, 500, 600$
29	2	Mumford0	Population size	$200, 300, 400, 500, 600$
29	3	Mumford1	Population size	$200, 300, 400, 500, 600$
29	4	Mumford2	Population size	$200, 300, 400, 500, 600$
29	5	Mumford3	Population size	$200, 300, 400, 500, 600$

By adopting a lower probability for crossover, and allowing some offspring solutions only to experience mutation, local optima may be exploited. A complete disruption of parent solution structures by the crossover procedure, on the other hand, mainly aids in search exploration. By balancing these two probabilities appropriately, the search may be improved significantly.

A larger population size naturally leads to increased diversity, but the challenge when deciding on a value for this parameter is finding an appropriate balance between maintaining a large enough population, and limiting the computational cost associated with maintaining such a population (when evaluating the objective function for candidate solutions). Figure A.22 indicates the improvement in the HV achievable when larger population sizes were chosen, showing that population sizes of 500 and 600 deliver better results than does a population size of 400, whereas population sizes of 200 and 300 yielded the poorest performance. The value of this parameter was selected as 400, however, so that an adequate balance may be achieved between the number of generations simulated and the computational burden of maintaining a larger population. Upon performing further tests, it became evident that simulating more generations was more beneficial than maintaining larger population sizes. This finding may be due to more mutations being performed and greater exploitation being achieved in the former case.

The total number of generations performed also plays a large part in the success achieved during the search process. Simulating too few generations may lead to premature termination of the algorithm. The challenge here is once again finding an appropriate trade-off between the computational resources required and the additional search performance gained. A point is typically reached when additional generations do not improve upon the best solutions found, and should be avoided as this wastes computational resources. This parameter may be set in various ways, depending on the availability of computational resources. The stopping criterion of observing no further improvement in HV within a pre-specified threshold over a pre-specified number of generations may be adopted for the longer runs, and then the maximum number of generations does not need to be set to a specific value.

A sensitivity analysis was conducted, following the aforementioned parameter tuning stage, to determine which of the parameters contribute to a greater degree to the performance of the NSGA II, and should therefore be afforded more attention when tuning the algorithm's parameters. The sensitivity analysis was conducted by taking the average of all the differences of the baseline performance and the extreme values for each parameter over all the instance sizes, in order to reach an overall conclusion. The results of the sensitivity analysis are presented in Figure 7.10, and indicate that the the crossover probability has the largest impact on the algorithm's performance, followed by the population size and lastly the mutation probability. It is further noted that the largest impact on the performance occurs when the crossover probability is chosen too low, rather than too high.

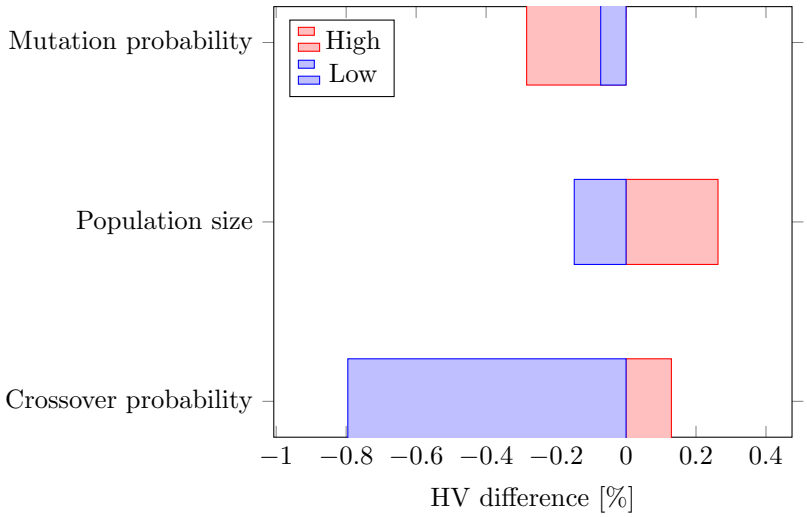


FIGURE 7.10: *Results of a sensitivity analysis conducted for the NSGA II of §6.1.8 with the various algorithmic parameters ordered on the vertical axis in increasing magnitude of impact on algorithmic performance, measured in terms of the difference in HV on the horizontal axis achieved when adopting the low and high values, respectively.*

7.3.3 Performance runs

All the NSGA II performance runs are numbered in Table A.3 of Appendix A for reference throughout this chapter. After having selected the NSGA II parameters as described above, 20 NSGA II runs of 2 000 generations each were conducted for each of the benchmark instances of §4.5.8, called Test set 30, and the attainment set of all these runs was computed. The final parameters and components of the NSGA II of §6.1.8 are tabulated in Table 7.14. These results can be found in Figure 7.11, plotted alongside the results obtained by John [89] and those of Mumford [121], as well as extreme point solutions reported by various other researchers. The NSGA II clearly performed favourably against these published results, dominating most of John's solutions, and all of Mumford's solutions. The HV achieved by the NSGA II of §6.1.8 for each benchmark instance may be found in Table 7.15, compared with the HV values obtained by John [89].

The mutation selection probabilities over the number of generations are also plotted graphically as stacked area charts in Figure 7.12. Figure 7.12(a) indicates that the cost-based trim LLH operator, on average, received greater computational resources as the search progressed for the Mandl6 instance. A similar trend is observed for Mumford0, but to a lesser extent. In general, it would seem from Figure 7.12 that the cost-based grow and cost-based trim operators were

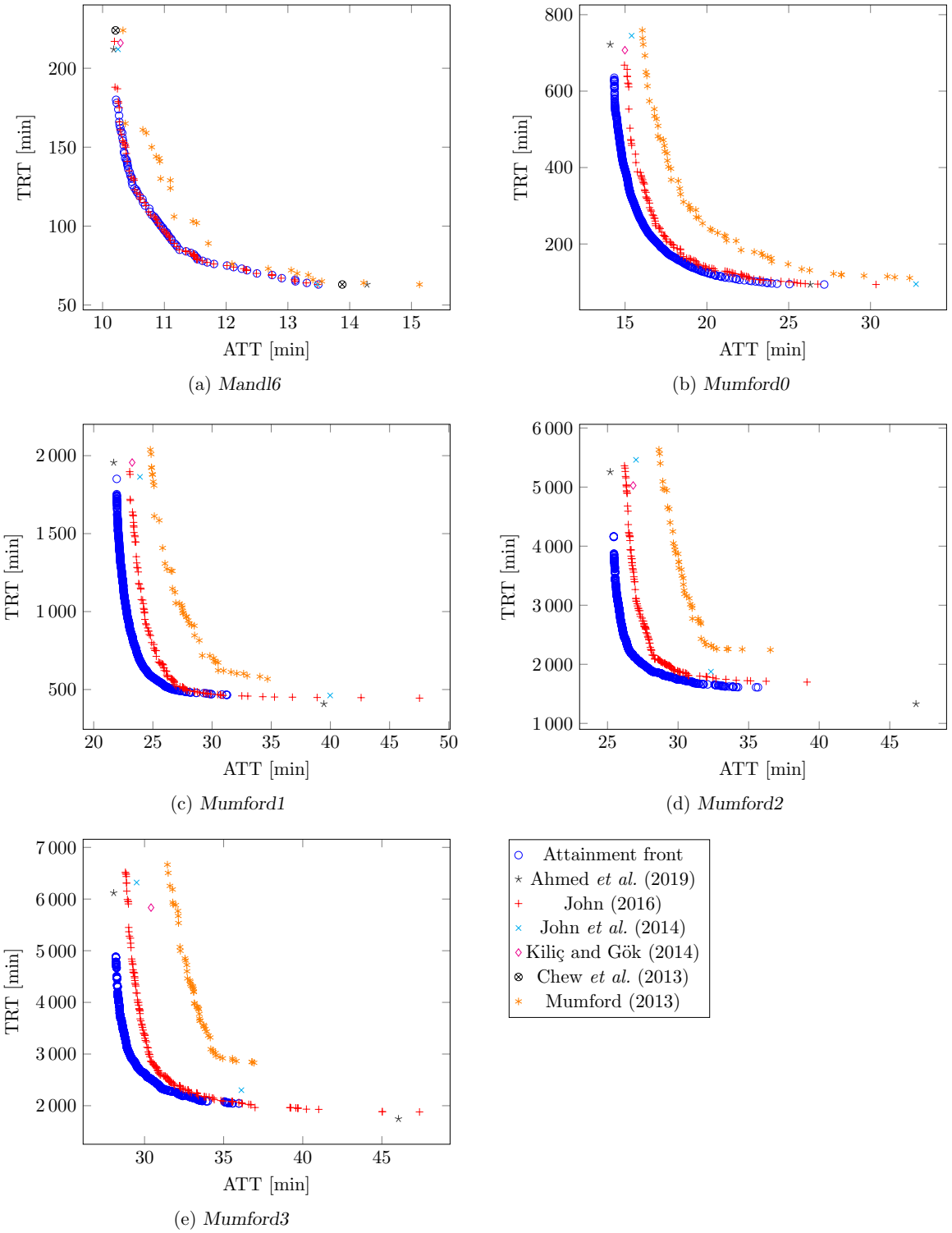
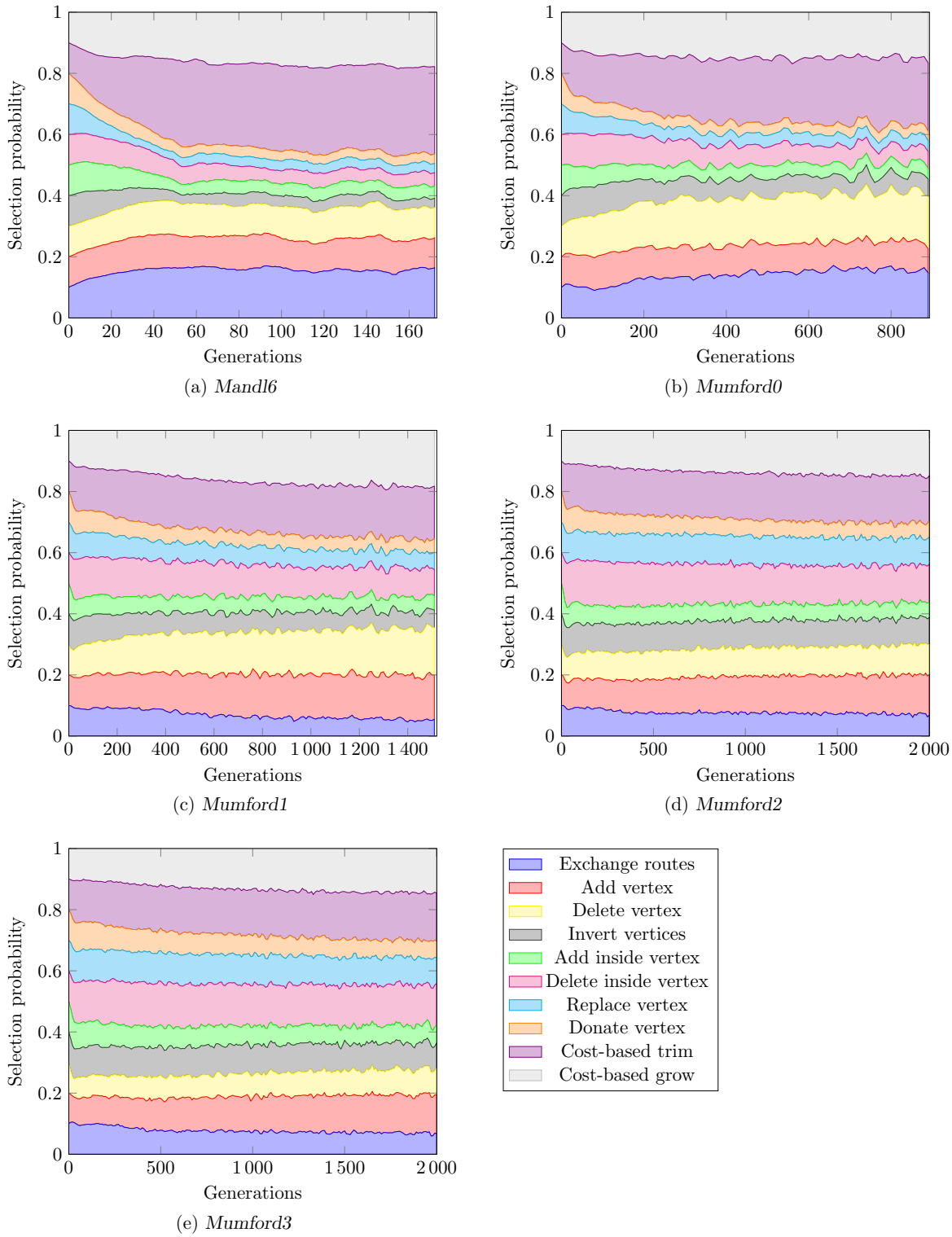


FIGURE 7.11: *Attainment fronts of solutions produced by the NSGA II of the previous chapter during the long runs of Test set 10, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8.*



(a) *Mandl6*

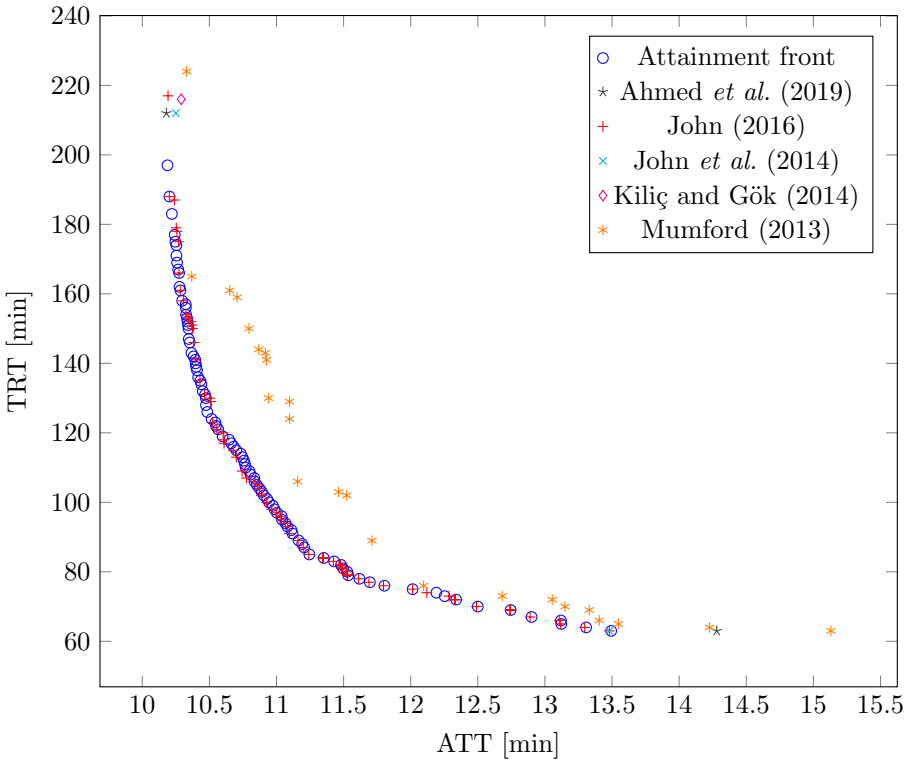
(b) *Mumford0*

(c) *Mumford1*

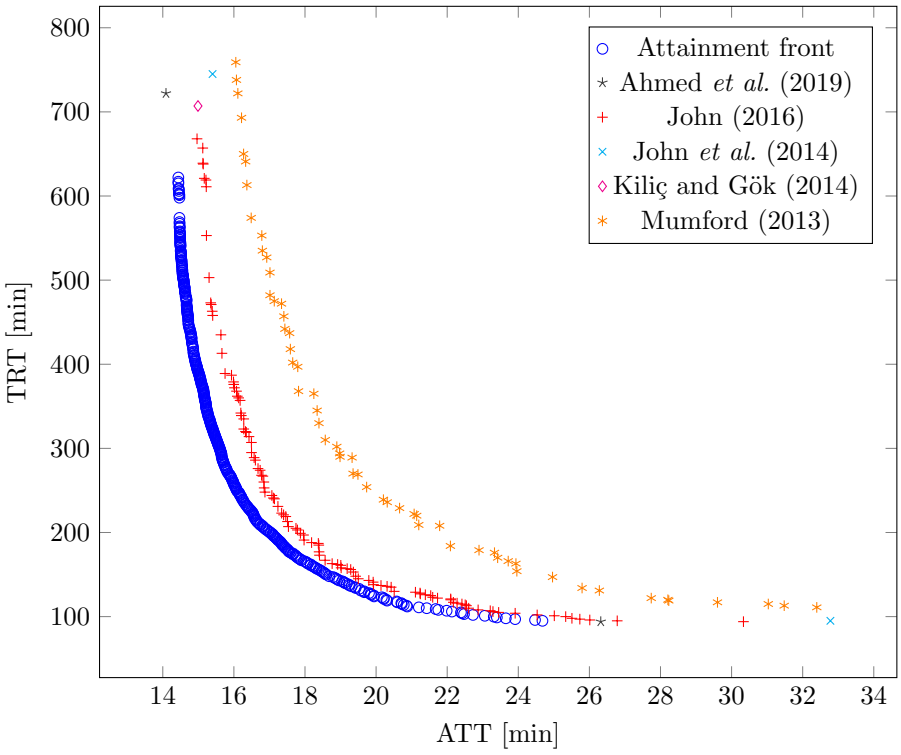
(d) *Mumford2*

(e) *Mumford3*

FIGURE 7.12: *Average mutation selection probability (Test set 10) results over 20 separate long runs of the NSGA II. Stacked area charts illustrate the changes in LLH selection probabilities on the vertical axes over the number of generations on the horizontal axes.*

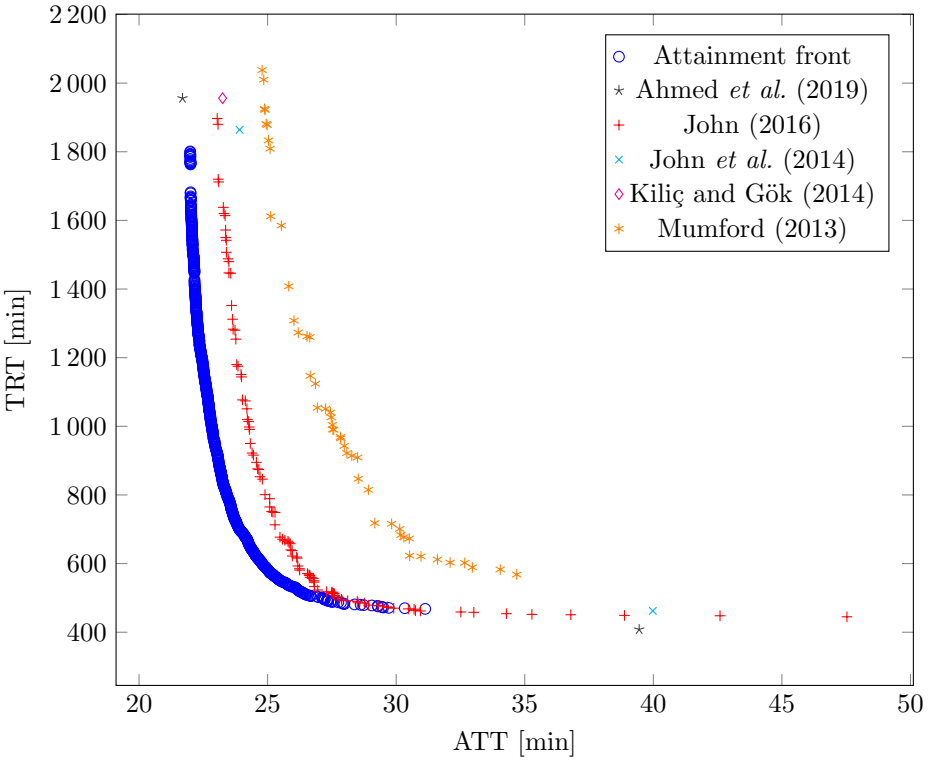


(a) *Mandl6*

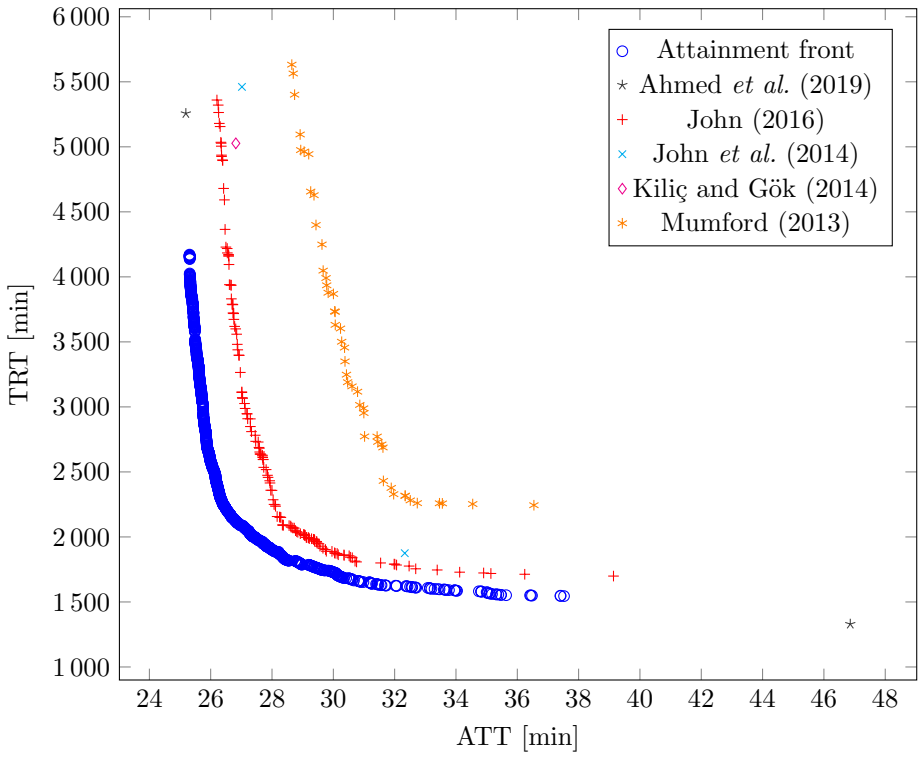


(b) *Mumford0*

FIGURE 7.13: *Attainment fronts for solutions produced by the NSGA II of the previous chapter during the uncapped runs of Test set 11, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. These are the first two subfigures of a continued figure.*



(c) *Mumford1*

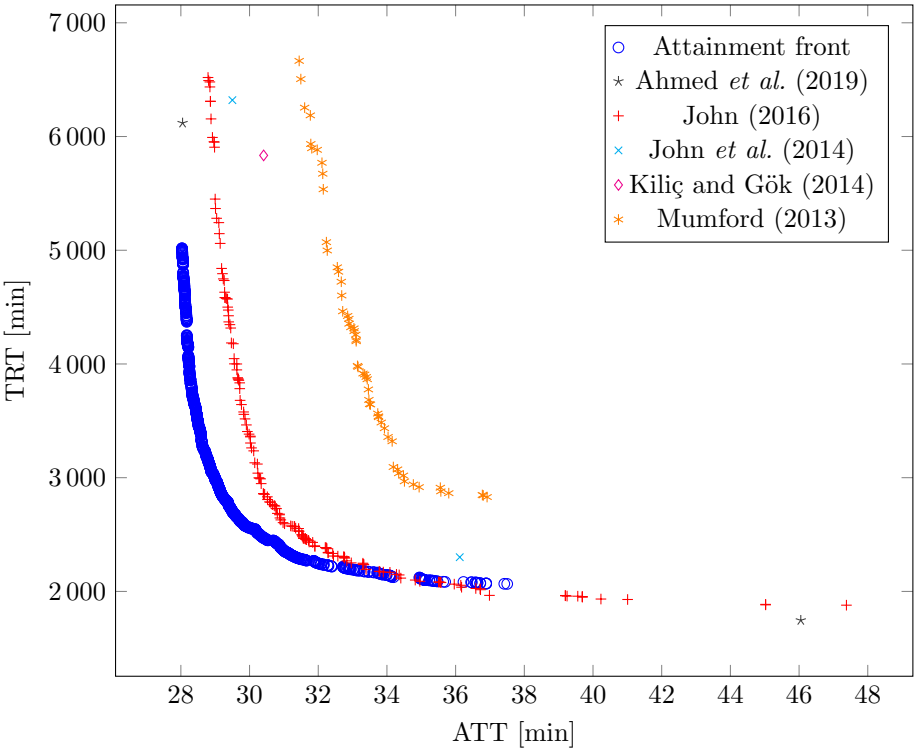


(d) *Mumford2*

FIGURE 7.13: *(continued) Attainment fronts for solutions produced by the NSGA II of the previous chapter during the uncapped runs of Test set 11, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. These are the third and fourth subfigures of a continued figure.*

TABLE 7.14: *The parameters considered during the performance tests of the NSGA II when solving the five UTRP benchmark instances of §4.5.8.*

Parameter	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
Initial solutions	Enhanced KSP-MMV-ICRSGP				
Crossover strategy	RSD crossover				
Mutation management strategy	AMALGAM mutation management strategy				
Mutation LLHs	$\{a, b, c, d, e, g, h, j, p, q\}$				
Number of runs	20	20	20	20	20
Mutation selection threshold	3%	3%	3%	3%	3%
Termination criterion	First to breach				
Maximum total iterations	800 000	800 000	800 000	800 000	800 000
Generations for HV improvement comparison	40	40	40	40	40
HV improvement threshold	0.01%	0.01%	0.01%	0.01%	0.01%
Generations	2 000	2 000	2 000	2 000	2 000
Population size	400	400	400	400	400
Crossover probability	0.5	0.4	0.4	0.5	0.6
Mutation probability	1	1	1	1	1



(e) *Mumford3*

FIGURE 7.13: *(continued) Attainment fronts for solutions produced by the NSGA II of the previous chapter during the uncapped runs of Test set 11, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. This is the last subfigure of a continued figure.*

favoured in all the instances. It should be noted that for different instances, it appears that different mutation operators are prioritised. For example, the delete inside vertex operator received the greatest prioritisation in the case of the Mumford3 instance, and very little in the Mandl6 and Mumford0 instances.

TABLE 7.15: *The HV values obtained during the performance tests of the NSGA II when solving the five UTRP benchmark instances of §6.1.8, along with the HV values obtained by John [89]. The HV values in boldface are the best from among the two approaches.*

Solution approach	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
The NSGA II of §6.1.8	**86.0690**	**85.5074**	**81.1267**	**70.4683**	**65.977**
John 2016 [89]	85.9769	80.8722	77.5761	64.9003	63.8547

TABLE 7.16: *Objective function and transfer percentage evaluations for the ATT extremal solutions in Figure 7.11 uncovered by the NSGA II in respect of the five benchmark instances of §4.5.8, with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mumford (2013) [121]	John *et al.* (2014) [90]	Kiliç and Gök (2014) [95]	Ahmed *et al.* (2018) [2]	NSGA II of §6.1.8
Mandl6	Passenger	10.27	10.25	10.29	**10.18**	10.19
	Operator	221	212	216	212	197
	d_0	95.38	—	95.5	**97.17**	97.36
	d_1	4.56	—	4.5	**2.82**	2.64
	d_2	0.06	—	0.00	**0.00**	0.00
	d_u	0.00	—	0.00	**0.00**	0.00
Mumford0	Passenger	16.05	15.40	14.99	**14.09**	14.34
	Operator	759	745	707	722	635
	d_0	63.20	—	69.73	**88.74**	86.94
	d_1	35.82	—	30.03	**11.25**	13.06
	d_2	0.98	—	0.24	**0.00**	0.00
	d_u	0.00	—	0.00	**0.00**	0.00
Mumford1	Passenger	24.79	23.91	23.25	**21.69**	21.94
	Operator	2 038	1 861	1 956	1 956	1 851
	d_0	36.60	—	45.10	**65.75**	62.11
	d_1	52.42	—	49.08	**34.18**	37.84
	d_2	10.71	—	5.76	**0.07**	0.05
	d_u	0.26	—	0.06	**0.00**	0.00
Mumford2	Passenger	28.65	27.02	26.82	**25.19**	25.31
	Operator	5 632	5 461	5 027	5 257	4 171
	d_0	30.92	—	33.88	**56.68**	52.56
	d_1	51.29	—	57.18	**43.26**	47.33
	d_2	16.36	—	8.77	**0.05**	0.11
	d_u	1.44	—	0.17	**0.00**	0.00
Mumford3	Passenger	31.44	29.50	30.41	28.05	**28.03**
	Operator	6 665	6 320	5 834	6 119	5 018
	d_0	27.46	—	27.56	50.41	**48.71**
	d_1	50.97	—	53.25	48.81	**51.1**
	d_2	18.79	—	17.51	0.77	**0.19**
	d_u	2.81	—	1.68	0.00	**0.00**

TABLE 7.17: *Objective function and transfer percentage evaluations for the TRT extremal solutions in Figure 7.11 uncovered by the NSGA II in respect of the five benchmark instances of §4.5.8, with the best overall performance for each instance highlighted in boldface.*

Instance	Performance metric	Mumford (2013) [121]	John *et al.* (2014) [90]	Ahmed *et al.* (2018) [2]	NSGA II of §6.1.8
Mandl6	Operator	**63**	**63**	**63**	**63**
	Passenger	**13.48**	**13.48**	14.28	13.49
	d_0	70.91	—	62.23	71.18
	d_1	25.50	—	27.16	25.21
	d_2	2.95	—	9.57	2.97
	d_u	0.64	—	1.028	0.64
Mumford0	Operator	111	95	**94**	**94**
	Passenger	32.4	32.78	26.32	27.17
	d_0	18.42	—	14.61	24.71
	d_1	23.40	—	31.59	38.31
	d_2	20.78	—	36.41	26.77
	d_u	37.40	—	17.37	10.20
Mumford1	Operator	568	462	**408**	465
	Passenger	34.69	39.98	39.45	31.26
	d_0	16.53	—	18.02	19.70
	d_1	29.06	—	29.88	42.09
	d_2	29.93	—	31.9	33.87
	d_u	24.66	—	20.19	4.33
Mumford2	Operator	2 244	1 875	**1 330**	1 545
	Passenger	36.54	32.33	46.86	37.52
	d_0	13.76	—	13.63	13.48
	d_1	27.69	—	23.58	36.79
	d_2	29.53	—	23.94	34.33
	d_u	29.02	—	38.82	15.39
Mumford3	Operator	2 830	2 301	**1 746**	2 043
	Passenger	36.92	36.12	46.05	35.97
	d_0	16.71	—	16.28	15.02
	d_1	33.69	—	24.87	48.66
	d_2	33.69	—	26.34	31.83
	d_u	20.42	—	32.44	4.49

Furthermore, the HV obtained over the generations are plotted in Figure A.23, and the mean objective function values of the ATT and the TRT over the number of generations are plotted in Figure A.24. When observing the HV over time plots of Figure A.23, it is clear that the HV gradients computed for the Mumford2 and Mumford3 instances are still growing steadily, indicating that the NSGA II terminated prematurely (*i.e.* that 2 000 generations were not enough). A number of additional uncapped runs were therefore performed in order to push the NSGA II to its limits in terms of uncovering the highest possible quality of solutions. The number of generations for these runs were therefore set to 10 000 and the stopping criterion was set to first to breach, specifying that the NSGA II should terminate when either 10 000 generations had been completed, or when the HV no longer improved by at least 0.01% over a moving window of 40 generations. The results returned by these uncapped runs are displayed in Figure 7.13. The average run times for the long and uncapped runs are shown in Table A.2 of Appendix A, with the average run time for the Mumford3 instance long runs being 54 hours, 53 minutes and 4 seconds, and 76 hours, 50 minutes and 30 seconds for the uncapped runs.

When considering the best results returned overall by the NSGA II of §6.1.8 in Table 7.16 for the best passenger cost and in Table 7.17 for the best operator cost, some interesting insights arise. When specifically considering the best passenger costs returned in Table 7.16 in conjunction with Figure 7.13, it is clear that at the extremal points, the ATT times are a few seconds longer than the best times reported in the literature, but in terms of the second objective, represent a considerable benefit over the TRT incurred when achieving these results. This is especially evident in the larger instances, namely Mumford2 and Mumford3, consisting of network sizes of 110 and 127 vertices, respectively.

For example, the best ATT achieved for Mumford2 is 25.31 min with a TRT of 4 171 min, while the best recorded results in the literature are due to Ahmed *et al.* [2], who reported an ATT of 25.19 min (which equates to a difference of 7.2 seconds, on average), but with a TRT of 5 257 min, which is a difference of 1 086 min of additional routes that have to be maintained for a relatively small benefit in ATT. The same is true for the Mumford3 instance. Here, however, the best ATT returned by the NSGA II of §6.1.8 outperformed the best ATT reported in the literature due to Ahmed *et al.* [2]. The solution returned achieved an ATT of 28.03 min, with a TRT of 5 018 min, whereas the ATT of Ahmed *et al.* [2] is 28.05 min, but with a TRT of 6 119 min, which represents an additional 1 101 min of routes that have to be maintained. This results in about one sixth of unnecessary bus routes according to Ahmed *et al.* [2], compared with the bus routes returned for the large instances by the NSGA II of §6.1.8. This once again elucidates the power of adopting a multi-objective modelling approach, where both objectives are minimised simultaneously. There is considerable benefit in such an approach, as it obtains excellent results in terms of route set quality, especially for the larger network instances, which is more practical for real world scenarios.

In terms of operator cost, the best TRT objective function values returned by the NSGA II of §6.1.8 are summarised in Table 7.17. Here, the NSGA II actually achieved the lower bounds for the Mandl6 and Mumford0 instances, although it could not achieve the corresponding lowest ATT reported by other researchers. The results are nevertheless of high quality when compared with other benchmark results. The NSGA II of §6.1.8 does seem to exhibit a better capability of exploiting the ATT objective than exploiting the TRT objective in a single-objective paradigm, but when keeping these two objectives in conflict with one another, it would seem that the algorithm performs especially well in the domain of the ATT objective function. It can be argued that of the two objective functions, the ATT is also the most important as it deals with passenger satisfaction.

ATT = 10.188 min
TRT = 197 min

Routes:
0-1-4-3-5-7-14-6
0-1-3-11-10-12-13
0-1-2-5-7-9-10-12
0-1-2-5-14-6
8-14-6-9-10-11-3-4
2-1-4-3-5-7-9-13

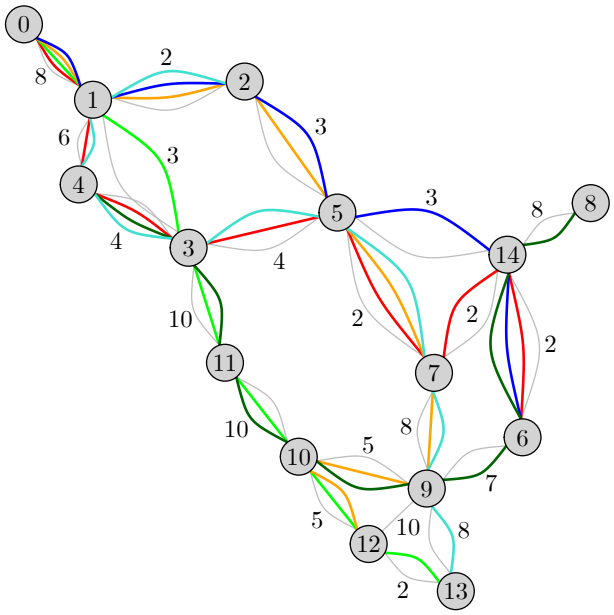


FIGURE 7.14: *Route set corresponding to the best ATT objective for the Mandl6 instance in Figure 7.13(a) returned by the NSGA II.*

ATT = 13.494 min
TRT = 63 min

Routes:
13-12
0-1-2-5-7-14-6-9
14-8
12-10-9
11-10
4-3-1

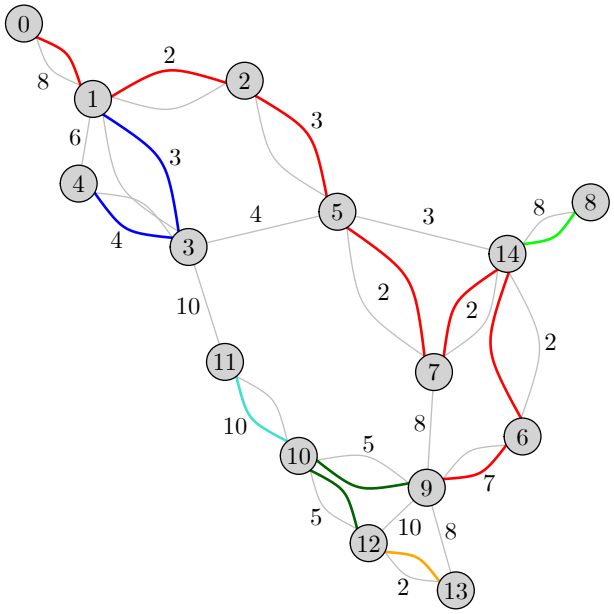


FIGURE 7.15: *Route set corresponding to the best TRT objective for the Mandl6 instance in Figure 7.13(a) returned by the NSGA II.*

Graphical representations of the ATT and TRT extremal solutions in Figure 7.13(a) are provided in Figures 7.14 and 7.15, respectively. The overall structures of these solutions are very similar to those returned by the DBMOSA algorithm. Finally, the objective function values of these solutions, as well as their transfer percentages, are once again very similar to those of the extreme solutions returned by the DBMOSA algorithm, as may be seen in Tables 7.16 and 7.17. These two tables contain summaries of the characteristics of the best overall extremal solutions returned by the NSGA II, compared against the best results reported in the literature. Finally, a last set of tests were conducted during which a static mutation management strategy was adopted, instead of employing the AMALGAM strategy. The results may be found in Figure A.25 of Appendix A. From these results, it is not clear which approach is better, and it may well be that similar results can be expected from both strategies.

7.4 Combined DBMOSA and NSGA II UTRP results

Having both a trajectory-based metaheuristic and a population-based metaheuristic available for solving instances of the UTRP may hold advantages, as explored in this section. One of the benefits (sometimes a drawback) of any SA algorithm is that it takes one solution as its current solution and, based on its various neighbouring solutions, explores the search space locally in the neighbourhood of this solution. This can be a time consuming task, but does offer flexibility of pursuing further improvements on solutions already obtained, such as solutions on the attainment front computed by another algorithm. The NSGA II, on the other hand, yields high-quality solutions in a reasonably short time. This population-based metaheuristic, however, does not possess particularly effective mechanisms for exploiting local optima, although it does yield a good overall spread of solutions along the approximate Pareto front, typically yielding a higher level of diversity of solutions than the DBMOSA algorithm.

In particular, the DBMOSA algorithm performed better in some instances with respect to obtaining high-quality solutions from the passengers' perspective for the five UTRP benchmark instances of §4.5.8, while the NSGA II yielded particularly high-quality solutions from the operator's perspective, as well as the passenger's perspective. The results returned by both the DBMOSA algorithm and the NSGA II are plotted alongside one another in Figure A.26 in Appendix A, together with the best results reported in the literature pertaining to the five benchmark instances mentioned in §4.5.8. The DBMOSA algorithm and the NSGA II can be combined by taking a subset of the solutions obtained in the attainment front of the NSGA II as the initial solutions of the DBMOSA algorithm, one at a time, thus hopefully leading to further improvement of the attainment front. An equispaced subset of 30 solutions from the NSGA II were therefore selected to seed the DBMOSA algorithm.

The attainment front computed by this method is provided in Figure 7.16, with HV values obtained reported in Table 7.18. From Figure 7.16 and Table 7.18 it is clear that no significant improvements were made for the three smaller instances, Mandl6, Mumford0, and Mumford1, while some significant improvements were evident for the larger two instances, Mumford2 and Mumford3. This is not a significant improvement in the extremal points, due to the NSGA II already producing results of a very high quality, which could not be improved upon significantly, but does yield an improvement in the middle region of the approximate Pareto set. These solutions in the middle range are solutions in which an operator may be specifically interested, and any improvement in this region may well be welcomed. Where no improvements were made, it can therefore be claimed, with some confidence, that local optima had been uncovered by the NSGA II.

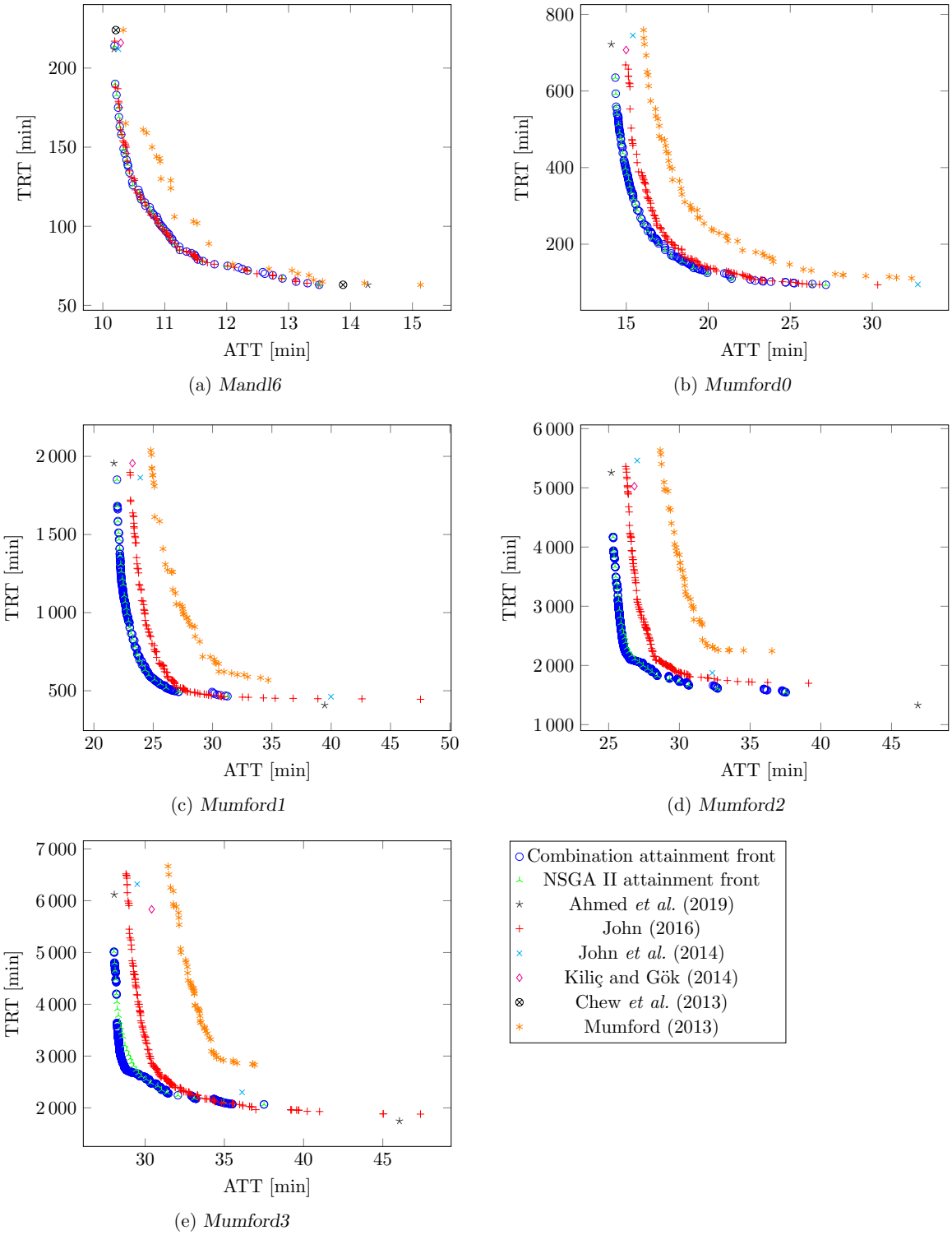


FIGURE 7.16: *Attainment fronts for solutions produced by the DBMOSA algorithm when initialised from a subset of the attainment front of the NSGA II of the previous chapter during the long runs of Test set 31, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8.*

TABLE 7.18: *The HV obtained during the performance tests when combining the NSGA II with the DBMOSA algorithm in order to solve the five UTRP benchmark instances of §4.5.8, along with the HV obtained by the DBMOSA of §6.1.7, that of the NSGA II of §6.1.8 and that of John [89]. The HV values in boldface are the best from among all the approaches.*

Solution approach	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
The DBMOSA of §6.1.7	85.3846	81.6712	78.7682	65.3447	58.2235
The NSGA II of §6.1.8	**86.0690**	**85.5074**	**81.1267**	70.4683	65.977
The combined approach	85.5263	85.0770	81.0378	**71.4094**	**66.2800**
John 2016 [89]	85.9769	80.8722	77.5761	64.9003	63.8547

7.5 UTRP with incremental redesign results

This section contains an introduction of the first set of results in the literature for the UTRP with incremental redesign based on the paradigm outlined in §5.2. The rationale for this extension is that previously, researchers were mainly concerned with redesigning an entire transit network anew. Recall that this novel approach, however, allows for providing as input a reference route set $\mathcal{R}_0$ from which the DP objective function is calculated. As no benchmark instances exist in the literature for this problem, five new benchmark instances are introduced in this section, along with numerical results pertaining to these benchmark instances when simultaneously minimising the two objective functions (5.5) and (5.7). The DBMOSA algorithm of §6.1.7 and the NSGA II of §6.1.8 were employed to solve these instances with the parameter values set to the values outlined in Tables 7.6 and 7.14, respectively. This once again showcases the flexibility of metaheuristics, especially when they are equipped with self-adaptive LLH management strategies, and may be employed readily on different problems.

For the reference route sets, a randomly generated initial feasible solution was chosen corresponding to each of the benchmark instances of §4.5.8, using the KSP-MMV-ICRSGP of §6.1.2. These route sets are presented in Table A.4 of Appendix A. The initial solution adopted for each run of the DBMOSA algorithm was the reference route set itself, and for the initial population for the NSGA II, the saved solutions returned by the enhanced KSP-MMV-ICRSGP were adopted, with the exception that one of the solutions was replaced with the reference route set. By having the reference route set within the population, subsequent generations would have at least one route set from which to draw solution quality. The reference route set naturally has a DP of 0%.

The non-dominated sets returned by the DBMOSA algorithm and the NSGA II are presented in Figure 7.17. The reference route set is the DBMOSA initial set. It is interesting to note the difference in shapes of the two non-dominated attainment fronts returned by the DBMOSA algorithm and the NSGA II. This may be ascribed to the different natures of the metaheuristics, the former being a trajectory-based search, and the latter being a population-based search. The trajectory-based search commences at the reference route set, but as the search progresses, the chances of revisiting routes that achieve lower disruption percentages decreases as the entire networks become increasingly disrupted. For the population-based search, on the other hand, the initial population is quite diverse, but as the search progresses, the population starts containing more solutions that more closely resemble the reference route set.

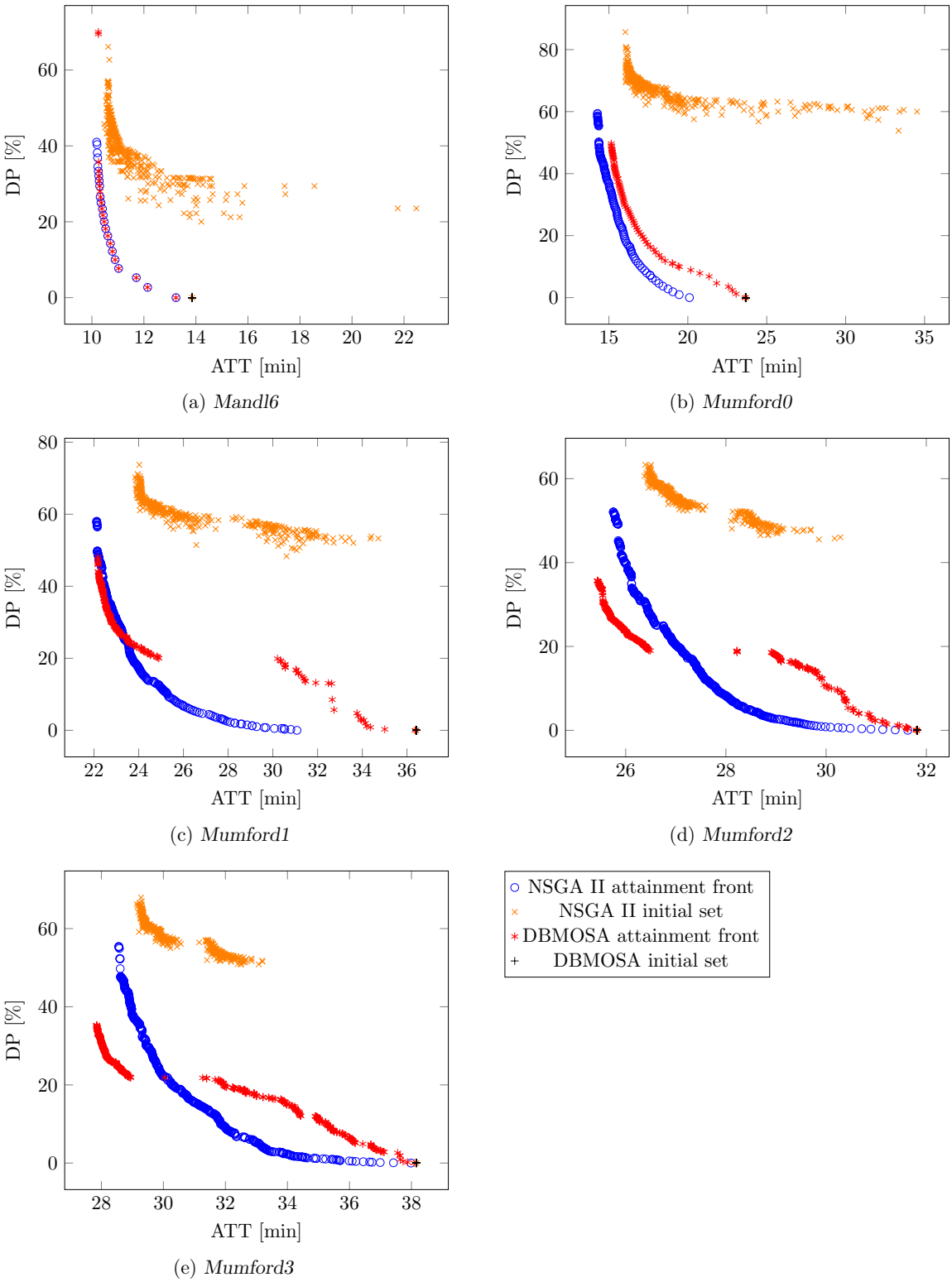
This may also explain why the DBMOSA does better in the ATT objective for the larger benchmark instances, as it does not converge prematurely due to being a trajectory-based search, while the NSGA II may converge prematurely by flooding the population with route sets that

are very similar, upon which it no longer effectively explores the search space. It is noteworthy that the DBMOSA algorithm managed to find a solution with an ATT of 27.8477 min, the lowest ATT encountered among all the runs reported in this chapter for the Mumford3 instance, when adopting the incremental redesigning paradigm. This solution corresponds to a TRT of 6 091 min. In terms of transfer percentage evaluation, the solution achieved 50.01%, 49.62% and 0.37% for the d_0, d_1 and d_2 values, respectively, and resulted in 0% unmet demand d_u. This solution outperformed the best solution from the passenger's perspective in the literature reported by Ahmed *et al.* [2] (having an ATT of 28.05 min and a TRT of 6 119 min) in terms of both ATT and TRT.

Another interesting feature of incorporating the DP objective function is that there may be route sets with similar or identical multi-sets of edges (*i.e.* achieving the same DP), and this is especially significant when a DP of 0% is achieved for a route set with a lower ATT than the reference route set. What this means is that there is a configuration of route sets, utilising the same edges, but ordered in a way that is more efficient, yet does not disrupt the entire transit system too much. From the results in Figure 7.17, it is clear that there exists a route set like this for each of the five benchmark instances introduced, identified as the solutions with a DP of 0% and achieving a lower ATT than that of the reference route set. Implementations of these routes could hold additional benefit for the passengers, at a very low cost to the operators.

A visualisation of the results achieved *via* this modelling approach and some of the route sets returned are presented for the smallest test instance, namely Mandl6, in Figure 7.18. Figure 7.18(a) contains the reference set, achieving an ATT of 13.86 min and a DP of 0%. Figure 7.18(b) contains a route set with the same multi-set edge composition as the reference set, but which obtains an ATT of 13.23 min instead, also with a DP of 0%. The difference between these two route sets is that in the latter, the light blue edge $\{v_0, v_1\}$ is replaced by a red edge, the orange edge $\{v_9, v_{10}\}$ is replaced by a red edge, the dark green edge $\{v_{10}, v_{12}\}$ is replaced by a red edge, and the red edge $\{v_9, v_{13}\}$ is replaced by an orange edge. These changes help facilitate the improvements required to obtain the route set in Figure 7.18(b) from the reference route set in Figure 7.18(a). The next two solutions achieving the smallest DP values found in Figure 7.17(a) are presented in Figures 7.18(c) and 7.18(d). Here, to transition from the solution in Figure 7.18(b) to the solution in Figure 7.18(c), a dark green edge $\{v_{10}, v_{11}\}$ is added, and this change leads to an ATT of 12.14 min, an improvement of 1.09 min, with a DP cost of 2.7%. Transitioning from the solution in Figure 7.18(c) to the solution in Figure 7.18(d) leads to an ATT of 11.71 min, which is a 0.43 min improvement at a total DP cost of 5.26%, and the change required is adding a dark green edge $\{v_6, v_{14}\}$. These are relatively small changes which lead to considerable improvements in ATT. By simply rearranging certain edges, and adding an additional two edges, the ATT can be reduced from 13.86 min to 11.71 min, which is a total improvement of 2.15 min at a DP cost of 5.26%.

The selection probabilities over the algorithmic iterations are presented in Figures A.27 and A.28, for the DBMOSA algorithm and for the NSGA II, respectively. It is clear that different prioritisations were assigned to the LLHs than to the selection probabilities for the solution implementations of the UTRP in the previous sections, §7.2 and §7.3, where the DBMOSA algorithm and the NSGA II were employed in a different context. Here, the cost-based trim, cost-based grow, add vertex, delete vertex and the delete inside vertex operators were favoured the most for the largest three benchmark instances in both the cases of the DBMOSA algorithm and the NSGA II.



(a) *Mandl6* (b) *Mumford0*

(c) *Mumford1* (d) *Mumford2*

(e) *Mumford3*

FIGURE 7.17: *Attainment fronts for solutions produced by the DBMOSA algorithm and the NSGA II of the previous chapter, where instances of the UTRP were solved upon having incorporated the DP objective function.*

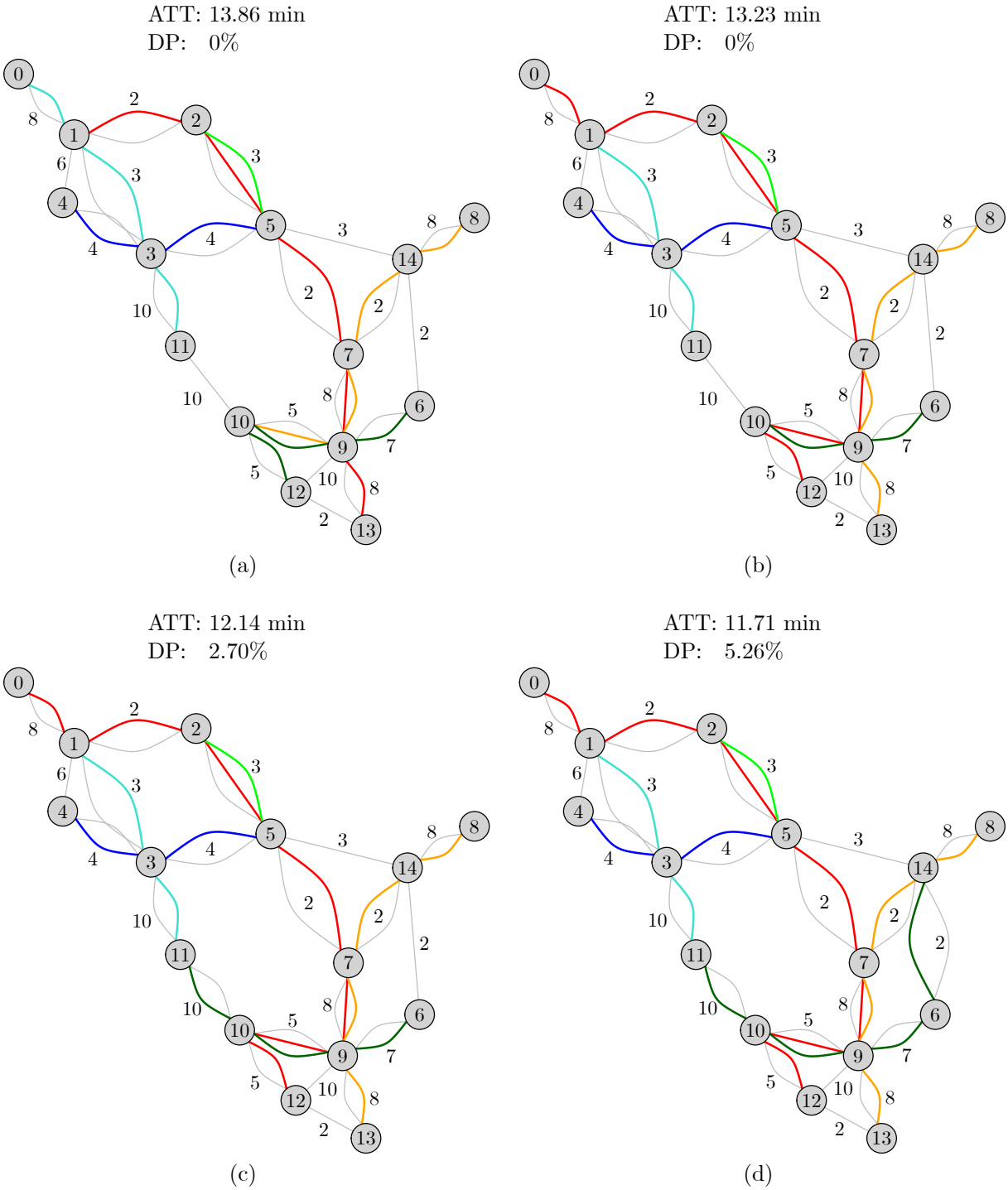


FIGURE 7.18: *Illustration of four solutions to the UTRP with incremental redesign, returned by the DBMOSA algorithm and the NSGA II for the Mandl6 instance. The initial reference route set is presented in (a), an improved route set in (b) with the same DP as that in (a), and two route sets in (c) and (d) with the 2nd and 3rd lowest DP returned by the algorithms. The ATT and DP objective function values achieved are indicated above each route set.*

7.6 UTFSP results returned by the NSGA II

Validation of the NSGA II of §6.2.3 is considered in this section. For the UTFSP, very few benchmark instances have been published in which the two objective functions described in §5.3, as discussed by John [89], are addressed simultaneously. John has, however, published results returned by his implementation of the NSGA II for solving instances of the UTFSP, and these results are used as validation data for the NSGA II of §6.2.3.

John's [89] approach towards solving the UTFSP involved selecting five solutions across his attainment front of the UTRP, namely the best solution for operators, the best solution for passengers, and then solutions closest to the first, second and third quartiles. Due to the long computation times associated with calculating the passenger objective function value, caused by having to solve the transit assignment problem for each new set of frequencies, the computational resources available for adopting this method are typically limited. It was therefore decided only to consider the best solution from the operator's perspective and to use those routes as the benchmark instance. This solution is $\mathcal{R}_O = \{\langle 4,3,1\rangle, \langle 13,12\rangle, \langle 8,14\rangle, \langle 9,10,12\rangle, \langle 9,6,14,7,5,2,1,0\rangle, \langle 10,11\rangle\}$, which attained an ATT of 13.480 min and a TRT of 63 min. A graphical representation of this route set is shown in Figure 7.19.

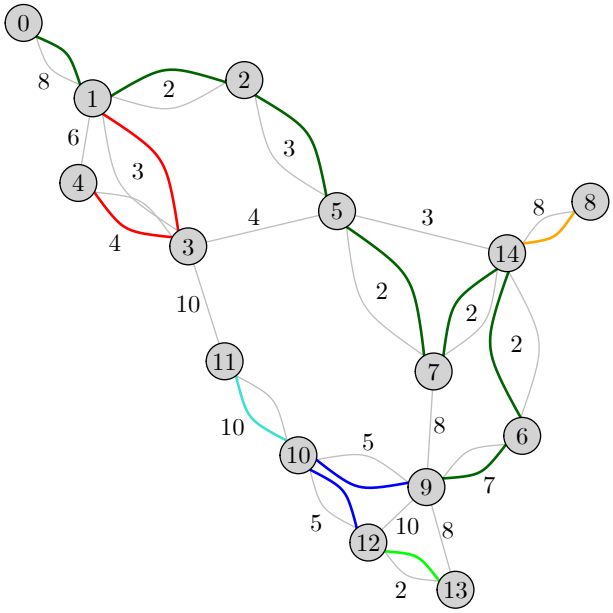


FIGURE 7.19: *Route set corresponding to the best operator cost, computed by John [89].*

Once again, the HV performance measure is used to compare the relative qualities of solution sets, and the reference point is selected by assigning the maximum and minimum frequencies to all the routes, respectively. The best possible AETT is achieved by setting the frequencies of all the routes equal to the largest value in the set $\theta = \left\{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\right\}$ (*i.e.* $\frac{1}{5}$), yielding an AETT of 12.933 min, and the corresponding worst objective function value for the TBR of 25.2 buses. Similarly, the best TBR-value is achieved by setting all the frequencies of the routes to the smallest value in θ (*i.e.* $\frac{1}{30}$), and this yields a TBR of 4.2 buses and an AETT of 17.068 min. These values were taken as reference points when computing normalised HV values, as mentioned in §2.6.2.

When longer routes are present, the transit graph increases in size, as described in §5.3, and therefore the computation time associated with the transit assignment decisions of passengers also increases. For testing and evaluation purposes, it makes sense to consider the smaller in-

stance so that more tests may be conducted in a shorter time frame in order to fine-tune the parameters of the NSGA II for solving the UTFSP instance. Similar to the general approach followed to obtain the NSGA II results in §7.3, the same GA parameters were chosen for consideration, namely the crossover probability, the mutation probability, the population size and the number of generations.

The main difference is that one function evaluation for the UTFSP, considering the smallest route network, takes about 6 seconds on a 3.4 GHz Intel ® Core™ i7-4770 processor with 16 GB of RAM running in the Windows 10 operating system, and one function evaluation for the UTRP takes about 0.786 milliseconds. It therefore takes approximately 7 630 times longer to compute the objective function values of UTFSP instances than those of UTRP instances. One algorithmic run of the NSGA II of §6.2.3 therefore took 7 hours and 21 minutes when 20 generations were simulated with a population size of 200 solutions. Computing 20 algorithmic runs for each combination of parameter values would become very cumbersome. One algorithmic run was therefore executed for each of the baseline and test value combinations, and the HV obtained over the entire number of generations was observed in order to ascertain which parameter values perform better, while simultaneously reducing the computational burden of the parameter tuning procedure. The values to which the baseline and test value combinations were set are summarised in Table 7.19.

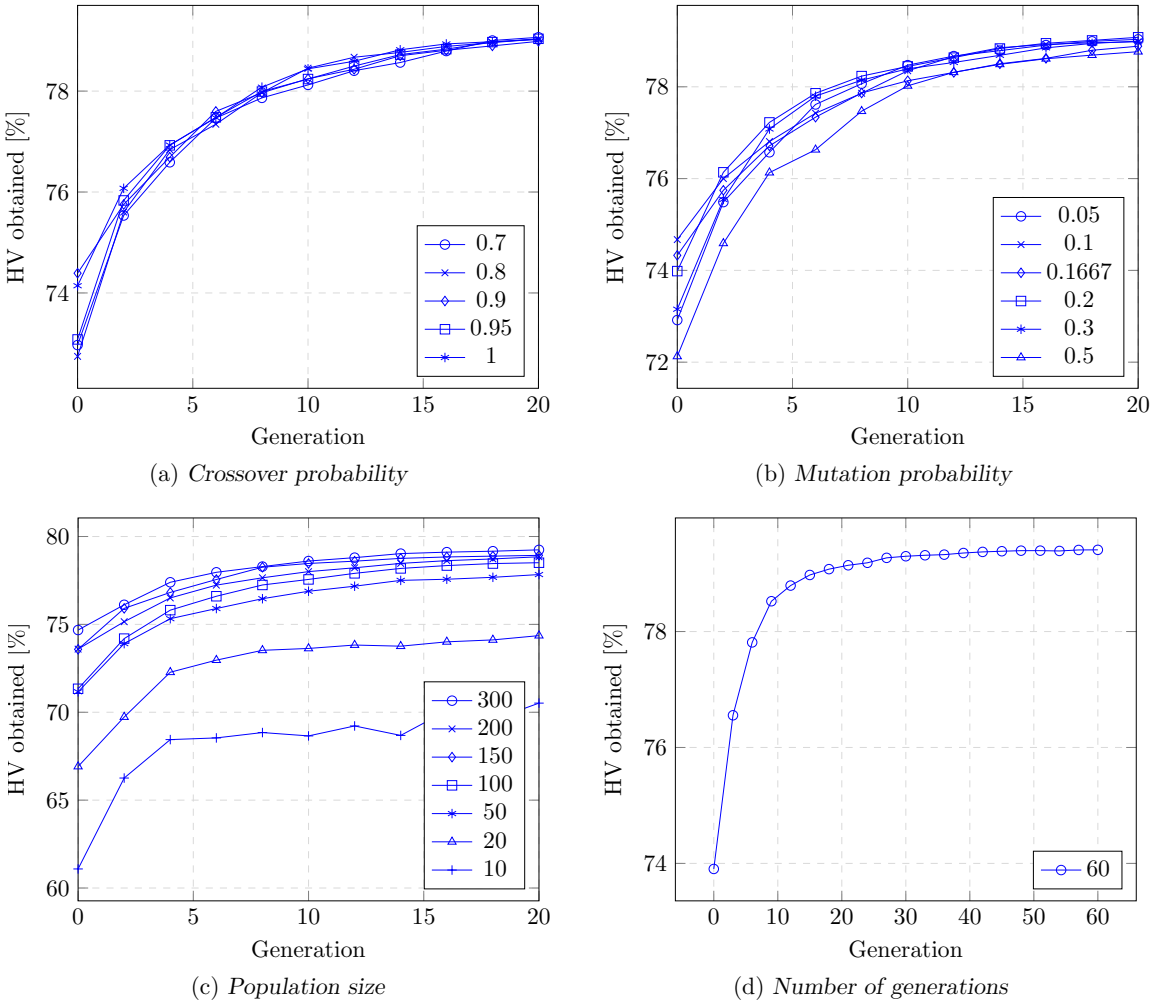
TABLE 7.19: *The design parameters considered when tuning the NSGA II for solving John's [89] instance of the UTFSP, the baseline value of each parameter, and the test values varied during each algorithmic run.*

Parameter	Baseline value	Test values
Crossover probability	0.9	0.7, 0.8, 0.95, 1
Mutation probability	0.1667	0.05, 0.1, 0.2, 0.3, 0.5
Population size	200	10, 20, 50, 100, 150, 200, 300
Number of generations	20	60

The baseline values for the crossover probability, the mutation probability and the population size were chosen as 0.9, $\frac{1}{|\mathcal{R}|}$ and 200, respectively, following John [89]. The number of generations was, however, set to 20 due to computational limitations and in pursuit of finding a good trade-off value. The mutation probability was calculated as 0.1667 for six routes. The test values, shown in Table 7.19, were chosen in such a way that a good spread of the influence of each parameter could be observed, and could be adjusted accordingly.

When considering the crossover probability, one model run was conducted for the base value and one for each test value. The resulting HV values obtained over the total number of generations is plotted in Figure 7.20(a). This figure reveals that among all the crossover probability values tested, the HV results returned converged to a similar performance, but a value of 0.7 performed marginally better than the other values, and therefore was chosen as the crossover probability for the NSGA II of §6.2.3. In terms of the mutation probability, Figure 7.20(b) shows that the best HV value was achieved when the mutation probability was set to 0.2. The value of 0.1667 performed marginally worse, and considering the fact that the expression $\frac{1}{|\mathcal{R}|}$ used to compute this value is scaleable when the number of routes changes, it makes sense to incorporate this value instead, again following John [89]. This expression was therefore selected for the mutation probability in the NSGA II of §6.2.3.

The population size and the number of generations are the two factors that have the largest impact on the computational cost associated with the model, and if smallest values of these parameters are found which do not compromise excessively on solution quality, it would amount



(a) *Crossover probability* (b) *Mutation probability*

(c) *Population size* (d) *Number of generations*

FIGURE 7.20: *The design analysis results for one run of the NSGA II in the context of the UTFSP, where Figures (a)–(d) indicate the plots of the HV over the generations obtained when varying each parameter mentioned, and the legend indicating the values to which the parameters were set.*

to a considerable computational cost benefit. Figure 7.20(c) reveals that taking a population size of 20 or below delivers poor quality solutions, due to not introducing enough diversity into the population, while a value of 50 or upwards delivers high-quality solutions. A population size of 300 delivered the best results, and this value may be used when considering the final design of transit networks, but the values of 150 and 200 performed only marginally worse than did 300, and so these could be viable values too. The value of 200 was therefore chosen for further testing purposes. This decision saves about a third of the time when solving instances of the UTFSP by means of the NSGA II. Figure 7.20(d), on the other hand, shows the effect of allowing the algorithm to perform more generations, up to 60. It may be observed that for more than 30 generations, the HV obtained starts to plateau, and that only incremental improvements are made from that point onwards. Once again, when considering a final set of parameter choices if algorithmic runtime is not a factor, it is suggested to let the algorithm run for more generations. While conducting preliminary testing, on the other hand, a value of 30 generations should be sufficient.

A sensitivity analysis was conducted during which the magnitude of the effect of each parameter on HV was evaluated. The results can be seen in Figure 7.21. It is clear that the population

size is the largest contributing factor, and this makes sense as population size plays a significant role in the diversity of the solutions, and therefore the potential quality of solutions uncovered later during the search. It is noted, however, that small values of the population size have much more severe consequences than setting the value too large. One should therefore gravitate towards setting the population size larger rather than smaller. The number of generations has the second largest impact on the quality of the results, and similar to the population size, more severe effects are observed when choosing its value too small. This parameter should be adjusted carefully so that the algorithm is allowed to run over a sufficient number of generations. The mutation probability and the crossover probability have the smallest effect on the algorithmic performance, but should also be adjusted carefully for improved performance.

Figure 7.22 contains two attainment fronts obtained by the NSGA II of §6.2.3, in which the walk factor was taken as 3 and 100, respectively. A walk factor of 100 will lead passengers to avoid all walk links based on the optimal strategies assignment, and therefore these results are comparable to those of John [89] and his modelling approach. It is clear in Figure 7.22 that when incorporating walk links, and imposing a relatively small walk cost factor on each link, such as a value of 3, for example, considerable flexibility may be afforded to passengers to choose even shorter routes based on the optimal strategies assignment model, therefore allowing the attainment front to perform better with respect to the AETT objective. When, on the other hand, the walk cost factor is set high, such as a value of 100, for example, the option of following walk links is in essence not considered due to passengers being able to reach their destinations faster by making use of the bus routes provided. The walk factor plays a larger role in the evaluation of the AETT when the frequencies of bus routes are set low, and waiting times for buses become long, in which case the option of walking will instead be exercised by passengers.

It should be noted that when incorporating walk links, passengers have the choice of not making use of the bus service provided; a scenario that operators would like to avoid in order to maximise their revenue. This modelling paradigm may be adopted when considering whether buses will be preferred over walking, and approximating the passenger demand that will be lost due to bus routes not sufficiently meeting passenger demand. By knowing the walk links that passengers may use, bus routes can be designed more effectively by ignoring routes that cannot be covered sufficiently due to the presence of walk links which allow passengers to reach their destinations in less time. Additionally, buses may be assigned to routes more appropriately by ensuring a sufficiently high bus frequency operating along bus routes so that passengers would hopefully prefer to make use of buses as opposed to walking.

The attainment front obtained by the NSGA II of §6.2.3, when the walk factor is set to 100, is almost identical to that obtained by John [89], but with the exception that a small offset is present due to the assumption that passengers take six seconds to board a bus and six seconds to alight a bus, whereas John assumed the time for each to be zero seconds. The route set considered by John [89] yields an ATT of 13.493 min when setting all routes' frequencies to $\frac{1}{10}$. The AETT objective function (5.8), however, returns an estimation of the average travel time for passengers with higher accuracy than the ATT objective function (5.5), due to taking bus arrival waiting time into consideration [89]. The AETT objective function evaluations are therefore, on average, larger than the ATT objective function evaluation of the same transit network.

The extremal solutions, Solutions A and B, in Table 7.20 indicates that the best frequency setting for passengers, resulting in the best AETT, is obtained when the frequencies are set high (Solution A), and that the best frequency setting for operators, resulting in the best TBR, is obtained when the frequencies are set low (Solution B).

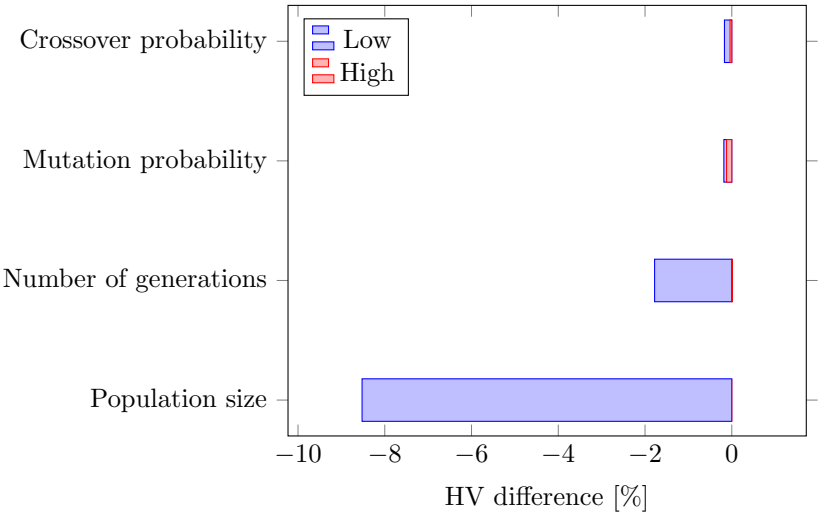


FIGURE 7.21: *Results of a sensitivity analysis conducted for the NSGA II of §6.2.3 with the various algorithmic parameters ordered on the vertical axis in increasing magnitude of impact on algorithmic performance, measured in terms of the difference in HV on the horizontal axis achieved when adopting the low and high values, respectively.*

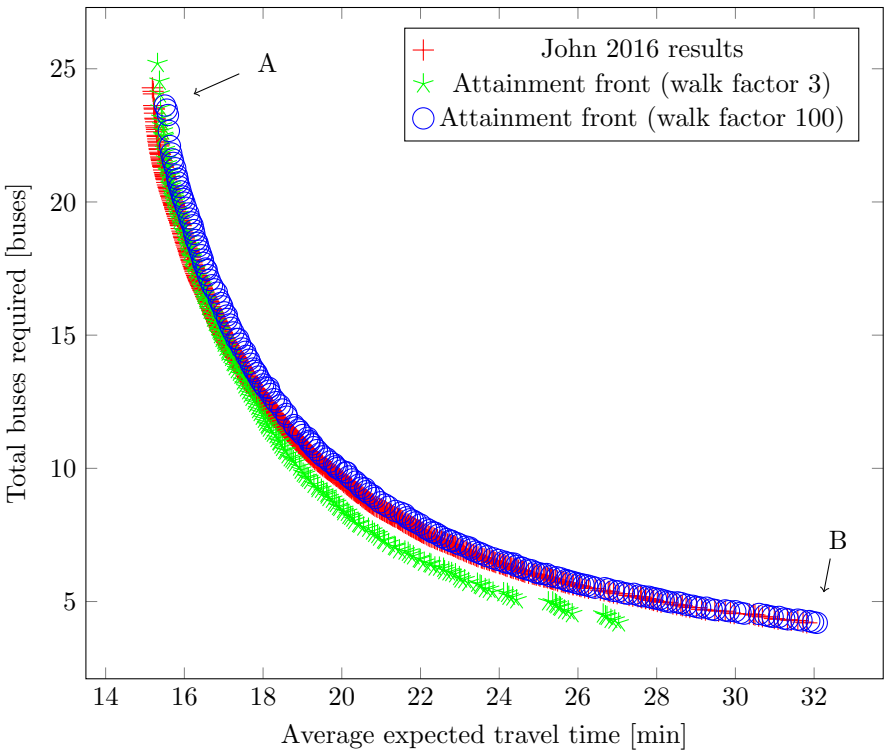


FIGURE 7.22: *Attainment front for the UTFSP instance in Figure 7.19 for sets containing six routes.*

TABLE 7.20: *The objective function evaluation and associated frequencies for the extremal solutions A and B in Figure 7.22, returned by the NSGA II when solving the UTFSP instance in Figure 7.19.*

Extreme solutions	AETT	TBR	Frequencies for routes
Solution A	15.507 min	23.619 buses	$\{\frac{1}{5}, \frac{1}{5}, \frac{1}{7}, \frac{1}{5}, \frac{1}{5}, \frac{1}{6}\}$
Solution B	32.068 min	4.2 buses	$\{\frac{1}{30}, \frac{1}{30}, \frac{1}{30}, \frac{1}{30}, \frac{1}{30}, \frac{1}{30}\}$

7.7 Chapter summary

This chapter was devoted to a validation of the algorithmic implementations of Chapter 6 for solving the models of Chapter 5. The chapter opened in §7.1 with a discussion on the initial solutions returned by the KSP-MMV-ICRSGP and the enhanced KSP-MMV-ICRSGP, the latter of which proved to deliver very favourable results of high quality. Thereafter, the UTRP model was considered and solved by invoking both a trajectory-based approach and a population-based approach, namely the DBMOSA algorithm and the NSGA II, respectively. The results returned by these implementations in the context of the five benchmark instances of §4.5.8 were discussed in §7.2 and §7.3, respectively. Both algorithms returned high-quality results, competing with results published in the literature. Next, these two algorithms were combined in a bid to further improve upon the quality of the results in §7.4, and this proved to be of benefit in view of the larger instances of the benchmark problem instances considered. Thereafter, in §7.5, benchmark instances for the UTRP with incremental redesign were introduced and solved by means of both the DBMOSA algorithm and the NSGA II of the previous chapter, which led to some interesting insights. The chapter finally closed in §7.6 with a validation of the NSGA II of §6.2.3 for solving instances of the UTFSP and involved comparing the solutions thus found with those previously published in the literature, which were found to be in close agreement.

Part III

Case study

CHAPTER 8

A case study

Contents

As mentioned in Chapter 1, the Matie Bus is a transport service provided by Stellenbosch University to its students [155]. The focus of the service is to provide transport to students between parking lots and lecture venues on the campus periphery, and lecture venues located centrally on campus. An effective redesign of the routes of such a bus service may lead to a larger portion of students potentially being serviced. This chapter is devoted to a case study on the Matie Bus service with a view to propose an alternative set of high-quality bus routes that may be implemented in service of a larger portion of the student population. The establishment of a new set of bus stops is addressed in §8.1, after which the focus in §8.2 shifts to the input data required to carry out the case study. The results obtained when applying the solution methodologies of Chapter 6 to solve the relevant UTRP and UTFSP instances in the context of this case study are discussed in §8.3. The chapter finally closes in §8.4 with a brief summary of its contents.

8.1 Bus stop locations

Currently the day shuttle of the Matie Bus services six bus stops between 07h00 and 17h30 on Mondays to Fridays. These are the Faculty of Food Science, a parking lot at the Launch Lab, a bus stop on Joubert Street, the Endler Hall of the Conservatory of Music, the Welgevallen Experimental Farm and the Coetzenburg Sports Centre [155], displayed in Figure 8.1 (with each stop numbered in the order that it was mentioned above). The main operating time of the University is from 08h00 to 17h00. After these hours, the buses take on a different function, servicing students by transporting them from the Neelsie Student Centre directly to their required destinations, leaving the Neelsie every hour (on the hour) from 18h00 to 02h00 (inclusive) on every day of the week [155].

Recently, Klink [98] published data, gathered in 2019, about movements of different modes of transport on the campus of Stellenbosch University. He used 44 custom-built Bluetooth enabled sensors to observe unique identification codes for mobile devices that had their Bluetooth

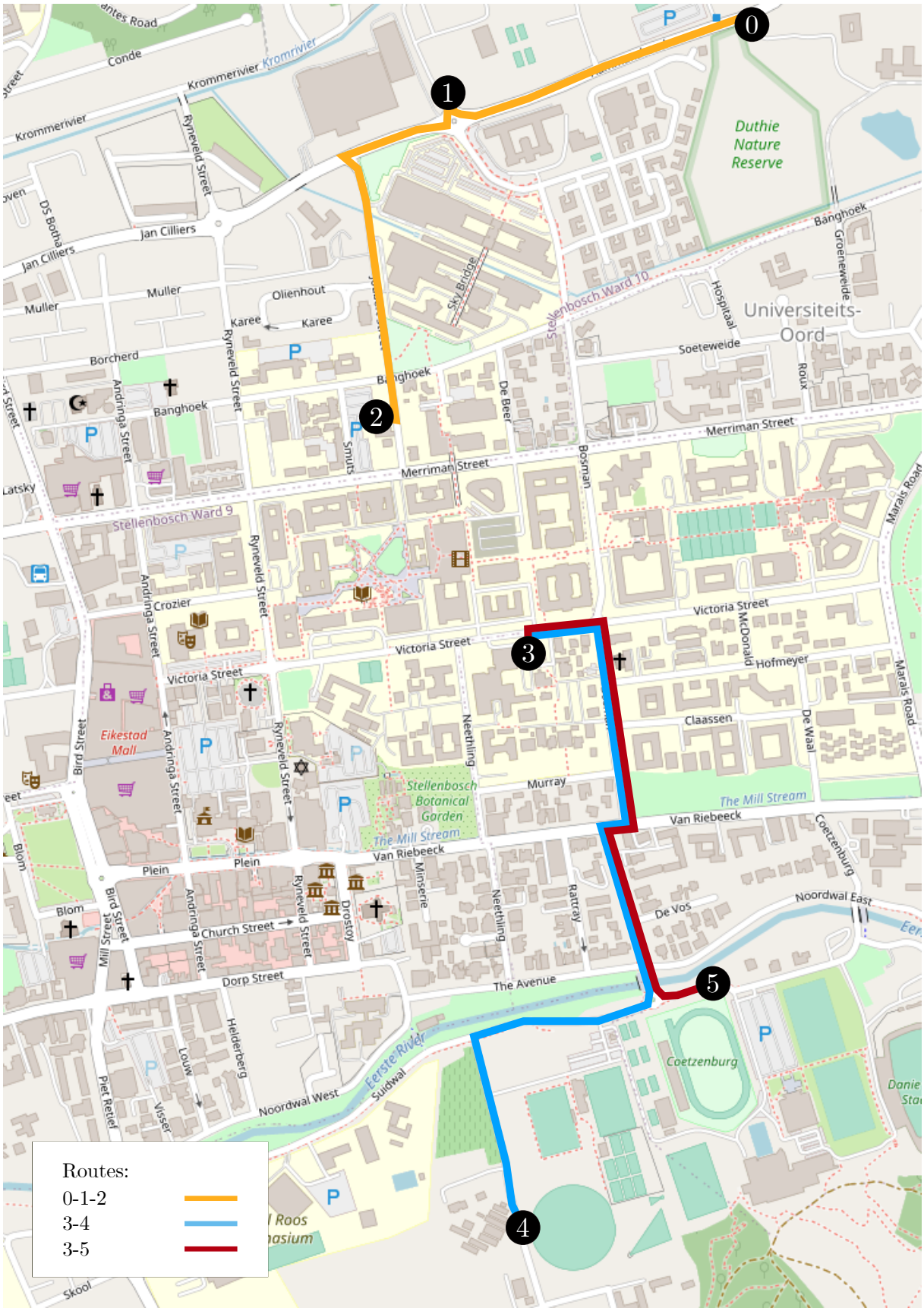


FIGURE 8.1: *Matie Bus routes for the day shuttle service [155].*

enabled. The data revealed the movement of both motorised and non-motorised transport users in terms of volume and spread, and were filtered into OD demand matrices for each hour of the day. These OD demand data are utilised in this case study. The locations of the 44 Bluetooth sensors were aggregated into ten zones, clustered according to common zoning attributes, such as commercial, educational, recreational or residential. These ten zones are indicated in Figure 8.2, superimposed on a map of Stellenbosch. The author subsequently decided upon locations for bus stops in order to satisfy travel demand between the zones. These bus stops are labelled 0–9 in Table 8.1 and Figure 8.2. The name, general type of stop and geographical coordinates of each stop are provided in Table 8.1.

TABLE 8.1: *Proposed Matie Bus service stop location names, their types, as well as their latitude and longitude coordinates.*

Label	Name	Type	Latitude	Longitude
0	Engineering faculty	Educational	−33.929 972	18.865 677
1	Cnr of Andringa St and Banghoek Rd	Residential	−33.930 799	18.859 930
2	Eikestad Mall parking lot	Commercial	−33.935 169	18.861 320
3	Cnr of Van Riebeeck Rd and Neetling St	Residential	−33.936 725	18.866 079
4	Cnr of De Waal Rd and Claassen St	Residential	−33.934 916	18.871 437
5	Cnr of Mcdonald Rd and Victoria St	Residential	−33.933 425	18.870 353
6	Bus stop in Merriman Avenue	Residential	−33.930 843	18.871 007
7	Central campus	Educational	−33.932 234	18.864 670
8	Cnr of Victoria St and Neetling St	Educational	−33.934 052	18.865 602
9	Stellenbosch University sport grounds	Recreational	−33.940 034	18.870 718

The bus stop locations were chosen in such a way as to allow for natural stops for buses, along with bi-directional accessibility based on prior knowledge of the town layout of Stellenbosch. The locations of Bus stops 4, 8 and 9 were chosen outside the aforementioned zones. In the case of Bus stop 4, the reasoning was that it is the most central bus access point for a number of female residences, and is located closer to the attractions to which people would congregate if walking from Zone 4. Although some residences are also present, Zone 8 is classified as educational due to its centrality, containing many educational buildings near the Southern boundary of the central campus. Bus stop 7 was chosen in such a way that it covers the Northern boundary of the central campus, therefore also covering both sides of the central campus, since people can walk between Bus stops 7 and 8. Bus stop 9 was chosen in such a way that it is close to the university sports grounds.

Thereafter, the author determined the allowed road links that could be considered in an instance of the UTRP as possible edges that may be traversed by buses. The reasoning followed during this choice was that the main attraction types are commercial, educational and recreational, and that all residential areas should be allowed direct access to these attractions. Therefore, no edges exist between the bus stops of residential zones, but instead there are edges between the bus stops of non-residential zones and all other bus stops, as illustrated in Figure 8.3. This road network can therefore be represented as a graph of order of 10 and size 36.

8.2 The input data

The traversal times associated with road links in the transport network of Figure 8.3 are a function of time and traffic density, and can be estimated by utilising satellite technology [73].

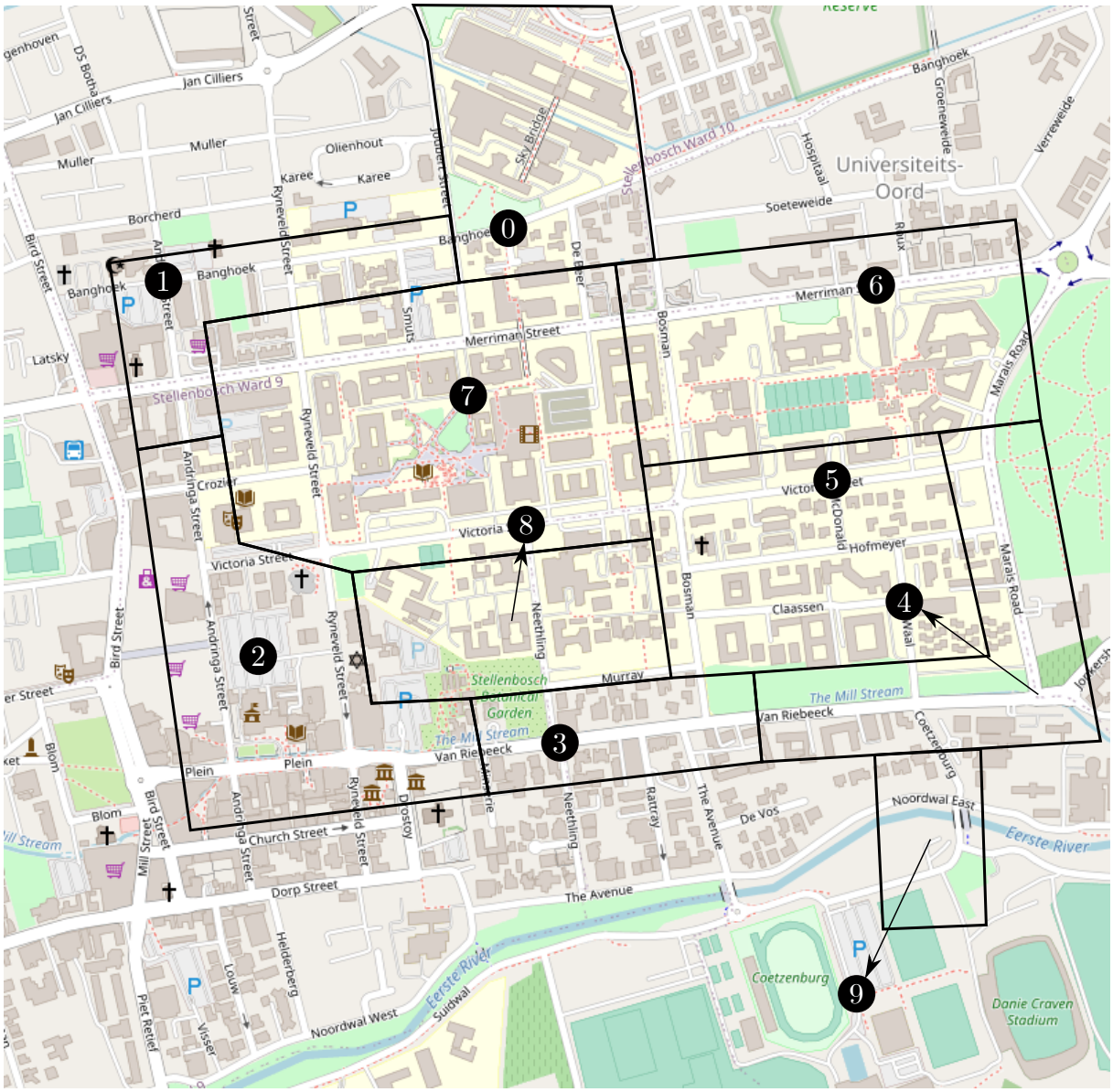


FIGURE 8.2: *An illustration of the zonal partitioning of the data gathered by Klink [98], indicated by the black boundaries. The numbered circles represent bus stop locations recommended by the author for these zones.*

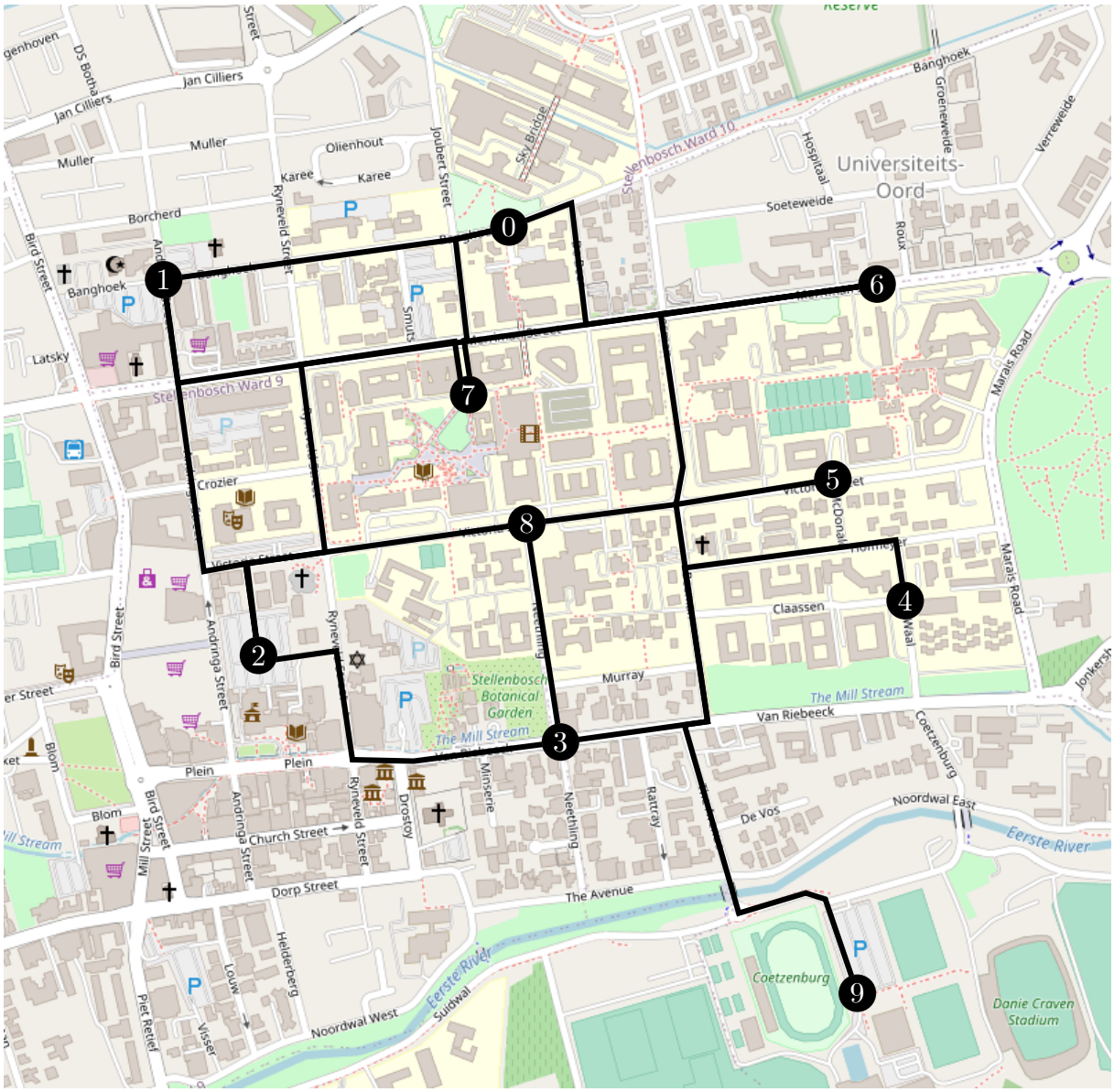


FIGURE 8.3: *The road network considered during the Matie Bus case study.*

In a previous study, time instants during the morning peak period (06h00, 06h15 and 06h30), the off-peak period (12h00 and 12h15) and the afternoon peak period (16h30) were considered for data gathering purposes [75]. In order to obtain a good approximation of the required distance matrix $\boldsymbol{W}$ for this case study, the 12h00 off-peak period was chosen for data gathering purposes as this period is representative of other periods during a week day. Estimated travel times were gathered from Google Maps [66] during the 12h00 off-peak period between all pairs of proposed bus stop locations in the network of Figure 8.3. The resulting distance matrix is shown in Figure 8.4 in the required *.csv* file format.

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	2	4	3	4	3	2	2	3	5
3	**1**	2	0	3	10000	10000	10000	10000	3	4	6
4	**2**	4	3	0	3	4	3	4	3	2	5
5	**3**	3	10000	3	0	10000	10000	10000	3	1	3
6	**4**	4	10000	4	10000	0	10000	10000	4	3	4
7	**5**	3	10000	3	10000	10000	0	10000	3	2	4
8	**6**	2	10000	4	10000	10000	10000	0	2	2	4
9	**7**	2	3	3	3	4	3	2	0	3	5
10	**8**	3	4	2	1	3	2	2	3	0	4
11	**9**	5	6	5	3	4	4	4	5	4	0

FIGURE 8.4: *The distance matrix at 12h00 for the Matie Bus case study. All values are measured in minutes, and the 10 000 values represent placeholders where no edge exists between two vertices.*

The accuracy of OD demand data is crucial during the design of a transit system aimed at meeting the demand of passengers effectively. These data are usually estimated, but with the emerging technology of Bluetooth and Wi-Fi sensors [48] mentioned in Chapter 1, it is now possible to generate accurate OD demand matrices. The SSML [153] has launched a project to install such sensors across the campus of Stellenbosch University that facilitate this type of data gathering.

The data thus gathered were made available to the author in 2019 [98] and may be used in conjunction with the UTRP model of Chapter 5 to design bus routes that can effectively meet the demand of the students. For the purpose of designing bus routes, it is desirable to establish a set of routes that remain fixed throughout the day. For any one of these routes the number of buses assigned to it throughout the day may, however, vary in order to accommodate fluctuating demand patterns. The OD demand matrix $\boldsymbol{D}_{\text{Day}}$ for the route design was established by aggregating the OD demand matrices gathered by Klink [98] over the operating hours of the Matie Bus service (07h00–17h00), so that the overall movement of students throughout the network could be computed, and the routes designed accordingly. This OD demand matrix is shown in Figure 8.5. While this demand matrix is not symmetric, as assumed for the UTRP model in §5.1, it is noted that both the DBMOSA algorithm of §6.1.7 and the NSGA II of §6.1.8 are also capable of handling asymmetric demand matrices.

Once the route set has been established for the Matie Bus transportation network, frequencies, or buses, may be assigned to each route so that sufficient provision is made for the demand along that route. In order to set up a UTFSP model instance for determining these frequencies, an appropriate route set should be chosen from the non-dominated solutions obtained when solving the corresponding UTRP model instance. Thereafter, the UTFSP instance may be solved for a specific time instant, in respect of its own OD demand matrix for that specific instant. The three main time instants, 08h00, 12h00, and 17h00, each represents unique circumstances during the day. Students predominantly move towards the campus at 08h00, students move to lunch

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	503	48	31	25	238	2131	2251	34	0
3	**1**	527	0	1852	13	15	19	178	3955	75	0
4	**2**	29	2025	0	909	137	144	61	9687	966	12
5	**3**	26	15	1001	0	763	1334	130	435	888	42
6	**4**	28	13	52	1011	0	1343	392	222	85	360
7	**5**	225	15	224	1722	1531	0	8154	2435	1354	16
8	**6**	2508	223	90	129	369	7870	0	3901	346	1
9	**7**	1873	3963	9077	417	107	2300	3647	0	12639	2
10	**8**	45	48	1367	869	70	2150	444	9859	0	5
11	**9**	0	0	17	13	102	12	6	5	1	0

FIGURE 8.5: *The aggregated OD demand matrix for each hour of the day between 07h00 and 17h00 for the Matie Bus case study.*

venues at 12h00, and then they return to their places of residence or some other attraction at 17h00. The OD demand matrices for each of these time instants are shown in Figure 8.6.

When choosing the desired number of stops along a route, it was noted that lectures start at every hour of the day between 08h00 and 17h00 (on the hour) and that students often have to be transported between different venues for these lectures. Lectures typically end 45–50 minutes after the hour and so there is only 10–15 minutes available for this transportation. The routes should therefore preferably not be more than 10 minutes in length, but if a route is too long, this problem can be remedied by setting the frequencies appropriately once the bus routes have been established. The number of stops were consequently set at 2–6 per route, allowing the algorithm to find a high-quality set of routes within this range. The number of bus routes sought was specified as eight in order to accommodate an appropriate service coverage.

8.3 Numerical results

This case study was performed by providing the data described in the previous section as input to the algorithms of Chapter 6. The route sets recommended by the algorithms when solving the UTRP instance are discussed briefly in this section. Figure 8.7 contains a graphical representation of the non-dominated attainment front in objective function space from which a decision maker can make a subjective trade-off selection for implementation purposes.

It is suggested that the routes be chosen in such a way as to accommodate the students' needs as the main goal, because the Matie Bus service exists to aid students in their transport needs. Therefore, the route set achieving the smallest ATT is preferred for implementation purposes. For this to be a viable option, however, the operators should be able to maintain the route set. The TRT required for the route set determines its viability in this sense, and so this is where a trade-off needs to be made. An assumption is made that an ATT of 3 minutes and 25 seconds provides a sufficient service level in the UTRP context — it should be kept in mind that the AETT will be larger than the ATT for any solution as soon as waiting times for buses are taken into consideration. The solution adhering to this requirement with the closest ATT is therefore chosen as the recommended route set, and shown in Figure 8.7 in objective space indicated by the label C. This solution achieves an ATT of 3 minutes and 23 seconds, and a TRT of 40 minutes. The route set corresponding to Solution C in Figure 8.7 is shown in decision space in Figure 8.8. This route set is anticipated to facilitate students reaching their respective destinations within the time constraints imposed when transitioning between lectures.

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	49	4	1	1	23	193	254	3	0
3	**1**	39	0	174	1	1	0	12	435	5	0
4	**2**	1	121	0	56	4	4	4	845	60	0
5	**3**	3	0	67	0	37	142	8	65	120	3
6	**4**	3	2	3	72	0	136	36	13	6	21
7	**5**	21	1	7	123	91	0	689	305	112	0
8	**6**	178	23	3	5	10	730	0	579	38	0
9	**7**	121	305	640	44	4	155	237	0	1027	0
10	**8**	1	5	114	60	1	144	26	907	0	0
11	**9**	0	0	2	0	3	1	0	0	0	0

(a) *08h00*

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	62	5	2	1	24	271	270	4	0
3	**1**	55	0	180	0	1	3	16	392	7	0
4	**2**	3	184	0	92	13	15	6	965	110	1
5	**3**	1	2	101	0	72	95	5	29	86	2
6	**4**	7	0	3	96	0	96	29	20	13	10
7	**5**	19	1	23	166	148	0	909	209	140	0
8	**6**	245	18	8	10	38	883	0	337	30	0
9	**7**	221	422	1077	42	10	277	448	0	1528	0
10	**8**	3	5	155	99	6	261	47	1103	0	1
11	**9**	0	0	0	1	4	1	0	0	0	0

(b) *12h00*

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	36	9	5	5	15	157	106	2	0
3	**1**	30	0	167	3	1	1	24	285	5	0
4	**2**	2	228	0	128	13	24	7	923	97	2
5	**3**	6	1	105	0	120	150	10	39	68	6
6	**4**	4	1	6	109	0	156	56	33	9	130
7	**5**	10	1	32	234	208	0	525	183	120	7
8	**6**	155	20	15	23	50	519	0	211	28	0
9	**7**	126	283	870	44	15	222	273	0	935	0
10	**8**	8	2	125	70	9	193	47	711	0	2
11	**9**	0	0	4	4	27	2	1	1	0	0

(c) *17h00*

FIGURE 8.6: *The OD demand matrices for the Matie Bus case study at different hours of the day.*

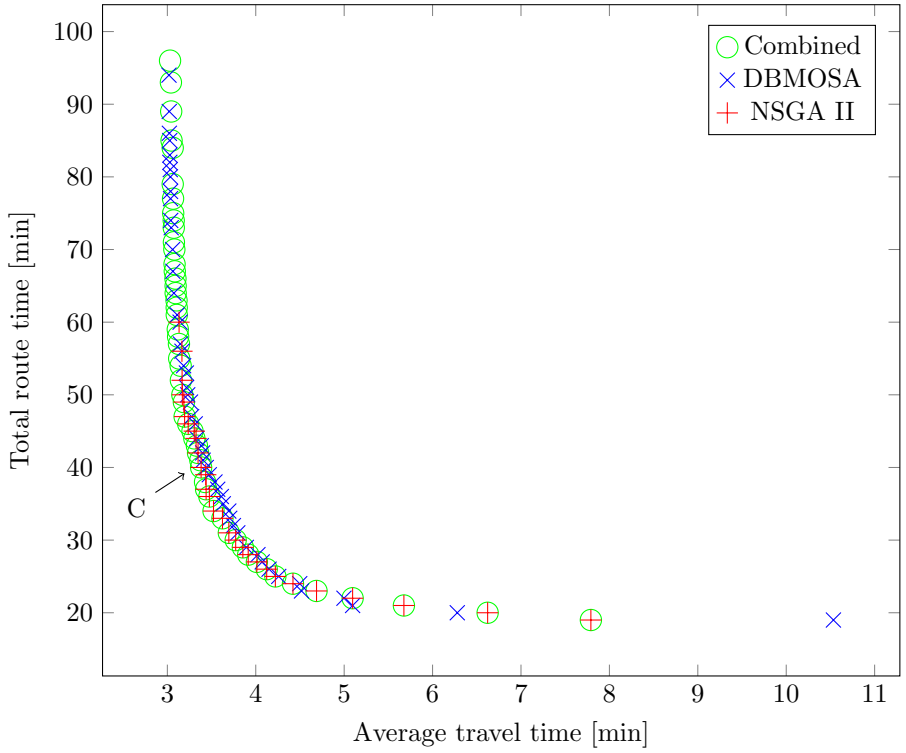


FIGURE 8.7: *The non-dominated attainment front obtained by each of the methods described in §6.1 when solving the UTRP instance for the Matie Bus case study. The combined method entails executing the DBMOSA, taking as initial solutions those solutions returned by the NSGA II one by one. Each solution contains eight routes of between 2 and 6 stops (inclusive).*

Table 8.2 depicts the full route set evaluation for solution C in Figure 8.7, containing its objective function values and its transfer percentages. According to Chakroborty [31], this route set would be classified as a good route set due to it satisfying all transit demand, with a large percentage of the demand being satisfied by direct connections.

TABLE 8.2: *The objective function values and transfer percentage evaluation for the chosen route set corresponding to Solution C in Figure 8.7, uncovered by the NSGA II in respect of Matie Bus case study depicted in Figure 8.3.*

Solution	ATT	TRT	d_0	d_1	d_2	d_u
Solution C	3.274 min	40 min	95.21	4.79	0.0	0.0

The route set depicted in Figure 8.8 was subsequently taken as input for the UTFSP instance of the Matie Bus case study. It was decided that the routes will be kept constant throughout the day, due to disruptions of route configuration variation potentially creating confusion among passengers. The frequencies of buses travelling along these routes may, however, be altered for different times in the day in order to accommodate fluctuating demand. Walk arcs were discussed in the context of the UTFSP in §5.3. A walk factor of three was assigned in §7.6 to these walk arcs in the context of a small network (Mandl's Swiss network) [115]. This assumption was required since no walking times are available for Mandl's benchmark instance, and neither are geographical data whereby these times could be determined. For the Matie Bus case study, however, the geographical locations of the bus stops may be used in conjunction with data retrieved from Google Maps [66] to determine the walking times between these stops. The

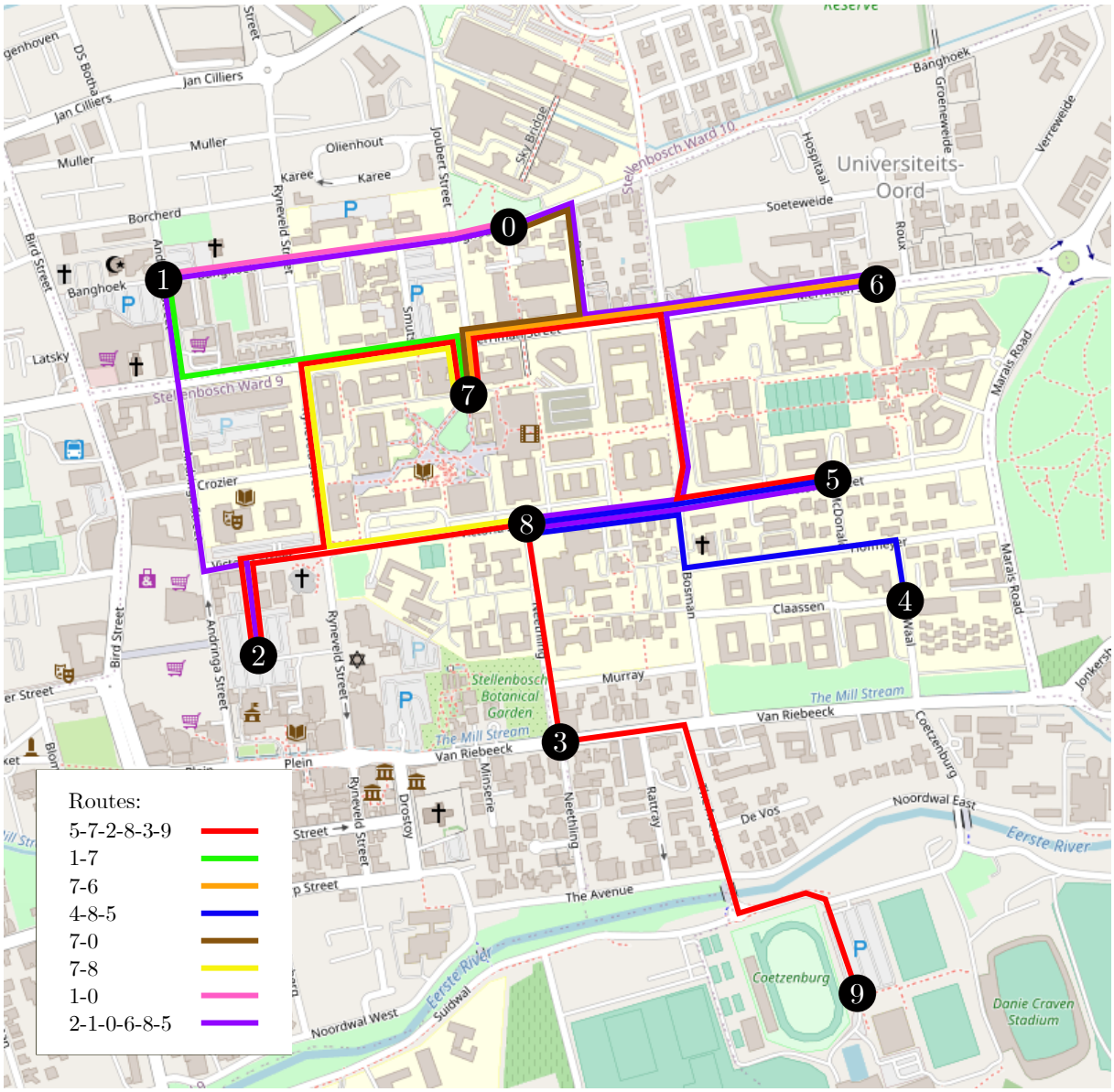


FIGURE 8.8: *The route set corresponding to Solution C in Figure 8.7 for the Matie Bus case study, superimposed on a map of Stellenbosch.*

walking times matrix thus generated are shown in Figure 8.9. The entry in row i and column j of this matrix may be used to determine the cost of the walk arc representing travelling from vertex v_i to vertex v_j for all the walk arcs of the UTFSP instance. The NSGA II of §6.2.3 may then be executed as normal to solve the resulting UTFSP instance.

	A	B	C	D	E	F	G	H	I	J	K
1		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	**0**	0	6	12	12	14	10	10	4	8	19
3	**1**	6	0	7	15	20	17	15	8	11	24
4	**2**	12	7	0	8	15	12	16	8	7	17
5	**3**	12	15	8	0	8	10	14	8	4	9
6	**4**	14	20	15	8	0	3	11	11	8	9
7	**5**	10	17	12	10	3	0	9	8	5	12
8	**6**	10	15	16	14	11	9	0	8	10	18
9	**7**	4	8	8	8	11	8	8	0	4	17
10	**8**	8	11	7	4	8	5	10	4	0	13
11	**9**	19	24	17	9	9	12	18	17	13	0

FIGURE 8.9: *The walking times matrix for the Matie Bus case study (measured in minutes).*

The OD demand matrix for 12h00 was selected for solving the UTFSP as it exhibited the largest total demand for any of the hours of the day, and assigning buses to accommodate the maximum demand would be sufficient for the lesser demand as well. The demand patterns are, however, similar for each of the different times represented in Figure 8.6. The attainment front achieved by the NSGA II of §6.2.3 can be seen in Figure 8.10 for the 12h00 UTFSP instance, corresponding to the demand matrix in Figure 8.6(b). For this instance, the set of possible frequencies was defined as $\theta = \left\{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\right\}$, similar to that employed by Martínez *et al.* [117]. A solution from the non-dominated front returned by the NSGA II of §6.2.3, taking into consideration walk arcs, was chosen instead of the non-dominated front corresponding to the situation where walk arcs are not taken into consideration. This choice was made due to the increased complexities accommodated by the NSGA II of §6.2.3 when considering walk arcs, thereby hopefully facilitating better decisions based on more accurate information. It was decided for the small network of the case study, and the associated short travel times of the buses, that it would make sense to select the frequency set corresponding to the best AETT objective function evaluation. This solution is denoted by the label D in Figure 8.10, and the objective function evaluation and the corresponding frequencies to which buses were assigned to routes according to this solution are indicated in Table 8.3.

TABLE 8.3: *Objective function evaluation and associated frequencies for Solution D in Figure 8.10, returned by the NSGA II when solving the UTFSP instance in Figure 8.8.*

Solution	AETT (no walk arcs)	AETT (with walk arcs)	TBR	Frequencies for routes
Solution D	6.185 min	5.370 min	14.333 buses	$\left\{\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{30}, \frac{1}{30}, \frac{1}{5}, \frac{1}{5}\right\}$

When frequencies are set for routes, buses have to be assigned to each route in order at least to achieve the required frequencies. Therefore, the number of buses that are required are rounded up to the nearest integer, thereby ensuring that the minimum required frequency is indeed met. For each route in the route set of Figure 8.8, the route time, the number of buses assigned to the route, the bus frequency assigned to it and the headway operating along the route are shown in Table 8.4.

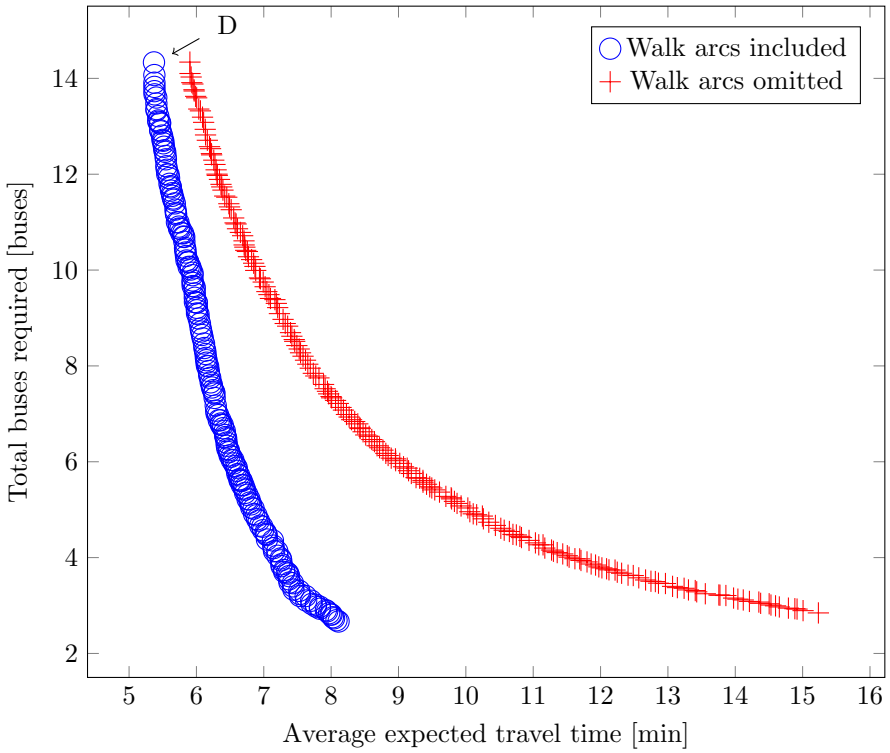


FIGURE 8.10: *The non-dominated attainment front obtained by the NSGA II of §6.2.3 when solving the UTFSP instance for the Matie Bus case study when adopting the route set in Figure 8.7 and the input data of §8.2.*

TABLE 8.4: *The associated route traversal times, bus assignments, frequencies achieved and operating headways for the routes in Solution D of Figure 8.10, returned by the NSGA II when solving the UTFSP instance in Figure 8.8.*

Route label	Route	Route time [min]	Buses assigned [buses]	Frequency [buses/min]	Headway [min/bus]
A	$\langle 5, 7, 2, 8, 3, 9 \rangle$	12	5	0.208	4.8
B	$\langle 1, 7 \rangle$	3	2	0.333	3.0
C	$\langle 7, 6 \rangle$	2	1	0.25	4.0
D	$\langle 4, 8, 5 \rangle$	5	2	0.2	5.0
E	$\langle 7, 0 \rangle$	2	1	0.25	4.0
F	$\langle 7, 8 \rangle$	3	1	0.167	6.0
G	$\langle 1, 0 \rangle$	2	1	0.25	4.0
H	$\langle 2, 1, 0, 6, 8, 5 \rangle$	11	5	0.227	4.4

When an integer number of buses are assigned to bus routes, however, the actual frequency of buses operating along each route will differ from those returned by the NSGA II of §6.2.3, as only the discretised values of θ were used in the latter case. Therefore, the AETT was recomputed after the actual number of buses had been assigned to each route (as depicted in Table 8.4) so as to achieve a more accurate AETT estimate, as shown in Table 8.5. It should be noted that the AETT measured when walk arcs are included and the AETT measured when walk arcs are omitted only differ by a few seconds — this is because the frequencies are set high enough that passengers would prefer making use of bus transport as opposed to walking. A total of 18 buses are required for this frequency configuration.

TABLE 8.5: *Objective function re-evaluation for the solution in Table 8.4 to the UTFSP instance in Figure 8.8.*

Solution	AETT (no walk arcs)	AETT (with walk arcs)	TBR
Implementation of Table 8.4	4.443 min	4.288 min	18 buses

A more in-depth analysis was subsequently conducted, based on the h-values returned by the optimal strategies algorithm (Algorithm 6.14) which represent the volumetric passenger flows over each arc for a specific destination vertex r. These h-values may be aggregated for each destination in the transit system, thus obtaining the total volume for each arc in the transit network. Each arc is shown in Table 8.6, together with its non-zero passenger volumes, its associated travel time cost and the actual bus frequency operating along it.

TABLE 8.6: *The associated route traversal time, frequency and volume of passengers for each non-zero volume arc v_i-v_j corresponding to the solution in Table 8.4 for the UTFSP instance in Figure 8.8.*

v_i	v_j	Cost [min]	Frequency [buses/min]	Volume [passengers/hour]
0	7	4.0	∞	270.0
0	E0	0.1	0.5	4.429
0	G0	0.1	0.5	61.179
0	H0	0.1	0.455	363.75
1	B1	0.1	0.667	397.5
1	G1	0.1	0.5	27.5
1	H1	0.1	0.455	229.0
2	A2	0.1	0.417	1 200.5
2	H2	0.1	0.455	193.0
3	A3	0.1	0.417	393.0
4	9	9.0	∞	10.0
4	D4	0.1	0.4	264.0
5	A5	0.1	0.417	226.667
5	D5	0.1	0.4	308.667
5	H5	0.1	0.455	1 079.667
6	C6	0.1	0.5	341.0
6	H6	0.1	0.455	1 228.0
7	0	4.0	∞	232.0
7	8	4.0	∞	1 585.0
7	A7	0.1	0.417	1 362.929
7	B7	0.1	0.667	362.143

TABLE 8.6 (continued): *The associated route traversal time, frequency and volume of passengers for each non-zero volume arc v_i-v_j corresponding to the solution in Table 8.4 for the UTFSP instance in Figure 8.8.*

v_i	v_j	Cost [min]	Frequency [buses/min]	Volume [passengers/hour]
7	C7	0.1	0.5	448.0
7	E7	0.1	0.5	60.357
8	7	4.0	∞	1 152.0
8	A8	0.1	0.417	595.833
8	D8	0.1	0.4	327.0
8	H8	0.1	0.455	283.5
9	4	9.0	∞	4.0
9	A9	0.1	0.417	2.0
A5	5	0.1	∞	281.929
A5	A7	3.0	∞	226.667
A7	7	0.1	∞	1 185.5
A7	A5	3.0	∞	281.929
A7	A2	3.0	∞	1 088.667
A2	2	0.1	∞	1 371.5
A2	A7	3.0	∞	966.5
A2	A8	2.0	∞	234.0
A8	8	0.1	∞	430.0
A8	A2	2.0	∞	282.833
A8	A3	1.0	∞	509.0
A3	3	0.1	∞	508.0
A3	A8	1.0	∞	392.0
A3	A9	3.0	∞	4.0
A9	9	0.1	∞	4.0
A9	A3	3.0	∞	2.0
B1	1	0.1	∞	362.143
B1	B7	3.0	∞	397.5
B7	7	0.1	∞	397.5
B7	B1	3.0	∞	362.143
C7	7	0.1	∞	341.0
C7	C6	2.0	∞	448.0
C6	6	0.1	∞	448.0
C6	C7	2.0	∞	341.0
D4	4	0.1	∞	289.0
D4	D8	3.0	∞	264.0
D8	8	0.1	∞	328.667
D8	D4	3.0	∞	289.0
D8	D5	2.0	∞	282.0
D5	5	0.1	∞	282.0
D5	D8	2.0	∞	308.667
E7	7	0.1	∞	4.429
E7	E0	2.0	∞	60.357
E0	0	0.1	∞	60.357
E0	E7	2.0	∞	4.429
G1	1	0.1	∞	61.179

TABLE 8.6 (continued): *The associated route traversal time, frequency and volume of passengers for each non-zero volume arc v_i-v_j corresponding to the solution in Table 8.4 for the UTFSP instance in Figure 8.8.*

v_i	v_j	Cost [min]	Frequency [buses/min]	Volume [passengers/hour]
G1	G0	2.0	∞	27.5
G0	0	0.1	∞	27.5
G0	G1	2.0	∞	61.179
H2	2	0.1	∞	185.0
H2	H1	3.0	∞	193.0
H1	1	0.1	∞	270.679
H1	H2	3.0	∞	185.0
H1	H0	2.0	∞	53.5
H0	0	0.1	∞	294.5
H0	H1	2.0	∞	87.179
H0	H6	2.0	∞	322.071
H6	6	0.1	∞	1 283.0
H6	H0	2.0	∞	286.5
H6	H8	2.0	∞	997.071
H8	8	0.1	∞	252.667
H8	H6	2.0	∞	1 016.5
H8	H5	2.0	∞	1 091.071
H5	5	0.1	∞	1 091.071
H5	H8	2.0	∞	1 079.667

When considering the maximum passenger volume flow along the routes, labelled A to H, as shown in Table 8.7, it is interesting to note that when the frequencies were initially set, routes E and F were assigned the lowest discretised frequency in the set θ, namely $\frac{1}{30}$, while all the other routes were assigned the highest discretised frequency in θ, namely $\frac{1}{5}$. It is natural to expect when the minimum AETT is computed in the context of the UTFSP, that all the frequencies should be set to the highest possible value. In this case, however, the elegance and power of multi-objective optimisation are elucidated, in that the other objective at play, the TBR, is also minimised, albeit without loss in quality of the AETT objective. Table 8.7 reveals that these specific routes, E and F, experience maximum passenger flows of 60.357 and 0, respectively, indicating that very few passengers, in fact, make use of route E, and no passengers at all make use of route F.

Table 8.8 contains the non-zero volumes of the walk arcs in the transit network, indicating that passengers prefer to walk between vertices v_0 and v_7, and between vertices v_7 and v_8. These are exactly the bus stop vertices covered by routes E and F. Although the frequencies may have been set at their maximum values by the NSGA II of §6.2.3, it became clear that the additional buses and increased frequency will decrease a passenger's travel time, and so the frequencies were set low by the algorithm along these routes so as to avoid unnecessarily wasting additional buses. Although experiencing a small maximum volume, as shown in Table 8.7, route G serves demand flowing between the vertices of the bus route, instead of passengers choosing to walk. Route G was therefore not removed like routes E and F, but a decision maker may choose to discard this route if it is not deemed profitable enough.

Due to the rounding up of bus assignments, however, one bus was assigned to each of the routes E and F. It is anticipated that the decision maker may not consider it beneficial to assign any

buses to these routes, as they would serve little, if any, demand. With the removal of routes E and F, a final recommended route set, together with its accompanying bus assignments, is presented in Figure 8.11 and Table 8.9, respectively, and the performance of this solution is summarised in Table 8.10. The final solution performs satisfactorily with respect to servicing passengers with an AETT of under 10 minutes, (in fact, even below 5 minutes), and this is achieved by assigning 16 buses to 6 different routes.

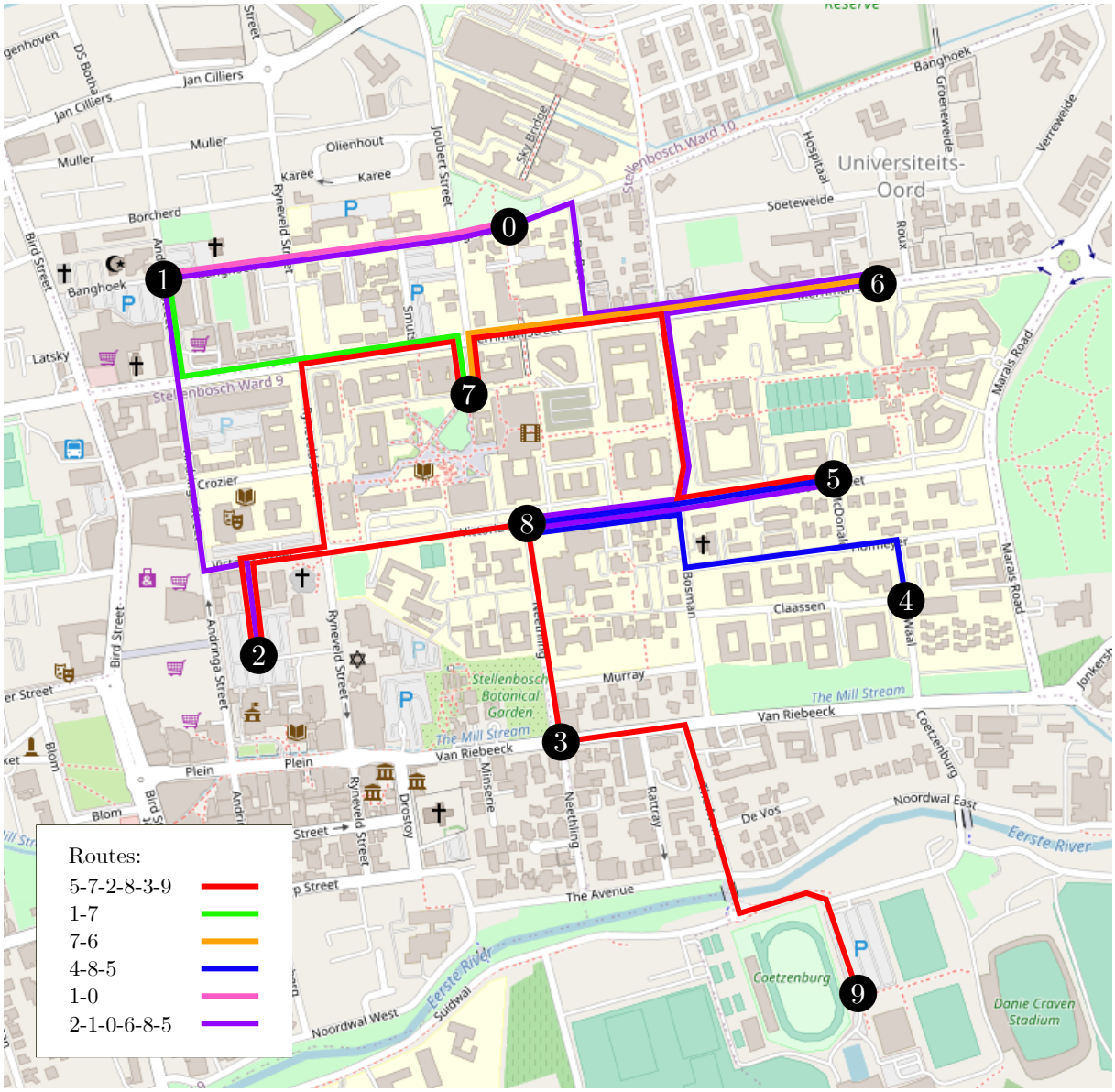


FIGURE 8.11: *The final set of routes for the Matie Bus case study, superimposed on a map of Stellenbosch, and representing the recommended set of routes, with the number of buses assigned to each route in Table 8.9.*

Upon considering the routes actually operated by the Matie Bus day shuttle service, depicted in Figure 8.1, a comparison may be made. An initial comment on the Matie Bus routes is made in respect of its connectivity. The Matie Bus day shuttle service routes are not all connected, and therefore passengers would need to walk in order to reach some areas of campus. This is understandable as the Matie Bus service aims mainly to serve students travelling between the Faculty of Food Science and the central campus (the bus stop on Joubert street), as well

TABLE 8.7: *The maximum volumetric passenger usage for the routes, labelled A to H, corresponding to the solution in Table 8.4.*

Route label	Route	Maximum volume
A	$\langle 5, 7, 2, 8, 3, 9\rangle$	1 371.5
B	$\langle 1, 7\rangle$	397.5
C	$\langle 7, 6\rangle$	448.0
D	$\langle 4, 8, 5\rangle$	328.667
E	$\langle 7, 0\rangle$	60.357
F	$\langle 7, 8\rangle$	0.0
G	$\langle 1, 0\rangle$	61.179
H	$\langle 2, 1, 0, 6, 8, 5\rangle$	1 283.0

TABLE 8.8: *The total number of passengers making use of walk arc v_i-v_j in the solution of Table 8.4.*

v_i	v_j	Volume
0	7	270
4	9	10
7	0	232
7	8	1 585
8	7	1 152
9	4	4

TABLE 8.9: *The associated route traversal time, bus assignment, frequency achieved and operating headway along each route in the final solution when solving the UTFSP instance in Figure 8.8.*

Route label	Route	Route time [min]	Buses assigned [buses]	Frequency [buses/min]	Headway [min/bus]
A	$\langle 5, 7, 2, 8, 3, 9\rangle$	12	5	0.208	4.8
B	$\langle 1, 7\rangle$	3	2	0.333	3.0
C	$\langle 7, 6\rangle$	2	1	0.25	4.0
D	$\langle 4, 8, 5\rangle$	5	2	0.2	5.0
G	$\langle 1, 0\rangle$	2	1	0.25	4.0
H	$\langle 2, 1, 0, 6, 8, 5\rangle$	11	5	0.227	4.4

TABLE 8.10: *The objective function evaluation of the final solution in Table 8.9 for the UTFSP instance in Figure 8.8.*

Solution	AETT (no walk arcs)	AETT (with walk arcs)	TBR
Implementation of Table 8.9	4.830 min	4.321 min	16 buses

as between the Coetzenburg Sports Centre and the central campus (the Endler Hall of the Conservatory of Music) [155]. This, however, serves only those students who travel between the aforementioned bus stops, and does not establish a high-quality bus network, as defined in §4.5.1, where the entire transit demand is satisfied, as well as including mostly direct routes between origins and destinations [31].

To quantify this statement, the Matie Bus day shuttle route set was evaluated within the context of the UTRP and UTFSP performance measures adopted for evaluating quality. In respect of the UTRP, the TRT and transfer percentages were evaluated. Evaluation of the ATT was omitted due to the network not being connected. In the context of the UTFSP, the AETT could be evaluated, due to the incorporation of walk arcs, as could the TBR. Furthermore, an analysis was conducted with respect to the arcs usage and volumetric flows along the arcs. Due to the Matie Bus case study culminating in the solution in Table 8.9, which differs from the routes actually operated by the Matie Bus day shuttle service (indicated in Figure 8.1), a number of assumptions had to be made in order to facilitate a comparative approach. First, the bus stops are not located at the same places and, therefore, the demand assignment and the route time had to be adapted in the case of the routes operated by the Matie Bus service so that it can be evaluated in terms of the objective functions of the AETT, TRT and TBR.

Therefore, vertices v_0 and v_1 in the route set of the Matie Bus day shuttle were combined into one bus stop and mapped to vertex v_0 of the Matie Bus case study, due to their close proximity. Furthermore, vertices v_2, v_3, v_4 and v_5 in the route set of the Matie Bus day shuttle were mapped to vertices v_7, v_8, v_3 and v_9 of the case study, respectively. The mapping once again allocated the bus stops of the day shuttle to the closest bus stops of the case study. Naturally, the demand for the respective vertices were also adopted according to the case study bus stops. Moreover, due to the unavailability of the actual operating frequencies of the buses along each route, a conservative approach was adopted by setting the frequencies of all the routes in the Matie Bus day shuttle to the highest frequency of the final solution in Table 8.9, namely $\frac{1}{3}$.

The evaluation of the Matie Bus day shuttle performance objectives is summarised in Table 8.11. For the UTRP and UTFSP Matie Bus instances, the AETT of 8.007 min was considerably worse (almost double) than the AETT of 4.288 min for the solution of Table 8.9. With respect to the operator cost, the Matie Bus day shuttle service performs better than the solution of Table 8.9 because less expenses are incurred, as may be inferred from the 7 min TRT and the 6 TBR in the former case, compared with the 40 min TRT and 16 TBR of the Matie Bus case study final solution. With respect to transfer percentages, the Matie Bus day shuttle service does not perform satisfactorily in meeting overall passenger demand, as indicated by the 97.11% unmet demand value, *i.e.* not satisfying demand with fewer than three transfers.

TABLE 8.11: *Objective function values and transfer percentage evaluation for the Matie Bus day shuttle route set corresponding to the routes in Figure 8.1, where a frequency of $\frac{1}{3}$ is assigned to each route.*

Solution	AETT	TRT	TBR	d_0	d_1	d_2	d_u
Matie Bus day shuttle	8.007 min	7 min	6 buses	2.87	0.02	0.0	97.11

When considering the actual routes, the maximum passenger flows for each route can be found in Table 8.12. These values are rather small in comparison with the maximum passenger flows of the Matie Bus case study final solution, shown in Table 8.7. The small maximum passenger flow of route C in Table 8.12 can be explained by the bus stop mapping of vertex v_4 of the day shuttle to vertex v_3 of the Matie Bus case study, and also due to limited demand data gathered by Klink [98] for vertex v_4 of the day shuttle service, as indicated in Figure 8.1.

TABLE 8.12: *The associated route traversal times, bus assignments, frequencies achieved, operating headways and maximum volumetric passenger usage for the Matie Bus day shuttle service routes of Figure 8.1.*

Route label	Route	Route time [min]	Buses assigned [buses]	Frequency [buses/min]	Headway [min/bus]	Maximum volume
A	$\langle 0, 7 \rangle$	2	2	0.5	2.0	281
B	$\langle 8, 3 \rangle$	1	1	0.5	2.0	416
C	$\langle 8, 9 \rangle$	4	3	0.375	2.667	4

8.4 Chapter summary

The solution approaches described in Chapter 6 were applied in this chapter to a case study involving the Matie Bus service of Stellenbosch University. The aim in the case study was to propose an alternative set of bus routes to the one currently in use that is able to expand service coverage to more students during the main operating hours of the university. Bus stops were recommended in §8.1, after which the input data utilised in the case study were described in §8.2. A discussion followed in §8.3 on the results returned by the algorithms when solving the case study instance of the UTRP, and a corresponding instance of the UTFSP chosen from the output route sets for the UTRP. This culminated in a high-quality alternative route set recommendation for the bus service, along with the number of buses that should be assigned to each route in order to achieve its stated aims. Thereafter, the Matie Bus day shuttle service routes were evaluated and compared with the stated recommendation, and found to perform much better with respect to the passengers' objective. If implemented, the recommended route set is expected to lead to greater satisfaction experienced by the students, as well as higher revenues due to increased transportation supply and passenger usage.

Part IV

Conclusion

CHAPTER 9

Dissertation summary and appraisal

Contents

This penultimate chapter serves as conclusion to the work documented in this dissertation. The dissertation content is summarised in §9.1, after which the contributions of the dissertation are critically assessed in §9.2.

9.1 Dissertation summary

Apart from the stand-alone Chapter 1 and the current Part IV, the other seven chapters of this dissertation were partitioned into three parts, titled *Literature study* (Chapters 2–4), *Mathematical modelling* (Chapters 5–7) and *Case study* (Chapter 8).

Chapter 1 served the purpose of providing the reader with the required background on the topic of the dissertation, an outline of the objectives pursued in the dissertation, a delimitation of the dissertation scope, and a description of the research methodology adopted during the execution of the work documented in this dissertation.

Part I was a literature review on topics relevant to the focus of this dissertation, and provided the reader with the necessary background and insight required to understand concepts employed in the subsequent Parts II and III. Chapter 2 was the first chapter in Part I, contained detailed descriptions of a number of mathematical prerequisites central to the material presented in the remainder of the dissertation and was included in the spirit of self-containment. The aim of the chapter was to facilitate a development on the part of the reader of an understanding of the mathematical concepts employed in later chapters. The main topics covered in Chapter 2 were an introduction to basic concepts in graph theory, brief discussions on Dijkstra's [45], Floyd's [59], and Yen's [179] well-known algorithms for finding shortest paths in weighted graphs, and an exposition on basic concepts in multi-objective optimisation (including a number of fundamental relations between candidate solutions to multi-objective optimisation problems). Methods for isolating non-dominated subsets from a given candidate solution set were also explored, as were the notions of non-dominated sorting of solution sets and multi-objective solution set quality evaluation.

Chapter 3 provided the reader with insight into popular approaches in the literature towards solving combinatorial optimisation problems, both in a single-objective optimisation context and in a multi-objective optimisation context, in fulfilment of Objective I(a) of §1.3. This material was included so that an uninitiated reader might gain a sufficient understanding of the nature and complexity associated with various types of combinatorial optimisation problems, and appropriate solution techniques that are applicable to various different cases. First, exact solution approaches were considered, then heuristics, metaheuristics and finally hyperheuristics. Metaheuristics were classified into two main classes, namely trajectory-based metaheuristics and population-based metaheuristics. One popular metaheuristic was examined in detail in each of these classes, both in a single-objective context and in a multi-objective context. In the class of trajectory-based metaheuristics, the method of SA was considered, together with its DBMOSA variant in the context of multi-objective optimisation, while in the class of population-based metaheuristics, the GA was considered, together with its NSGA II variant in the multi-objective optimisation context. These two solution approaches were elaborated upon so that the reader might gain an understanding of their fundamental working when applied to various optimisation problem instances encountered later in this dissertation. A short discussion on hyperheuristics was presented towards the end of the chapter, providing the reader with the required background on the incorporation of heuristics and metaheuristics into hyperheuristic optimisation approaches.

Chapter 4 was devoted to a review of the literature on transit network planning, in fulfilment of Objectives I(b)–(e) of §1.3. The topics reviewed were UTNDPs in general, the complexity of the UTNDP in particular, the phases of the transit planning process, and classifications of different types of UTNDPs. Thereafter, the three specific sub-problems of the UTNDP considered in this dissertation, called the UTRP, the UTFSP and the UTRFSP, were considered in more detail. Key characteristics of the UTRP were reviewed, namely its objectives, decision variables, network structure, demand patterns, demand characteristics and constraints. Additionally, appropriate solution methodologies adopted in the literature for solving instances of the UTRP were discussed, namely exhaustive searches, mathematical programming techniques, heuristics, trajectory-based metaheuristics, population-based metaheuristics and hybrid approaches. A discussion was also included on the benchmark problem instances available in the literature, along with the best results reported for each instance. Next, the objectives, decision variables, constraints, and the notion of transit assignment were described in the context of the UTFSP, closing with a discussion on the various solution approaches adopted in the literature toward solving instances of the UTFSP. The last sub-problem considered, the UTRFSP, involves the simultaneous design of bus routes and the setting of frequencies at which buses should operate along these routes, and therefore shares the key characteristics of the UTRP and the UTFSP. The emphasis of the discussion on the UTRFSP was on solution approaches that have been adopted to solve instances of the UTRFSP. The chapter closed with a short discussion on commercial software available in practice for transit modelling and design. None of these software alternatives supports the capability of designing an entire transit network anew.

Part II was the heart of the dissertation in which mathematical models were established for the UTRP and the UTFSP. The implementations of solution techniques for these models were discussed, and a validation of these techniques was carried out in fulfilment of Objectives II—IV of §1.3. Chapter 5 was devoted to the establishment of three mathematical models adopted in this dissertation for solving instances of the standard UTRP, the UTRP with incremental redesign, and the standard UTFSP, respectively. For the UTRP, a bi-objective minimisation model was established on a conceptual level in graph theoretic terms, in partial fulfilment of Objective II of §1.3. The model for the UTRP with incremental redesign closely followed that of the standard UTRP, with the exception that the operator cost objective function was replaced

with the minimisation of a novel DP objective function. Another bi-objective minimisation model was also documented in graph theoretic terms for the UTFSP, in final fulfilment of Objective II.

Chapter 6 contained detailed discussions on the approaches adopted in this dissertation towards solving the aforementioned models. Discussions were included on the implementations of algorithms by the author in the programming language Python for solving the UTRP and the UTFSP models of Chapter 5, in fulfilment of Objectives III and IV of §1.3.

Chapter 7 was devoted to the validation and verification of the aforementioned models and corresponding algorithmic solution implementations. This was achieved by comparing the solutions returned by the various algorithms with those documented in the literature for well-known benchmark problem instances, in fulfilment of Objective V of §1.3. The various algorithmic parameters were appropriately set during an extensive design analysis and a sensitivity analysis was conducted for each of these parameters in terms of the associated solution quality, as measured by HV.

Part III contained a single chapter, Chapter 8, and was devoted to a special case study involving the Matie Bus service of Stellenbosch University. A description was provided of how the model and solution implementations of Chapter 6 were applied to the specific UTRP and UTFSP instances pertaining to this case study, and the routes and bus frequencies generated by the algorithms as recommendations to the Matie Bus operators were described, in fulfilment of Objective VI of §1.3.

Objective VII of §1.3 is achieved in the remainder of this chapter, in the form of an appraisal of the dissertation contributions, whereas the next chapter stands in fulfilment of Objective VIII, containing a number of suggestions for potential future follow-up work.

9.2 Appraisal of dissertation contributions

This section contains a critical appraisal of the contributions of this dissertation, in fulfilment of Objective VII of §1.3.

Contribution I *A revised taxonomy of the UTNDP found in the literature.*

The literature pertaining to the UTNDP is today still an ongoing and active field of study [2, 82, 85]. Various methodologies exist for solving instances of the UTNDP, as no single best solution method or model representation has been established [85]. The vast research body related to the UTNDP contains a variety of terminologies used to describe its concepts, and some authors have aimed to unify these terminologies by establishing a standard lexicon [68]. The literature review of Chapter 4 was aimed at synthesising the numerous terminologies and problem components encountered when dealing with the UTNDP, so that an uninitiated reader may find the literature easier navigable.

More specifically, the literature review opened with a description of transit network design problems in general, providing the reader with an overall idea of what such problem instances typically entail, the various trade-offs at play, and a general mathematical model that is applicable to instances of the UTNDP. Thereafter, various sources contributing to the complexity of the UTNDP were described, providing insight into why the UTNDP is classified as NP-hard. Next, typical stages in the overall transit planning process were elaborated upon, so that the reader might develop a context for the UTNDP, and understand the various independent inputs required, the planning activities represented and the outputs that are generated by the different phases of the UTNDP. The exposition was aimed at developing a perspective on how all

these phases fit together and the relevant constituent considerations required during each phase. Finally, it was described how the UTNDP may be categorised into the popular sub-problems that exist in the literature, forming an overarching modelling perspective of the various components that should be considered when attempting to solve instances of these sub-problems. An emphasis was placed on the UTRP, the UTFSP and the UTRFSP, as these are the specific problems relevant to this dissertation. A review also followed of solution methodologies that have been adopted in the literature for each of the sub-problems, categorised according to broad methodological classes.

Contribution II *The proposal of eight novel LLHs for incorporation into the solution approach for solving instances of the UTRP.*

Various LLH operators have been proposed in the literature for use when solving instances of the UTRP [2, 50, 89, 115], as summarised in Table 6.3 and illustrated graphically in Figure 6.6. None of these LLH operators, except for one, takes into consideration the underlying demand or cost components inherent to the UTRP.

This gap in the literature was bridged by introducing eight novel LLH operators in §6.1.5 of this dissertation, summarised in Table 6.3, and illustrated graphically in Figure 6.8. The first, the remove low demand terminal (n) operator, involves removing a terminal vertex from a route set that serves the lowest demand. Secondly, the remove large cost terminal (o) operator is based on removing a terminal vertex from a route set that has the largest edge weight associated with it.

Next, the cost-based grow (p) and cost-based trim (q) operators were introduced, described in detail in §6.1.3. These two operators may be seen as counterparts of one another, and work on the principle of stochastically selecting a grow or trim to be applied to a route set, based on demand per cost ratios or on cost per demand ratios, respectively. The remaining four operators incorporate the cost-based trim and cost-based grow operators in their working.

The random cost-based grow/trim (r)/(s) operator applies the cost-based grow/trim operator to one route a random number of times between one and the maximum number of vertices that may be added/deleted to/from the route without loosing feasibility. The random all cost-based grow/trim (t)/(u) operator applies the random cost-based grow/trim operator to all the routes in a route set.

After testing these operators empirically in Chapter 7, it was concluded that the cost-based grow and the cost-based trim operators contribute most significantly during the search procedures of the DBMOSA algorithm and the NSGA II when solving instances of the UTRP, aiding these algorithms in finding high-quality solutions.

Contribution III *The proposal of three novel perturbation selection management strategies for trajectory-based searches aimed at solving instances of the UTRP.*

Due to the complexity of the UTRP, various researchers [2, 89, 150] have adopted numerous LLH operators when solving instances of the UTRP in recent years, and have reported considerable success in obtaining high-quality results. When so many LLHs are incorporated into a search approach, an effective LLH management strategy is required to weigh the selection importance of each LLH appropriately during the search. Three novel perturbation management strategies for trajectory-based searches were proposed in §6.1.6 in order to manage the aforementioned LLH operators effectively, namely a stochastic roulette-wheel perturbation strategy, an AMALGAM perturbation strategy, and an every n-th AMALGAM perturbation strategy.

These perturbation strategies were evaluated empirically in Chapter 7, and it was found that the stochastic roulette-wheel perturbation strategy outperformed the other two perturbation management strategies, as well as a static approach towards perturbation selection, in the context of solving instances of the UTRP in combination with a DBMOSA algorithm. This approach, therefore, holds considerable potential for future implementations of trajectory-based searches, as it has been proven successful in the context of a DBMOSA metaheuristic when solving instances of the UTRP.

Contribution IV *The proposal of a novel mutation selection management strategy for population-based searches aimed at solving instances of the UTRP.*

Based on the same reasoning as in the motivation behind Contribution III, a novel mutation selection management strategy was proposed to aid in the selection of LLHs within the context of population-based searches aimed at solving instances of the UTRP. The AMALGAM approach [167] was applied to manage mutations during a population-based search, instead of managing search substrategies within the context of evolutionary searches, as explained in §6.1.6. This approach was adopted as the mutation management strategy for LLHs in the context of an NSGA II implementation for solving instances of the UTRP and it was found, upon empirical experimentation reported in Chapter 7, that it is not clear whether this approach outperforms a static mutation management strategy for the NSGA II — further tests are required in this respect.

Contribution V *The proposal of five novel crossover operators for the UTRP.*

The application of a crossover operator is the main exploration mechanism in the NSGA II. The relative performances of the crossover operator proposed by Mumford [121] and five new crossover operators proposed in §6.1.8 of this dissertation were analysed in the context of the UTRP. These five crossover operators exhibit two main qualities. The first is the stochastic or deterministic nature of selecting routes, based on the proportions of potential insertions of vertices not present in the route set to the number of the total number of vertices not present in the route set. The second involves the replacement of subroutes found in offspring solutions after having performed crossover, as well as the method of choosing a replacement route. One option of choosing a replacement route involves randomly selecting a route from one of the parent solutions not yet contained in the offspring solution, and another option is choosing a replacement route randomly from the set of initial candidate solutions of K shortest paths generated by the KSP-MMV-ICRSGP. After comparing these five novel crossover operators with each other and with the crossover operator of Mumford in §7.3.1, it was found that the RSD crossover operator performed the best overall. This operator draws its inspiration from the deterministic nature of the crossover operator of Mumford [121] in combination with replacing subroutes with other routes from one of the parent solutions not present in the offspring. This novel contribution may aid future researchers in overcoming the obstacles associated with finding duplicates of subroutes that may emerge in offspring solutions as a result of applying the original crossover procedure proposed by Mumford. Having subroutes present in any solution to a UTRP instance will always be suboptimal as they do not contribute to the ATT objective function, but increase the value of the TRT objective function.

Contribution VI *The proposal of a novel ICRSGP and a route set enhancement procedure for the UTRP.*

Generating high-quality initial feasible solutions to instances of the UTRP may prove challenging, especially in the context of larger instances. A novel ICRSGP was proposed in this dissertation, as was an enhancement procedure for improving route set quality in terms of spread

across the attainable portion of the objective space of an UTRP instance, as elaborated upon in §6.1.2 and §6.1.3, respectively. Route generation by maximising missing vertices involved adopting the approach of Mumford's crossover procedure of 2013 [121] in which the concept of maximising the proportion of missing vertices added to the route set was employed, as described in pseudo-code form in Algorithm 6.4. This allows for consistently and quickly generating feasible initial route sets, even for large networks, as all vertices of the network are included each time. Some researchers have resorted to applying repair strategies in order to generate initial feasible solutions [2, 121], which is not required in this procedure. The initial set of solutions provided as input to Algorithm 6.4, of course, has an influence on the quality of routes eventually generated. Utilising the filtered K shortest paths generated by Algorithm 6.3 for this purpose proved capable of establishing high-quality initial solutions.

After having generated a set of initial feasible solutions to a UTRP instance by invoking the aforementioned procedure, the set of routes may be enhanced in terms of quality with a view to attain improved spread in objective space. A procedure for this type of enhancement was described in §6.1.3. The novel cost-based grow and cost-based trim operators were used as foundations to improve upon the route set quality in terms of the ATT objective and the TRT objective, respectively. The results returned by this enhancement procedure were validated against the results reported by other researchers in §7.1, and is was concluded that high-quality initial candidate route sets were indeed returned by this approach. This novel solution enhancement approach may hold benefit for other researchers aiming to solve instances of the UTRP, as this approach is intuitive, easy to implement, and consistently produces high-quality, feasible results for any UTRP instance size.

Contribution VII *The proposal of a novel DBMOSA solution implementation for the UTRP.*

Since the dawn of the new millennium, metaheuristics have become increasingly popular when solving instances of the UTRP. Various types of metaheuristics have been applied to this problem [85]. It is, however, only very recently that instances of the UTNDP have been solved in their multi-objective optimisation contexts (as is appropriate for this type of problem) — earlier solution methodologies mainly scalarised a number of objective functions [85, 133].

In this dissertation, a novel computer-based DBMOSA solution implementation for solving instances of the UTRP was successfully designed and implemented. This implementation is capable of finding high-quality bus routes for a public transport system, aimed at minimising the ATT of passengers and the TRT of the system simultaneously. The algorithm takes as input demand and distance data and produces as output sets of high-quality trade-off bus routes from which a decision maker can select one subjectively for implementation purposes. The algorithm was shown to deliver high-quality results in the context of well-known benchmark problem instances and a real-world case study, and can thus be accepted as a potentially useful algorithm for transit network optimisation. It was further applied to solve instances of the UTRP with incremental redesign in which the TRT objective was replaced by the DP objective function, thereby demonstrating the flexibility of this solution approach.

The DBMOSA algorithm was equipped with the stochastic roulette-wheel selection hyperheuristic of §6.1.6 for managing the selection of the perturbation operators presented as input to the algorithm. This is a novel methodological contribution to the DBMOSA solution framework, and was shown to outperform a static perturbation selection probability approach within the context of the UTRP.

Contribution VIII *The proposal of an NSGA II solution implementation for the UTRP.*

A computerised NSGA II solution implementation was proposed for the UTRP, based on the work of John [89]. The NSGA II of §6.1.8, however, applied ten mutation operators, instead of

eight as was the case in the original implementation of John [89]. Furthermore, a design analysis was documented in §7.3 for establishing suitable values for the parameters of the NSGA II, something that was not made public by John [89]. In the process, unexpected, yet appropriate, values for the crossover probability and mutation probability were established for well-known benchmark problem instances. The NSGA II implementation included a novel component introduced in this dissertation, namely the AMALGAM mutation management strategy of §6.1.6 for managing the selection of the ten mutations provided as input to the algorithm.

The NSGA II of §6.1.8 returned high-quality results, even dominating most of the results reported in the literature (including most of the solutions returned by John's NSGA II implementation [89]).

Contribution IX *An evaluation of combining a DBMOSA and an NSGA II solution implementation for the UTRP.*

Trajectory-based metaheuristics are typically well-suited for exploiting local-optima, whereas population-based metaheuristics are typically well-suited for exploring solution spaces. A combination of metaheuristics from each class may therefore hold additional benefit over applying only one algorithm. A combination of the DBMOSA algorithm of §6.1.7 and the NSGA II of §6.1.8 was explored in §7.4. The approach involved taking thirty equispaced solutions returned by an NSGA II run as starting solutions for a DBMOSA algorithmic run, and forming a final non-dominated set from the solutions returned by the DBMOSA runs.

A comparison between this combined method and separately executing the DBMOSA algorithm and the NSGA II revealed that the former method of combining the two algorithms returned the best results for the large benchmark instances considered in this dissertation. Marked improvements were observed in the mid portions of the approximate Pareto set in objective space. The combination therefore holds potential if a transit network operator is interested in these mid-range approximate Pareto efficient solutions, which are typically realistic in terms of practical implementation.

Contribution X *Improved results for five of the UTRP benchmark instances in the literature.*

The best results reported for five popular UTRP benchmark instances in the literature were summarised in §4.5.8, of which only the non-dominated solution sets of John [89] and Mumford [121] are publicly available. To the best of the author's knowledge, the non-dominated sets of John [89] were the best results available in the literature in terms of HV performance at the time of writing this dissertation. When the results returned by the DBMOSA algorithm of §6.1.7 and the NSGA II of §6.1.8 were compared with the best results reported in the literature, as summarised in Table 7.18, it became clear that the quality of the non-dominated solutions sets returned by the NSGA II of §6.1.8 outperformed the DBMOSA algorithm of §6.1.7, and also those results reported by John [89] and Mumford [121] in terms of the HV obtained by the non-dominated sets. The non-dominated sets returned by the algorithms of this dissertation are illustrated in objective space in Figure A.26 of Appendix A, plotted alongside the best results reported in the literature. Furthermore, after combining the DBMOSA algorithm with the NSGA II, improved results were observed for the Mumford2 and Mumford3 benchmark instances over those returned by the NSGA II of §6.1.8, as illustrated in Figure 7.16.

For the Mumford3 UTRP benchmark instance, the largest of the benchmark instances, an improved extremal solution in terms of the ATT objective was contributed in this dissertation, as indicated in Table 7.16. This solution achieved an ATT of 28.03 min and a TRT of 5 018 min, whereas the best reported solution in the literature achieved an ATT of 28.05 min and a TRT of 6 119 min, which is about a one sixth improvement in terms of TRT objective function values.

Even for the Mumford2 instance, the ATT of 25.31 min was very close to the best value in the literature of 25.19 min, reported by Ahmed *et al.* [2], but the corresponding TRT objective function value for the best solution of Ahmed *et al.* was 1 086 min more than that of the solution returned by the NSGA II of §6.1.8. This indicates that for larger UTRP instances, the NSGA II of §6.1.8 has the ability to beat state-of-the-art algorithms, improving significantly on especially the TRT objective.

Contribution XI *The proposal of a novel objective function and model for the UTRP with incremental redesign.*

As mentioned by John [89], it might often be more practical to reconfigure only partial routes within a transit network, instead of attempting a complete redesign, which is the approach adopted by almost all researchers. When an entire transit system redesign is pursued, this might negatively affect its customers due to familiar routes being disrupted, not to mention the possible confusion, dissatisfaction and potential loss of customers that may result. For the most part, *ad hoc* approaches have been applied to instances in which only a partial route configuration is required. There is, therefore, a lack of suitable approaches in the literature which may be adopted towards establishing a standardised solution methodology for solving this type of incremental design problem.

John [89] proposed a similarity measure between two route sets, expressed as a proportion. A new objective function was established in §5.2, called the DP, based on this similarity concept, in order to capture the *system disruption* anticipated when altering route sets, and it was also measured as a proportion. The current route set of a transit network may be taken as an anchor for similarity calculations pertaining to each new route set uncovered during an algorithmic search. As mentioned, the literature mainly addresses the issue of complete network redesign, where entirely new route sets, bearing little resemblance to the current operational route set, is proposed for the implementation by operators. The novel objective function may, however, be incorporated into the standard UTRP model of §5.1. By solving the resulting UTRP model with incremental redesign, in which the ATT and DP objectives are minimised simultaneously, solutions may be produced which resemble trade-offs between route similarity with a reference or current route set, and the ATT of passengers.

On the one extreme, the reference set itself would achieve the minimum DP of 0%, although other route sets may also achieve a DP of 0%, but performing differently in terms of the ATT objective. These alternative route set configurations are especially significant, as they resemble routes that, while theoretically containing the same edges as the reference route set, are nevertheless able to achieve better ATT. This means that the operator is able to provide a potentially better service to passengers at the cost of not disrupting the transit network too much, which would be highly beneficial to both operators and passengers.

On the other extreme of the results obtained when solving the UTRP model with incremental redesign, a route set achieving the lowest ATT may be found, with its corresponding DP objective function value typically rather high and the solutions in-between representing route sets with varying degrees of similarity to the reference set. One could even view the results returned as a road map of potential subsequent steps of improvements that may be introduced within the transit system by incrementally changing the configuration over time until a final "best" solution in terms of ATT is reached, if that were to be the end goal. The other perspective that may be taken is that the operator receives a set of trade-off route sets from which (s)he is able to choose a route set with an improved ATT along with a level of disruption (s)he is willing to tolerate, represented by the value of the DP objective function.

The UTRP model with incremental redesign was solved for five new benchmark instances introduced in this dissertation, based on the five popular benchmark instances mentioned in §4.5.8. High-quality results were obtained. The DBMOSA algorithm of §6.1.7 and the NSGA II of §6.1.8, for instance, both returned almost exactly the same solutions for the Mandl6 benchmark instance. This is indicative of the fact that, with high confidence, local optima were found for this instance. Furthermore, as the benchmark problem sizes increased, the quality of the results started to differ for the two different solutions implementations, with the NSGA II performing better in the region of small DP objective function values along the approximate Pareto set, and the DBMOSA algorithm performing better in the region of small ATT objective function values. The latter algorithm even returned a solution with an ATT of 27.8477 min for the Mumford3 instance, with a corresponding TRT of 6 091 min, thus outperforming the best ATT-extremal solution in the literature.

This novel contribution may be one of the most significant practical contributions of this dissertation, in the sense that the solutions returned when solving UTRP instances following the incremental redesign approach may be more easily implemented in the real world, instead of only remaining theoretical constructs. This approach would lead to decreased disruption for passengers using a transit system when route sets are chosen for implementation by the operator that achieve small values for the DP objective function.

Contribution XII *The proposal of a novel UTFSP model together with an NSGA II solution implementation for this model.*

When modelling instances of the UTFSP, the transit assignment of passengers is critical when evaluating the passengers' objective function. Transit assignment models are often oversimplified so that they can be solved easily in a personal computer environment. Such oversimplified underlying modelling assumptions may, however, lead to models that are not practical or representative of real-world situations.

It is, for instance, common not to model potential walk options that passengers might have in a transit system, as these are not always relevant when large transit networks are considered. When smaller transit networks are considered, on the other hand, the option of walking from an origin to a destination may be a viable option for a passenger, especially when an insufficient number of buses are allocated to routes. Spiess and Florian [151] indicated how walk arcs may be included in an optimal strategies transit assignment model, but to the best of the author's knowledge, this concept has not yet been applied in the context of the UTFSP.

In order to address this shortcoming, walk arcs were included in the modelling approach adopted for the UTFSP in §5.3, and a term *walk factor* was introduced for networks in which the real walk times cannot be determined due to unavailable data. This walk factor was multiplied by the cost of a bus traversing an arc, measured in units of time, and the product was subsequently taken as the cost of a passenger choosing to walk along that arc instead of making use of a bus. For validation purposes, the walk factor was set to 100, rendering the walk arcs too expensive for passengers to adopt, and therefore forcing the modelling approach to exclude the option of walking. When the walk factor was set to 3, however, resembling a small transit network as in the case of some of the benchmark problem instances utilised in Chapter 7, walking was found to become a viable option for some passengers. If sufficient number of buses are allocated to routes, passengers will most likely opt to make use of buses. If too few buses are allocated to routes, on the other hand, and waiting times become excessively large, passengers might well prefer to walk to their destinations instead.

In §6.2.3, an NSGA II solution implementation was adopted to solve instances of the novel UTFSP. This NSGA II implementation is flexible in the sense that different OD demand matri-

ces, route network structures, and route sets may be presented to the algorithm by a decision maker in order to establish approximate UTFSP solutions for different problem instances. A decision maker may subjectively choose a bus frequency set from among the approximate Pareto set of solutions returned, based on a desirable trade-off between passenger cost and operator cost.

The bi-objective modelling approach adopted in §5.3 took into account two very realistic transport bus route frequency setting design criteria, namely the AETT and the TBR. Moreover, the NSGA II solution implementation described in §6.2.3 was a novel contribution of this dissertation in the context of the UTFSP. Models for these types of problems usually omit walk arcs — a consideration which becomes increasingly important as the scale of the bus transit system decreases. An article on this contribution has been accepted for publication [80].

Contribution XIII *Application of the UTRP and UTFSP models to a real-world case study for designing a transit network in Stellenbosch.*

The practical applicability of the UTRP and UTFSP models, as well as the corresponding solution implementations of Chapters 5 and 6, respectively, was validated in Part III of this dissertation (Chapter 8), in respect of real-world case study data. The practical working of the algorithms of Chapter 6, together with their required input data, were showcased by systematically following all the steps required by a decision maker in order to facilitate the route design and frequency setting components of transit network design. The context of the case study was the Stellenbosch University Matie Bus day shuttle service, mainly providing transport to students attending lectures or parking at the periphery of the university campus, towards and back from the central campus. The data considered in the case study captured the movement of students over the campus during the main operating hours of the university.

First, the transit route design stage was completed by gathering the OD demand data of student movement over the campus, and relevant road network data in the form of a weight matrix. Both the DBMOSA of §6.1.7 and the NSGA II of §6.1.8 were applied to the resulting UTRP instance established by the input data. High-quality bus route sets were returned by these algorithms, and one was subjectively selected for recommendation purposes, based on a minimum ATT required assumption (a decision maker may choose an ATT value deemed to be acceptable).

This route set was then presented as input to the NSGA II of §6.2.3, along with the same aforementioned OD and demand matrices, in order to establish an UTFSP instance. A walk cost matrix was additionally presented as input to the algorithm in order to facilitate the incorporation of walk arcs. A set of high-quality frequencies were returned by the algorithm. A further analysis was conducted, based on the h-values returned by the optimal strategies transit assignment [151]. More specifically, the passenger volumes along the arcs were analysed, taking into account walk arc usage. The analysis revealed interesting insights, such as that passengers would have preferred walking over the use of two of the proposed routes returned by the route design stage. This insight would not have been intuitive had the walk arcs not been included, due to the small scale of the Matie Bus case study.

For a decision maker, such an insight may help save cost by omitting ineffective routes due to the improved accuracy of the AETT objective incorporated into the modelling approach, along with the walk arcs. In the case study, the aforementioned two ineffectual routes were omitted from the final recommended route set. A comparison was attempted between the relative performances of the resulting route set and of the current Matie bus day shuttle routes, and it was found that the establishment of the suggested transit network could potentially save minutes in AETT due to sufficient transportation being available to students, as opposed to having to walk.

CHAPTER 10

Future work

Contents

In fulfilment of Objective VIII of §1.3, this final chapter contains a selection of suggestions in respect of potential future work that can be pursued in follow-up to this dissertation. These suggestions are partitioned into five sections, namely a suggestion pertaining to a taxonomy of the literature relevant to the topic of this dissertation, suggestions related to modelling approaches that may be adopted for the UTNDP, suggestions as to solution approaches for solving UTNDP instances, a suggestion in respect of a decision support framework for the UTNDP, and lastly, suggestions related to various case study-specific investigations. For each suggestion, a formal statement is given, which is then briefly discussed and motivated.

10.1 A suggestion related to a taxonomy of the relevant literature

This section contains a suggestion for future work that may be pursued by building on the proposed taxonomy of Chapter 4.

Proposal I *Updating the taxonomy of the literature pertaining to the urban transit network design problem.*

The literature on the UTNDP is still active and constitutes a large body of research that is growing in terms of methodology and application with respect to designing transit networks and, more specifically, bus route design, bus frequency setting, timetabling and the operational tasks mentioned in §4.3. New models and techniques for solving various instances of the sub-problems of the UTNDP mentioned in §4.4 are regularly introduced. Chapter 4 contained a categorisation and exposition on research in the literature related to these topics, with an emphasis on the models and solution methodologies relevant to the topic of this dissertation. Due to the relatively active field of study, new research related to the UTNDP may have been introduced which is not documented in the taxonomy presented in Chapter 4. A further review may, therefore, be conducted in order to include any additional literature into the proposed taxonomy, in the contexts of the UTRP, the UTFSP and the UTRFSP.

10.2 Suggestions related to modelling approaches

This section contains three suggestions related to modelling approaches that may be adopted as future work, building on the work of this dissertation, as presented in Chapters 5 and 6.

Proposal II *Modelling and solving the UTRP based on trade-offs between three objective functions.*

The UTRP has recently been cast as a bi-objective minimisation model in which the two objectives pursued are minimisation of the ATT and minimisation of the TRT. The novel DP objective function proposed in this dissertation may, however, be included as a third objective function in addition to the ATT and TRT objectives of Chapter 5. The trade-offs between the ATT, the TRT and the newly introduced system disruption objective can then be considered. Such a tri-objective model could conceivably be solved by means of similar solution methodologies as those considered in this dissertation, but producing as output a set of non-dominated solutions based on the three objective functions. A decision maker may alternatively wish to specify a percentage of allowed disruption that (s)he is willing to tolerate and adopt that as a constraint when subjectively choosing a route set from among the non-dominated set of routes returned for the two objectives considered in this dissertation.

The incorporation of the aforementioned DP objective as a third objective would nevertheless reflect real scenarios more realistically, due to complete redesigns of transit systems not always being practical. The system disruption objective would ensure that non-dominated solutions uncovered, which exhibit small system disruption values, be retained during the search, as opposed to only minimising for the ATT and the TRT, and retaining the non-dominated solutions based on trade-offs between these two objectives. Although route sets associated with small system disruption values may possibly perform worse than other route sets obtained in pursuit of minimising the ATT and the TRT only, such route sets may lay out a roadmap of incremental steps that could be adopted in order to improve the transit network systematically and iteratively without causing too much disruption and confusion for its users.

Proposal III *Incorporating machine learning techniques when solving instances of the UTNDP.*

Machine learning has become a popular methodological trend in recent years. Although the concept is not new, it might hold potential for applications in the UTNDP context. In 1992, Xiong and Schneider [175] incorporated a neural network as proxy for solving the costly transit assignment problem (discussed in §4.6.4), and found that the trained neural network could predict total travel time values of customers with impressive speed and accuracy. John [89] also created surrogate models in 2016 which he used to approximate the passenger objective function value of a given route set. The long computation run times typically associated with solution implementations for large instances of the UTNDP pose a considerable obstacle for solving these instances within a reasonable computational time. Costly passenger transit assignment algorithms are the main cause of these computationally expensive run times, and one way of mitigating this problem may be to incorporate machine learning techniques to approximate these values instead. Further applications of combining machine learning and the UTNDP may also be explored due to a general lack of research that has been conducted in this regard.

Proposal IV *Combining transit modelling software with algorithmic solution techniques.*

One of the advantages of the commercial software alternatives mentioned in §4.8 used in practice during transit system design is the incorporation of more detail and complexities which reflects reality and real-world problems more accurately. When designing transit network systems purely

based on mathematical models with numerous underlying assumptions, the solutions returned might be far from reflecting reality. Such oversimplifications of UTNDP models may be addressed to some extent by a combination of optimisation models with software specialised in transportation modelling, in order to capture intricacies that could not be modelled in Chapters 5 and 6 of this dissertation.

One of the disadvantages, on the other hand, of the commercial software alternatives mentioned in §4.8 is that they do not have the capability to design a transit route network anew, but can typically only facilitate the use of *ad hoc* methods for designing and testing a transit route network manually by means of a variety of modelling tools available in these commercial software toolboxes. The models and solution approaches of Chapters 5 and 6 may therefore be exploited to remedy this shortcoming, while benefiting from the detail and sophistication that commercial software suites may add in terms of realism.

In this way, parameters that were not considered in this dissertation may be incorporated stochastically into the modelling approach, such as modelling passenger arrival rates and waiting times, bus departure and travel times, traffic congestion and passenger transfers. The strategic layouts of routes may thus be determined by the solution methodologies adopted in this dissertation, and then fine-tuned in the commercial software environment, hopefully creating synergy.

10.3 Suggestions related to solution approaches

Four suggestions are provided in this section pertaining to solution approaches for UTNDP instances.

Proposal V *Solving the UTRP and UTFSP simultaneously, and comparing the benefit achieved.*

As Ceder [29] pointed out, the simultaneous planning of the various stages of urban transit network design is desirable in order to exploit the full system capacity with a view to maximise productivity and efficiency. In the literature, various approaches have been proposed in which the UTRP is solved first, and then the UTFSP thereafter, instead of solving the UTRFSP as a whole. Separation of the routing and frequency setting aspects of the UTRFSP is, of course, aimed at reducing the computational expenditure associated with solving instances. The UTRP and the UTFSP are, in fact, rarely solved simultaneously (such as in the approach of Buba and Lee [22]), although they should theoretically not be uncoupled.

A question therefore arises as to whether the additional computational effort associated with a simultaneous solution approach is warranted in terms of the potentially improved quality of solutions over those solution approaches that involve solving the two sub-problems in stages. This is a worthy question to consider as unnecessary computational power might be wasted if no benefit of note is to be gained during the simultaneous solution approach. If, however, large gains in solution quality are thus achievable, it would certainly be worthwhile rather to focus more energy on developing solution methodologies capable of designing bus routes and setting bus frequencies simultaneously, instead of following the uncoupled approach that is most common in the literature.

The relative merits of the two approaches may be compared by measuring the performances of the approaches, both implemented in the same computing environment. The models and solution implementations of Chapters 5 and 6 may be used to evaluate the performance of solving the UTRP and UTFSP sub-problems separately. Another model and solution implementation would, however, be required to handle a decision variable containing both routes and frequencies

simultaneously. The AETT and TBR objective functions of §5.3 would be the most appropriate objectives on which to base such a methodological comparison, because they take into account frequencies, unlike the ATT objective function of §5.1, and encapsulate a more realistic estimate of both the passenger and operator objectives, albeit at a higher computational cost. The HV performance measure of §2.6.2 may be used to compare the solution qualities of the sets of non-dominated solutions returned by the two approaches.

Proposal VI *Solving the UTRP and UTFSP for time-varying OD demand.*

Although the case study based on the Matie Bus service in Chapter 8 was only conducted for one time instant during a weekday, the nature of the UTNDP is in reality dynamic, rather than static. Considering a version of the problem that is dynamic in terms of OD demand data would, admittedly, cause further complexities to be added to the modelling approach of this dissertation in order to capture the full essence of dynamic demand. One alternative to adding these extra complexities is to use the solution implementations of Chapter 6 on a regular basis by providing as input the OD demand matrices from the different time instants, and thus creating bus routes or setting bus frequencies appropriate for time-varying demand. This might be particularly useful when the demand pattern varies significantly over the duration of the day or over different days of the week. A decision maker would be able to monitor the demand patterns of the passengers, and as soon as a major shift in these patterns occurs, establish new routes or frequencies by invoking the algorithms of Chapter 6 in order to meet the passenger demand while also minimising operating cost.

Proposal VII *Implementing and testing other hyperheuristic approaches towards solving instances of the UTRP.*

Based on the considerable success achieved by Ahmed *et al.* [2] when using hyperheuristics to solve instances of the UTRP, and the high-quality results obtained in Chapter 7 of this dissertation, it is becoming evident that hyperheuristics are especially effective in the context of the UTRP, and should be considered during future solution implementations when solving instances of the UTRP. There are a vast number of hyperheuristics available in the literature, both for trajectory-based and population-based searches, which are yet to be explored. Both the approaches of Ahmed *et al.* [2] and that adopted in this dissertation aimed to manage the LLHs, but instead applying higher-level hyperheuristics aimed at choosing and managing different metaheuristic solution implementations, like the original AMALGAM methodology of Vrugt and Robinson [167], could be interesting avenues to explore.

Proposal VIII *Establishing an automated parameter design framework for the solution implementations considered in this dissertation.*

The parameter testing and design of both the DBMOSA algorithm and the NSGA II conducted in §7.2 and §7.3, respectively, may become a cumbersome task and requires much detailed planning and execution time. It is therefore proposed that the parameter design for the UTRP be automated by introducing a general framework for the effective fine-tuning of parameter values for the DBMOSA algorithm and the NSGA II aimed at improved performance. It is evident from the NSGA II employed by John [89], when compared with the NSGA II of §6.1.8, that a stark difference in performance is achievable by optimising NSGA II parameter settings. Establishing an automated approach towards finding high-quality parameter values may be a very useful research contribution.

10.4 A suggestion related to a decision support framework

This section contains a single suggestion related to the establishment of a generalised solution framework that may be of benefit to decision makers in terms of reaching high-quality decisions during the design of bus routes or when setting bus frequencies.

Proposal IX *Designing a general decision support framework for the models and solution implementations considered in this dissertation.*

Bagloee and Ceder [9] have claimed that many public transit networks have gone without reappraisal for anything between 20 and 50 years. All the while land-use patterns have been changing, as have migration patterns from towns to suburban areas. Public transit systems have, however, not kept up with the pace of change, prompting the authors to claim that it is desirable to establish automated tools for keeping up with the ever-changing public transport environment.

A user-friendly, computerised *decision support system* (DSS) may be a useful tool capable of effectively aiding operators in designing bus routes with speed and of high quality, while keeping up with the times.

The models and solution implementations of Chapters 5 and 6 may compactly be packaged into such a DSS, which may serve as a tool for bus route design and bus frequencies setting. Decision makers without prior knowledge of coding or mathematical modelling would then be able to use the DSS effectively in support of their decision making processes.

Such a DSS should have the option of establishing a user-specified UTRP instance or a UTFSP instance. If the UTRP instance option is selected, the DSS should be able to take as input the required OD demand data, a weight matrix representing the road network, and the coordinates of the bus stops in the form of *.csv* files. As output, the DSS should be able to recommend the solutions returned by the algorithm, displayed graphically in a two-dimensional plane with each axis representing an objective function, and to visualise any user-selected solution in decision space in a graphical and intuitive manner with the bus stops displaced according to the input coordinates (such as Figure 7.6, for example). The decision maker should also be able to download and save the solutions together with their corresponding objective function values.

When the option is selected for frequency setting, the user should be able to provide all the input as required for the route design option, as well as a specific route set for which frequencies are required. As output, the DSS should display solutions graphically as described above, and allow a user to select a solution and view its various recommended frequencies. Furthermore, analysis tables should be presented when a solution is selected, similar to Tables 8.6, 8.7 and 8.8. The solutions, as well as their objective function values, should again be downloadable by the user.

10.5 Suggestions related to case studies

Three suggestions are presented in this section for future work that may be pursued, building upon the work presented in Chapters 5–8, specifically pertaining to case studies.

Proposal X *Creating more benchmark problem instances in which walk arcs are incorporated into the modelling approach.*

The concept of walk arcs was explored in this dissertation, mainly in Chapters 5 and 8, and was shown to add an additional element of realism when modelling passenger movements. This

becomes particularly insightful when small UTFSP instances are established, and passengers may prefer walking above travelling by bus. Previous research has not taken into account walk arcs, as this is not always necessary when large distances prevail between bus stops, in which case passengers would not prefer to walk. When small transit networks are, however, considered, the models may return inaccurate results due to no walk options having been included, while in reality the passengers might well choose the option of walking when bus routes are not well designed (exhibiting long detours and/or requiring multiple transfers), or insufficient bus assignments are made to routes (inducing long waiting times or overcrowding at bus stops). These are important elements to take into consideration, as an unrealistic model will return results of limited value, and thereby limit effective decision making based on accurate information.

Proposal XI *Creating more benchmark problem instances of the UTRP with incremental redesign.*

The new approach towards incremental redesign of transit networks, introduced in §5.2 and showcased in §7.5, may hold considerable potential for the urban transit network design community, and is an avenue that deserves further exploration. Introducing new real-world case studies for which current operational route sets are provided as reference route sets for the UTRP with incremental redesign, and following the aforementioned solution approach, may contribute towards building on this new line of study.

Proposal XII *Incorporating a minibus taxi service during the off-peak hours of the day for the Matie Bus case study.*

In the context of Stellenbosch University, student mobility may be improved through a possible co-operation between minibus taxi services and the Maties Bus service of the university, whereby any free capacity in minibus taxis can be utilised to serve student demand during the off-peak hours of the day. As Klink [98] discovered, the demand peaks for students and the off-peak hours of taxis, representing surplus supply, coincide within the same timeslots, and could therefore be matched so that the surplus supply may be utilised to meet the unmet demand of students. Potential routes may therefore be designed and proposed as another case study for designing bus routes which may be maintained by an operator during off-peak hours. This may be performed in a fashion similar to the case study of Chapter 8, but taking into account the various intricacies and design requirements for designing routes and bus frequencies specific to every hour of the day in the context of the Stellenbosch University student population.

Proposal XIII *Conducting case studies on larger real-world instances.*

As explained in the dissertation, it is more challenging, and takes longer, to obtain high-quality solutions to larger instances of the UTRP than for the smaller networks typically considered in the literature. Considerable insight might be gained if larger real-world case studies were to be conducted.

References

[1] Aarts E, Korst J & Michiels W, 2005, *Simulated annealing*, pp. 187–210 in Burke EK & Kendall G (Eds), *Search methodologies: Introductory tutorials in optimization and decision support techniques*, Springer, Boston (MA).

[2] Ahmed L, Mumford C & Kheiri A, 2019, *Solving urban transit route design problem using selection hyper-heuristics*, European Journal of Operational Research, **274(2)**, pp. 545–559.

[3] Ahmed L, Soares PH, Mumford C & Mao Y, 2019, *Optimising bus routes with fixed terminal nodes: Comparing hyper-heuristics with NSGA II on realistic transportation networks*, Proceedings of the Genetic and Evolutionary Computation Conference, Prague, pp. 1102–1110.

[4] Arbex RO & da Cunha CB, 2015, *Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm*, Transportation Research — Part B: Methodological, **81**, pp. 355–376.

[5] Baaj MH & Mahmassani HS, 1991, *An AI-based approach for transit route system planning and design*, Journal of Advanced Transportation, **25(2)**, pp. 187–209.

[6] Baaj MH & Mahmassani HS, 1995, *Hybrid route generation heuristic algorithm for the design of transit networks*, Transportation Research — Part C: Emerging Technologies, **3(1)**, pp. 31–50.

[7] Back T, 1996, *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, Oxford.

[8] Bader J & Zitzler E, 2011, *HypE: An algorithm for fast optimization*, Evolutionary Computation, **19(1)**, pp. 45–76.

[9] Bagloee SA & Ceder AA, 2011, *Transit-network design methodology for actual-size road networks*, Transportation Research — Part B: Methodological, **45(10)**, pp. 1787–1804.

[10] Behnel S, Bradshaw R, Dalcin L, Florisson M, Makarov V & Seljebotn DS, 2021, *Cython C-extensions for Python*, [Online], [Cited September 28th, 2021], Available from `https://cython.org/`.

[11] Benn H, 1995, *TCRP Synthesis 10: Bus route evaluation standards*, (Unpublished) Technical Report, Transportation Research Board & National Research Council, Washington (DC).

[12] Bertsimas D & Tsitsiklis J, 1993, *Simulated annealing*, Statistical Science, **8(1)**, pp. 10–15.

[13] Beume N, 2009, *S-metric calculation by considering dominated hypervolume as Klee's measure problem*, Evolutionary Computation, **17(4)**, pp. 477–492.

[14] Beume N, Fonseca CM, López-Ibáñez M, Paquete L & Vahrenhold J, 2009, *On the complexity of computing the hypervolume indicator*, IEEE Transactions on Evolutionary Computation, **13(5)**, pp. 1075–1082.

[15] Bilgin B, Özcan E & Korkmaz EE, 2006, *An experimental study on hyper-heuristics and exam timetabling*, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, pp. 394–412.

[16] Bozkurt B, Fowler JW, Gel ES, Kim B, Köksalan M & Wallenius J, 2010, *Quantitative comparison of approximate solution sets for multicriteria optimization problems with weighted Tchebycheff preference function*, Operations Research, **58(3)**, pp. 650–659.

[17] Bringmann K & Friedrich T, 2012, *Approximating the least hypervolume contributor: NP-hard in general, but fast in practice*, Theoretical Computer Science, **425**, pp. 104–116.

[18] Bringmann K & Friedrich T, 2010, *Approximating the volume of unions and intersections of high-dimensional geometric objects*, Computational Geometry, **43(6-7)**, pp. 601–610.

[19] Bringmann K & Friedrich T, 2013, *Approximation quality of the hypervolume indicator*, Artificial Intelligence, **195**, pp. 265–290.

[20] Brockhoff D, Wagner T & Trautmann H, 2012, *On the properties of the R2 Indicator*, Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, Philadelphia (PA), pp. 465–472.

[21] Buba AT & Lee LS, 2016, *Differential evolution for urban transit routing problem*, Journal of Computer and Communications, **4(14)**, pp. 11–25.

[22] Buba AT & Lee LS, 2018, *A differential evolution for simultaneous transit network design and frequency setting problem*, Expert Systems with Applications, **106**, pp. 277–289.

[23] Bunte S & Kliewer N, 2009, *An overview on vehicle scheduling models*, Public Transport, **1(4)**, pp. 299–317.

[24] Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E & Qu R, 2013, *Hyper-heuristics: A survey of the state of the art*, Journal of the Operational Research Society, **64(12)**, pp. 1695–1724.

[25] Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E & Woodward JR, 2010, *A classification of hyper-heuristic approaches*, pp. 449–468 in Gendreau M & Potvin JY (Eds), *Handbook of metaheuristics*, Springer, New York (NY).

[26] Busetti F, 2003, *Simulated annealing overview*, [Online], [Cited August 12th, 2020], Available from `http://www.cs.ubbcluj.ro/~csatol/mestint/pdfs/Busetti_AnnealingIntro.pdf`.

[27] Byrne BF, 1975, *Public transportation line positions and headways for minimum user and system cost in a radial case*, Transportation Research, **9(2-3)**, pp. 97–102.

[28] Ceder A, 1984, *Bus frequency determination using passenger count data*, Transportation Research — Part A: General, **18(5-6)**, pp. 439–453.

[29] Ceder A, 2007, *Public transit planning and operation: Theory, modelling and practice*, 1st Edition, Elsevier Ltd, London.

[30] Ceder A & Wilson NH, 1986, *Bus network design*, Transportation Research — Part B: Methodological, **20(4)**, pp. 331–344.

[31] Chakroborty P, 2003, *Genetic algorithms for optimal urban transit network design*, Computer-aided Civil and Infrastructure Engineering, **18(3)**, pp. 184–200.

[32] Chakroborty P & Wivedi T, 2002, *Optimal route network design for transit systems using genetic algorithms*, Engineering Optimization, **34(1)**, pp. 83–100.

[33] Chew JSC, Lee LS & Seow HV, 2013, *Genetic algorithm for bi-objective urban transit routing problem*, Journal of Applied Mathematics, **2013**, pp. 1–15.

[34] Citilabs, 2020, *Cube Voyager*, [Online], [Cited October 30th, 2020], Available from `https://www.citilabs.com/software/cube/cube-voyager/`.

[35] Coello Coello CA & Sierra MR, 2004, *A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm*, Proceedings of the 3rd Mexican International Conference on Artificial Intelligence, Mexico City, pp. 688–697.

[36] Constantin I & Florian M, 1995, *Optimizing frequencies in a transit network: A nonlinear bi-level programming approach*, International Transactions in Operational Research, **2(2)**, pp. 149–164.

[37] Davis L, 1985, *Applying adaptive algorithms to epistatic domains*, Proceedings of the 9th International Joint Conference on Artificial Intellingence, Los Angeles (CA), pp. 162–164.

[38] Deb K, 1999, *An introduction to genetic algorithms*, Sadhana, **24(4-5)**, pp. 293–315.

[39] Deb K, 2001, *Multi-objective optimisation using evolutionary algorithms*, John Wiley & Sons, New York (NY).

[40] Deb K & Jain S, 2002, *Running performance metrics for evolutionary multi-objective optimization*, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, pp. 13–20.

[41] Deb K, Miettinen K & Chaudhuri S, 2010, *Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches*, IEEE Transactions on Evolutionary Computation, **14(6)**, pp. 821–841.

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

[43] Desaulniers G & Hickman MD, 2007, *Public transit*, pp. 69–120 in Barnhart C & Laporte G (Eds), *Handbooks in operations research and management science*, Elsevier, Amsterdam.

[44] Dial RB, 1971, *A probabilistic multipath traffic assignment model which obviates path enumeration*, Transportation Research, **5(2)**, pp. 83–111.

[45] Dijkstra EW, 1959, *A note on two problems in connexion with graphs*, Numerische Mathematik, **1(1)**, pp. 269–271.

[46] Dorigo M, 1992, *Optimisation, learning and natural algorithms*, PhD Thesis, Politecnico di Milano, Milan.

[47] Dréo J, Pétrowski A, Siarry P & Taillard E, 2006, *Metaheuristics for hard optimization: Methods and case studies*, 1st Edition, Springer, Berlin.

[48] Dunlap M, Li Z, Henrickson K & Wang Y, 2016, *Estimation of origin and destination information from Bluetooth and Wi-fi sensing for transit*, Transportation Research Record: Journal of the Transportation Research Board, **2595**, pp. 11–17.

[49] Eiben ÁE, Hinterding R & Michalewicz Z, 1999, *Parameter control in evolutionary algorithms*, IEEE Transactions on Evolutionary Computation, **3(2)**, pp. 124–141.

[50] Fan L & Mumford CL, 2010, *A metaheuristic approach to the urban transit routing problem*, Journal of Heuristics, **16(3)**, pp. 353–372.

[51] Fan W & Machemehl RB, 2004, *Optimal transit route network design problem: Algorithms, implementations, and numerical results*, (Unpublished) Technical Report, Center for Transportation Research, University of Texas, Austin (TX).

[52] Fan W & Machemehl RB, 2006, *Optimal transit route network design problem with variable transit demand: Genetic algorithm approach*, Journal of Transportation Engineering, **132(1)**, pp. 40–51.

[53] Fan W & Machemehl RB, 2006, *Using a simulated annealing algorithm to solve the Transit Route Network Design Problem*, Journal of Transportation Engineering, **132(2)**, pp. 122–132.

[54] Fan W & Machemehl RB, 2008, *Tabu search strategies for the public transportation network optimizations with variable transit demand*, Computer-Aided Civil and Infrastructure Engineering, **23(7)**, pp. 502–520.

[55] Fan W, Machemehl RB & Lownes NE, 2008, *Some computational insights on the optimal bus transit route network design problem*, Journal of the Transportation Research Forum, **47(3)**, pp. 60–75.

[56] Farhang-Mehr A & Azarm S, 2003, *Minimal sets of quality metrics*, Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization, Berlin, pp. 405–417.

[57] Faulkenberg SL & Wiecek MM, 2010, *On the quality of discrete representations in multiple objective programming*, Optimization and Engineering, **11(3)**, pp. 423–440.

[58] Fleischer M, 2003, *The measure of Pareto optima applications to multi-objective metaheuristics*, Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization, Berlin, pp. 519–533.

[59] Floyd RW, 1962, *Algorithm 97: Shortest path*, Communications of the Association for Computing Machinery, **5(6)**, p. 345.

[60] Gao Z, Sun H & Shan LL, 2003, *A continuous equilibrium network design model and algorithm for transit systems*, Transportation Research — Part B: Methodological, **38(3)**, pp. 235–250.

[61] Geem ZW, Kim JH & Loganathan GV, 2001, *A new heuristic optimization algorithm: Harmony search*, Simulation, **76(2)**, pp. 60–68.

[62] Glover F, 1986, *Future paths for integer programming and links to artificial intelligence*, Computers and Operations Research, **13(5)**, pp. 533–549.

[63] Gogna A & Tayal A, 2013, *Metaheuristics: Review and application*, Journal of Experimental and Theoretical Artificial Intelligence, **25(4)**, pp. 503–526.

[64] Goldberg DE & Lingle R, 1985, *Alleles, loci, and the traveling salesman problem*, Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh (PA), pp. 154–159.

[65] Gomory RE, 1958, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society, **64(5)**, pp. 275–278.

[66] Google, 2020, *Google Maps*, [Online], [Cited October 20th, 2020], Available from `https://www.google.com/maps/`.

[67] Guan J, Yang H & Wirasinghe SC, 2006, *Simultaneous optimization of transit line configuration and passenger line assignment*, Transportation Research — Part B: Methodological, **40(10)**, pp. 885–902.

[68] Guihaire V & Hao JK, 2008, *Transit network design and scheduling: A global review*, Transportation Research — Part A: Policy and Practice, **42(10)**, pp. 1251–1273.

[69] Han AF & Wilson NH, 1982, *The allocation of buses in heavily utilized networks with overlapping routes*, Transportation Research — Part B: Methodological, **16(3)**, pp. 221–232.

[70] Hansen MP & Jaszkiewicz A, 1998, *Evaluating the quality of approximations to the non-dominated set*, (Unpublished) Technical Report, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby.

[71] Hansen P, Mladenović N & Pérez JAM, 2008, *Variable neighbourhood search: Methods and applications*, 4OR, **6(4)**, pp. 319–360.

[72] Hasselström D, 1981, *Public transportation planning: A mathematical programming approach*, PhD Dissertation, University of Gothenburg, Gothenburg.

[73] Henning MA & Van Vuuren JH, 2021, *Graph and network theory — An applied approach using* Mathematica, Springer, In press.

[74] Hillier FS & Lieberman GJ, 2010, *Introduction to operations research*, 9th Edition, McGraw-Hill, New York (NY).

[75] Hitge G & Vanderschuren M, 2015, *Comparison of travel time between private car and public transport in Cape Town*, Journal of the South African Institution of Civil Engineering, **57(3)**, pp. 35–43.

[76] Hoffman KL & Padberg M, 1991, *Improving LP-representations of zero-one linear programs for branch-and-cut*, ORSA Journal on Computing, **3(2)**, pp. 121–134.

[77] Holland JH, 1992, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, Cambridge (MA).

[78] Huang M, Romeo F & Sangiovanni-Vincentelli A, 1986, *An efficient general cooling schedule for simulated annealing*, Proceedings of the 1986 IEEE International Conference on Computer-Aided Design, Santa Carla (CA).

[79] Huband S, Hingston P, While L & Barone L, 2003, *An evolution strategy with probabilistic mutation for multi-objective optimisation*, Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Canberra, pp. 2284–2291.

[80] Hüsselmann G, Van Vuuren JH & Andersen SJ, 2021, *Walk modelling when designing small bus transit networks*, South African Journal of Industrial Engineering, In press.

[81] Huurne D & Andersen J, 2014, *A quantitative measure of congestion in Stellenbosch using probe data*, Proceedings of the 1st International Conference on the use of Mobile Informations and Communication Technology in Africa, Stellenbosch, pp. 62–67.

[82] Ibarra-Rojas OJ, Delgado F, Giesen R & Munoz JC, 2015, *Planning, operation, and control of bus transport systems: A literature review*, Transportation Research — Part B: Methodological, **77**, pp. 38–75.

[83] Ikeda K, Kita H & Kobayashi S, 2001, *Failure of Pareto-based MOEAs: Does non-dominated really mean near to optimal?*, Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, pp. 957–962.

[84] Iles R, 2005, *Public transport in developing countries*, Emerald Group Publishing Limited, Bingley.

[85] Iliopoulou C, Kepaptsoglou K & Vlahogianni E, 2019, *Metaheuristics for the transit route network design problem: A review and comparative analysis*, Public Transport, **11(3)**, pp. 487–521.

[86] INRO, 2020, *Emme*, [Online], [Cited October 30th, 2020], Available from `https://www.inrosoftware.com/en/products/emme/`.

[87] Ishibuchi H, Masuda H & Nojima Y, 2015, *A study on performance evaluation ability of a modified inverted generational distance indicator*, Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, pp. 695–702.

[88] Jha SB, Jha JK & Tiwari MK, 2019, *A multi-objective meta-heuristic approach for transit network design and frequency setting problem in a bus transit system*, Computers and Industrial Engineering, **130**, pp. 166–186.

[89] John MP, 2016, *Metaheuristics for designing efficient routes & schedules for urban transportation networks*, PhD Thesis, Cardiff University, Cardiff.

[90] John MP, Mumford CL & Lewis R, 2014, *An improved multi-objective algorithm for the urban transit routing problem*, Evolutionary Computation in Combinatorial Optimisation, Lecture Notes in Computer Science, **8600**, pp. 49–60.

[91] Kechagiopoulos PN & Beligiannis GN, 2014, *Solving the urban transit routing problem using a particle swarm optimization based algorithm*, Applied Soft Computing, **21**, pp. 654–676.

[92] Kennedy J & Eberhart R, 1995, *Particle swarm optimization*, Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, pp. 1942–1948.

[93] Kepaptsoglou K & Karlaftis M, 2009, *Transit Route Network Design Problem: Review*, Journal of Transportation Engineering, **135(8)**, pp. 491–505.

[94] Kernighan BW & Ritchie DM, 1988, *The C programming language*, Prentice-Hall, Englewood Cliffs (NJ).

[95] Kiliç F & Gök M, 2014, *A demand based route generation algorithm for public transit network design*, Computers and Operations Research, **51**, pp. 21–29.

[96] Kirkpatrick S, Gelatt CD & Vecchi MP, 1983, *Optimization by simulated annealing*, Science, **220(4598)**, pp. 671–680.

[97] Kirlik G & Sayin S, 2015, *Computing the nadir point for multiobjective discrete optimization problems*, Journal of Global Optimization, **62(1)**, pp. 79–99.

[98] Klink HP, 2020, *The characterisation of non-motorised transportation on the Stellenbosch campus*, MA Thesis, Stellenbosch University, Stellenbosch.

[99] Knowles J & Corne D, 2002, *On metrics for comparing nondominated sets*, Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Honolulu (HI), pp. 711–716.

[100] Knowles J & Corne D, 2003, *Properties of an adaptive archiving algorithm for storing nondominated vectors*, IEEE Transactions on Evolutionary Computation, **7(2)**, pp. 100–116.

[101] Knowles JD, 2002, *Local-search and hybrid evolutionary algorithms for Pareto optimization*, PhD Thesis, University of Reading, Reading.

[102] Knowles JD, Thiele L & Zitzler E, 2006, *A tutorial on the performance assessment of stochastic multiobjective optimizers*, (Unpublished) Technical Report, Computer Engineering and Networks Laboratory, ETH Zürich, Zürich.

[103] Kora P & Yadlapalli P, 2017, *Crossover operators in genetic algorithms: A review*, International Journal of Computer Applications, **162(10)**, pp. 34–36.

[104] Kramer O, 2017, *Genetic algorithm essentials*, Springer, New York (NY).

[105] Kung HT, Luccio F & Preparata FP, 1975, *On finding the maxima of a set of vectors*, Journal of the Association for Computing Machinery, **22(4)**, pp. 469–476.

[106] Kutzbach M, 2010, *Megacities and megatraffic*, [Online], [Cited November 6th, 2020], Available from `https://www.accessmagazine.org/fall-2010/megacities-megatraffic/`.

[107] Land A & Doig A, 1960, *An automatic method of solving discrete programming problems*, Econometrica, **28(3)**, pp. 497–520.

[108] LeBlanc LJ, 1988, *Transit system network design*, Transportation Research — Part B: Methodological, **22(5)**, pp. 383–390.

[109] Lee YJ & Vuchic VR, 2005, *Transit network design with variable demand*, Journal of Transportation Engineering, **131(1)**, pp. 1–10.

[110] Li M, Chen T & Yao X, 2018, *A critical review of: "A practical guide to select quality indicators for assessing Pareto-based search algorithms in search-based software engineering": Essay on quality indicator selection for SBSE*, Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, Gothenburg, pp. 17–20.

[111] Li M & Yao X, 2019, *Quality evaluation of solution sets in multiobjective optimisation: A survey*, Association for Computing Machinery Computing Surveys, **52(2)**, pp. 1–38.

[112] Liu Y, Bunker J & Ferreira L, 2010, *Transit users' route-choice modelling in transit assignment: A review*, Transport Reviews, **30(6)**, pp. 753–769.

[113] Lombard M & Hugo J, 2002, *Public transport in Cape Town: Customer opinions, attitudes and revealed preferences*, (Unpublished) Technical Report, TRC Africa (Pty) Ltd, City of Cape Town Metropolitan Municipality, Cape Town.

[114] Magnanti TL & Wong RT, 1984, *Network design and transportation planning: Models and algorithms*, Transportation Science, **18(1)**, pp. 1–55.

[115] Mandl CE, 1979, *Applied network optimization*, Academic Press, London.

[116] Mandl CE, 1980, *Evaluation and optimization transportation networks*, European Journal of Operational Research, **5**, pp. 396–404.

[117] Martínez H, Mauttone A & Urquhart ME, 2014, *Frequency optimization in public transportation systems: Formulation and metaheuristic approach*, European Journal of Operational Research, **236(1)**, pp. 27–36.

[118] Martins EQ & Pascoal MM, 2003, *A new implementation of Yen's ranking loopless paths algorithm*, Quarterly Journal of the Belgian, French and Italian Operations Research Societies, **1(2)**, pp. 121–133.

[119] Mauttone A & Urquhart ME, 2009, *A multi-objective metaheuristic approach for the transit network design problem*, Public Transport, **1(4)**, pp. 253–273.

[120] Mladenović N & Hansen P, 1997, *Variable neighborhood search*, Computers and Operations Research, **24(11)**, pp. 1097–1100.

[121] Mumford CL, 2013, *New heuristic and evolutionary operators for the multi-objective urban transit routing problem*, Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cardiff, pp. 939–946.

[122] Murray AT, Davis R, Stimson RJ & Ferreira L, 1998, *Public transportation access*, Transportation Research — Part D: Transport and Environment, **3(5)**, pp. 319–328.

[123] Nel GS, 2021, *A hyperheuristic approach towards the training of artificial neural networks*, PhD Thesis, Stellenbosch University, Stellenbosch.

[124] Newell GF, 1979, *Some issues related to the optimal design of bus routes*, Transportation Science, **13(1)**, pp. 20–35.

[125] Ngamchai S & Lovell DJ, 2003, *Optimal time transfer in bus transit route network design using a genetic algorithm*, Journal of Transportation Engineering, **129(5)**, pp. 510–521.

[126] Nikolić M & Teodorović D, 2013, *Transit network design by bee colony optimization*, Expert Systems with Applications, **40(15)**, pp. 5945–5955.

[127] Nikolić M & Teodorović D, 2014, *A simultaneous transit network design and frequency setting: Computing with bees*, Expert Systems with Applications, **41(16)**, pp. 7200–7209.

[128] Okabe T, Jin Y & Sendhoff B, 2003, *A critical survey of performance indices for multi-objective optimisation*, Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Canberra, pp. 878–885.

[129] Oliver I, Smith D & Holland JR, 1987, *Study of permutation crossover operators on the traveling salesman problem*, Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge (MA), pp. 224–230.

[130] Özcan E, Bilgin B & Korkmaz EE, 2008, *A comprehensive analysis of hyper-heuristics*, Intelligent Data Analysis, **12(1)**, pp. 3–23.

[131] Park SJ, 2005, *Bus network scheduling with genetic algorithms and simulation*, MA Thesis, University of Maryland, College Park (MD).

[132] Pattnaik S, Mohan S & Tom V, 1998, *Urban bus transit route network design using genetic algorithm*, Journal of Transportation Engineering, **124(4)**, pp. 368–375.

[133] Possel B, Wismans LJ, Van Berkum EC & Bliemer MC, 2018, *The multi-objective network design problem using minimizing externalities as objectives: Comparison of a genetic algorithm and simulated annealing framework*, Transportation, **45(2)**, pp. 545–572.

[134] PTV Group, 2020, *PTV Visum*, [Online], [Cited October 30th, 2020], Available from `https://www.ptvgroup.com/en/solutions/products/ptv-visum/`.

[135] Rardin RL, 1998, *Optimization in operations research*, Prentice Hall, Upper Saddle River (NJ).

[136] Ravber M, Mernik M & Črepinšek M, 2017, *The impact of quality indicators on the rating of multi-objective evolutionary algorithms*, Applied Soft Computing, **55**, pp. 265–275.

[137] Riquelme N, Von Lücken C & Baran B, 2015, *Performance metrics in multi-objective optimization*, Proceedings of the 2015 Latin American Computing Conference (CLEI), Arequipa, pp. 1–11.

[138] Royal HaskoningDHV, 2016, *Comprehensive integrated transport plan 2016–2020*, (Unpublished) Technical Report, Stellenbosch Municipality, Stellenbosch.

[139] Saferspaces, 2017, *The state of public transport in South Africa*, [Online], [Cited November 6th, 2020], Available from `https://www.saferspaces.org.za/understand/entry/the-state-of-public-transport-in-south-africa`.

[140] Sastry K & Goldberg D, 2005, *Genetic algorithms*, pp. 97–125 in Burke E & Kendall G (Eds), *Search methodologies*, Springer, New York (NY).

[141] SATURN, 2020, *Saturn: Congested highway assignment software*, [Online], [Cited October 30th, 2020], Available from `https://saturnsoftware2.co.uk/`.

[142] Sayin S, 2000, *Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming*, Mathematical Programming, Series B, **87(3)**, pp. 543–560.

[143] Schéele S, 1980, *A supply model for public transit services*, Transportation Research — Part B: Methodological, **14(1-2)**, pp. 133–146.

[144] Schott JR, 1995, *Fault tolerant design using single and multicriteria genetic algorithm optimization*, MA Thesis, Massachusetts Institute of Technology, Cambridge (MA).

[145] Sen S & Sherali HD, 1985, *A branch and bound algorithm for extreme point mathematical programming problems*, Discrete Applied Mathematics, **11(3)**, pp. 265–280.

[146] Shih MC & Mahmassani HS, 1994, *A design methodology for bus transit route networks with coordinated operations*, (Unpublished) Technical Report 1, Center for Transportation Research, University of Texas at Austin, Austin (TX).

[147] Smats Traffic Solutions, 2018, *TrafficBox — The portable and secure scanner*, [Online], [Cited April 12th, 2018], Available from `http://www.smatstraffic.com/products/trafficbox/`.

[148] Smith KI, Everson RM, Fieldsend JE, Murphy C & Misra R, 2008, *Dominance-based multiobjective simulated annealing*, IEEE Transactions on Evolutionary Computation, **12(3)**, pp. 323–342.

[149] Soares HP, 2020, *Three steps towards practical application of public transport route optimisation in urban areas*, PhD thesis, University of Nottingham, Nottingham.

[150] Soares PH, Ahmed L, Mao Y & Mumford CL, 2021, *Public transport network optimisation in PTV Visum using selection hyper-heuristics*, Public Transport, **13(1)**, pp. 163–196.

[151] Spiess H & Florian M, 1989, *Optimal strategies: A new assignment model for transit networks*, Transportation Research — Part B: Methodological, **23(2)**, pp. 83–102.

[152] Srinivas N & Deb K, 1994, *Multiobjective optimization using nondominated sorting in genetic algorithms*, Evolutionary Computation, **2(3)**, pp. 221–248.

[153] Stellenbosch Smart Mobility Lab, 2018, *Stellenbosch Smart Mobility Lab — About SSML*, [Online], [Cited March 15th, 2018], Available from `http://www.sun.ac.za/english/faculty/eng/ssml/Pages/default.aspx`.

[154] Stellenbosch Unit for Operations Research in Engineering, 2018, *SUnORE Home*, [Online], [Cited September 30th, 2018], Available from `http://sunore.co.za/`.

[155] Stellenbosch University Sustainability, 2020, *Stellenbosch University Systemic Sustainability Campus shuttle service*, [Online], [Cited October 19th, 2020], Available from `http://www0.sun.ac.za/sustainability/pages/services/transport/campus-shuttle-service.php`.

[156] Stewart TJ, 2007, *The essential multiobjectivity of linear programming*, ORiON, **23(1)**, pp. 1–15.

[157] Suppapitnarm A, Seffen KA, Parks GT & Clarkson PJ, 2000, *Simulated annealing algorithm for multiobjective optimization*, Engineering Optimization, **33(1)**, pp. 59–85.

[158] Tan KC, Lee TH & Khor EF, 2002, *Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons*, Artificial Intelligence Review, **17(4)**, pp. 251–290.

[159] The Association of European Operational Research Societies, 2020, *The Association of European Operational Research Societies*, [Online], [Cited March 9th, 2020], Available from `https://www.euro-online.org/web/pages/301/or-and-euro`.

[160] Tian Y, Cheng R, Zhang X, Cheng F & Jin Y, 2018, *An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility*, IEEE Transactions on Evolutionary Computation, **22(4)**, pp. 609–622.

[161] Toth P & Vigo D, 2002, *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[162] Van Rossum G & Drake FL, 2009, *Python 3 reference manual*, CreateSpace, Scotts Valley (CA).

[163] Van Veldhuizen DA & Lamont GB, 1998, *Evolutionary computation and convergence to a Pareto front*, Proceedings of the Late Breaking Papers at the 1998 Genetic Programming Conference, Madison (WI), pp. 221–228.

[164] Van Veldhuizen DA, 1999, *Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations*, PhD Thesis, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Dayton (OH).

[165] Van Vuuren JH, 2019, *What is operations research?*, pp. 1–12 in Kruger HA & Van Vuuren JH (Eds), *Operations research in South Africa — The first 50 years*, AFRICAN SUN MeDIA, Stellenbosch.

[166] Vigeh A, 2011, *Investigation of a simulated annealing cooling schedule used to optimise the estimation of the fiber diameter distribution in a peripheral nerve trunk*, MA Thesis, California Polytechnic State University, San Luis Obispo (CA).

[167] Vrugt JA & Robinson BA, 2007, *Improved evolutionary optimization from genetically adaptive multimethod search*, Proceedings of the National Academy of Sciences, **104(3)**, pp. 708–711.

[168] Wan QK & Lo HK, 2003, *A mixed integer formulation for multiple-route transit network design*, Journal of Mathematical Modelling and Algorithms, **2(4)**, pp. 299–308.

[169] Wang S, Ali S, Yue T, Li Y & Liaaen M, 2016, *A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering*, Proceedings of the 38th International Conference on Software Engineering, Austin (TX), pp. 631–642.

[170] While L, Bradstreet L & Barone L, 2012, *A fast way of calculating exact hypervolumes*, IEEE Transactions on Evolutionary Computation, **16(1)**, pp. 86–95.

[171] Winston WL, 2004, *Operations research: Applications and algorithms*, 4th Edition, Thomson Brooks/Cole, Belmont (CA).

[172] Wolpert DH & Macready WG, 1997, *No free lunch theorems for optimization*, IEEE Transactions on Evolutionary Computation, **1(1)**, pp. 67–82.

[173] Wren A & Rousseau JM, 1995, *Bus driver scheduling — An overview*, Proceedings of the 6th International Workshop on Computer-Aided Scheduling of Public Transport, Berlin, pp. 173–187.

[174] Wu J, Song R, Wang Y, Chen F & Li S, 2014, *Modeling the coordinated operation between bus rapid transit and bus*, Mathematical Problems in Engineering, **2015**, pp. 1–7.

[175] Xiong Y & Schneider JB, 1992, *Transportation network design using a cumulative genetic algorithm and neural network*, Transportation Research Record, **1364**, pp. 37–44.

[176] Xu GM, Shi F & Wang P, 2014, *Model and algorithm of optimizing bus transit network based on line segment combination*, Proceedings of the 14th COTA International Conference of Transportation Professionals, Changsha, pp. 1514–1525.

[177] Yan Y, Liu Z, Meng Q & Jiang Y, 2013, *Robust optimization model of bus transit network design with stochastic travel time*, Journal of Transportation Engineering, **139(6)**, pp. 625–634.

[178] Yang XS, 2009, *Harmony search as a metaheuristic algorithm*, pp. 1–14 in Geem Z (Ed), *Music-inspired harmony search algorithm: Theory and applications*, Springer, Berlin.

[179] Yen JY, 1971, *Finding the K shortest loopless paths in a network*, Management Science, **17(11)**, pp. 712–716.

[180] Yu B, Yang ZZ, Jin PH, Wu SH & Yao BZ, 2012, *Transit route network design-maximizing direct and transfer demand density*, Transportation Research — Part C: Emerging Technologies, **22**, pp. 58–75.

[181] Yu B, Yang Z & Yao J, 2010, *Genetic algorithm for bus frequency optimization*, Journal of Transportation Engineering, **136(6)**, pp. 576–583.

[182] Zhao F & Gan A, 2003, *Optimization of transit network to minimize transfers*, (Unpublished) Technical Report, Lehman Center for Transportation Research, Department of Civil & Environmental Engineering, Florida International University, Miami (FL).

[183] Zhao F & Ubaka I, 2004, *Transit network optimization — Minimizing transfers and optimizing route directness*, Journal of Public Transportation, **7(1)**, pp. 63–82.

[184] Zhao F, Ubaka I & Gan A, 2005, *Transit network optimization: Minimizing transfers and maximizing service coverage with an integrated simulated annealing and tabu search method*, Transportation Research Record, **1923(1)**, pp. 180–188.

[185] Zhao F & Zeng X, 2006, *Optimization of transit network layout and headway with a combined genetic algorithm and simulated annealing method*, Engineering Optimization, **38(6)**, pp. 701–722.

[186] Zhao H, Xu W & Jiang R, 2015, *The memetic algorithm for the optimization of urban transit network*, Expert Systems with Applications, **42(7)**, pp. 3760–3773.

[187] Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN & Zhangd Q, 2011, *Multiobjective evolutionary algorithms: A survey of the state of the art*, Swarm and Evolutionary Computation, **1(1)**, pp. 32–49.

[188] Zitzler E, Deb K & Thiele L, 2000, *Comparison of multiobjective evolutionary algorithms: Empirical results*, Evolutionary Computation, **8(2)**, pp. 173–195.

[189] Zitzler E, Knowles J & Thiele L, 2008, *Quality assessment of Pareto set approximations*, pp. 373–404 in Branke J, Deb K, Miettinen K & Słowiński R (Eds), *Multiobjective optimization: Interactive and evolutionary approaches*, Springer, Berlin.

[190] Zitzler E & Künzli S, 2004, *Indicator-based selection in multiobjective search*, Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, pp. 832–842.

[191] Zitzler E & Thiele L, 1999, *Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach*, IEEE Transactions on Evolutionary Computation, **3(4)**, pp. 257–271.

[192] Zitzler E & Thiele L, 1998, *Multiobjective optimization using evolutionary algorithms — A comparative case study*, Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Amsterdam, pp. 292–301.

[193] Zitzler E, Thiele L, Laumanns M, Fonseca CM & Da Fonseca VG, 2003, *Performance assessment of multiobjective optimizers: An analysis and review*, IEEE Transactions on Evolutionary Computation, **7(2)**, pp. 117–132.

APPENDIX A

Further numerical results

This appendix contains the remainder of the results that were referenced in Chapter 7.

TABLE A.1: *Performance runs test cases for the UTRP DBMOSA approach.*

Test set	Test number	Instance	Test parameter
13	1	Mandl6	Long run
13	2	Mumford0	Long run
13	3	Mumford1	Long run
13	4	Mumford2	Long run
13	5	Mumford3	Long run
14	1	Mandl6	Static long run
14	2	Mumford0	Static long run
14	3	Mumford1	Static long run
14	4	Mumford2	Static long run
14	5	Mumford3	Static long run
15	1	Mandl6	Post optimisation
15	2	Mumford0	Post optimisation
15	3	Mumford1	Post optimisation
15	4	Mumford2	Post optimisation
15	5	Mumford3	Post optimisation
16	1	Mandl6	DP objective
16	2	Mumford0	DP objective
16	3	Mumford1	DP objective
16	4	Mumford2	DP objective
16	5	Mumford3	DP objective

TABLE A.2: *Average run times for various long runs conducted in this dissertation. The format of the times are hh:mm:ss, and the UTRP-IR indicates the UTRP with incremental redesign.*

Instance	UTRP DBMOSA long run	UTRP NSGA II long run	UTRP NSGA II uncapped run	UTRP-IR DBMOSA long run	UTRP-IR NSGA II long run
Mandl6	00:03:55	00:12:02	00:14:04	00:02:27	00:03:01
Mumford0	00:06:07	00:48:02	00:44:12	00:03:16	00:26:32
Mumford1	00:35:26	03:12:58	04:52:04	00:24:35	02:29:45
Mumford2	05:26:59	33:30:10	55:16:11	03:44:26	18:55:17
Mumford3	07:52:56	54:53:04	76:50:30	09:03:48	25:47:19

TABLE A.3: *Performance runs test cases for the UTRP NSGA II approach.*

Test set	Test number	Instance	Test parameter
30	1	Mandl6	Long run
30	2	Mumford0	Long run
30	3	Mumford1	Long run
30	4	Mumford2	Long run
30	5	Mumford3	Long run
31	1	Mandl6	Uncapped run
31	2	Mumford0	Uncapped run
31	3	Mumford1	Uncapped run
31	4	Mumford2	Uncapped run
31	5	Mumford3	Uncapped run
32	1	Mandl6	Static long run
32	2	Mumford0	Static long run
32	3	Mumford1	Static long run
32	4	Mumford2	Static long run
32	5	Mumford3	Static long run
33	1	Mandl6	DP objective
33	2	Mumford0	DP objective
33	3	Mumford1	DP objective
33	4	Mumford2	DP objective
33	5	Mumford3	DP objective

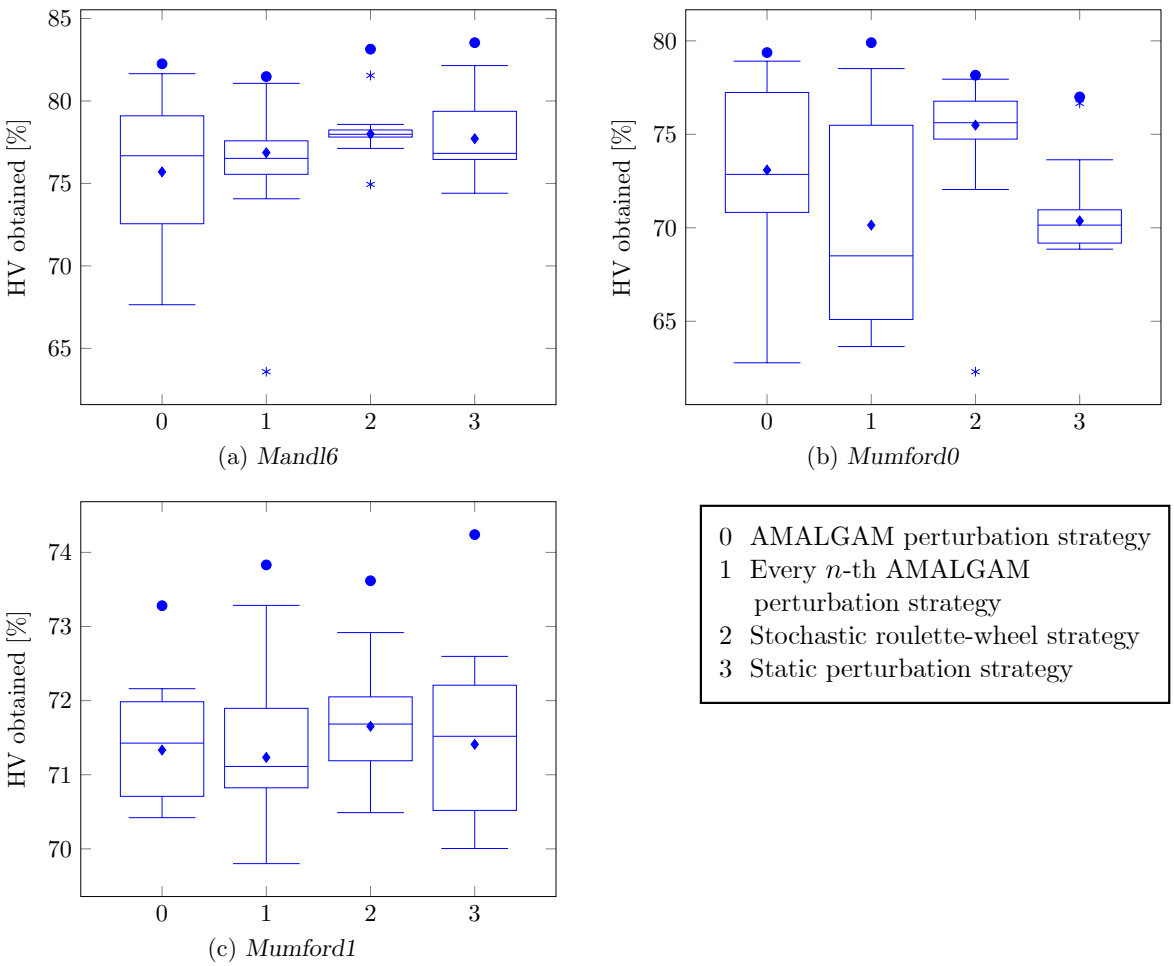


FIGURE A.1: *Dynamic perturbation analysis (Test set 2) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the perturbation management strategy (according to §6.1.6), as indicated on the horizontal axes.*

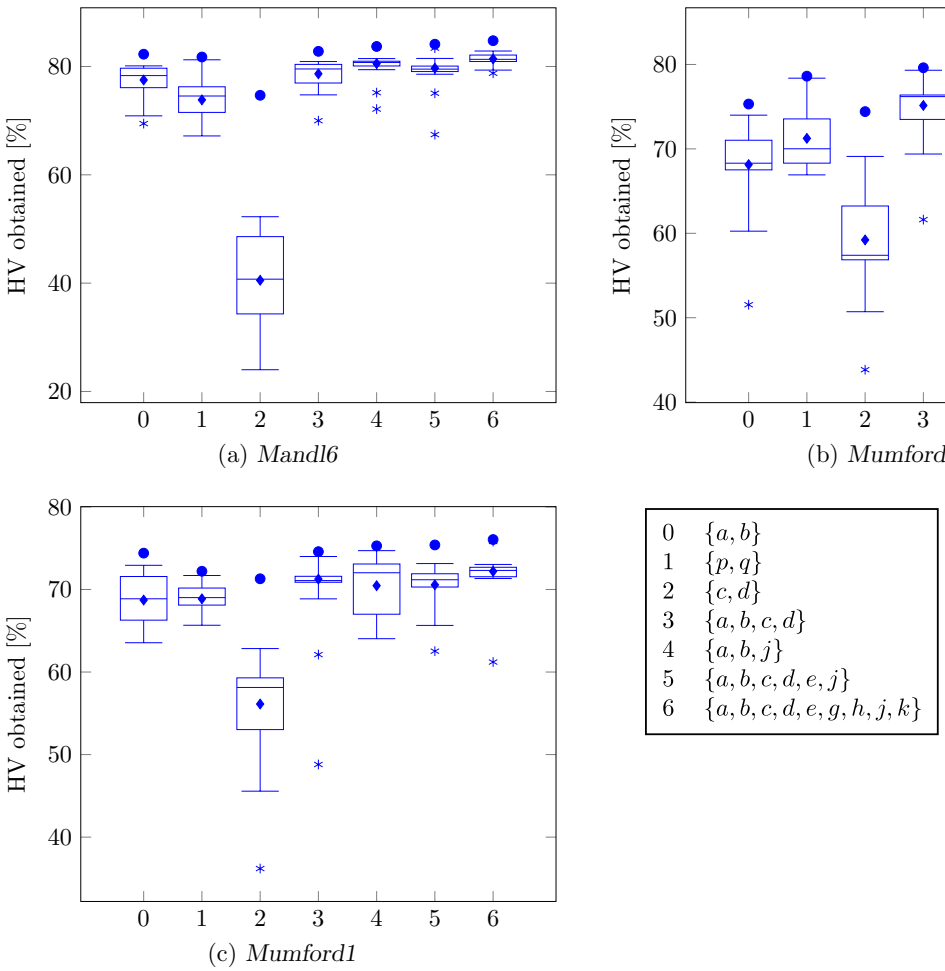


FIGURE A.2: *LLH combinations analysis (Test set 3) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the LLH combinations incorporated into the search, as indicated on the horizontal axes.*

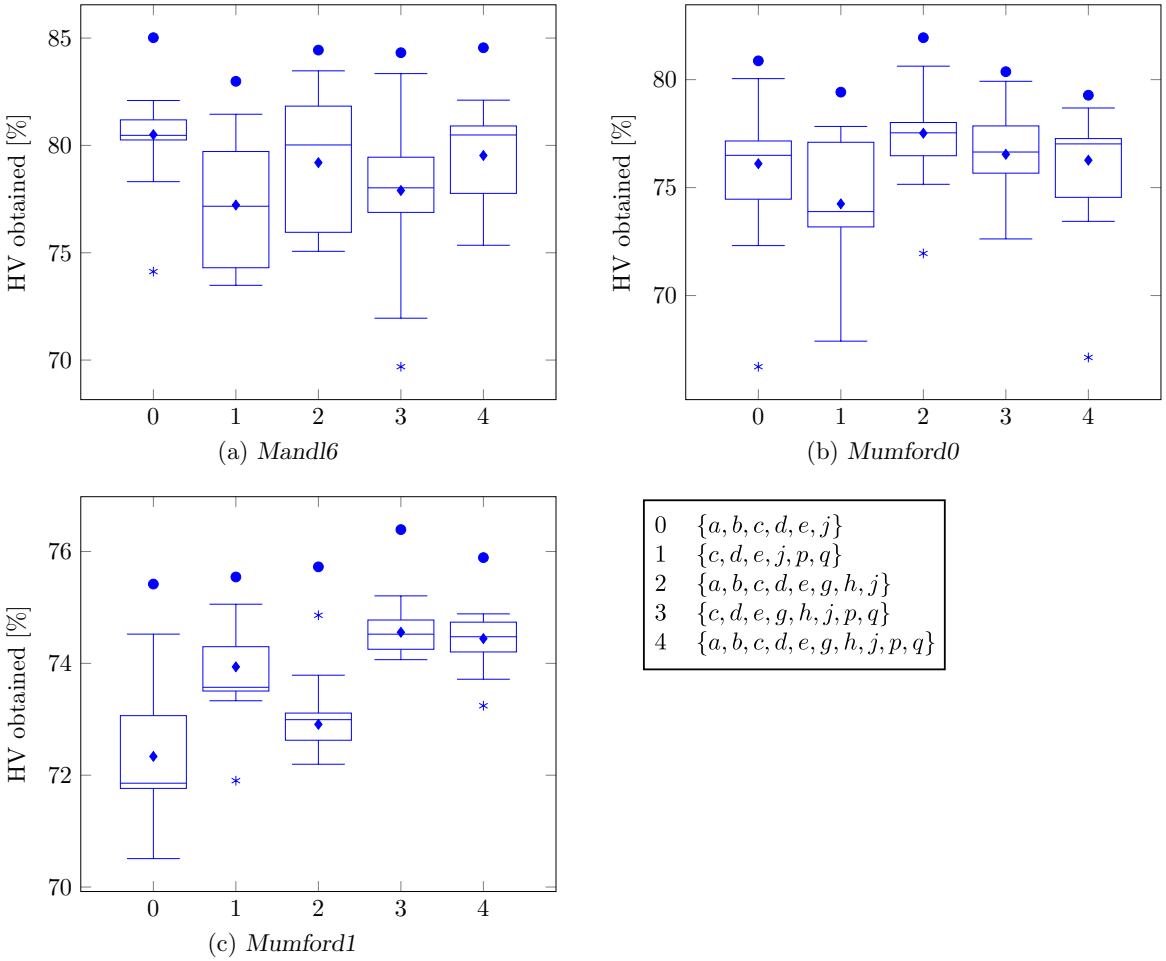
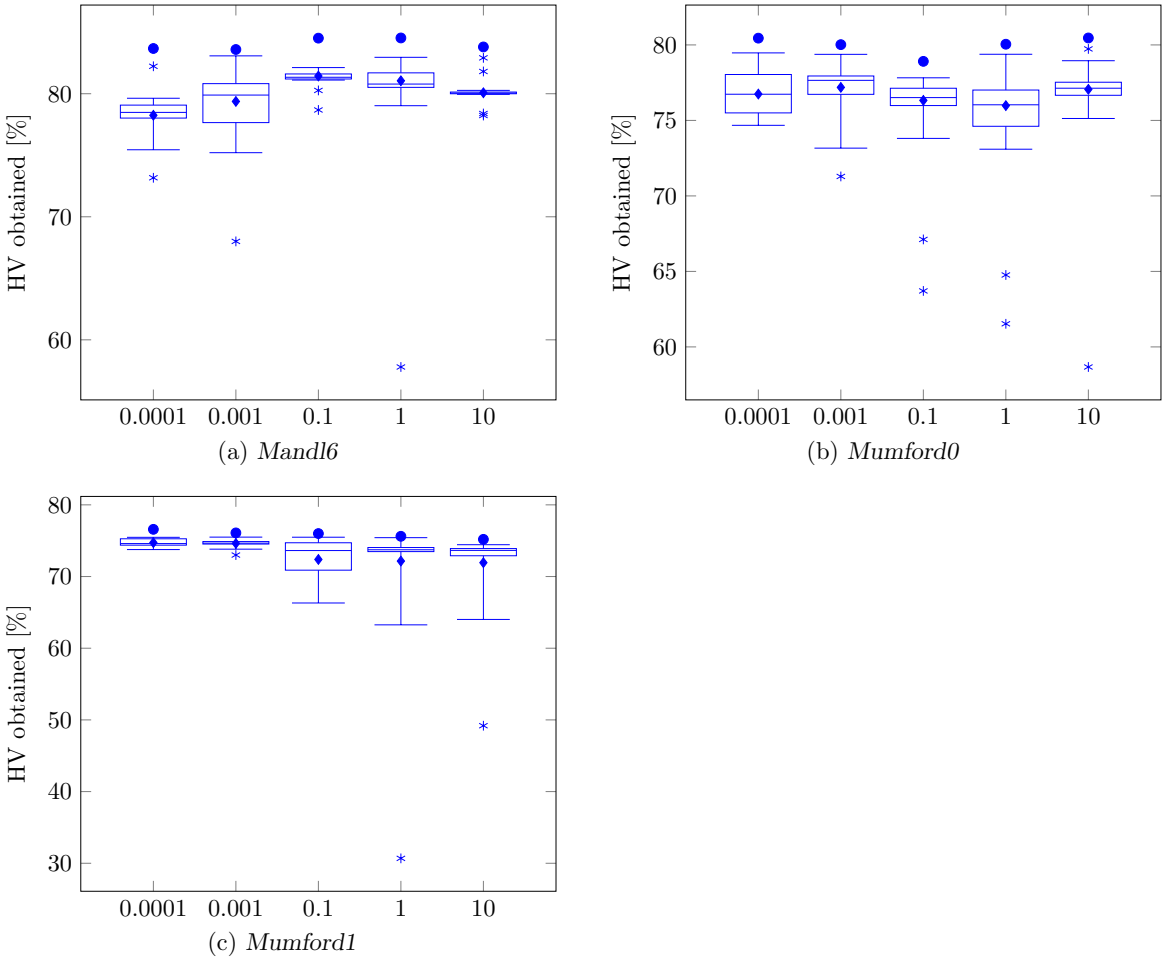


FIGURE A.3: *LLH combinations analysis (Test set 4) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the LLH combinations incorporated into the search, as indicated on the horizontal axes.*



(a) *Mandl6* (b) *Mumford0*

(c) *Mumford1*

FIGURE A.4: *Initial temperature design analysis (Test set 5) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the initial temperature as indicated on the horizontal axes.*

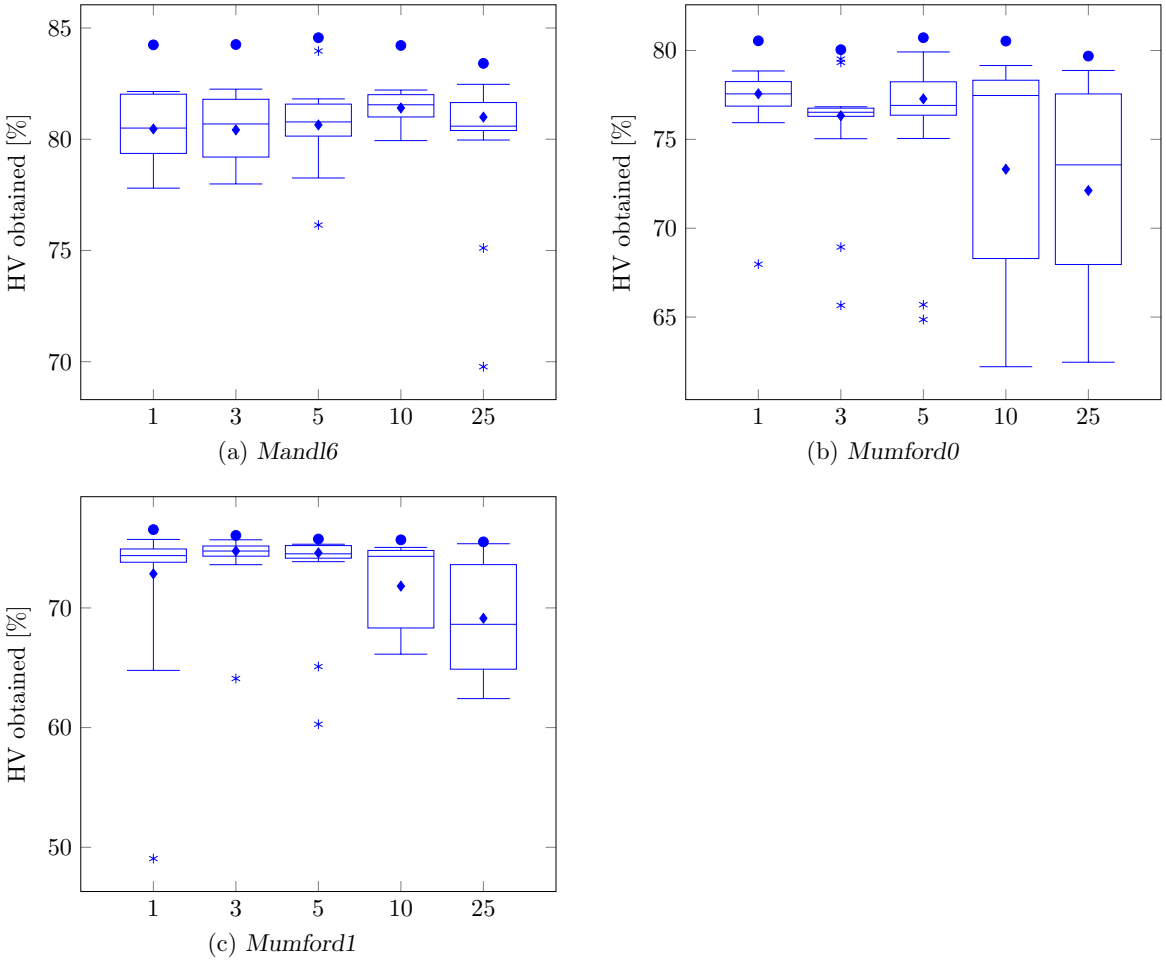
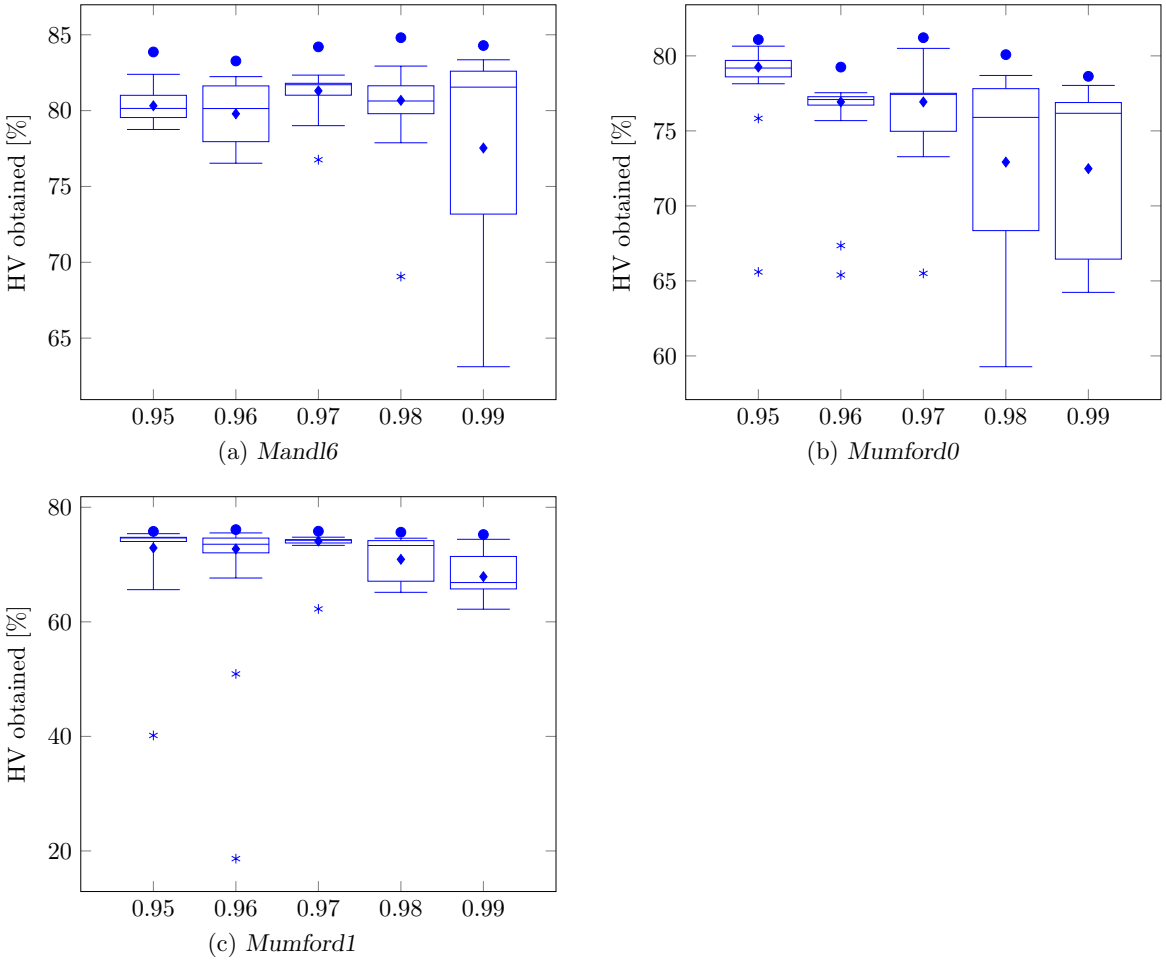


FIGURE A.5: *Maximum reheatings design analysis (Test set 6) results for 10 separate runs of the DB-MOSA algorithm, with box plots of the HV obtained when varying the maximum number of reheatings as indicated on the horizontal axes.*

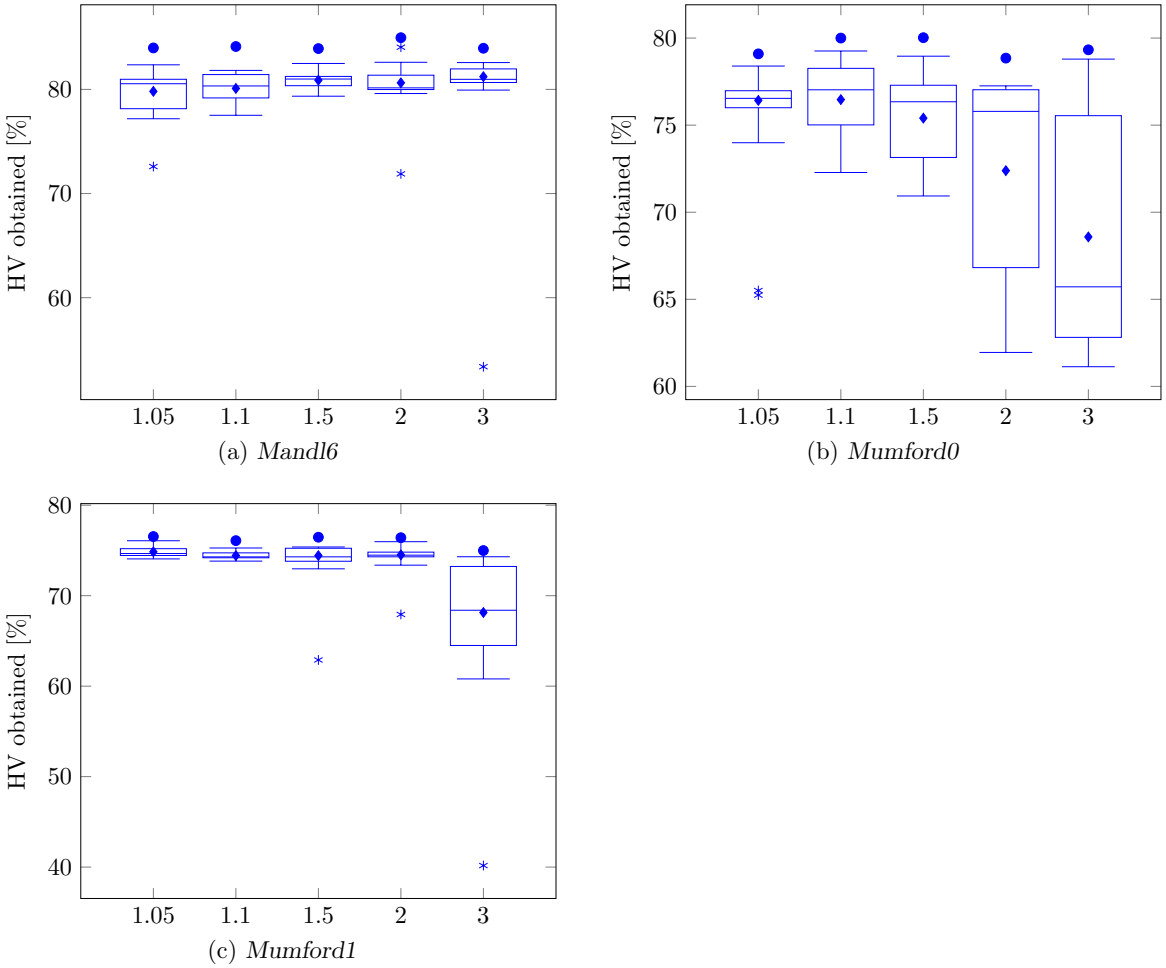


(a) *Mandl6*

(b) *Mumford0*

(c) *Mumford1*

FIGURE A.6: *Cooling rate design analysis (Test set 7) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the cooling rate as indicated on the horizontal axes.*

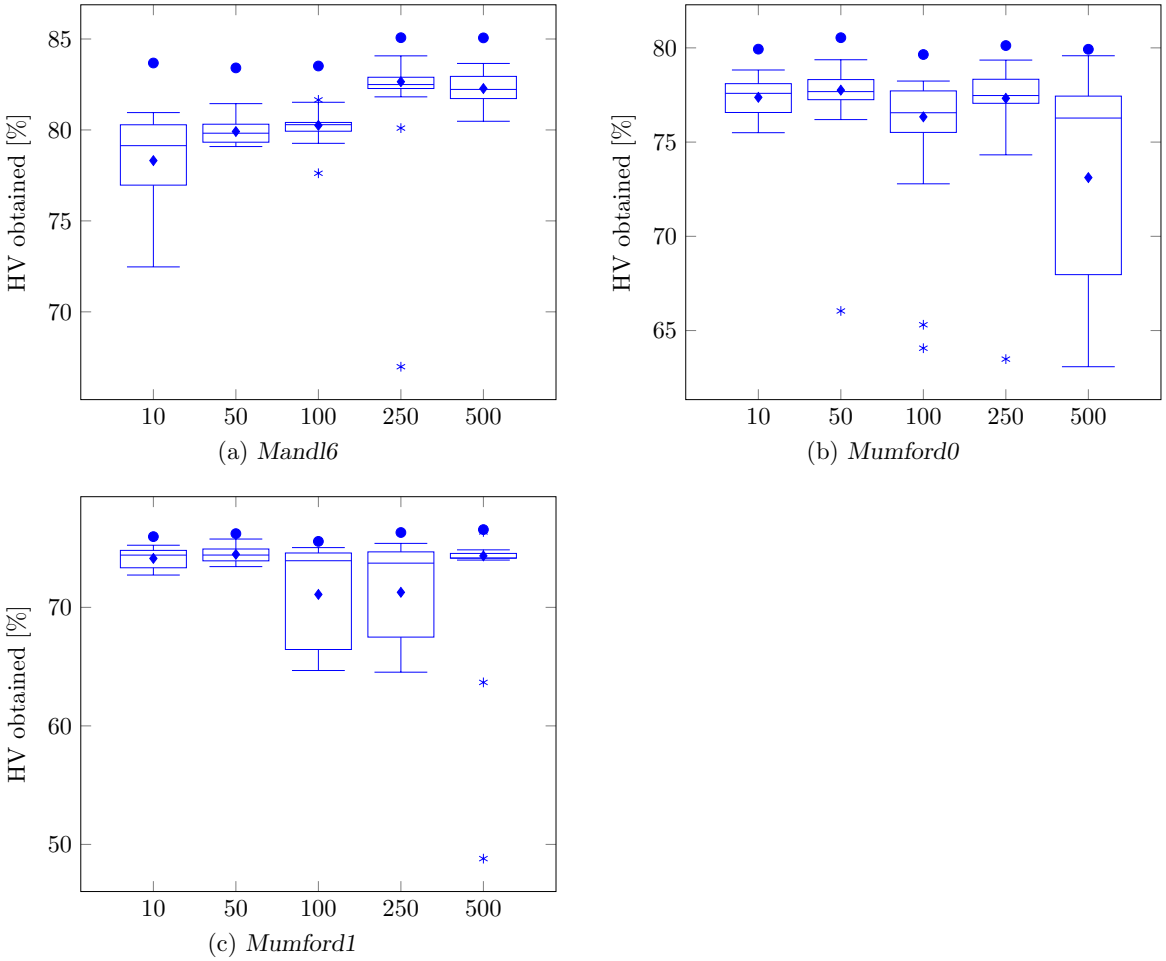


(a) *Mandl6*

(b) *Mumford0*

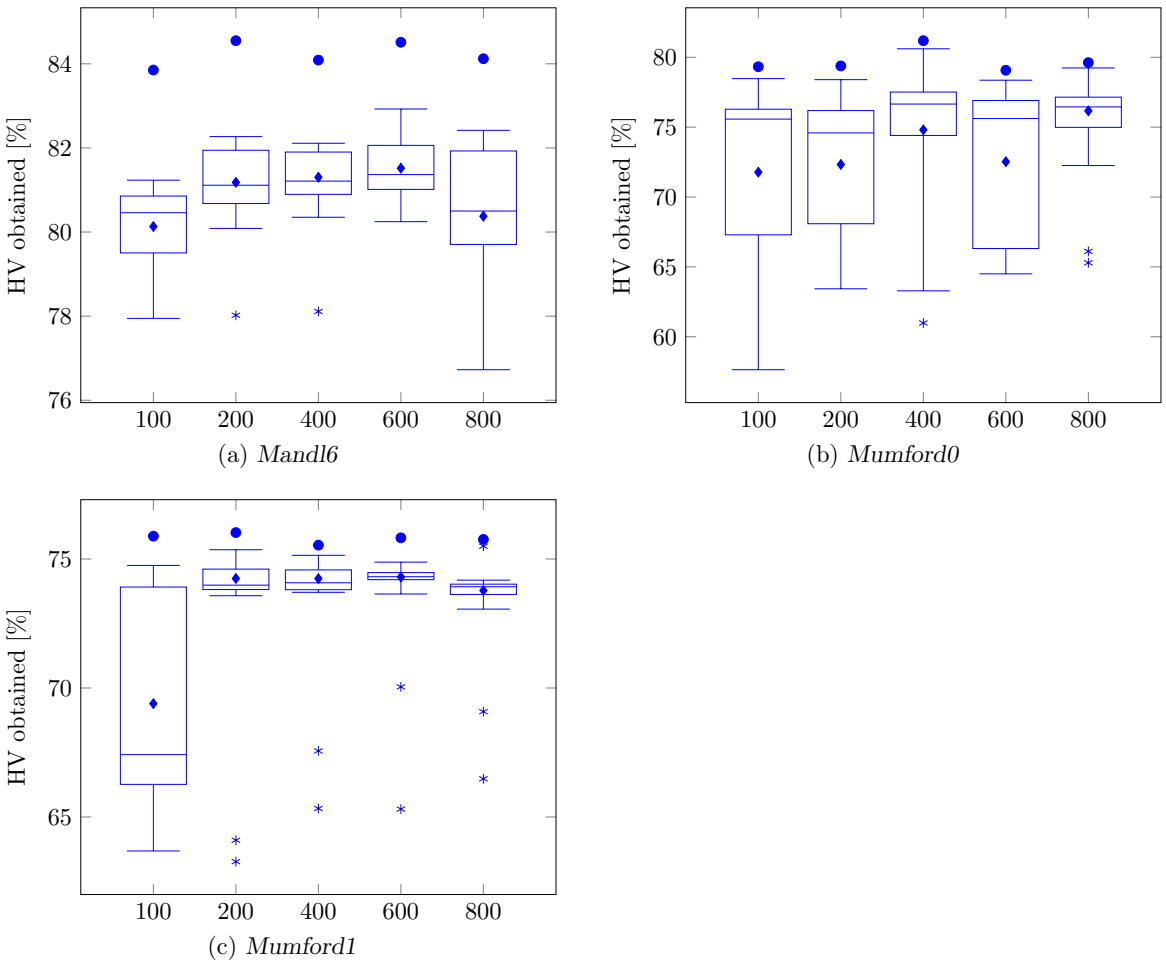
(c) *Mumford1*

FIGURE A.7: *Reheating rate design analysis (Test set 8) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the reheating rate as indicated on the horizontal axes.*



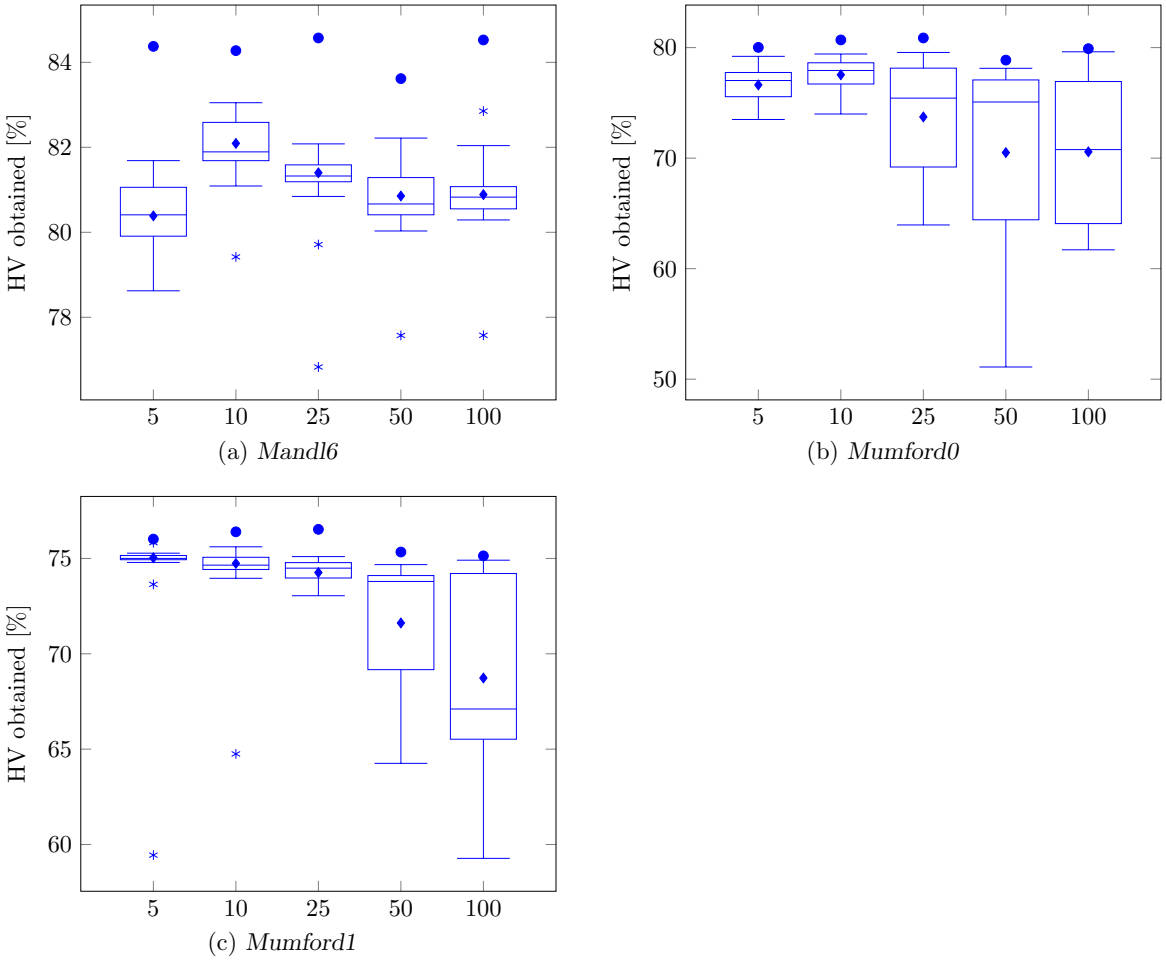
(a) *Mandl6* (b) *Mumford0* (c) *Mumford1*

FIGURE A.8: *Maximum iterations per epoch design analysis (Test set 9) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the maximum number of iterations per epoch as indicated on the horizontal axes.*



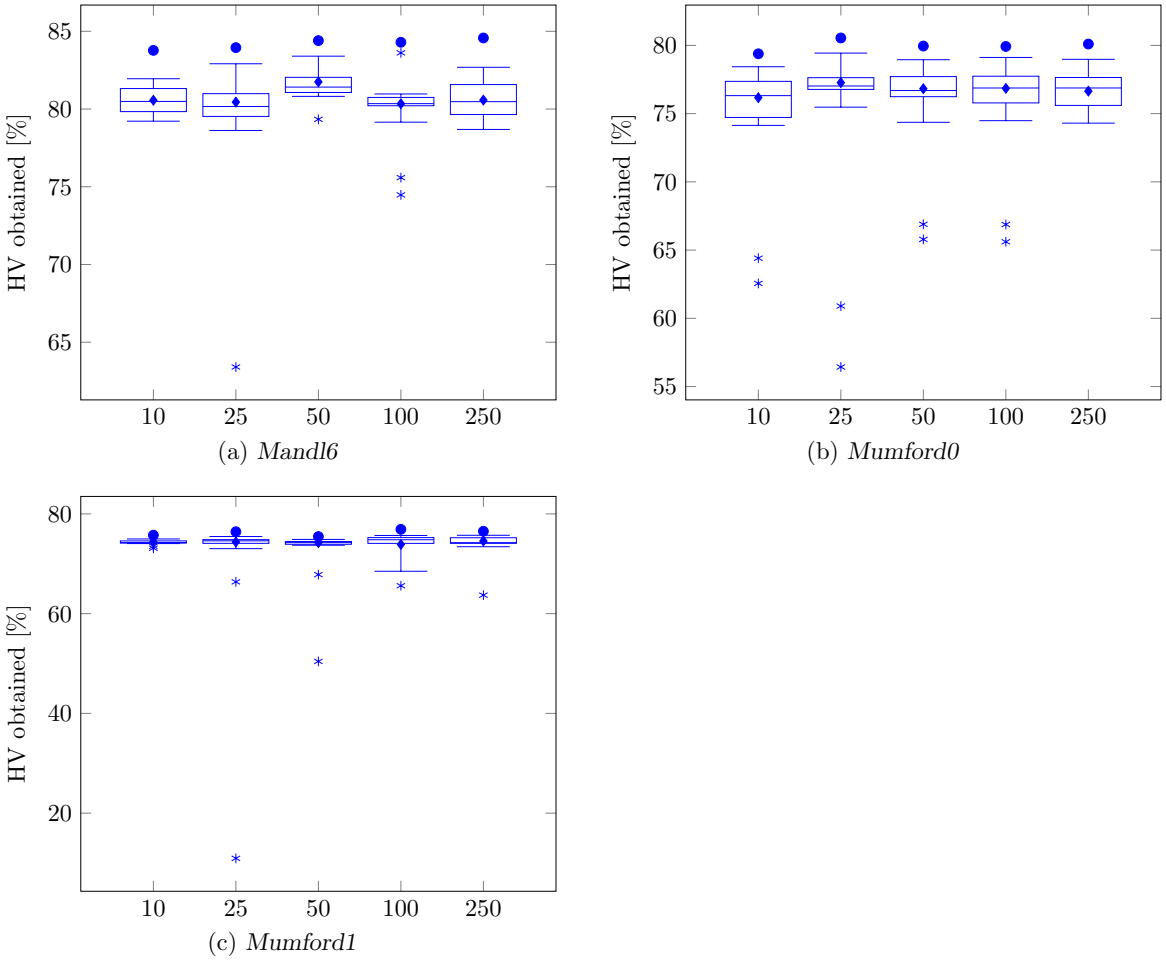
(a) *Mandl6* (b) *Mumford0* (c) *Mumford1*

FIGURE A.9: *Maximum poor epochs design analysis (Test set 10) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the maximum number of poor epochs as indicated on the horizontal axes.*



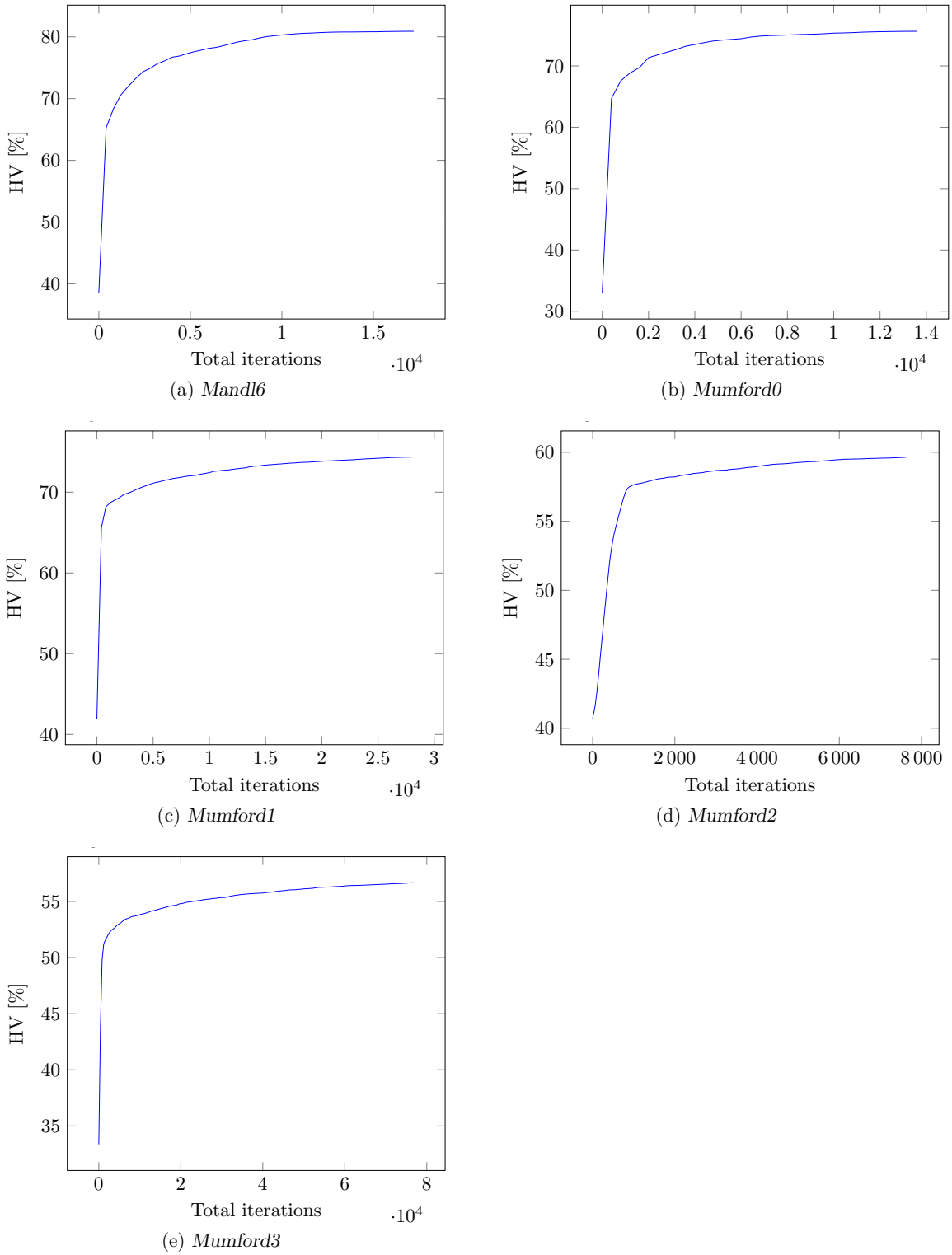
(a) *Mandl6* (b) *Mumford0* (c) *Mumford1*

FIGURE A.10: *Minimum accepts design analysis (Test set 11) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the maximum number of accepts as indicated on the horizontal axes.*



(a) *Mandl6* (b) *Mumford0* (c) *Mumford1*

FIGURE A.11: *Maximum attempts design analysis (Test set 12) results for 10 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the maximum number of attempts as indicated on the horizontal axes.*



(a) *Mandl6* (b) *Mumford0*

(c) *Mumford1* (d) *Mumford2*

(e) *Mumford3*

FIGURE A.12: *Mean HV analysis (Test set 13) results for 30 separate long runs of the DBMOSA, with line graphs of the HV obtained over the total number of iterations on the horizontal axes.*

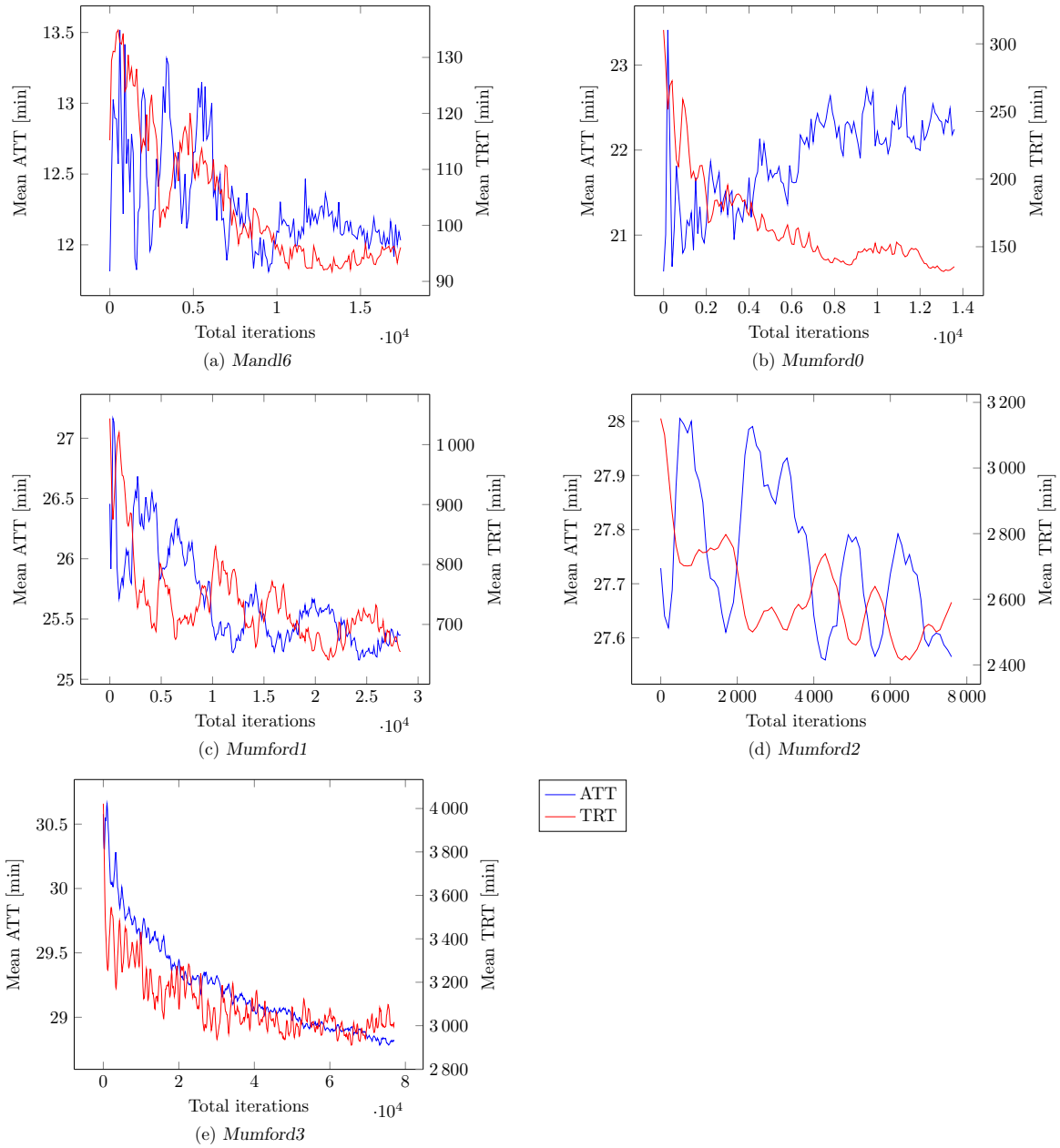
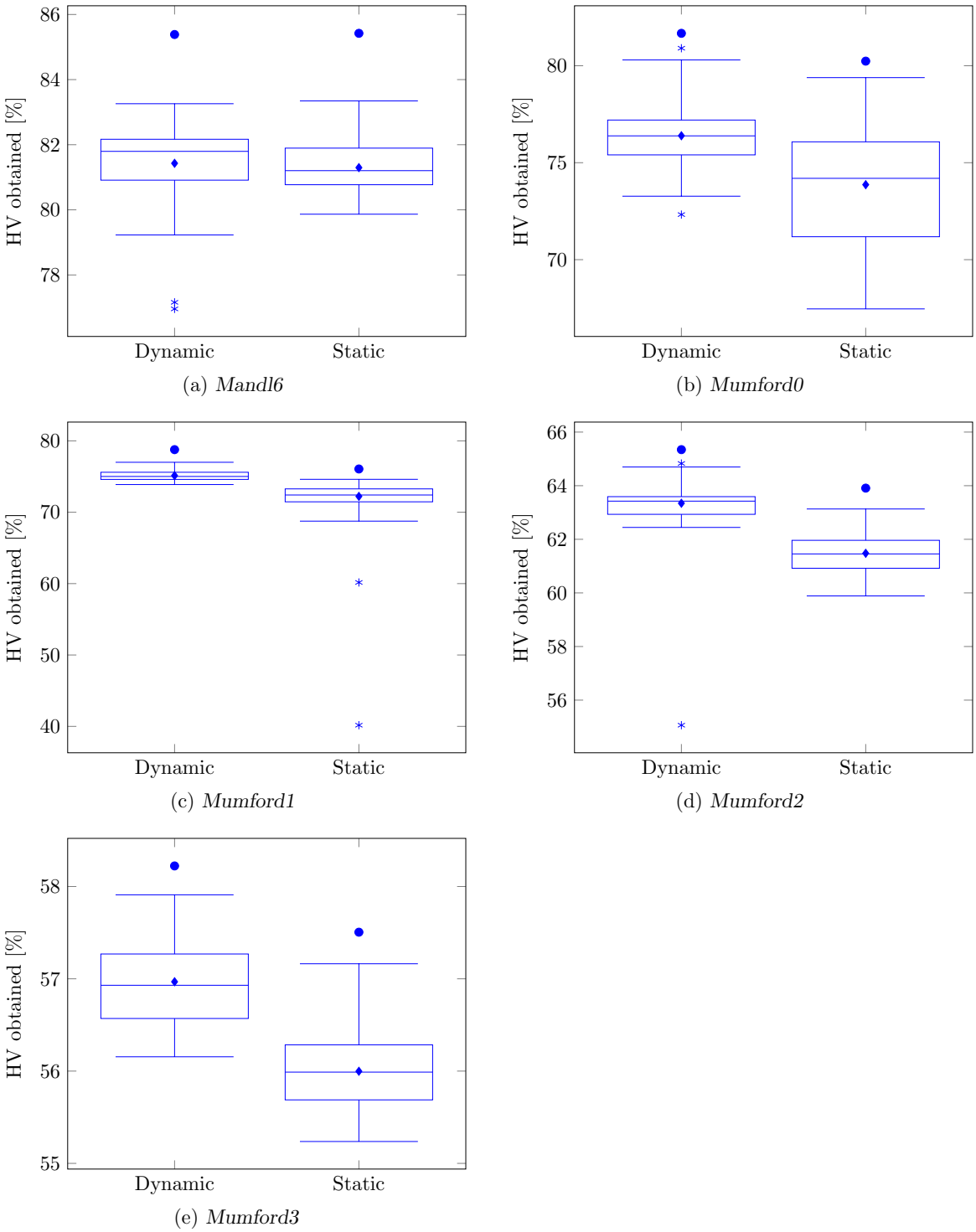


FIGURE A.13: *Mean objective function values analysis (Test set 13) results for 30 separate long runs of the DBMOSA algorithm, with line graphs of the mean ATT and TRT obtained, plotted along the left and right vertical axes, respectively, over the total number of iterations on the horizontal axes.*



(a) *Mandl6*

(b) *Mumford0*

(c) *Mumford1*

(d) *Mumford2*

(e) *Mumford3*

FIGURE A.14: *Dynamic versus static perturbations analysis (Test set 14) results for 30 separate runs of the DBMOSA algorithm, with box plots of the HV obtained when varying the dynamic or static nature of the mutation strategy employed as indicated on the horizontal axes. The dynamic mutation management strategy was described in §6.1.6.*

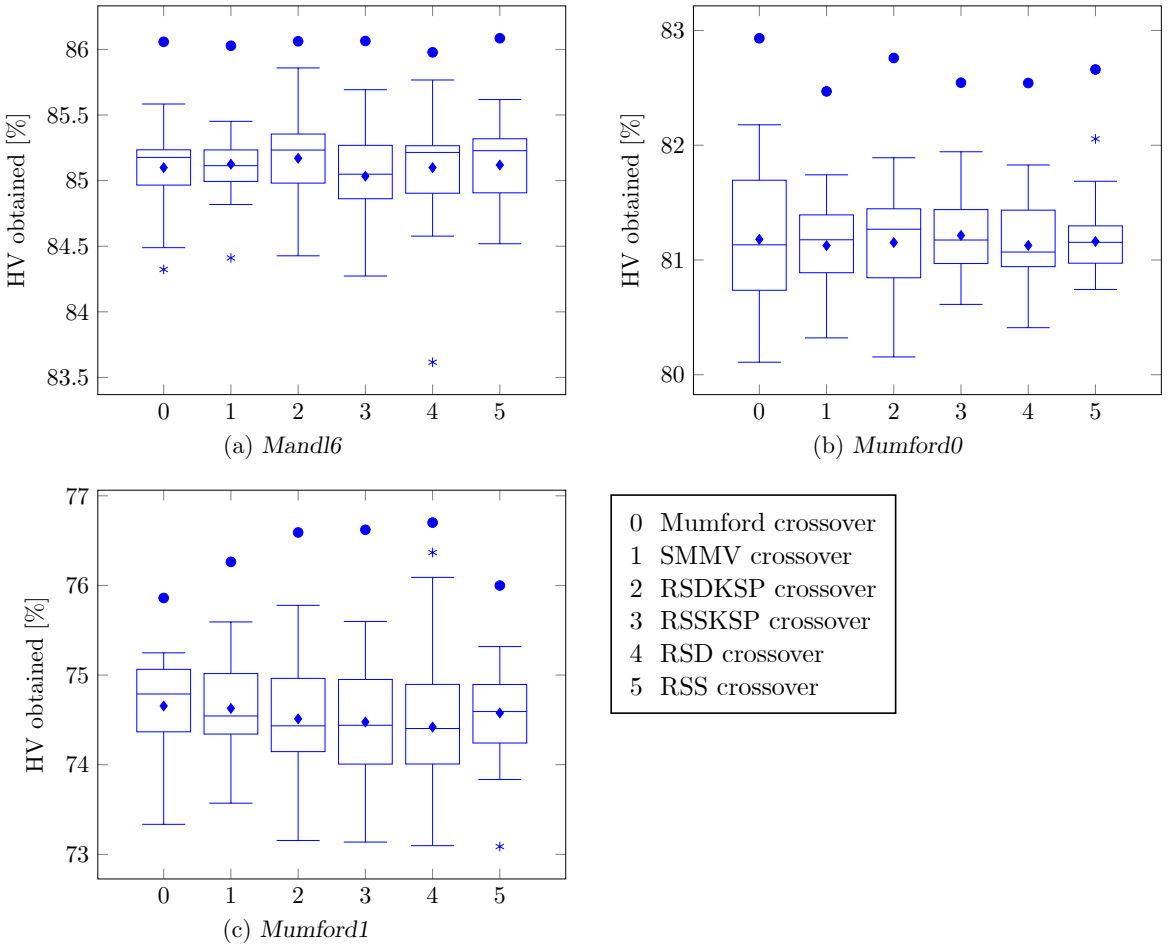
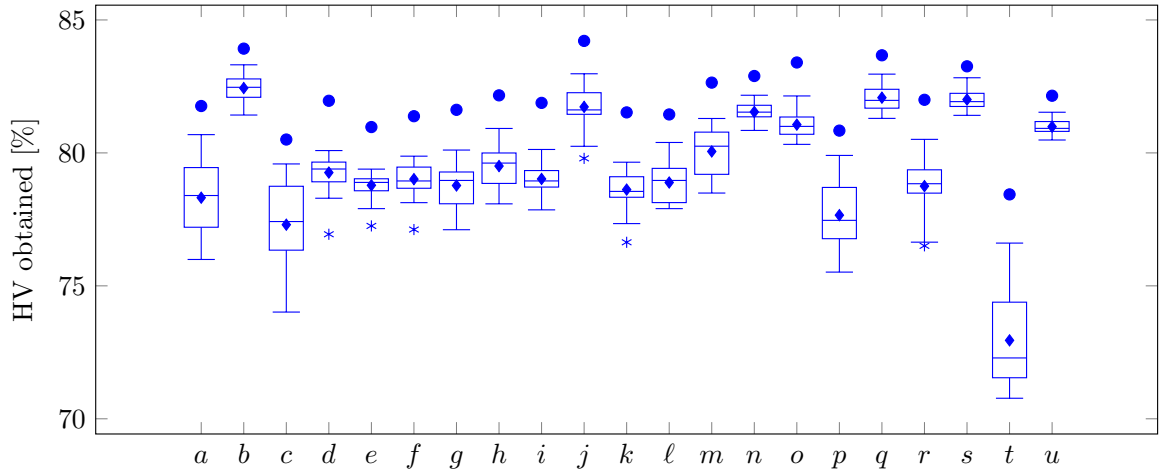
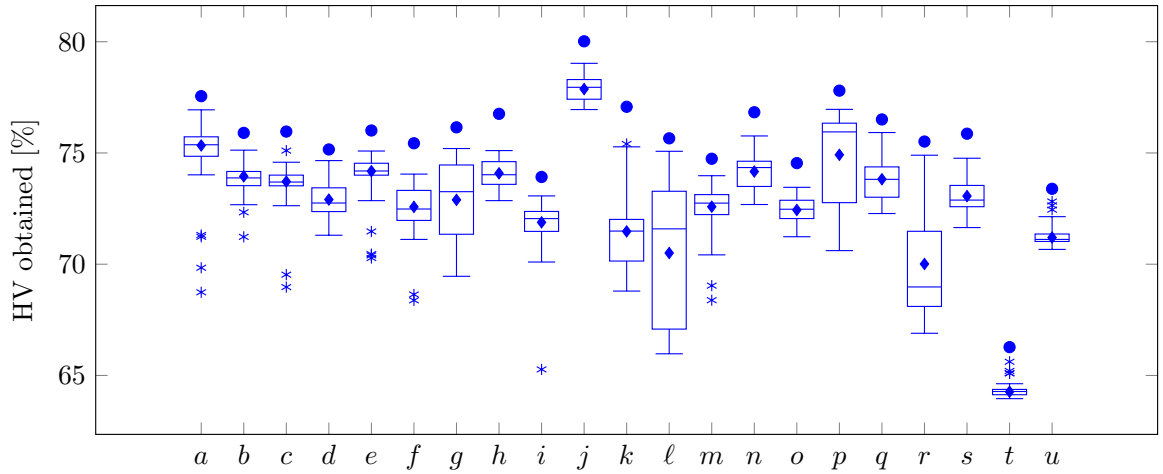


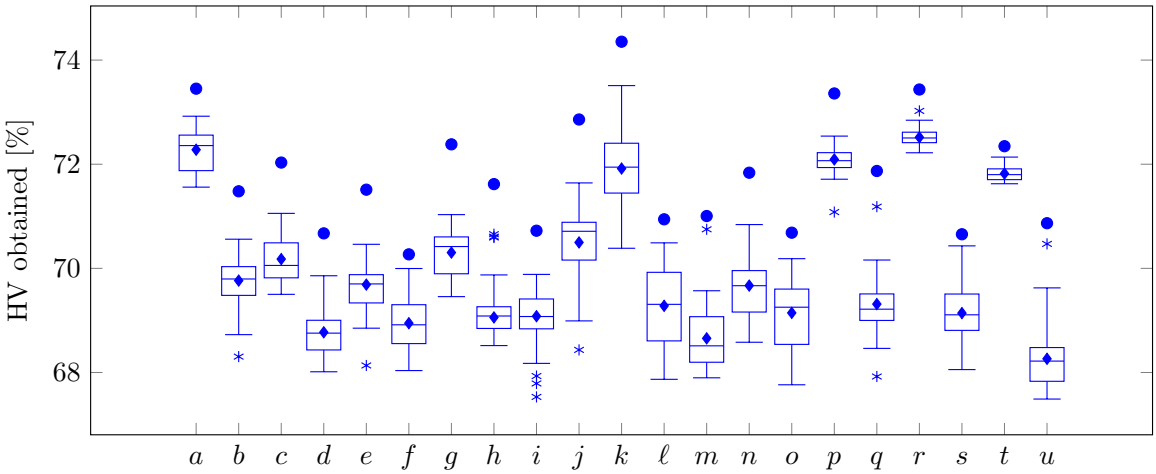
FIGURE A.15: *Crossover operator analysis (Test set 22) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying the crossover operator employed, as described in §6.1.8.*



(a) *Mandl6*

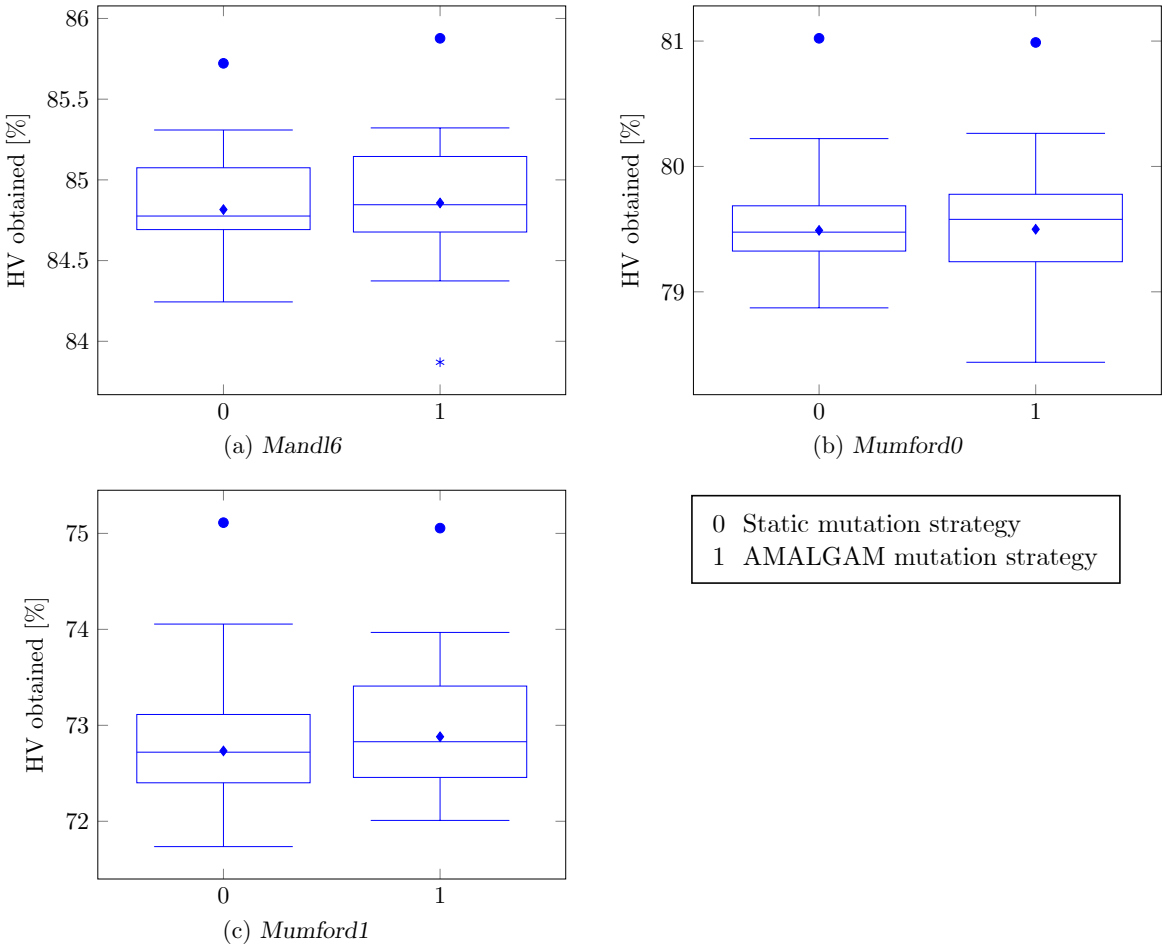


(b) *Mumford0*



(c) *Mumford1*

FIGURE A.16: *Mutation operator analysis (Test set 23) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying the mutation operator employed, as described in §6.1.5. The letters on the horizontal axes correspond to the LLH numbers in Table 6.3.*

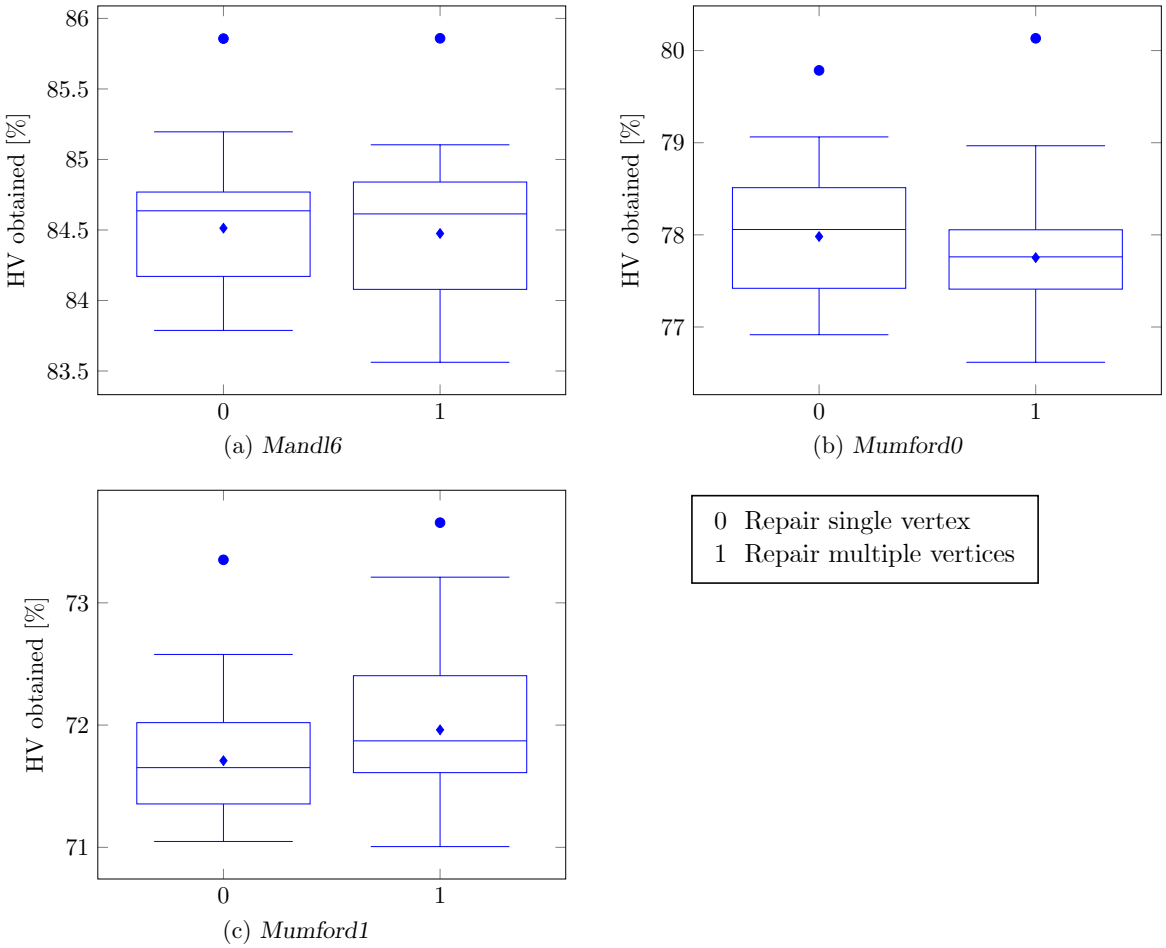


(a) *Mandl6*

(b) *Mumford0*

(c) *Mumford1*

FIGURE A.17: *Dynamic mutations analysis (Test set 24) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying whether static mutation probabilities or dynamic mutation probabilities (according to the description in §6.1.6), as indicated on the horizontal axes.*



(a) *Mandl6*

(b) *Mumford0*

(c) *Mumford1*

FIGURE A.18: *Repair analysis (Test set 25) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying whether single vertex repair or multiple vertices repair used, as indicated on the horizontal axes.*

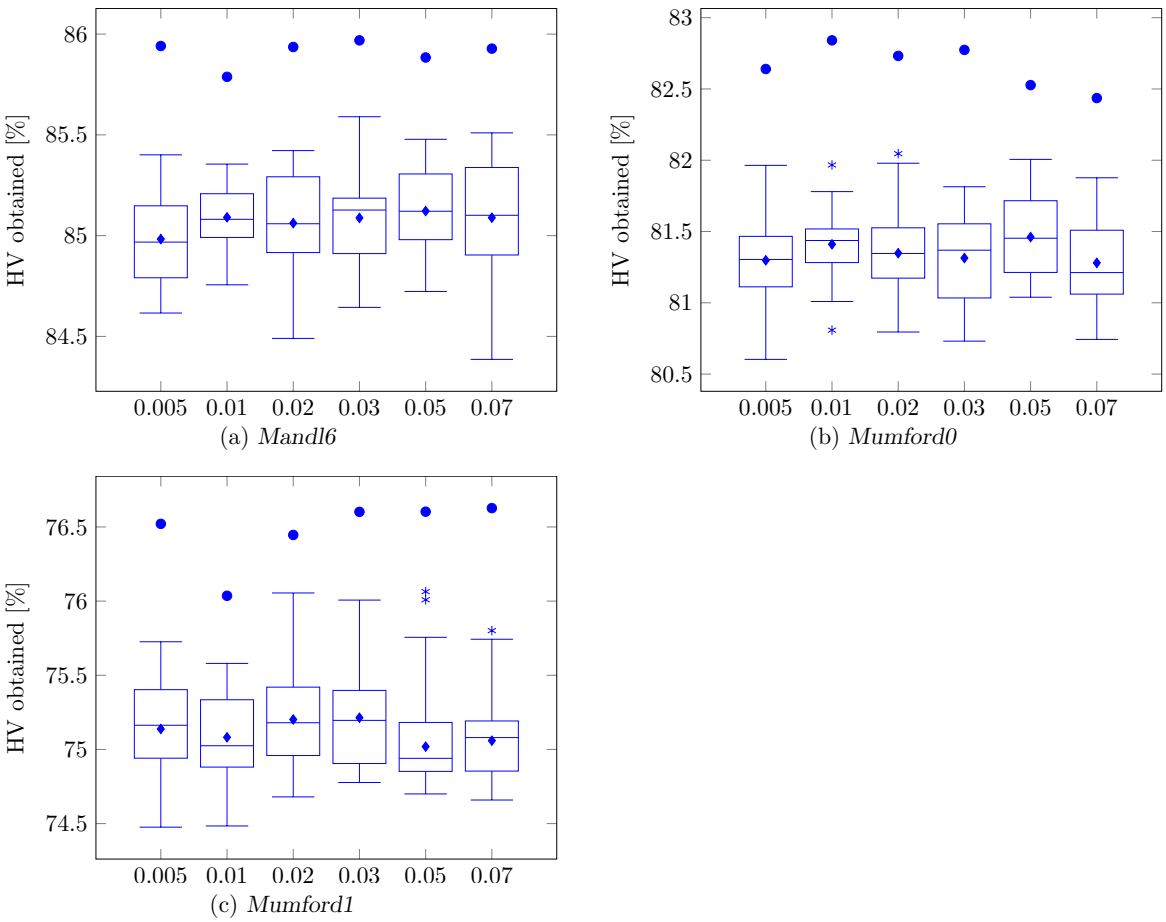


FIGURE A.19: *Selection probability threshold analysis (Test set 26) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying the selection probability as indicated on the horizontal axes.*

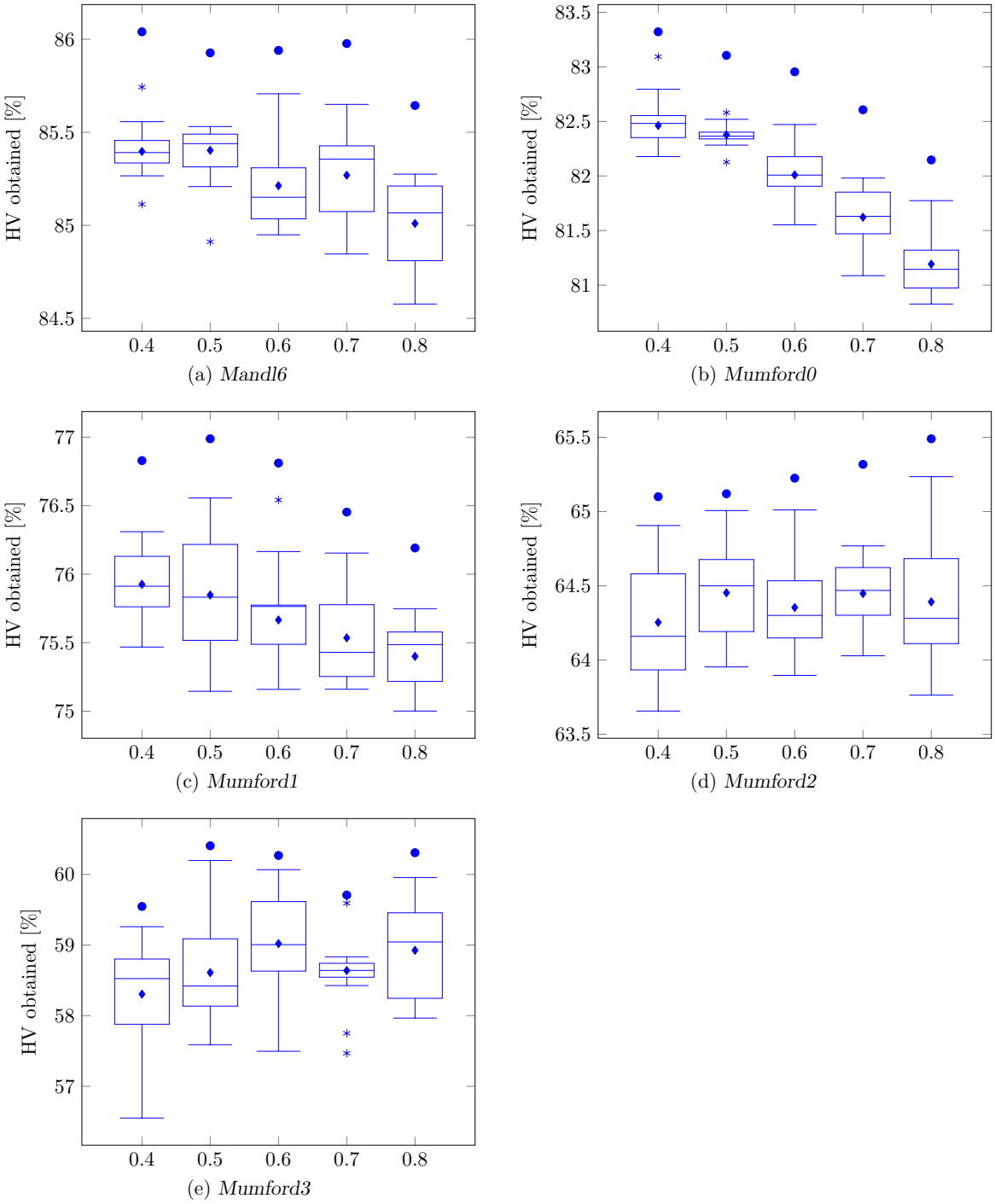


FIGURE A.20: *Crossover probability analysis (Test set 27) results for 10 separate runs of the NSGA II, with box plots of the HV obtained when varying the crossover probability as indicated on the horizontal axes.*

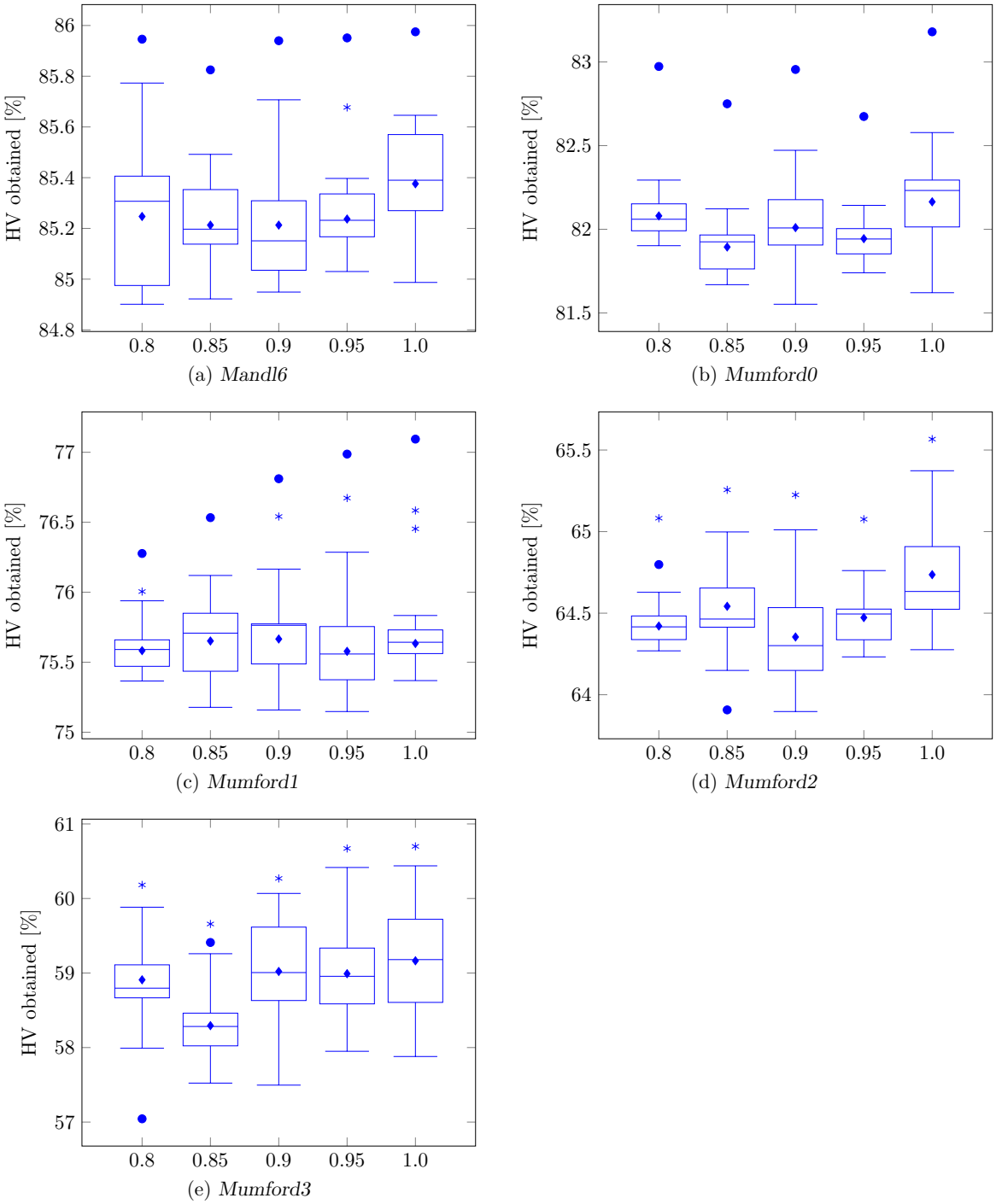


FIGURE A.21: *Mutation probability analysis (Test set 28) results for 10 separate runs of the NSGA II, with box plots of the HV obtained when varying the mutation probability as indicated on the horizontal axes.*

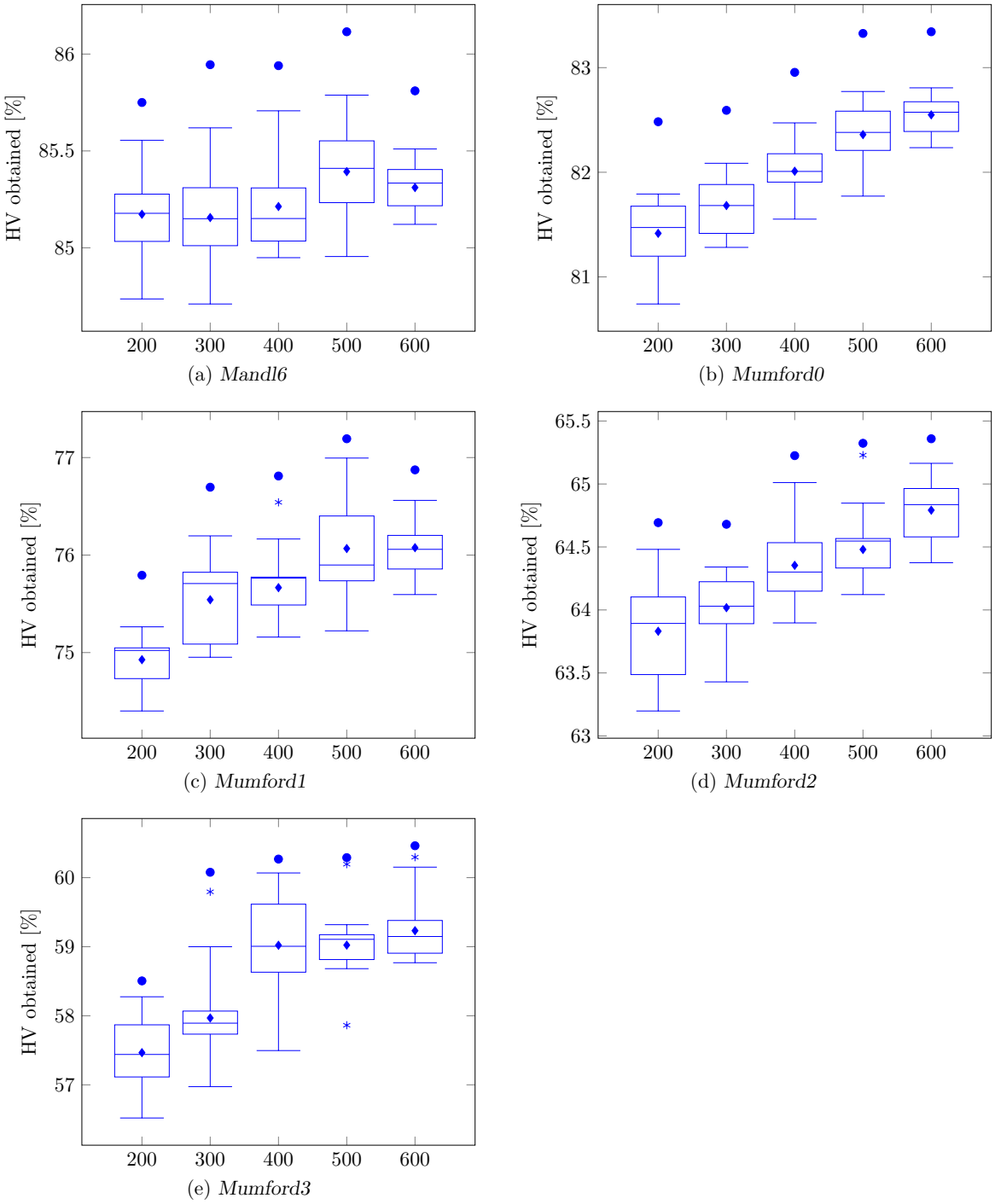
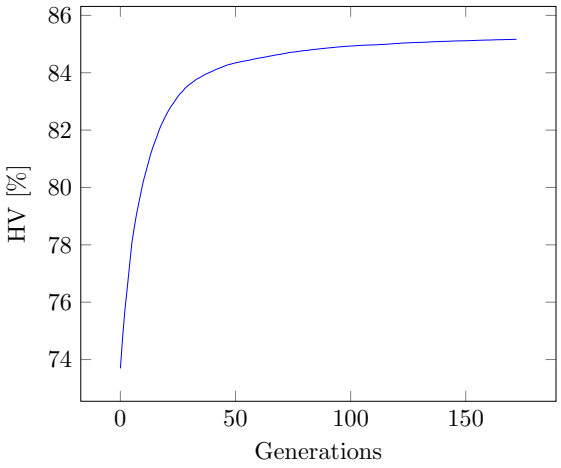
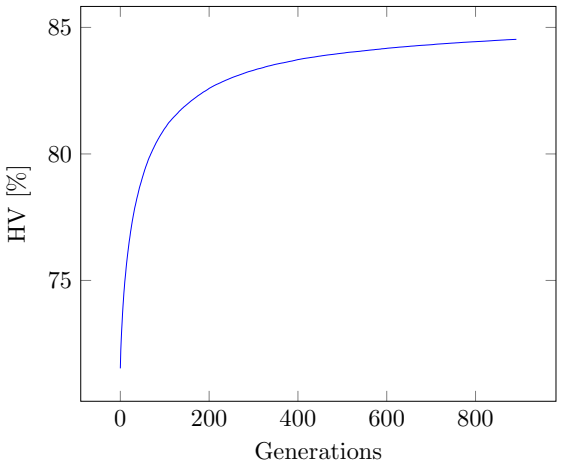


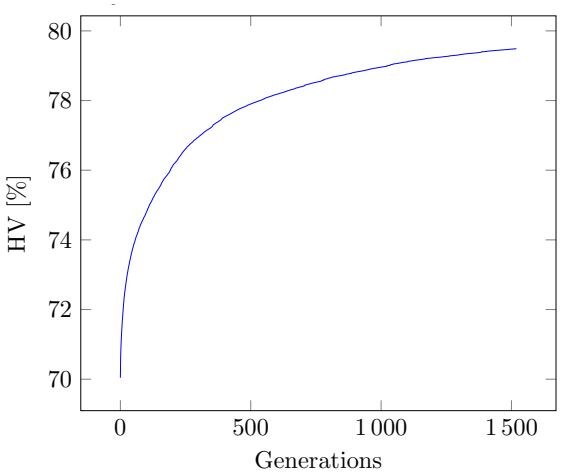
FIGURE A.22: *Population size analysis (Test set 29) results for 10 separate runs of the NSGA II, with box plots of the HV obtained when varying the population size as indicated on the horizontal axes.*



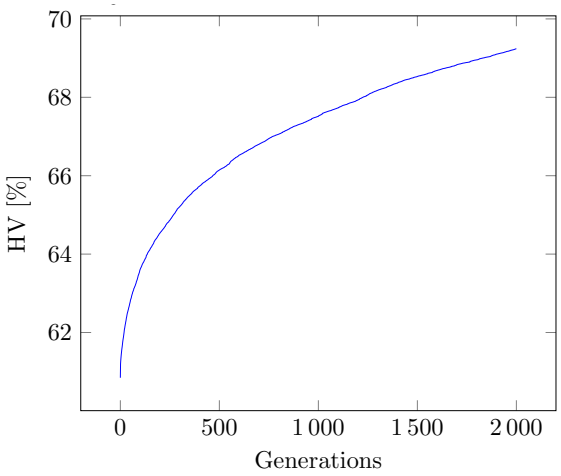
(a) *Mandl6*



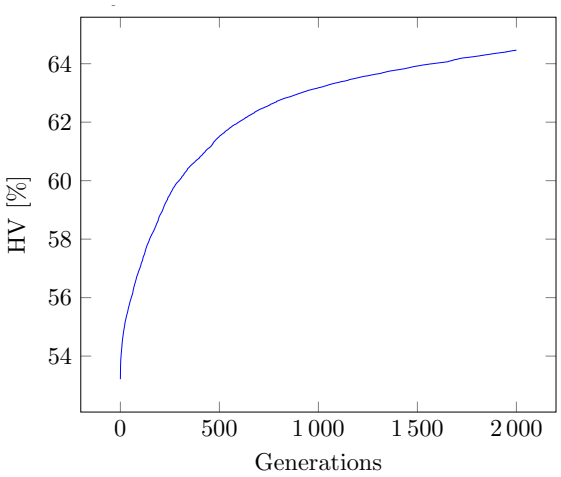
(b) *Mumford0*



(c) *Mumford1*

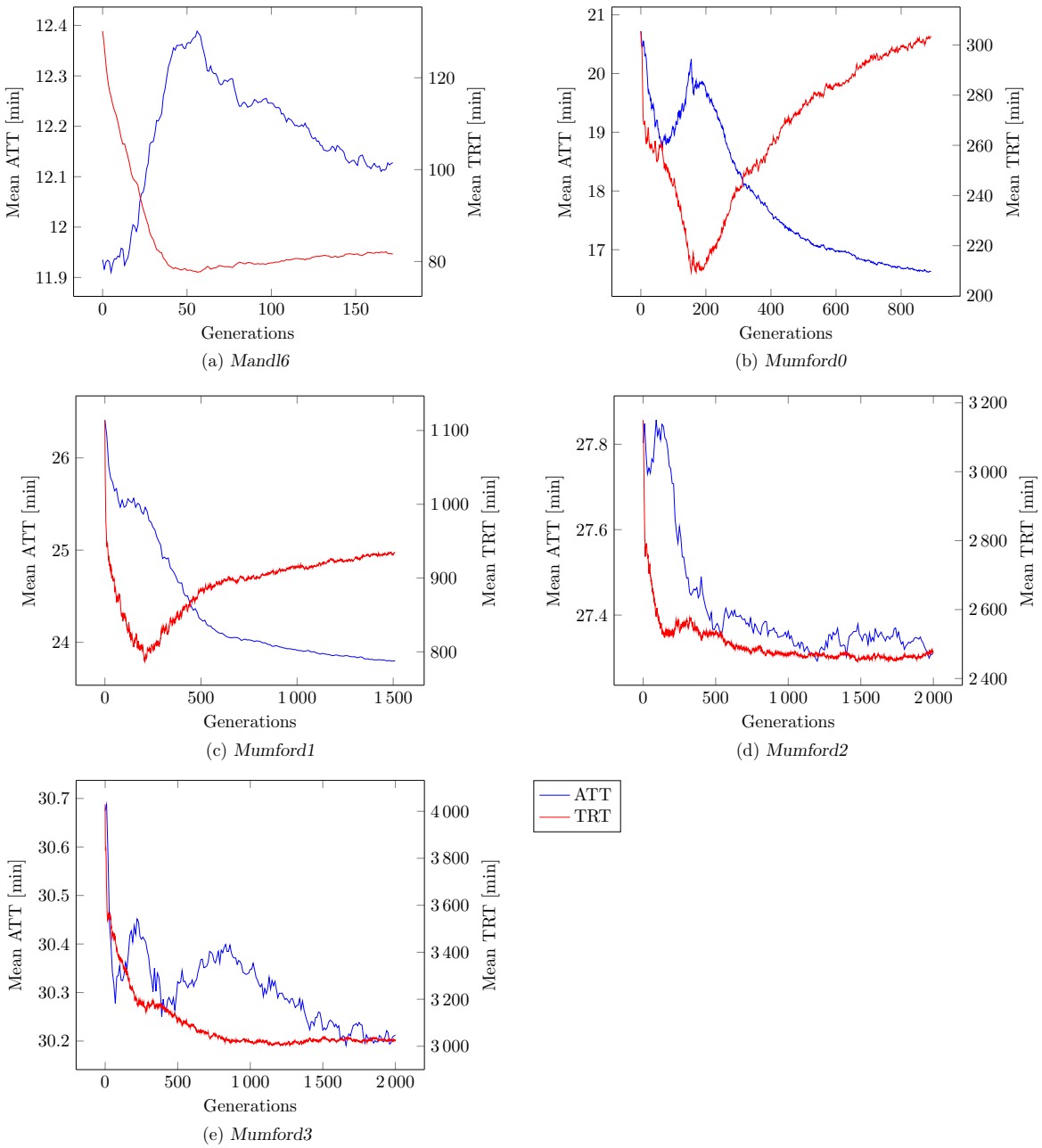


(d) *Mumford2*



(e) *Mumford3*

FIGURE A.23: *Mean HV analysis (Test set 30) results for 20 separate long runs of the NSGA II, with line graphs of the HV obtained over the number of generations on the horizontal axes.*



(a) *Mandl6*

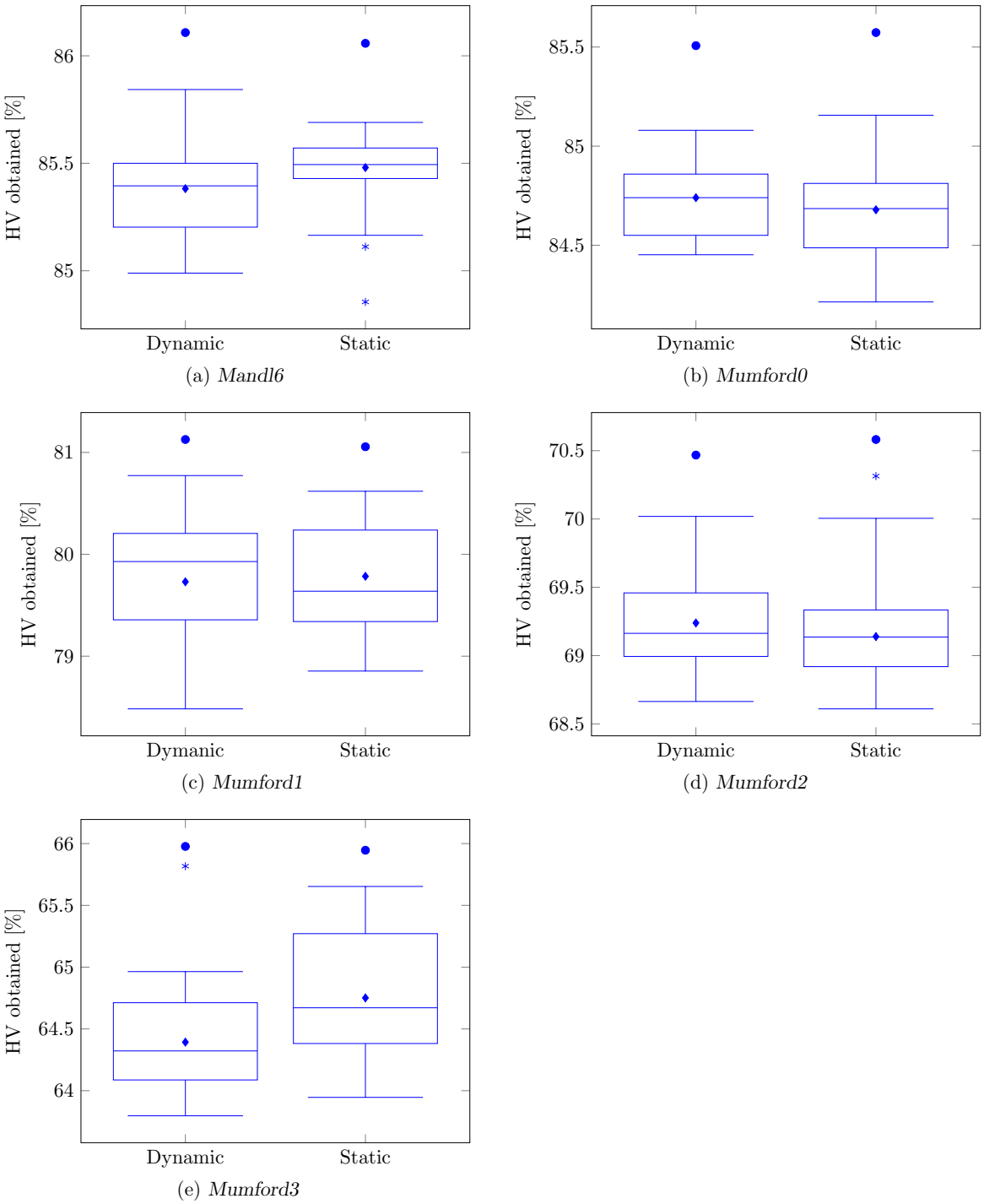
(b) *Mumford0*

(c) *Mumford1*

(d) *Mumford2*

(e) *Mumford3*

FIGURE A.24: *Mean objective function values analysis (Test set 30) results for 20 separate long runs of the NSGA II, with line graphs of the mean ATT and TRT obtained, plotted along the left and right vertical axes, respectively, over the number of generations on the horizontal axes.*

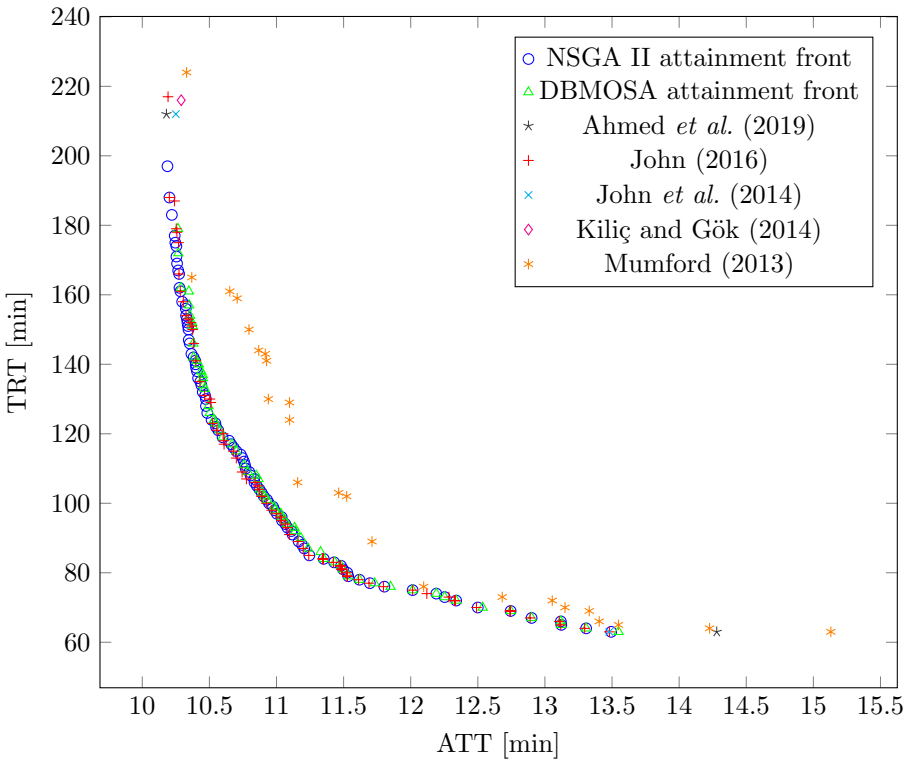


(a) *Mandl6* (b) *Mumford0*

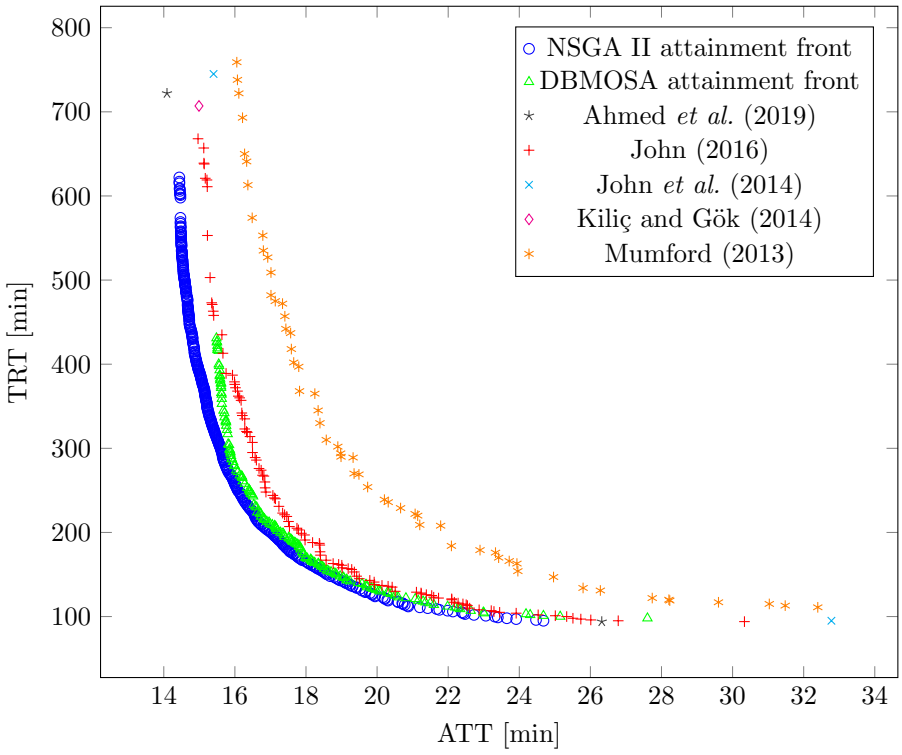
(c) *Mumford1* (d) *Mumford2*

(e) *Mumford3*

FIGURE A.25: *Dynamic versus static mutations analysis (Test set 32) results for 20 separate runs of the NSGA II, with box plots of the HV obtained when varying the dynamic or static nature of the mutation strategy employed as indicated on the horizontal axes. The dynamic mutation management strategy was described in §6.1.6.*

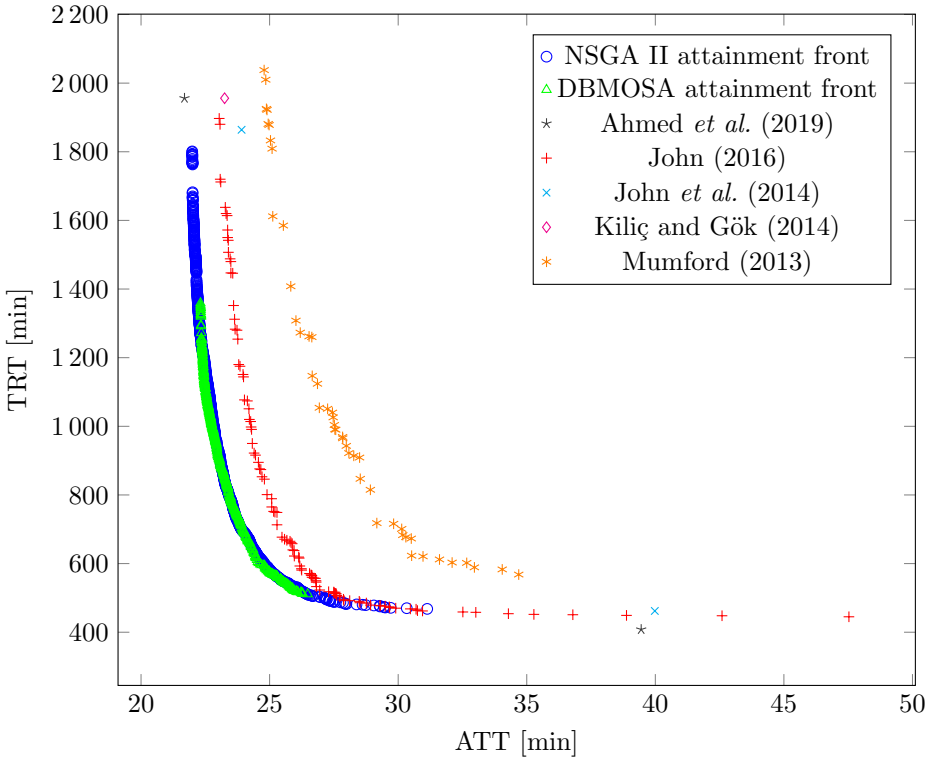


(a) *Mandl6*

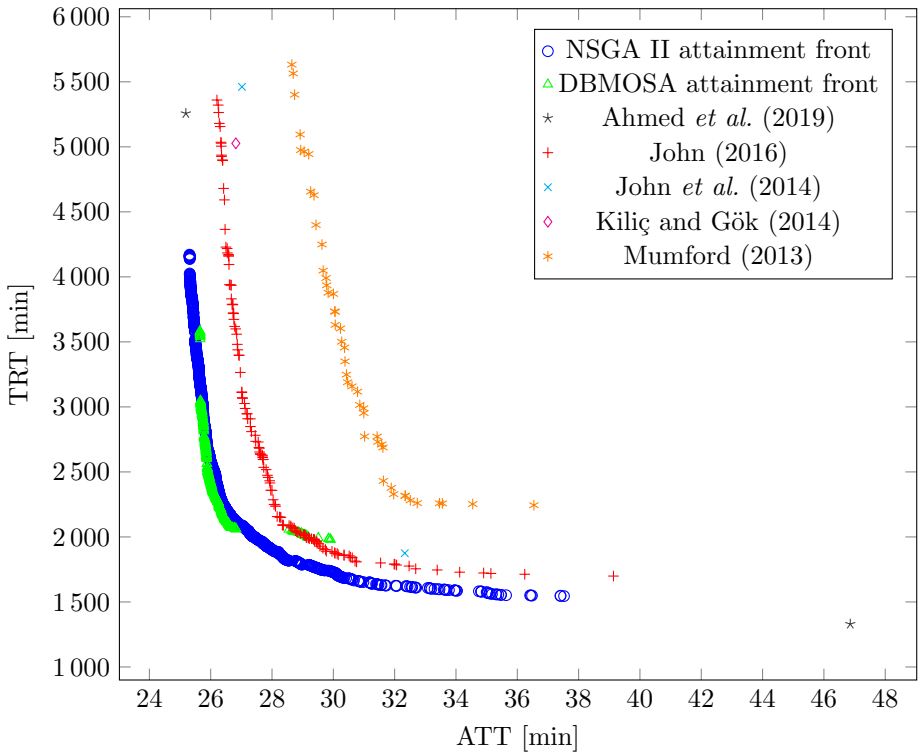


(b) *Mumford0*

FIGURE A.26: *Best attainment fronts for solutions produced by the NSGA II and the DBMOSA individually of Chapter 6, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. These are the first two subfigures of a continued figure.*

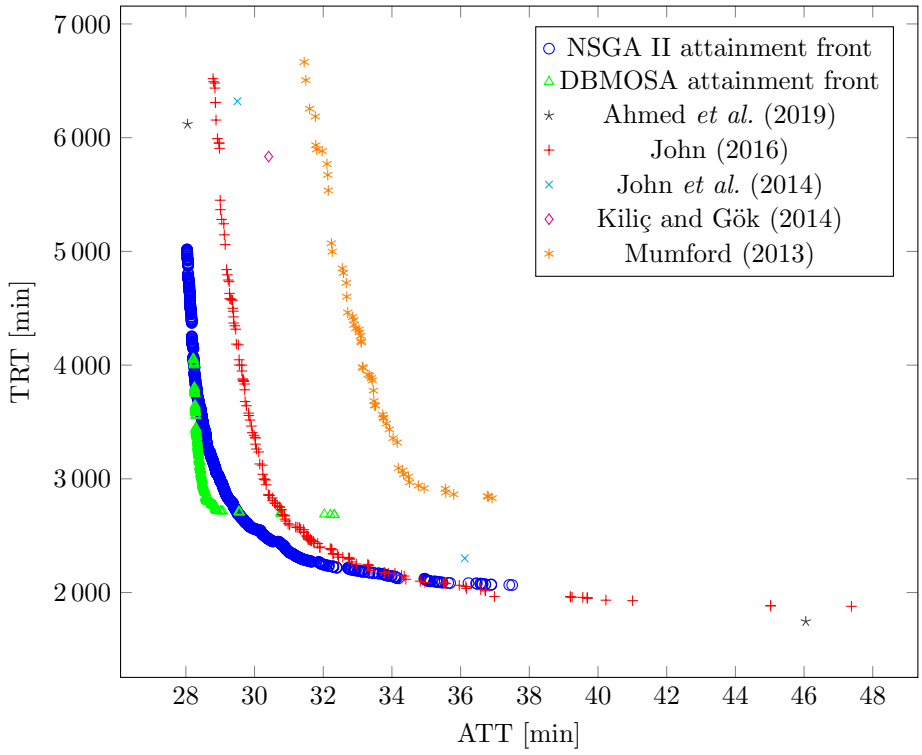


(c) *Mumford1*



(d) *Mumford2*

FIGURE A.26: *(continued) Best attainment fronts for solutions produced by the NSGA II and the DBMOSA individually of Chapter 6, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. These are the third and fourth subfigures of a continued figure.*



(e) *Mumford3*

FIGURE A.26: *(continued) Best attainment fronts for solutions produced by the NSGA II and the DBMOSA individually of Chapter 6, together with results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in §4.5.8. This is the last subfigure of a continued figure.*

TABLE A.4: *Reference route sets for the UTRP with incremental redesign, where asterisks indicate the end of a route and a dash an edge.*

Instance	Route set
Mandl6	1-2-5-7-9-13*2-5*8-14-7-9-10*4-3-5*6-9-10-12*0-1-3-11*
Mumford0	1-24-7-10-21*1-3-11-17-22-0-13*8-12-0-28-16-27-29*5-6-13-18-19*23-14-11-25*2-7-4*9-14-20*5-15*8-26*1-14-7-16-6-5*14-11-17-22-18*2-7-20-23*
Mumford1	23-29-48-66-19-22-33-20-55-16-26-62-21-6-35-64-52*39-18-54-36-45-58-51-8-26-49-42*11-9-17-20-33-22-38-54-5-37*24-63-6-35-34-45-36-19-66-48*31-1-16-10-20-33-22-19-18-32*37-5-54-19-22-33-20-17-59-68*15-49-26-8-51-41-36-54-18-39*28-56-51-57-25-12-40-46-7-47-6-21-30-26-16-55-20-33-22-38-66-48-3*42-49-26-8-51-34-45-36-54-65*11-9-17-20-33-22-19-54-5-37*31-1-16-55-20-33-22-19-54-32*0-69-18-54-36-45-34-51-8-26-49-42*60-44-61-14-2-50-13-43-17-9-31-1-16-55-20-33-22-19-18-32*42-49-26-8-51-34-45-36-54-65*27-67-5-54-36-45-58-51-8-26-49-15-30-63-6-47-53-4*
Mumford2	17-39-55-45-53-90-49-52-89-106*4-36-72-92-18-15-83-94-27-17-75*29-26-44-92-67-93-35-8-62-103*58-38-63-2-15-3-19-33-99-69*13-28-86-24-19-82-44-48-107-77*30-43-72-79-11-89-52-10-70-41*20-95-27-105-83-34-18-92-64-96*5-42-89-52-91-90-53-38-87-56-61*69-99-33-19-85-94-101-55-80-97*13-50-65-39-55-63-12-103-70-104*40-57-106-73-79-72-44-82-99-76*41-103-62-8-35-15-54-24-102-71-86-81*68-59-27-105-83-34-18-92-64-109*13-32-27-94-83-15-18-92-72-108*25-70-103-12-63-55-101-27-21-28-46*37-7-103-12-63-55-39-17-95-68*47-12-8-35-34-3-19-33-99-69*1-87-45-2-15-18-92-72-108-4*69-99-33-19-3-83-2-63-38-74*78-39-55-63-53-90-91-52-89-96*14-99-82-44-72-79-73-106-57-40*0-87-45-2-15-18-92-72-108-4*29-26-44-92-67-93-35-8-12-60*41-103-62-8-35-15-54-24-102-88-31*16-12-8-35-93-67-92-72-43-29*13-50-17-39-55-63-12-103-70-84-7-52-100*23-9-93-98-3-22-66-6-86-51-102-24-54-34-35-8-62-103-70*4-36-72-92-18-34-83-105-27-95-20*13-50-17-39-55-63-12-103-70-104*5-42-89-52-49-90-8-63-55-39-78*20-95-27-105-83-15-18-92-72-36*5-42-89-52-49-8-63-55-39-17-50*13-32-27-94-83-34-18-92-48-107-77*0-87-55-101-94-85-19-82-26-29*57-10-52-49-90-53-63-55-39-65*4-36-72-92-18-34-83-94-27-32*61-0-80-38-53-90-49-52-89-106*5-42-89-52-91-8-63-55-39-65*68-95-27-94-83-35-49-52-89-73*17-39-55-63-8-90-49-52-89-42*11-89-52-91-8-63-55-39-65-50*1-87-38-53-90-49-52-89-64-36*5-42-89-52-91-90-53-45-55-39*5-11-89-52-49-90-8-63-55-39*36-72-92-18-15-83-105-27-95-50*11-64-92-18-15-83-94-27-59-68*17-39-55-63-8-90-49-52-89-106*5-11-64-92-18-34-83-105-27-59*4-36-72-92-18-34-83-94-27-95-68*5-11-89-52-49-90-53-38-87-56*11-89-52-49-90-8-63-55-39-17-75*5-11-89-52-49-90-53-38-80-0-61*50-17-39-55-63-8-91-52-89-106*5-42-89-52-91-8-63-55-39-78*30-43-72-79-64-89-52-10-70-41*75-17-39-55-63-53-90-49-52-89-109*
Mumford3	20-32-60-73-48-2-49-66-80-63-58-104-18-109*85-92-126-26-11-1-118-52-89-59-43-69-109*10-117-24-5-29-27-67-11-7-40-50-13*35-91-88-101-0-79-2-23-53-61-87-84-56-3-71-100-96*34-56-3-78-114-2-11-46-39-30-22-119*10-86-24-36-9-72-49-2-48-126-92-76*83-50-116-93-88-39-31-89-113-43-69-109*45-73-48-97-11-1-118-52-89-17-64-111*75-60-73-48-0-77-1-118-52-89-42-98*99-55-16-62-112-6-73-48-2-49-124-9-63-105-104-18-109*57-8-35-125-88-101-0-79-37-23-53-61-87-110-84-56-15-107-3-115-108-14-106*19-92-126-26-11-1-118-52-89-113-90-103*20-50-65-82-88-39-31-89-42-43-69-109*34-56-12-78-114-2-79-0-101-88-91-123-125-121-25-41-30-54-33*10-117-24-5-80-81-67-11-7-40-50-13*4-43-113-89-52-118-1-11-26-126-92-76*28-42-89-52-118-1-11-97-48-73-60-75*21-22-30-39-46-11-2-114-78-3-56-34-68*19-32-60-73-48-2-49-124-9-36-24-86-109*10-117-24-36-80-66-49-2-48-126-92-20*6-73-48-0-11-1-118-52-89-17-64-111*76-92-126-26-11-1-118-52-89-42-90-103*74-114-16-55-60-73-48-2-49-72-9-63-58-104-18-109*32-60-73-48-2-49-72-9-5-24-86-109*10-86-24-36-9-72-49-2-48-126-92-83*10-117-24-36-9-72-49-2-48-126-92-76*20-32-60-73-48-2-49-124-9-5-24-86*69-43-113-89-52-118-1-11-26-126-92-76*19-32-60-73-48-2-49-72-9-36-24-86*45-73-48-0-11-1-118-52-89-17-64-111*10-117-24-36-80-66-49-2-48-73-60-75*10-86-24-36-9-124-49-2-48-73-60-32*10-117-24-36-80-66-49-2-48-126-92-50-65-120-44*85-92-126-26-11-1-118-52-89-42-43-69-109*18-104-105-63-80-66-49-2-48-73-60-32-19*10-86-24-5-80-66-49-2-48-126-92-13*20-92-126-26-11-1-118-52-89-42-43-69-109*10-117-24-36-80-66-49-2-48-126-92-83*32-60-73-48-2-49-72-9-63-105-43-103*20-92-126-48-2-49-66-80-36-24-86-109*18-104-105-63-9-124-49-2-48-73-60-32-20*76-50-40-82-88-39-31-89-42-43-69-109*20-32-60-73-48-2-49-72-9-36-24-117*69-43-105-63-9-124-49-2-48-73-60-75*20-92-126-26-11-1-118-52-89-59-43-103*20-92-126-26-11-1-118-52-89-113-43-109*20-50-116-93-88-39-31-89-113-43-69-109*10-86-24-36-29-27-67-11-26-40-50-13*19-32-60-73-48-2-49-66-80-63-105-104-59-89-51-38-47-31-52-122-94-81-9-70*13-83-92-126-48-2-49-124-9-5-24-86*102-24-86-18-104-58-63-9-72-49-2-48-73-60-32-20*20-32-60-73-48-2-49-66-80-63-105-104-18-109*95-13-50-92-126-48-2-49-72-9-5-24-86*20-32-60-73-48-2-49-124-9-63-58-104-18-109*20-32-60-73-48-2-49-124-9-36-24-86-109*20-50-40-93-88-39-31-89-42-43-69-109*10-86-24-36-29-81-1-11-26-40-50-13*10-86-24-36-80-66-49-2-48-126-92-83*4-43-42-89-52-118-1-11-26-126-92-85*76-50-116-93-88-39-31-89-59-43-69-109*

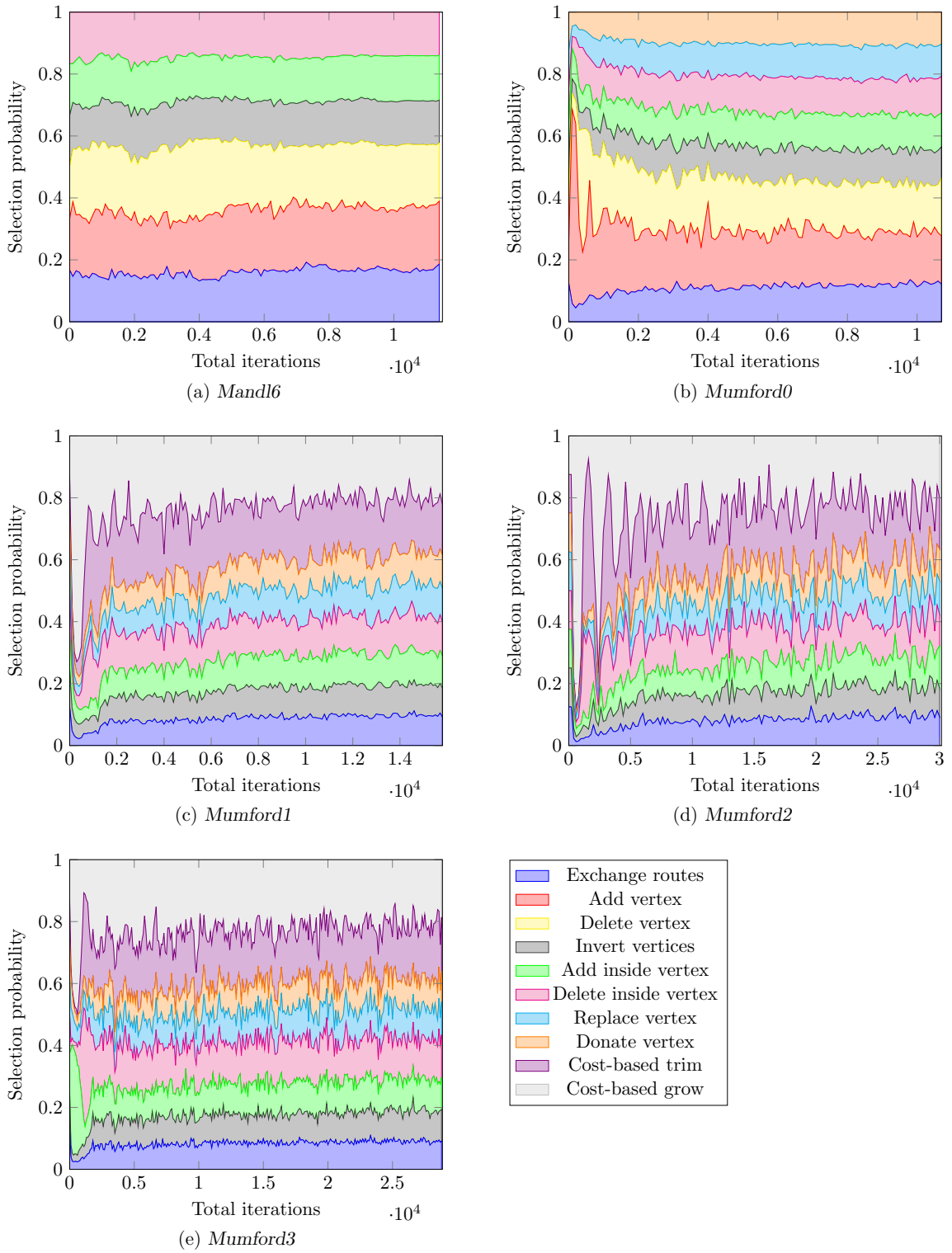


FIGURE A.27: *Perturbation selection probability analysis (Test set 16) results for each of the 30 separate long runs of the DBMOSA algorithm for solving the UTRP with incremental redesign, with stacked area charts of the change in LLH selection probabilities on the vertical axes over the total number of iterations on the horizontal axes.*

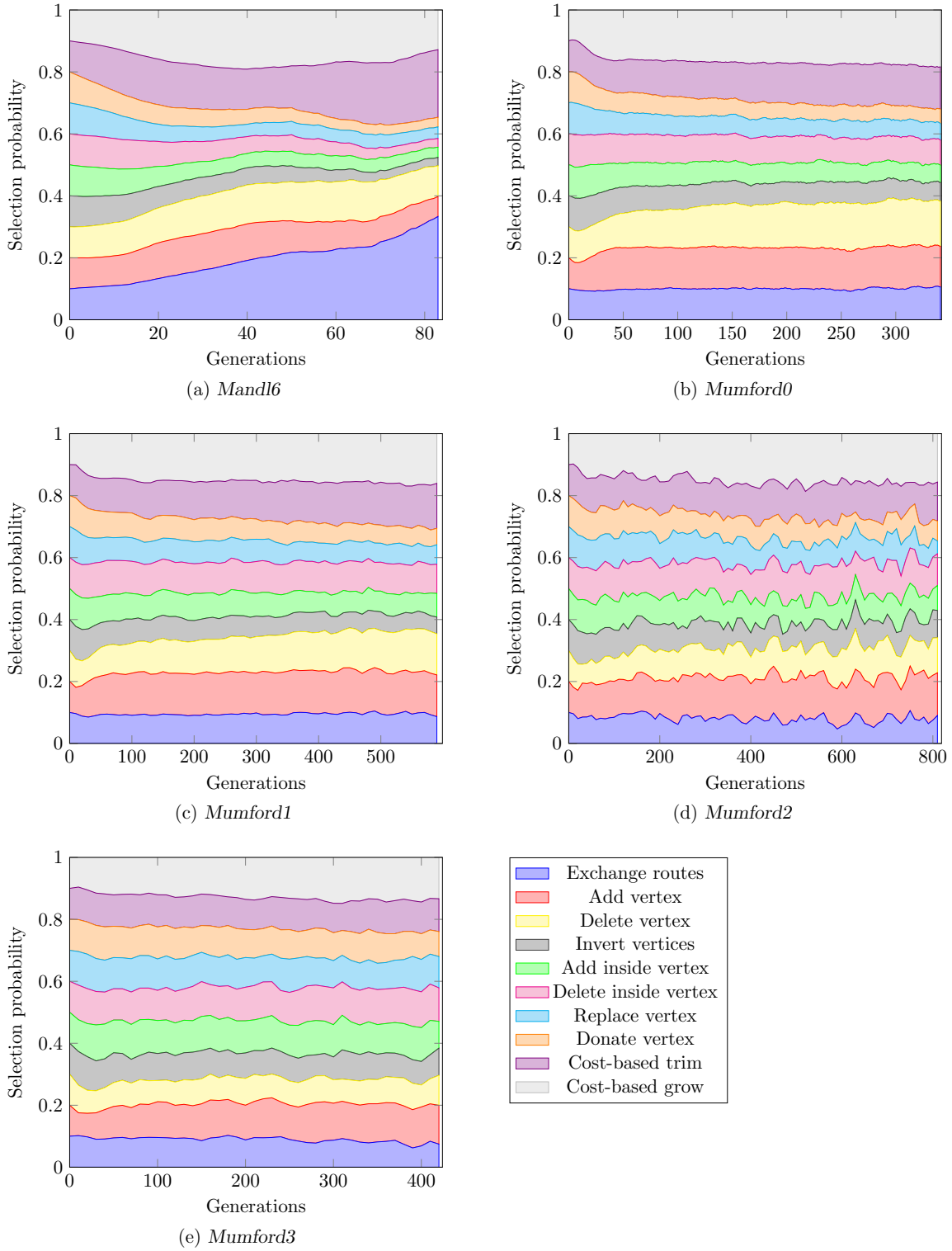


FIGURE A.28: *Mutation selection probability analysis (Test set 33) results for each of the 20 separate long runs of the NSGA II for solving the UTRP with incremental redesign, with stacked area charts of the change in LLH selection probabilities on the vertical axes over the number of generations on the horizontal axes.*