Multiobjective Routing and Transportation Problems

A thesis submitted in partial fulfilment of the requirements

for the Degree of Doctor of Philosophy

Andrea Raith

Supervised by
Associate Professor Matthias Ehrgott,
Dr. Stuart Mitchell,
and Dr. Judith Y.T. Wang

Department of Engineering Science

School of Engineering

The University of Auckland

New Zealand

2009

Abstract

Real life decision making takes into account multiple, often conflicting, criteria. This gives rise to multi-objective optimisation. We discuss a range of different bi-objective routing or transportation problems, in particular the shortest path, integer minimum cost flow, and traffic assignment problems.

The first part of this thesis is dedicated to the bi-objective shortest path problem. The problem is presented together with several well-known solution algorithms, namely bi-objective labelling, ranking, and the Two Phase method. Computational experiments highlight the strengths and weaknesses of the algorithms for different types of problem instances. We introduce new variations of the algorithms above and propose an easily implemented improvement for bi-objective labelling. Cyclist route choice in road networks is presented as a new application of the bi-objective shortest path problem, where cyclists aim to reach their destination in minimal travel time, but also along a safe route.

For the bi-objective integer minimum cost flow problem, we introduce one of the first solution algorithms based on the Two Phase method and demonstrate its performance based on different types of problem instances. An improvement of the algorithm for the transportation problem, a special case, is also discussed.

Finally, multi-objective traffic assignment is discussed. Traffic assignment is the process of modelling route choice of users of a (road) network in order to determine the total traffic on each road of the network. This is an equilibrium problem as the route choice of one traveller affects other travellers. We show how, traditionally, multiple objectives are treated in traffic assignment, often by combining them into a generalised cost function which may entail strong assumptions on road user behaviour. Instead, we suggest to explicitly distinguish the objectives in a multi-objective framework. We discuss existing

literature, which is of mainly theoretical nature, and comment on several articles presenting erroneous results. We contribute to the understanding of the theoretical concepts of vector equilibrium, vector variational inequalities, and vector optimisation. For the solution of bi-objective traffic assignment, we propose novel heuristic algorithms based on bi-objective shortest path algorithms as an important building block.

Acknowledgements

First and foremost I express my gratitude towards Associate Professor Matthias Ehrgott, Dr. Judith Wang, and Dr. Stuart Mitchell for supervising my thesis. Matthias has been an excellent supervisor. Without his great support and enthusiasm this project would not have been possible. Judith provided invaluable hands-on experience and knowledge of the transport modelling world and also the necessary industry contacts. Stuart provided advice, guidance, and encouragement whenever I needed it.

I would also like to thank the New Zealand Institute of Mathematics and its Applications (NZIMA) for the scholarship that allowed me to take up this PhD research.

Thank you to colleagues all over the world whom I got the chance to visit and work with. I thank José Figueira, Instituto Superior Técnico in Portugal, and Augusto Eusbío, Instituto Politécnico de Leira in Portugal, for an interesting collaboration on bi-objective integer minimum cost network flow problems, which started during a short visit to Portugal. I also thank Xavier Gandibleux, Anthony Przybylski, and all the PhD students at LINA, Université de Nantes in France, for welcoming me during a three month visit to Nantes.

Thank you to Chris Van Houtte and Tom Caiger, both students at the Department of Civil & Environmental Engineering, the University of Auckland, for an exciting collaboration on modelling cyclist route choice. I would like to thank John Davies from the Auckland Regional Council for providing the Auckland Regional Transport Planning Model network data that was vital for the cyclist route choice study.

I would like to thank everyone at the Department of Engineering Science for providing such a good working climate. Thank you to Lizhen, Eylem, Sepideh,

John, Anders, Richard, Oliver, and all the other PhD students at my department for making all those hours spent in our office (and the many hours we spent outside it!) much more enjoyable. Thanks also to all the staff members whose ideas and comments increased the quality of this thesis.

Heartfelt thanks also go out to those who support me outside university. I thank my friends in New Zealand and all the friends I left behind in Germany when I came to New Zealand (and those friends from other parts of the world) for their encouragement and patience – sometimes it is not easy being friends half-way around the world.

Besonderer Dank geht an meine Familie – an meine Eltern Angelika und Thomas, meinen Bruder Michael und meine Großeltern Brigitte, Karl-Heinz, Maria und Georg – ohne eure Unterstützung wäre dieses Projekt sicherlich sehr viel schwieriger gewesen. Ich denke an euch!

Last, but not least, I extend my love and gratitude to Oliver for having been at my side for all those years. Oliver is the one who lived with me through the ups and downs of my thesis, he was confident in my abilities whenever I wasn't.

Contents

List of Figures

List of Tables

Nomenclature

∇f	gradient of f: $\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m}\right)$, page 191
∂	partial derivative, page 142
Δ	upper bound on weighted sum objective, page 30
Δ'	upper bound on weighted sum objective, page 30
Δ_i	upper bound on weighted sum objective in triangle T_i, page 38
Λ	set of weights, $\Lambda := \{\lambda \in \mathbb{R}^r : \lambda_i \geqq 0, \sum_{i=1}^{r} \lambda_i = 1\}$, page 173
Φ	arc-path incidence matrix with entries ϕ_{ap}, page 141
α	VOT parameter, page 151
α_a	attractiveness rating of arc a, page 93
α_r	attractiveness rating of path r, page 94
δ	upper bound on weighted sum objective, page 29
δ_i	upper bound on weighted sum objective in triangle T_i, page 38
η_w	minimal travel time at equilibrium for OD pair w, page 144
π_i^k	dual variable at node i for objective k, page 24
ρ	cardinality of $\mathcal{R}$, page 141
σ	amount of flow change around cycle, page 16
ϕ_{ar}	has value 1 if arc a is part of path r, 0 otherwise, page 141
ω_i	weighting factor for $\overline{g}_a$ function, page 151
2LCOR	Two Phase Method with initialisation D, Phase 1 DDIC, and Phase 2 LCOR, page 76
2LSET	Two Phase Method with initialisation D, Phase 1 DDIC, and Phase 2 LSET, page 76
$\mathcal{A}$	set of arcs, page 11
$\mathcal{A}_x$	arc set of incremental graph G_x, page 121
A	$n \times m$ constraint matrix in linear programmes, page 11
A^G	node-arc incidence matrix of graph G, page 12
$A_{\mathcal{T}}^G$	basis matrix corresponding to basis $\mathcal{T}$, page 13

a^+	forward arc in incremental graph G_x, page 121
a^-	backward arc in incremental graph G_x, page 121
AON	all-or-nothing assignment for TA, page 147
b	right-hand-side vector $\in \mathbb{R}^m$ in linear programmes, page 11
$bd(\cdot)$	boundary of a set, page 173
BCMCF	bi-objective continuous minimum cost network flow problem, page 20
BFS	basic feasible solution, page 14
BIMCF	bi-objective integer minimum cost network flow problem, page 35
BIT	bi-objective integer transportation problem, page 132
bLCOR	bi-objective bounded label correcting algorithm, page 81
bLSET	bi-objective bounded label setting algorithm, page 81
BOP	bi-objective optimisation problem in the context of TA, page 169
BSP	bi-objective shortest path problem, page 41
BSPT	bi-objective shortest path tree problem, page 57
BUE	bi-objective user equilibrium for TA, page 167
C	$p \times m$ objective matrix in linear programmes with rows $(c^k)^\top, k = 1, \ldots, p$, page 11
C	cycle, page 16
$C(f)$	path cost objective matrix, $C : \mathbb{R}^\rho \to \mathbb{R}^{p \times \rho}$, page 168
$\overline{C}(\overline{f})$	arc cost objective matrix, $C : \mathbb{R}^m \to \mathbb{R}^{p \times m}$, page 168
c	arc cost vector, page 11
$\overline{c}$	reduced cost vector, page 14
c^λ	weighted sum objective vector, page 28
c^x	arc cost vector of incremental graph G_x, page 121
c^{λ^i}	weighted sum objective vector in triangle T_i, page 37
$c(f)$	vector of path cost functions, $c : \mathbb{R}^\rho \to \mathbb{R}^\rho$, page 142
$\overline{c}(\overline{f})$	vector of arc cost functions, $\overline{c} : \mathbb{R}^m \to \mathbb{R}^m$, page 142
$c_r(f)$	path cost function for path r, page 142
$\overline{c}_a(\overline{f})$	arc cost function for arc a, page 140
$\overline{c}^1(\overline{f}), \ldots, \overline{c}^p(\overline{f})$	arc cost vectors, p objectives, page 158
$\overline{c}^1_a(\overline{f}), \ldots, \overline{c}^p_a(\overline{f})$	arc cost for arc a, p objectives, page 151
$conv(\cdot)$	convex hull of a set, page 9
CTS	cost per unit time saving, page 216
d_w	travel demand for OD pair w, page 140
d^i_w	travel demand for OD pair w and user class i, page 152
D	single-objective label setting algorithm (Dijkstra's algorithm), page 56

DDIC	dichotomic Phase 1 approach using D to solve sub problems, page 57
EQS	equal share assignment, page 214
f	vector of path flow, page 141
$f(x)$	a function $f : \mathcal{X} \to \mathbb{R}$, page 26
f_r	flow on path r, page 141
f_r^i	flow on path r for user class i, page 152
$\overline{f}$	vector of arc flow, page 141
$\overline{f}_a$	flow on arc a, page 141
$\overline{f}_a^i$	flow on arc a for user class i, page 152
FIFO	first in first out strategy for inserting and removing elements from list, page 50
FSM	four-stage strategic transport planning model, page 136
FW	Frank-Wolfe algorithm for TA, page 148
G	graph, page 11
G_x	incremental graph in G for feasible solution x, page 120
$\overline{g}_a(\overline{f})$	weighted sum arc function, page 151
$\overline{g}_a^c(\overline{f})$	generalised arc cost function, page 151
$\overline{g}_a^t(\overline{f})$	generalised arc time function, page 151
$h(a)$	head node of arc a, page 11
$\mathcal{I}$	set of all user classes, page 152
$\mathcal{K}$	set of feasible path flow vectors, page 141
$\hat{\mathcal{K}}$	set of feasible capacitated path flow vectors, page 144
$\mathcal{K}_\mathcal{A}$	set of feasible arc flow vectors, page 141
$\hat{\mathcal{K}}_\mathcal{A}$	set of feasible capacitated arc flow vectors, page 144
l	distance label in shortest path algorithm, page 49
l	lower bound vector in $\mathbb{R}^m$, page 11
L	single-objective label correcting algorithm, page 56
LCOR	bi-objective label correcting algorithm, page 49
LDIC	dichotomic Phase 1 approach using L to solve sub problems, page 57
LSET	bi-objective label setting algorithm, page 52
m	dimension of decision space, page 7
m	number of arcs, page 11
$\overline{m}_a(\overline{f})$	arc cost function, page 151
MCF	single-objective minimum cost network flow problem, page 12
MIMCF	multi-objective integer minimum cost network flow problem, page 111
MOCO	multi-objective combinatorial optimisation problems, page 8

MOIO	multi-objective integer optimisation problems, page 8
MOZOO	multi-objective zero-one optimisation problems, page 8
MSA	method of successive averages for TA, page 148
MUE	multi-objective user equilibrium for TA, page 168
N	network, page 11
n	number of nodes, page 11
NSP	near shortest path algorithm (adapted to BSP), page 53
NSPD	near shortest path algorithm using Dijkstra's algorithm, page 55
OD	origin-destination pair, page 140
p	number of objectives, page 7
$\mathcal{R}$	set of all paths connecting OD pairs in $\mathcal{W}$, page 140
$\mathcal{R}_w$	set of all paths connecting OD pair w, page 140
$\mathbb{R}$	set of real numbers, page 7
RPT	reference point assignment, page 220
$\mathcal{S}_j$	set of candidate entering arcs in simplex iteration j, page 21
SEQ	single-objective equilibrium, page 143
SEQC	single-objective capacitated equilibrium, page 144
SPAR	parametric network simplex Phase 1 approach, page 58
$\mathcal{T}$	basis, contains all indices of basic columns, also basic tree, page 13
$\overline{\mathcal{T}}$	indices of non-basic columns, page 14
$\overline{\mathcal{T}}_l$	non-basic variables with flow at lower bound, page 14
$\overline{\mathcal{T}}_u$	non-basic variables with flow at upper bound, page 14
$t(a)$	tail node of arc a, page 11
t_a	travel time on arc a, page 91
t_r	travel time on path r, page 91
$\overline{t}_a(\overline{f})$	arc travel time function, page 151
TA	traffic assignment, page 135
u	upper bound vector in $\mathbb{R}^m$, page 11
UE	single-objective user equilibrium, page 143
$\mathcal{V}$	set of nodes or vertices, page 11
v	valuation function to transform time units into their monetary equivalent; often also v_w, page 160
$\tilde{v}$	valuation function to transform monetary units into their time equivalent, page 160
VEQ	vector equilibrium (multi-objective), page 169
VI	single-objective variational inequality, page 145

VI_ξ	scalarised variational inequality derived from VVI, page 171
VIa	arc based VI formulation of TA, page 145
VIp	path based VI formulation of TA, page 145
VOP	vector (multi-objective) optimisation problem in context of TA, page 169
VOT	value of time, page 151
VVI	vector variational inequality for TA, page 170
VVIa	arc based vector variational inequality for TA, page 170
$\mathcal{W}$	set of all OD pairs, page 140
WVEQ	weak vector equilibrium (multi-objective), page 169
WVOP	weak VOP, solutions are only weakly efficient, page 170
WVVI	weak VVI, page 170
$\mathcal{X}$	set of feasible solutions in decision space, page 7
$\mathcal{X}_E$	set of all efficient solutions, page 8
$\mathcal{X}_E^w(f)$	set of efficient paths with image in $\mathcal{Z}_N^w(f)$, page 172
$\mathcal{X}_{NE}$	set of all non-supported efficient solutions, page 9
$\mathcal{X}_{SE}$	set of all supported efficient solutions, page 9
$x_{lex(k,l)}$	a $lex(k,l)$-best solution of $\min_{x\in\mathcal{X}}(z_1(x), z_2(x))^\top$, page 10
$\mathcal{Z}$	set of feasible objective vectors in objective space, page 7
$\mathcal{Z}_N$	set of all non-dominated points, page 8
$\mathcal{Z}_{NN}$	set of all non-supported non-dominated points, page 9
$\mathcal{Z}_{SN}$	set of all supported non-dominated points, page 9
$\mathcal{Z}^w(f)$	set of all path cost vectors for OD pair w at flow f, page 172
$\mathcal{Z}_N^w(f)$	non-dominated points in $\mathcal{Z}^w(f)$, page 172
$\mathbb{Z}$	set of integer numbers, page 8
$z(x)$	objective function, page 7
$z_1(x), \ldots, z_p(x)$	components of objective function $z(x)$, page 7
z^N	nadir point, page 28
z^{LN}	local nadir point, page 29

Introduction

Most decisions made in real life are not of single-objective nature. Decisions often involve different, mostly conflicting, objectives. This fact is portrayed by the increasing popularity of bi-objective or multi-objective optimisation reflected by the number of publications on the topic. There are different approaches to solve multi-objective problems. Our aim in this thesis, in particular in Chapters 2 and 3, is to discuss algorithms capable of finding all *efficient* solutions, which are those solutions whose objectives cannot all be improved by another solution. The different approaches discussed are able to find all efficient solutions. As the problems considered are integer optimisation problems, we may distinguish two different types of efficient solutions: *supported* solutions that can be obtained as optimal solutions to weighted sum problems, whereas *non-supported* solutions are more difficult to find as their image lies in the interior of the convex hull of all feasible solutions of the problem (which means they cannot be obtained by a weighted-sum scalarisation).

In Chapter 2 we discuss bi-objective shortest path problems. Single-objective shortest path problems are very important as they have many applications. Obvious applications are, of course, searching for the shortest path between two points in a network as used in many route planning applications or when routing information through a computer network. There are many other applications, often not easily identifiable as applications of shortest path problems, such as approximating piecewise linear functions; knapsack problems (also a dynamic programming problem – many of them can be formulated as shortest path problems); basic scheduling problems; facility location; dynamic lot sizing (all listed in Ahuja et al. 1993). Furthermore, the shortest path problem appears as a sub-problem in many important optimisation problems such as the traffic assignment problem (see also Chapter 4) or scheduling problems

solved with the aid of column generation algorithms. Often, column generation approaches are based on solving resource constrained shortest path problems which are closely related to multi-objective shortest path problems (by identifying each separate resource with an objective). There is an extensive range of publications on the single-objective shortest path problem, where algorithmic performance is improved by sophisticated data structures and speed-up techniques.

The literature on the bi-objective shortest path problem is not as extensive (yet). A few applications of bi-objective or multi-objective nature have been discussed such as Satellite scheduling (Gabrel and Vanderpooten 2002) and computing shortest paths for passengers in railway networks (Müller-Hannemann and Weihe 2006). Different algorithms have been proposed, some of which are extensions of the well-known single-objective label setting (e.g. Dijkstra's algorithm) and label correcting algorithms (e.g. Bellman's algorithm). Other algorithms are based on shortest path ranking or follow the so-called Two Phase Method where supported and non-supported solutions are obtained separately, in Phase 1 and Phase 2, respectively.

Throughout the literature on bi-objective shortest path problems, it remains unclear which algorithm performs best in practice, although there appears to be a preference for bi-objective labelling algorithms, possibly due to their ease of implementation. We fill this gap by creating efficient implementations of all known solution strategies. We then compare these strategies for different types of problem instances. This allows us to show that algorithms based on the Two Phase Method are competitive with, if not better than, bi-objective labelling algorithms. By using three different types of problem instances we are able to show that the underlying structure of the instance determines which algorithm performs best. This constitutes the most extensive run-time study on bi-objective shortest path problems to date. We also propose an easily implemented improvement of bi-objective labelling algorithms that is, surprisingly, unreported in the literature so far.

We suggest a new application of bi-objective shortest paths in modelling the route choice of cyclists in traffic networks. It is widely acknowledged that motorists choose their travel route through a network with the main aim of minimising their travel time (or a generalised cost function that combines

travel time and other route choice factors). For cyclists, however, it is evident that route choice is not only based on the travel time along the path. Cyclists share road infrastructure with motorised vehicles, which can be a dangerous undertaking at times. Next to the travel time objective, we also formulate an attractiveness objective that combines path safety and other factors that make a path attractive to cyclists. This enables us to obtain paths that represent good trade-offs between travel time and path attractiveness. The attractiveness objective is not linear, therefore we need to adapt the solution approach to this objective.

A generalisation of shortest path problems are minimum cost network flow problems. In Chapter 3 we study the bi-objective integer minimum cost flow problem. There are many practical applications of network flow problems and therefore the study of the multi-objective problem becomes increasingly important. Typical applications studied for the single-objective problem involve the distribution of products through a network from points where they are in supply to other points where they are in demand. A cost-minimal way to move them through the network is sought. Applications include the planning of radiation therapy treatment to fight cancer (Ahuja and Hamacher 2005); models for building evacuation; distribution of empty rail cars; warehouse layout problem; production and inventory planning; scheduling problems (all listed in Ahuja et al. 1993).

In the literature, only few publications on bi- or multi-objective integer minimum cost network flow problems exist. There are articles on solution algorithms, but none that discuss an application of the multi-objective problem. We are among the first to publish a correct solution algorithm to compute all efficient solutions of an integer minimum cost flow problem. While it is well-known how to obtain all supported solutions of the problem, several algorithms proposed to obtain all non-supported solutions have some flaws. We present a correct solution algorithm based on the Two Phase Method. On the basis of a large set of test instances, we demonstrate the speed of the algorithm and what kinds of instances can be solved within reasonable time. We observe that the effort of computing all efficient solutions is large, with the computation of non-supported solutions being the most time-consuming part. We also comment on a refinement of the developed algorithm for solving the bi-objective transportation problem, which achieves good run-time improvements over our

original algorithm.

Finally, in Chapter 4, we move away from linear network optimisation problems to the traffic assignment problem, which is a network equilibrium problem. The more general trip assignment problem determines the route choice of network users within a network. We focus on traffic assignment, the problem of determining motorists' route choice within a road network. This underlying network is a road network where travel time for each arc is flow dependent – more traffic on a road means slower travel speed along the road. As the travel time increases with more and more flow on an arc (a road), we speak of traffic congestion.

Traffic assignment is the last component of the conventional four-stage strategic transport planning model. The first three steps of this transport planning model determine how much travel demand originates from different sections of the network, where it goes, and the mode choice of network users. Those travel options may include travelling by car, by bus, by train, by bike, and others. The final trip assignment step models the actual route choice of each traveller, one major component being traffic assignment for vehicular traffic. From this, a prediction of the actual traffic on each road of the network is obtained. It is generally assumed that network users aim to minimise their travel time or a generalised cost function that includes travel time and other route choice factors. There are several feasible approaches to solving the single-objective traffic assignment problem. Almost all of them take advantage of an equivalent re-formulation of the problem. For instance, the traffic assignment problem has (under certain assumptions as discussed in Chapter 4) an equivalent formulation as an optimisation problem or as a variational inequality problem.

It is acknowledged throughout the related literature that route choice is not solely based on the criterion of choosing a route with minimal travel time. As mentioned above, monetary costs and other objectives (scenery, safety) may be included into a generalised cost function. The problem of traffic assignment with two or more separate objectives has been studied with the aim of proposing solution algorithms. Conventionally, these different objectives are added together into a generalised cost function using a weighted sum, which may be a very strong behavioural assumption. Some approaches assume there is a single weighting factor, whereas others acknowledge that different network

users have different weightings and split users into different user classes – each with different weighting factors. Alternative approaches assume a distribution of weighting factors across the users. Instead, we propose to consider the main objectives in route choice, travel time and monetary cost, to be treated independently, leading to a bi-objective traffic assignment problem. The arising traffic assignment problem is a bi-objective network equilibrium problem. In Chapter 4 we give a definition of bi-objective traffic assignment that is not based on a weighted cost function assumption.

In the literature, the class of vector (or multi-objective) equilibrium problems, of which bi-objective traffic assignment is a special case, is studied on a purely theoretical level. Existence of solutions is discussed. Attempts are made to relate the vector equilibrium to other multi-objective problems, similar to what has been done for single-objective equilibrium problems. Unfortunately, equivalence of the vector equilibrium problem to another multi-objective problem, the vector variational inequality problem, can only be established under very strong assumptions.

We discuss some properties of the vector equilibrium problem and show that it is not equivalent to other multi-objective problems. We highlight the properties of solutions of the vector equilibrium problem that prevent this equivalence. This greatly enhances understanding of vector equilibrium problems and helps differentiate them from other multi-objective problems. It also enables us to relax the conditions for the equivalence of vector equilibrium problems and vector variational inequalities slightly – they still remain too restrictive to enable the use of this equivalence within a solution algorithm. We are furthermore able to clarify some erroneous results from the literature.

We extend the existing algorithms known for the single-objective traffic assignment problem to the bi-objective case. Thus, we present several heuristic solution approaches to obtain bi-objective equilibrium solutions. Our heuristics utilise a bi-objective shortest path algorithm as part of the solution approach, namely to solve arising sub-problems in each iteration. We believe this is a first step towards algorithms to solve the bi-objective traffic equilibrium problems, and thus vector equilibrium problems.

As this thesis covers three main topics, bi-objective shortest path, bi-objective integer minimum cost flow, and bi-objective traffic assignment, one chapter

is dedicated to each of them. This thesis is organised as follows. In the first chapter we give an overview of the relevant mathematical background. We introduce the terms of bi- and multi-objective optimisation and some solution algorithms and strategies as a basis for later chapters. Chapter 2 covers the bi-objective shortest path problem. In Chapter 3 the more general bi-objective integer minimum cost flow problem is discussed. Finally, Chapter 4 is dedicated to the bi-objective traffic assignment problem.

Each chapter contains its own introduction and conclusion as the chapters may be seen as individual entities. Nevertheless, the problems discussed in Chapter 3 are strongly related to those from Chapter 2 and so are the solution approaches. Also, bi-objective shortest path algorithms are an essential element of the heuristic solution algorithms presented for bi-objective traffic assignment in Chapter 4.

Chapter 1

Preliminaries: Mathematical Background

In this chapter the basics of bi- and multi-objective optimisation problems are introduced to the extent needed within the scope of this thesis. A comprehensive coverage of these topics can be found in Ehrgott (2005).

1.1 Multi-objective Optimisation Problems

A *multi-objective optimisation problem* has the form

$$\begin{aligned} \min \quad & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \\ \vdots \\ z_p(x) \end{pmatrix} \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned}$$

with $p \geqq 2$ and $\mathcal{X} \subset \mathbb{R}^m, m \geqq 1$. Whenever the number of objectives is $p > 2$, we talk about *multi-objective* or *multi-criteria* optimisation, whereas reducing p to two leads to a *bi-objective* or *bi-criteria* problem. The *set of feasible solutions* (*feasible set*) in *decision space* is denoted by $\mathcal{X}$ and its image is $\mathcal{Z} := \{z(x) : x \in \mathcal{X}\}$, which lies in *objective space*.

The structure of the feasible set $\mathcal{X}$ entails a further classification of the multi-objective optimisation problem. *Multi-objective zero-one optimisation (MOZOO)* problems have binary variables, i.e. $\mathcal{X} \subseteqq \{0,1\}^m$. An example for a MOZOO problem is the bi-objective shortest path problem, see Chapter 2. The variables may be restricted to integers with $\mathcal{X} \subset \mathbb{Z}^m$ leading to *multi-objective integer optimisation (MOIO)* problems. The bi-objective integer minimum cost flow problem discussed in Chapter 3 is a MOIO problem. We speak of *multi-objective continuous optimisation* problems when the set $\mathcal{X}$ is a continuous subset of $\mathbb{R}^m$. $\mathcal{X}$ may also be a combination of the above, for instance an optimisation problem with some integer variables and some continuous variables, i.e. $\mathcal{X} \subseteqq \mathbb{Z}^{m_1} \times \mathbb{R}^{m_2}$. *Multi-objective combinatorial optimisation (MOCO)* problems have a special constraint structure, such as for example the shortest path and the flow problems mentioned above. Large parts of this thesis deal with MOCO problems, and the special structure of the problems is exploited, whenever possible.

Let us define some order relations for $y^1, y^2 \in \mathbb{R}^p, p \geqq 2$:

$$\begin{aligned} y^1 \leqq y^2 &\Leftrightarrow y^1_k \leqq y^2_k \text{ for } k = 1,2,\ldots,p, \\ y^1 \leq y^2 &\Leftrightarrow y^1_k \leqq y^2_k \text{ for } k = 1,2,\ldots,p; \text{ and } y^1 \neq y^2, \\ y^1 < y^2 &\Leftrightarrow y^1_k < y^2_k \text{ for } k = 1,2,\ldots,p. \end{aligned}$$

The symbols $\geqq, \geq, >$ are defined analogously. Furthermore, for any of the symbols $\succ \in \{\geqq, \geq, >\}$, we define $\mathbb{R}^p_\succ$ as $\mathbb{R}^p_\succ := \{x \in \mathbb{R}^p : x \succ 0\}$.

In general the p objectives considered in multi-objective optimisation problems are conflicting with each other, they do not obtain their individual minima for the same value of x. Therefore, we seek those feasible solutions that do not allow to improve one component of the objective vector $z(x)$ without deteriorating another one.

Definition 1.1.1 A feasible solution $\hat{x} \in \mathcal{X}$ is called *efficient* if there does not exist any $x' \in \mathcal{X}$ with $(z_1(x'), z_2(x'), \ldots, z_p(x')) \leq (z_1(\hat{x}), z_2(\hat{x}), \ldots, z_p(x))$. The image $z(\hat{x})$ of $\hat{x}$ is called *non-dominated*. Let $\mathcal{X}_E$ denote the *set of all efficient solutions* and let $\mathcal{Z}_N$ denote the *set of all non-dominated points*. If there exist $x', x^* \in \mathcal{X}$ with $z(x') \leq z(x^*)$, we say that x^* *is dominated* and also that x' *dominates* x^*.

We distinguish two different kinds of efficient solutions.

- *Supported* efficient solutions are those efficient solutions that can be obtained as optimal solutions to a (single objective) weighted sum problem

$$\min_{x \in \mathcal{X}} \lambda_1 z_1(x) + \lambda_2 z_2(x) + \ldots + \lambda_p z_p(x) \tag{1.1}$$

  for some $\lambda \in \mathbb{R}^p_>$. The set of all supported efficient solutions is denoted by $\mathcal{X}_{SE}$, its non-dominated image by $\mathcal{Z}_{SN}$. The supported non-dominated points lie on the boundary of the convex hull $conv(\mathcal{Z})$ of the feasible set in objective space.

- Supported efficient solutions which define an extreme point of $conv(\mathcal{Z})$ are called *extreme* supported efficient solutions.

- The remaining efficient solutions in $\mathcal{X}_{NE} := \mathcal{X}_E \backslash \mathcal{X}_{SE}$ are called *non-supported* efficient solutions. They cannot be obtained as solutions of a weighted sum problem as their image lies in the interior of $conv(\mathcal{Z} + \mathbb{R}^p_{\geqq})$. The set of non-supported non-dominated points is denoted by $\mathcal{Z}_{NN}$. There is no known characterisation of non-supported efficient solutions that leads to a polynomial time algorithm for their computation.

All efficient solutions of a continuous linear multi-objective problem are supported (Isermann 1974, Theorem 1), whereas an integer linear multi-objective optimisation problem may have non-supported solutions.

The objective functions $z_1, z_2, \ldots, z_p$ do generally not attain their individual optima for the same value of $\hat{x}$. We assume in the following that there exists no $\hat{x}$ such that $\hat{x} \in \operatorname{argmin}\{z_1\} \cap \operatorname{argmin}\{z_2\} \cap \ldots \cap \operatorname{argmin}\{z_p\}$ for a multi-objective optimisation problem to ensure that we deal with multi-objective problems that actually have conflicting objectives and therefore more than one non-dominated point in objective space.

Definition 1.1.2 Two feasible solutions x and x' are called *equivalent* if $z(x) = z(x')$. A *complete set* $\mathcal{X}_E$ is a set of efficient solutions such that all $x \in \mathcal{X} \backslash \mathcal{X}_E$ are either dominated or equivalent to at least one $x \in \mathcal{X}_E$.

In this thesis, we only consider solution approaches that compute (at least) a complete set $\mathcal{X}_E$.

Another notion of optimality that is used in the context of bi-objective optimisation is *lexicographic minimisation*. In the bi-objective case, among all optimal feasible solutions for the *preferred* component k of the objective vector, we choose one that is optimal for the other component l.

Definition 1.1.3 Let $k \in \{1,2\}$ and $l \in \{1,2\}\backslash\{k\}$. Then $z(\hat{x}) \leqq_{lex(k,l)} z(x')$ if either $z_k(\hat{x}) < z_k(x')$ or both $z_k(\hat{x}) = z_k(x')$ and $z_l(\hat{x}) \leqq z_l(x')$. We call $\hat{x}$ a $lex(k,l)$-best solution if $z(\hat{x}) \leqq_{lex(k,l)} z(x)$ for all $x \in \mathcal{X}$. Let $x_{lex(k,l)}$ denote a $lex(k,l)$-best solution.

In the more general case with $p > 2$, assuming that objectives are numbered according to their preference, lexicographic minimisation can be defined as follows:

Definition 1.1.4 For $y^1 \neq y^2$, let $k^* := min\{k : y^1_k \neq y^2_k\}$. Then,

$$y^1 \leqq_{lex} y^2 \Leftrightarrow y^1 = y^2 \text{ or } y^1_{k^*} \leqq y^2_{k^*}.$$

1.2 Multi-objective Linear Programmes

An important class of multi-objective optimisation problems are the linear ones, where the objective functions as well as the feasible set are described by linear functions. Such a *multi-objective linear programme* has the form

$$\begin{aligned} \min \quad & Cx = \begin{pmatrix} (c^1)^\top x \\ (c^2)^\top x \\ \vdots \\ (c^p)^\top x \end{pmatrix} \\ \text{s.t.} \quad & Ax = b \\ & x \geqq 0, \end{aligned}$$

where C is a $p \times m$ *objective matrix* that has rows $(c^k)^\top, k = 1, \ldots, p$, which represent the k^{th} cost function. Feasible solutions $x \in \mathbb{R}^m$ are required to be non-negative and satisfy constraints $Ax = b$ given by the $n \times m$ constraint matrix A and the right-hand-side $b \in \mathbb{R}^m$.

Multi-objective linear programmes can be solved using a multi-objective version of the simplex algorithm, which is derived from the single-objective simplex algorithm (Dantzig and Thapa 1997, for instance). A whole chapter of Ehrgott (2005) is dedicated to multi-objective linear programming. In this thesis, all linear programmes considered have a special structure: they are network flow problems. For network flow problems a special version of the simplex algorithm, the *network* simplex algorithm, exists.

We first introduce single-objective network flow problems together with the network simplex algorithm to solve them. Subsequently, a bi-objective version of this network simplex algorithm, a parametric network simplex algorithm, is introduced.

1.2.1 Single-objective Network Flow Problems and the Network Simplex Algorithm

First, we introduce the single-objective network flow problem. We define a *directed network* by $N = (G, c, l, u)$, where the graph $G = (\mathcal{V}, \mathcal{A})$ consists of a set of *vertices* or *nodes* $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of m *arcs* $\mathcal{A} \subseteqq \mathcal{V} \times \mathcal{V}$. For $a \in \mathcal{A}$, denote by $t(a)$ the tail node of arc a and by $h(a)$ its head node. The vectors $c, l, u \in \mathbb{R}^m$ represent cost, lower bounds, and upper bounds and are explained below. The single-objective network flow problem has the form

$$\begin{array}{lll} \min & \displaystyle\sum_{a \in \mathcal{A}} c_a x_a & \\ \text{s.t.} & \displaystyle\sum_{\{a \in \mathcal{A}: t(a) = i\}} x_a - \sum_{\{a \in \mathcal{A}: h(a) = i\}} x_a = b_i & \text{for all } i \in \mathcal{V} \\ & u_a \geqq x_a \geqq l_a & \text{for all } a \in \mathcal{A}. \end{array} \tag{1.2}$$

In this model, the variable $x \in \mathbb{R}^m$ represents flow in the network. The flow on arc a is x_a and its cost per unit of flow c_a. The first set of constraints ensures flow conservation at the different nodes, and we assume that $\sum_{i \in \mathcal{V}} b_i = 0$ since otherwise the problem is infeasible. A *balance* of $b_i > 0$ represents *surplus*

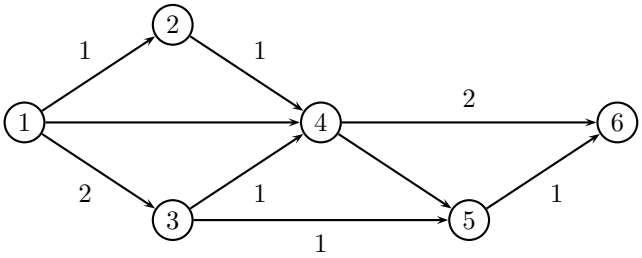


Figure 1.1. Network of Example 1.2.1.

and a balance of $b_i < 0$ represents *demand* of a commodity at node i, whereas $b_i = 0$ indicates that i is a *trans-shipment node* that has neither demand nor supply. The second set of constraints ensures that flow on each arc a is between its lower bound l_a and upper bound u_a with $u_a \geqq l_a \geqq 0$.

The objective leads to cost-minimal shipment of the commodity between supply and demand nodes with minimal cost, hence the problem is referred to as *minimum cost network flow (MCF) problem*. In case of positive lower bound capacities, the network can be transformed into a network with zero lower bound capacities as explained in Ahuja et al. (1993). Therefore, we assume $l = 0$ in the following.

Example 1.2.1 *Figure 1.1 shows the network of an MCF problem consisting of* 6 *nodes and* 9 *arcs. All arcs have lower bound* 0 *and upper bound* 2*. Node* 1 *has balance* 3 *and node* 6 *has balance* -3*, all other nodes have balance* 0*. This means that* 3 *units of a commodity are shipped through the network. The numbers next to an arc represent the flow on the arc (zero flow if there is no number). The indicated flow solution is a feasible solution. Arc costs are omitted at the moment.*

A $n \times m$-dimensional *node-arc incidence matrix* A^G of graph $G = (\mathcal{V}, \mathcal{A})$ in the MCF problem is constructed by

$$A^G_{ia} = \begin{cases} +1 & \text{if } t(a) = i, \\ -1 & \text{if } h(a) = i, \\ 0 & \text{otherwise,} \end{cases}$$

for $1 \leqq i \leqq n$ and $1 \leqq a \leqq m$ with $n = |\mathcal{V}|, m = |\mathcal{A}|$. The matrix A^G is an equivalent representation of the graph G above. With the vectors $c^\top = (c_1, c_2, \ldots, c_m)$, $l^\top = (l_1, l_2, \ldots, l_m) = (0, 0, \ldots, 0)$, $u^\top = (u_1, u_2, \ldots, u_m)$, and $b^\top = (b_1, b_2, \ldots, b_n)$ the above MCF problem can be re-written as

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & A^G x = b \\ & u \geqq x \geqq 0. \end{aligned} \tag{1.3}$$

This linear programme associated with the MCF problem (1.2) can be solved using the standard simplex algorithm with upper and lower bound constraints on the variable x. However, there exists a version of the simplex algorithm, called *network simplex algorithm*, which is dedicated to solving network flow problems. Special data structures allow a very efficient implementation of the network simplex algorithm. Helgason and Kennington (1995) develop a detailed derivation of the network simplex algorithm and how it relates to the standard simplex algorithm. We outline a derivation of this network simplex algorithm with upper and lower bound constraints in the following.

Network Simplex Algorithm

From now on, it is assumed that the graph G is *connected*, i.e. every pair of nodes i, j is connected by an un-directed path. A subset of columns of A^G is linearly dependent whenever the arcs represented by the columns contain an un-directed cycle. As the network is connected, a maximal set of linearly independent columns represents a set of arcs $\mathcal{A}^* \subseteqq \mathcal{A}$ such that the undirected graph $(\mathcal{V}, \mathcal{A}^*)$ is a *spanning tree*. A spanning tree is an un-directed subset of arcs so that there is exactly one path between every pair of nodes. In particular, a spanning tree does not contain a cycle. A spanning tree has $n-1$ arcs. Therefore, the matrix A^G in (1.3) has rank $n-1$. We add an additional arc to A^G represented by a column vector with only one non-zero entry in row r. This makes node r a *root node* of G, which has an in-going arc that does not have a tail node. The obtained matrix A^G now has full rank n.

An $n \times n$ sub-matrix of A^G, $A^G_{\mathcal{T}}$, is called *basis matrix* if $A^G_{\mathcal{T}}$ is invertible. The set of column indices $\mathcal{T}$ that identify the columns of $A^G_{\mathcal{T}}$ in A^G is called *basis*.

The set of indices of all *non-basic* columns of A^G is $\overline{\mathcal{T}} = \{1, 2, \ldots, m\} \setminus \mathcal{T}$. A^G, c, and x can be split into basic and non-basic components

$$A^G = (A^G_{\mathcal{T}}, A^G_{\overline{\mathcal{T}}}),\ c = \begin{pmatrix} c_{\mathcal{T}} \\ c_{\overline{\mathcal{T}}} \end{pmatrix}, \text{ and } x = \begin{pmatrix} x_{\mathcal{T}} \\ x_{\overline{\mathcal{T}}} \end{pmatrix}.$$

$A^G x = b$ can be re-written as

$$(A^G_{\mathcal{T}}, A^G_{\overline{\mathcal{T}}}) \begin{pmatrix} x_{\mathcal{T}} \\ x_{\overline{\mathcal{T}}} \end{pmatrix} = b \iff x_{\mathcal{T}} = (A^G_{\mathcal{T}})^{-1} \left(b - A^G_{\overline{\mathcal{T}}} x_{\overline{\mathcal{T}}}\right). \tag{1.4}$$

We distinguish those non-basic variables with flow at lower bound and at upper bound, indexed by $\overline{\mathcal{T}}_l$ and $\overline{\mathcal{T}}_u$, respectively, and $\overline{\mathcal{T}} = \overline{\mathcal{T}}_l \cup \overline{\mathcal{T}}_u$. The non-basic variables have value $x_{\overline{\mathcal{T}}_l} = 0$ and $x_{\overline{\mathcal{T}}_u} = u_{\overline{\mathcal{T}}_u}$, where $u_{\overline{\mathcal{T}}_u}$ is obtained by splitting the vector u into basic and non-basic components $u = (u^\top_{\mathcal{T}}, u^\top_{\overline{\mathcal{T}}_l}, u^\top_{\overline{\mathcal{T}}_u})$. The matrix A^G and x are split up accordingly. Then, any *basic solution* x is given by

$$x = \begin{pmatrix} x_{\mathcal{T}} \\ x_{\overline{\mathcal{T}}_l} \\ x_{\overline{\mathcal{T}}_u} \end{pmatrix} \text{ with } x_{\mathcal{T}} = (A^G_{\mathcal{T}})^{-1} \begin{pmatrix} b - \underbrace{A^G_{\overline{\mathcal{T}}_l} x_{\overline{\mathcal{T}}_l}}_{=0} - \underbrace{A^G_{\overline{\mathcal{T}}_u} x_{\overline{\mathcal{T}}_u}}_{=u_{\overline{\mathcal{T}}_u}} \end{pmatrix}.$$

If $0 \leqq x \leqq u$ also holds, x is called *basic feasible solution (BFS)*. Now the cost $c^\top x$ can be rewritten as

$$\begin{aligned} &\left(c^\top_{\mathcal{T}}, c^\top_{\overline{\mathcal{T}}_l}, c^\top_{\overline{\mathcal{T}}_u}\right) \begin{pmatrix} x_{\mathcal{T}} \\ x_{\overline{\mathcal{T}}_l} \\ x_{\overline{\mathcal{T}}_u} \end{pmatrix} \\ &= c^\top_{\mathcal{T}} (A^G_{\mathcal{T}})^{-1} b \\ &\quad + \underbrace{\left(c^\top_{\overline{\mathcal{T}}_l} - c^\top_{\mathcal{T}} (A^G_{\mathcal{T}})^{-1} A^G_{\overline{\mathcal{T}}_l}\right)}_{\overline{c}_l} x_{\overline{\mathcal{T}}_l} + \underbrace{\left(c^\top_{\overline{\mathcal{T}}_U} - c^\top_{\mathcal{T}} (A^G_{\mathcal{T}})^{-1} A^G_{\overline{\mathcal{T}}_u}\right)}_{\overline{c}_u} x_{\overline{\mathcal{T}}_u}, \end{aligned}$$

where the vector $\overline{c} = (\overline{c}^\top_l, \overline{c}^\top_u)$ is called *reduced cost* vector. One way to reduce the cost $c^\top x$ is increasing one of the non-basic variables with flow at its lower bound, i.e. x_s with $s \in \overline{\mathcal{T}}_l$, that has negative reduced cost $\overline{c}_s < 0$. Similarly, $c^\top x$ decreases as the flow of a non-basic variable with flow at its upper bound

with positive reduced cost decreases. Therefore, a variable with index $s \in \left\{i \in \overline{\mathcal{T}}_l : \bar{c}_i < 0\right\} \cup \left\{i \in \overline{\mathcal{T}}_u : \bar{c}_i > 0\right\}$ is chosen to enter the basis $\mathcal{T}$.

In exchange for the new variable with index s, another variable needs to leave the basis. If $s \in \overline{\mathcal{T}}_l$, assume that the value of x_s increases to $\sigma \leqq u_s$. In (1.4), rewrite $\tilde{A} = (A^G_{\mathcal{T}})^{-1} A^G_{\overline{\mathcal{T}}}$. It follows that the basic variable x_i with $i \in \mathcal{T}$ changes to $x_i - \sigma \tilde{A}_{is}$. To ensure feasibility of the basic variable $x_i - \sigma \tilde{A}_{is}$, one has to ensure that

$$0 \leqq x_i - \sigma \tilde{A}_{is} \leqq u_i \quad \Leftrightarrow \quad \sigma \leqq \begin{cases} \frac{x_i}{\tilde{A}_{is}} & \text{if } \tilde{A}_{is} > 0 \\ \frac{u_i - x_i}{-\tilde{A}_{is}} & \text{if } \tilde{A}_{is} < 0 \end{cases} \tag{1.5}$$

Therefore, whenever $s \in \overline{\mathcal{T}}_l$, the variable x_s increases to a value of $\sigma \leqq u_s$ for which (1.5) is satisfied for all $i \in \mathcal{T}$. The largest possible value is

$$\sigma = \min\left\{u_s, \min\left\{\frac{x_i}{\tilde{A}_{is}} : i \in \mathcal{T}, \tilde{A}_{is} > 0\right\}, \min\left\{\frac{u_i - x_i}{-\tilde{A}_{is}} : i \in \mathcal{T}, \tilde{A}_{is} < 0\right\}\right\}.$$

A basis leaving variable $t \in \mathcal{T}$ is then chosen from the set of indices for which the minimal σ is attained. In a similar fashion it can be determined what the maximal decrease in variable x_s is, if x_s is a variable at its upper limit, i.e. $s \in \overline{\mathcal{T}}_u$. Here, a decrease of variable x_s by $0 \leqq \sigma \leqq u_s$ yields the new value of the i^{th} basic variable $x_i + \sigma \tilde{A}_{is}, i \in \mathcal{T}$. To ensure feasibility of the new basic variable $x_i + \sigma \tilde{A}_{is}$, one has to ensure that

$$0 \leqq x_i + \sigma \tilde{A}_{is} \leqq u_i \Leftrightarrow \sigma \leqq \begin{cases} -\frac{x_i}{\tilde{A}_{is}} & \text{if } \tilde{A}_{is} < 0 \\ \frac{u_i - x_i}{\tilde{A}_{is}} & \text{if } \tilde{A}_{is} > 0 \end{cases}$$

Therefore, σ is chosen as

$$\sigma = \min\left\{u_s, \min\left\{-\frac{x_i}{\tilde{A}_{is}} : i \in \mathcal{T}, \tilde{A}_{is} < 0\right\}, \min\left\{\frac{u_i - x_i}{\tilde{A}_{is}} : i \in \mathcal{T}, \tilde{A}_{is} > 0\right\}\right\}.$$

Again, a variable $t \in \mathcal{T}$ that leaves the basis is chosen from the set of indices for which the minimal σ is attained.

One iteration of the simplex algorithm is complete once the entering variable x_s replaces the leaving variable x_t in the basis, which is called a *pivot*. The basis is now $(\mathcal{T} \setminus \{t\}) \cup \{s\}$. The BFS $x_{\mathcal{T}}$ is updated by adjusting it by σ, the

amount of change in x_s.

Up to this point, this discussion is a derivation of the simplex algorithm with upper and lower bounds. Next, the special features of the network simplex algorithm are introduced.

For the network simplex algorithm any BFS can be interpreted as a spanning tree as the matrix $A_{\mathcal{T}}^{G}$ has full rank and thus represents a spanning tree rooted at node r (the column corresponding to the root node is always part of $A_{\mathcal{T}}^{G}$). As the basis represents a tree we also refer to it as *basic tree*. Introducing a new variable x_s into the basis means that an additional arc s is added to this spanning tree. This additional arc is part of an undirected *cycle* C in the tree. The flow conservation constraints imply that increasing the flow on arc s entails increasing flow on each arc of cycle C with the same orientation as s, and decreasing flow on each arc of C with the opposite orientation. On the other hand, decreasing the flow on arc s means decreasing the flow on each arc of C with same orientation as s and increasing it on those arcs with opposite orientation. In this manner, the flow around the cycle changes by σ, which is determined as biggest possible flow change respecting the lower and upper bounds on flow for each arc. Thus, σ increases until the flow on at least one arc reaches its lower or upper limit. Any of those arcs in the cycle that reach their upper or lower limit after changing the flow on the cycle by σ can be selected as leaving arc. We later explain how the leaving arc is selected to prevent the algorithm from cycling.

Example 1.2.1 continued: *The process of adding a non-basic arc to the network, updating the flow along the arising cycle and removing a basic arc from it, is illustrated for Example 1.2.1. The network from Example 1.2.1 is repeated in Figure 1.2. One BFS of the solution in Figure 1.2 is represented by the tree $\mathcal{T}$ in Figure 1.3. Node 3 was chosen as root node. In this figure, all basic arcs have flow 1. The green dashed non-basic arcs have flow equal to their upper bounds, namely 2, and the green dotted non-basic arcs have flow equal to their lower bound 0. Introducing the non-basic arc $s = 4 \to 6$ with flow 2 into the basic tree yields the cycle highlighted in Figure 1.4. When introducing s into the tree and decreasing the flow on s by $\sigma = 1$, arcs $5 \to 6$ and $3 \to 5$ reach their upper bound 2, and arc $3 \to 4$ reaches its lower bound 0. Any of the three arcs are candidates for leaving the basis. Selecting arc $5 \to 6$ to leave*

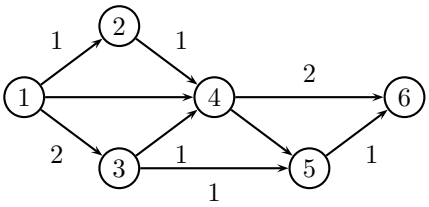


Figure 1.2. One feasible solution for the network from Example 1.2.1.

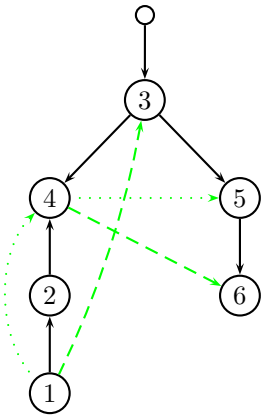


Figure 1.3. Corresponding basic tree structure $\mathcal{T}$ representing a BFS. Non-basic arcs at upper bound are dashed, those at lower bound are dotted.

the basis, we obtain the new BFS displayed in Figure 1.6. The network with new flow values is shown in Figure 1.5

Compared with the simplex algorithm, one advantage of the network simplex algorithm is that only arcs in the cycle need to be considered when identifying the basis leaving variable (or arc), rather than all variables (arcs) in the basis. The same applies when the BFS is updated, the flow changes only on arcs along cycle C. The network simplex algorithm is summarised in Algorithm 1 assuming the MCF problem is bounded. If it is not known whether a MCF problem is bounded, it is sufficient to check whether $\sigma < \infty$ in Step 8 or 12.

An initial solution as required by Algorithm 1, if it exists, can be obtained as a solution of an auxiliary problem such as a Two-Phase approach (not to be confused with the Two Phase Method to solve multi-objective integer problems also discussed in this chapter) or a Big-M approach (e.g. Bazaraa et al. 2005).

Similar to the simplex algorithm, the network simplex algorithm may *cycle*, i.e. move back and forth infinitely between BFSs with the same objective value. When this happens, we talk about *degeneracy* in the network. In case of the network simplex, *strongly feasible* trees (Cunningham 1976) are a tool to prevent cycling. A basic tree is called strongly feasible if it is possible to send one unit of flow from every node in the tree to the root node. If every arc with

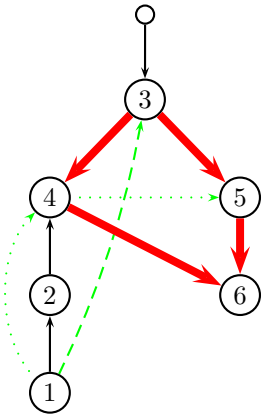


Figure 1.4. Basic tree structure $\mathcal{T}$ with non-basic arc $s = 4 \to 6$ and $C = 6 \leftarrow 5 \leftarrow 3 \to 4 \to 6$.

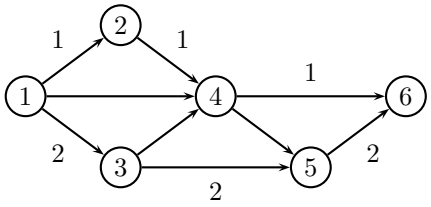


Figure 1.5. Feasible solution for network from Example 1.2.1 after performing a simplex pivot with $s = 4 \to 6$.

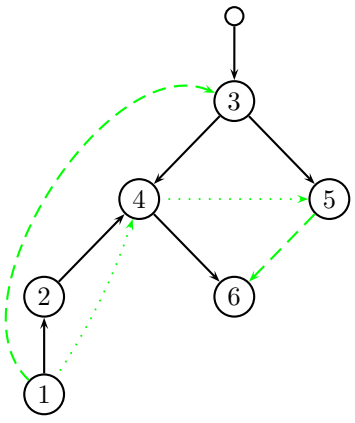


Figure 1.6. Basic tree structure $\mathcal{T}$ after simplex pivot with entering arc $4 \to 6$ and leaving arc $5 \to 6$.

Algorithm 1 Single-objective Network Simplex Algorithm

input: Initial BFS $x_{\mathcal{T}}$ and basic tree $\mathcal{T}$.
Compute reduced cost $\overline{c}$.
while $\{i \in \overline{\mathcal{T}}_l : \overline{c}_i < 0\} \cup \{i \in \overline{\mathcal{T}}_u : \overline{c}_i > 0\} \neq \emptyset$ **do**
  Choose $s \in \{i \in \overline{\mathcal{T}}_l : \overline{c}_i < 0\} \cup \{i \in \overline{\mathcal{T}}_u : \overline{c}_i > 0\}$.
  Find C, the basic arcs in the cycle obtained by adding arc s to tree $\mathcal{T}$.
  if $s \in \mathcal{T}_l$ **then**
    $\sigma_1 = \min\left\{\frac{x_i}{\tilde{A}_{is}} : i \in C, \tilde{A}_{is} > 0\right\}, \sigma_2 = \min\left\{\frac{u_i - x_i}{-\tilde{A}_{is}} : i \in C, \tilde{A}_{is} < 0\right\}$
    $\sigma = \min\{u_s, \sigma_1, \sigma_2\}$
    $\mathcal{L} = \left\{i \in C : \tilde{A}_{is} > 0, \sigma = \frac{x_i}{\tilde{A}_{is}}\right\} \cup \left\{i \in C : \tilde{A}_{is} < 0, \sigma = \frac{u_i - x_i}{-\tilde{A}_{is}}\right\}$
  else
    $\sigma_1 = \min\left\{-\frac{x_i}{\tilde{A}_{is}} : i \in C, \tilde{A}_{is} < 0\right\}, \sigma_2 = \min\left\{\frac{u_i - x_i}{\tilde{A}_{is}} : i \in C, \tilde{A}_{is} > 0\right\}$
    $\sigma = \min\{u_s, \sigma_1, \sigma_2\}$
    $\mathcal{L} = \left\{i \in C : \tilde{A}_{is} < 0, \sigma = -\frac{x_i}{\tilde{A}_{is}}\right\} \cup \left\{i \in C : \tilde{A}_{is} > 0, \sigma = \frac{u_i - x_i}{\tilde{A}_{is}}\right\}$
  end if
  if $s \in \mathcal{T}_l$ **then**
    $x_s = \sigma$; increase flow x_i on arc $i \in C$ by σ if i has the same orientation along the cycle as s, otherwise decrease flow on i by σ.
  else
    $x_s = u_s - \sigma$; decrease flow x_i on arc $i \in C$ by σ if i has the same orientation along the cycle as s, otherwise increase flow on i by σ.
  end if
  if $\sigma \neq u_s$ **then**
    Insert s into basic tree $\mathcal{T}$.
    Choose $t \in \mathcal{L}$; remove t from basic tree $\mathcal{T}$.
  end if
end while
output: Optimal BFS $x_{\mathcal{T}}$.

flow at its lower bound points towards the root, and every arc with flow at its upper bound points away from the root, the tree is strongly feasible. Trees remain strongly feasible if the basis leaving arc is chosen as the last eligible arc in the cycle starting from the node of the cycle that is closest to the root and traversing the cycle in direction of increasing flow.

An excellent implementation of the network simplex algorithm by Löbel (2004) is called `MCF`. This implementation constructs an initial feasible solution following the Big-M approach and also takes advantage of strongly feasible trees to prevent cycling. Whenever the network simplex algorithm is used in computational experiments in this thesis, it is derived from the `MCF` code for the

primal network simplex and adapted to the particular problem.

1.2.2 Parametric Network Simplex Algorithm for Continuous Bi-objective MCF Problems

Multi-objective continuous linear programmes can be solved using the simplex algorithm in an approach also known as *parametric network simplex algorithm.* The parametric simplex for continuous linear multi-objective programmes is discussed in Ehrgott (2005). Next, we will introduce the parametric network simplex algorithm for bi-objective MCF problems.

Using the symbols from the single-objective MCF Model (1.3), the linear programming formulation of MCF is extended to a *bi-objective continuous minimum cost flow (BCMCF)* problem

$$\begin{aligned} \min \quad & z(x) = \begin{pmatrix} (c^1)^\top x \\ (c^2)^\top x \end{pmatrix} \\ \text{s.t.} \quad & A^G x = b \\ & u \geqq x \geqq 0, \end{aligned} \tag{1.6}$$

where $c^1, c^2 \in \mathbb{R}^m$ now represent two different cost components.

A single-objective problem is derived from BCMCF by replacing the objectives by $\lambda_1 (c^1)^\top x + \lambda_2 (c^2)^\top x$. Solving this weighted sum problem for $\lambda_1, \lambda_2 > 0$ yields an efficient solution of the bi-objective problem. In fact, all efficient solutions can be obtained by varying $\lambda_1, \lambda_2 \in \mathbb{R}_>$. With the aim of describing a complete set of efficient solutions, it is sufficient to just obtain all extreme points so that the corresponding objective vectors (non-dominated points) are extreme points of the set of feasible objective vectors $\mathcal{Z} = \{z(x) \in \mathbb{R}^2 : x \in R^m, A^G x = b, u \geqq x \geqq 0\}$. A complete set of efficient solutions is then derived by convex combinations of all extreme points that define non-dominated faces of the polyhedron $\mathcal{Z}$.

In order to obtain those extreme efficient solutions, the parametric network simplex algorithm can be used. Such a network simplex algorithm is proposed by Sedeño-Noda and González-Martín (2000) and corrected in Sedeño-Noda et al. (2005).

The algorithm starts from the initial (efficient) solution $x^0 = x_{lex(1,2)}$. As there are two cost components (c_a^1, c_a^2) associated with each arc a, in the network simplex algorithm the reduced cost of each arc also consists of two components $(\overline{c}_a^1, \overline{c}_a^2)$. An arc to enter the basis is selected with *the smallest ratio of deterioration of* c^1 *and improvement of* c^2. Moving from solution $x_{lex(1,2)}$ to $x_{lex(2,1)}$ in this manner ensures that all extreme efficient solutions are passed on the way. In iteration j of the network simplex algorithm, candidate entering arcs (contained in the set $\mathcal{S}_j$) are selected with minimal ratio of their reduced costs derived from the current supported efficient solution x^j as indicated in Procedure 1. Again, we refer to $\overline{\mathcal{T}}_l$ and $\overline{\mathcal{T}}_u$ as the set of non-basic arcs at lower and upper bound with respect to the current BFS.

Procedure 1 `compute_entering_arcs`$[\overline{c}, x^j]$

1: **input**: Reduced costs $\overline{c} = (\overline{c}^1, \overline{c}^2)$ derived from current efficient solution x^j.
2: $\mu_j = \min \Big\{ \frac{\overline{c}_a^2}{\overline{c}_a^1} : a \in \overline{\mathcal{T}}_l \text{ with } \overline{c}_a^2 < 0 \text{ and } \overline{c}_a^1 > 0,$
$\frac{\overline{c}_a^2}{\overline{c}_a^1} : a \in \overline{\mathcal{T}}_u \text{ with } \overline{c}_a^2 > 0 \text{ and } \overline{c}_a^1 < 0 \Big\}$
3: Let $\mathcal{S}_j \subseteqq \overline{\mathcal{T}}_l \cup \overline{\mathcal{T}}_u$ be the set of non-basic arcs for which μ_j is attained.
4: **output**: Minimal ratio μ_j and set of non-basic candidate basic-entering arcs $\mathcal{S}_j$.

One of the candidate arcs $s \in \mathcal{S}_j$ is removed from $\mathcal{S}_j$ and enters the basis. By performing a simplex-pivot with entering arc s, the reduced costs may change. The reduced costs of all arcs remaining in $\mathcal{S}_j$ are updated according to the BFS obtained by pivoting s into x^j. As long as there are arcs remaining in $\mathcal{S}_j$ with

$$\overline{c}_a^2 < 0, \overline{c}_a^1 > 0 \text{ and } a \in \overline{\mathcal{T}}_l \text{ or } \overline{c}_a^2 > 0, \overline{c}_a^1 < 0 \text{ and } a \in \overline{\mathcal{T}}_u, \tag{1.7}$$

the process repeats as in Procedure 2. We show in the proof of Theorem 1.2.1 why those arcs in $\mathcal{S}_j$ satisfying (1.7) remain candidate entering arcs.

The next BFS x^{j+1} might define an extreme point $z(x^{j+1}) \in \mathcal{Z}$. We denote the last extreme efficient solution that was found by x^k. If for the new minimal ratio μ_{j+1} we have $\mu_{j+1} \neq \mu_k$, then x^{j+1} corresponds to an extreme efficient solution. If, on the other hand, $\mu_{j+1} = \mu_k$, then x^{j+1} is not extreme, i.e. $z(x^{j+1})$ corresponds to a supported non-extreme non-dominated point.

The parametric network simplex approach is summarised in Algorithm 2. The algorithm finishes when no candidate arcs to enter the basis can be found,

Procedure 2 `compute_next_BFS`$[\mathcal{S}_j, \overline{c}, x^j]$

1: **input**: Set of candidate basic-entering arcs $\mathcal{S}_j$ and reduced costs $\overline{c} = (\overline{c}^1_a, \overline{c}^2_a)$ derived from current efficient solution x^j.
2: **while** $\mathcal{S}_j \neq \emptyset$ **do**
3: Let s be the first arc in $\mathcal{S}_j$, set $\mathcal{S}_j = \mathcal{S}_j \setminus \{s\}$.
4: **if** $\overline{c}^2_s < 0, \overline{c}^1_s > 0$ and $s \in \overline{\mathcal{T}}_l$ OR $\overline{c}^2_s > 0, \overline{c}^1_s < 0$ and $s \in \overline{\mathcal{T}}_u$ **then**
5: Perform simplex-pivot with entering arc s.
6: Update $\overline{c}$.
7: **end if**
8: **end while**
9: **output**: Next BFS x^{j+1}.

i.e. when $\mathcal{S}_j = \emptyset$ in Step 7 of Algorithm 2. This indicates that the $lex(2,1)$-best solution is obtained.

Algorithm 2 Parametric Network Simplex Algorithm

1: **input**: Network (G, c, l, u) with $c = (c^1, c^2)$.
2: Compute $x^1 = x_{lex(1,2)}$.
3: $\mathcal{E}_{ex} = \{x^1\}$
4: Compute reduced costs $\overline{c}$ for x^1.
5: $(\mu_1, \mathcal{S}_1) =$ `compute_entering_arcs`$[\overline{c}, x^1]$
6: $j = 1, k = 1$
7: **while** $\mathcal{S}_j \neq \emptyset$ **do**
8: $x^{j+1} =$ `compute_next_BFS`$[\mathcal{S}_j, \overline{c}, x^j]$
9: Update $\overline{c}$ for x^{j+1}
10: $(\mu_{j+1}, \mathcal{S}_{j+1}) =$ `compute_entering_arcs`$[\overline{c}, x^{j+1}]$
11: **if** $\mu_{j+1} \neq \mu_k$ **then**
12: $\mu_{k+1} = \mu_{j+1}$ and $x^{k+1} = x^{j+1}$
13: $\mathcal{E}_{ex} = \mathcal{E}_{ex} \cup \{x^{k+1}\}$
14: $k = k + 1$
15: **end if**
16: $j = j + 1$
17: **end while**
18: **output**: Complete set of extreme efficient solutions $\mathcal{E}_{ex} = \{x^1, x^2, \ldots, x^{l-1}, x^l\}$.

The algorithm was originally proposed by Sedeño-Noda and González-Martín (2000). The authors do not include a check whether an efficient solution is extreme in their algorithm, but claim that every solution with $x^{j+1} \neq x^j$ obtained by one execution of Procedure 2 is an extreme efficient solution. It can be easily seen that this is not true. A BFS with $x^{j+1} \neq x^j$ is not necessarily an extreme efficient solution, i.e. $z(x^{j+1})$ is not necessarily an extreme point in

$\mathcal{Z}$. The BFS x^{j+1} might represent a supported non-extreme efficient solution with $z(x^{j+1})$ on the facet between two extreme points in $\mathcal{Z}$. It is therefore incorrect that the computation of *one* set $\mathcal{S}_j$ per extreme efficient solution x^j is sufficient. In Sedeño-Noda et al. (2005) the algorithm is corrected as follows: the computation of a new BFS (Procedure 2) is modified by *updating* $\mathcal{S}_j$ when updating the reduced costs $\bar{c}$ as long as the ratio μ_j does not change. For completeness, we include a proof of the correctness of Algorithm 2.

Theorem 1.2.1 *The set $\mathcal{E}_{ex}$ generated by Algorithm 2 is a complete set of extreme efficient solutions of BCMCF.*

Proof *Correctness of the parametric (network) simplex:* By definition, $\mathcal{X}_{SE}$ is the set of optimal solutions to (1.1) for $\lambda_1, \lambda_2 > 0$. Dividing (1.1) by λ_1, we obtain that (1.1) is equivalent to

$$\min\{(c^1 + \theta c^2)x : x \in \mathcal{X}\} \text{ and } \theta > 0, \tag{1.8}$$

where $\mathcal{X}$ is the set of feasible solutions that satisfy the constraints of the BCMCF Model (1.6). Therefore, it is sufficient to find one optimal solution of (1.8) for each $\theta > 0$. This is a parametric linear programme which can be solved by the parametric simplex algorithm (Dantzig and Thapa 1997).

As an initial solution, we use $x_{lex(1,2)}$, an extreme efficient solution. Clearly, $x_{lex(1,2)}$ is efficient and an optimal solution to (1.1) with $\lambda_1 = 1$ and $\lambda_2 = \varepsilon$ for sufficiently small $\varepsilon > 0$ (Isermann 1974), i.e. it is a supported efficient solution. Thus, there exists $\theta > 0$ such that $x_{lex(1,2)}$ is an optimal solution to (1.8).

The entering variables in the parametric network simplex algorithm are chosen from $\mathcal{S}_j$. At termination, the algorithm yields, for each $\theta > 0$ one optimal solution (a BFS) to (1.8), the parametric linear programme.

We obtain a set of candidate arcs $\mathcal{S}_j$ and introduce the arcs contained in $\mathcal{S}_j$ into the basis one after the other (if they are still eligible to enter the basis). Therefore, the correctness of the algorithm critically depends on the fact that after pivoting $s \in \mathcal{S}_j$ into the BFS x^j, the remaining arcs $a \in \mathcal{S}_j \setminus \{s\}$ that are still eligible, i.e. satisfy $\bar{c}^2_a < 0, \bar{c}^1_a > 0$ for $a \in \overline{\mathcal{T}}_l \cap \mathcal{S}_j$ or $\bar{c}^2_a > 0, \bar{c}^1_a < 0$ for $a \in \overline{\mathcal{T}}_u \cap \mathcal{S}_j$ (according to updated reduced costs), still have the minimal ratio

μ_j. Note that when pivoting an arc s with minimal ratio into the BFS of x^j, the new minimal ratio of all eligible arcs is equal to or greater than the previous one, i.e. $\mu_j \leqq \mu_{j+1}$. Again, this follows from Dantzig and Thapa (1997) as the optimal value of the parametric linear programme (1.8) is a continuous piecewise linear concave function, $f(\theta) = \min\{(c^1 + \theta c^2)x : x \in \mathcal{X}\}$, of the parameter θ (in our case with slopes given by the different values of μ).

We need to show that $a \in \mathcal{S}_j \setminus \{s\}$ either remains eligible with minimal ratio after the pivot, or is not eligible to enter the basis any more. Reduced costs for arc a are computed as $\overline{c}_a^k = c_a^k - \pi_{t(a)}^k + \pi_{h(a)}^k, k = 1, 2$, where π_i^k is the dual variable at node i for objective k. The duals are only updated for nodes in one of the two sub-trees obtained by removing arc s from the basic tree. The value of the dual at all nodes that are updated, changes by the same amount, namely the reduced cost of the basis entering arc. If none or both of the dual variables $\pi_{t(a)}$ and $\pi_{h(a)}$ change, the reduced costs computed by $\overline{c}_a^k = c_a^k - \pi_{t(a)}^k + \pi_{h(a)}^k, k = 1, 2$ do not change (as both $\pi_{t(a)}$ and $\pi_{h(a)}$ change by the same amount), and therefore the ratio remains the same.

If only one of the dual variables $\pi_{t(a)}$ changes to $\left(\pi_{t(a)}^1 + \overline{c}_s^1, \pi_{t(a)}^2 + \overline{c}_s^2\right)$, updating reduced costs yields

$$(\overline{c}_a^k)^{new} = c_a^k - \left(\pi_{t(a)}^k + (\overline{c}_s^k)^{old}\right) + \pi_{h(a)}^k = (\overline{c}_a^k)^{old} - (\overline{c}_s^k)^{old} \text{ for } k = 1, 2.$$

For both $s, a \in \mathcal{S}_j$, with respect to the "old" reduced costs, we have:

$$\mu_j = \frac{(\overline{c}_s^2)^{old}}{(\overline{c}_s^1)^{old}} = \frac{(\overline{c}_a^2)^{old}}{(\overline{c}_a^1)^{old}} \Rightarrow (\overline{c}_a^2)^{old} = \mu_j (\overline{c}_a^1)^{old} \text{ and } (\overline{c}_s^2)^{old} = \mu_j (\overline{c}_s^1)^{old}.$$

We assume that the updated reduced costs of a still satisfy $\overline{c}_a^2 < 0, \overline{c}_a^1 > 0$ for $a \in \overline{\mathcal{T}}_l \cap \mathcal{S}_j$ or $\overline{c}_a^2 > 0, \overline{c}_a^1 < 0$ for $a \in \overline{\mathcal{T}}_u \cap \mathcal{S}_j$. It follows for the new ratio of $\overline{c}_a^2$ and $\overline{c}_a^1$:

$$\frac{(\overline{c}_a^2)^{new}}{(\overline{c}_a^1)^{new}} = \frac{(\overline{c}_a^2)^{old} - (\overline{c}_s^2)^{old}}{(\overline{c}_a^1)^{old} - (\overline{c}_s^1)^{old}} = \frac{\mu_j \left((\overline{c}_a^1)^{old} - (\overline{c}_s^1)^{old}\right)}{(\overline{c}_a^1)^{old} - (\overline{c}_s^1)^{old}} = \mu_j.$$

A complete set of extreme efficient solutions $\mathcal{E}_{ex}$ is obtained by Algorithm 2: For every non-dominated extreme point $y \in \mathcal{Z}_N$ there exists $x \in \mathcal{X}$ with $z(x) = y$ and x is an optimal solution to (1.1) for some $\lambda_1, \lambda_2 > 0$. Hence, x is an optimal solution to (1.8) with $\theta = \frac{\lambda_2}{\lambda_1}$. Hence, the algorithm does find x or

a BFS $x' \in \mathcal{X}$ with $z(x) = z(x') = y$.

□

A complete set of efficient solutions of the continuous bi-objective MCF problem is obtained as convex combinations of all pairs of consecutive extreme efficient solutions $x^i, x^{i+1}, i = 0, \ldots, l-1$.

Remark 1.2.1 For single-objective (network) simplex algorithms, different speed-up techniques for the selection of basic entering variables (arcs) are known such as partial pricing. In partial pricing only a subset of all non-basic arcs is scanned and the basic entering arc is selected from this subset. This may have the advantage of a much faster pricing step to select the entering variable but at the cost of possibly necessitating more simplex iterations. Often, the advantage of a fast selection of the entering arcs outweighs the disadvantages.

It should be noted that within the parametric network simplex, *all* non-basic arcs have to be considered when choosing a basic entering arc, as an arc with minimal ratio of reduced costs has to be chosen. Thus speed-up techniques such as partial pricing cannot be used with the parametric simplex algorithm as all non-basic arcs need to be considered in every iteration.

1.3 Solving Bi-objective Integer Optimisation Problems

The two problems we study in Chapters 2 and 3 are both bi-objective *integer* network flow problems. Such bi-objective integer problems may have non-supported solutions. Non-supported solutions cannot be obtained by an algorithm that identifies solutions to the weighted sum problem (1.1), such as the parametric network simplex algorithm for continuous linear problems discussed in the previous section.

There is no alternative characterisation that helps identify such non-supported solutions. In this section two general strategies to find a complete set of efficient solutions of bi-objective integer problems with non-supported solutions are

discussed.

Some solution algorithms are problem-specific, in that they are dedicated to a particular problem type such as bi-objective labelling algorithms for bi-objective shortest path problems. Such problem specific algorithms will be explained in the corresponding chapters, whereas general solution schemes that are applicable to a wide range of bi- or multi-objective integer problems and relevant within the scope of this thesis are introduced in the following. One strategy for solving bi- or multi-objective integer problems is ranking, and another one is given by the Two Phase Method.

We now consider a *bi-objective integer* optimisation (MOIO) problem

$$\begin{aligned} \min \quad & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned} \tag{1.9}$$

with feasible set $\mathcal{X} \subseteqq \mathbb{Z}^n$.

1.3.1 Ranking

Among the first to present a ranking approach to solve a MOCO problem, namely the bi-objective shortest path problem, were Martins and Clímaco (1981); Clímaco and Martins (1982). The approach is easily explained for the bi-objective case. We will introduce bi-objective ranking and comment on multi-objective ranking at the end of this section.

Single-objective *ranking* or so-called *k-best* algorithms generate solutions for a minimisation problem ordered by non-decreasing objective values. The algorithm is initiated with an optimal solution x^0 with objective value $f(x^0)$. The ranking algorithm then generates solutions with non-decreasing values of the objective f:

$$x^0, x^1, x^2, \ldots \quad \text{with} \quad f(x^0) \leqq f(x^1) \leqq f(x^2) \leqq \ldots.$$

This process typically continues until the k^{th} solution is reached, hence the name k-best algorithm.

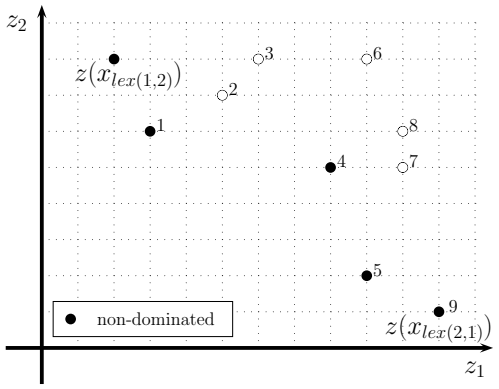


Figure 1.7. Illustration of ranking non-dominated solutions with $lex(1,2)$ objective: the ranking algorithm finds solutions with objective values ordered as numbered. Among them, the non-dominated points are highlighted.

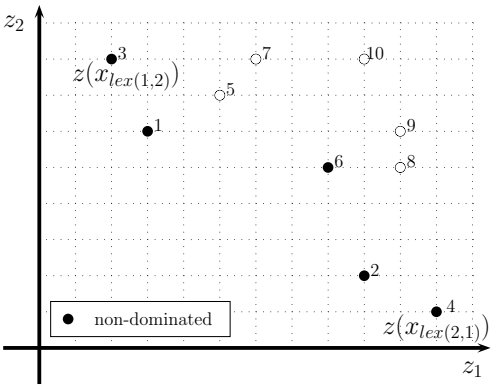


Figure 1.8. Illustration of ranking non-dominated solutions with weighted sum objective: the ranking algorithm finds solutions with objective values ordered as numbered. Among them, the non-dominated points are highlighted.

In order to apply such a ranking algorithm to solve a MOIO problem, a few adaptations need to be made, but the general ranking scheme remains. Firstly, the MOIO problem is reduced to a single-objective problem. Two possibilities are either using a lexicographic objective (Definition 1.1.3) or a weighted sum (1.1) of the two objectives. Secondly, one needs to ensure that ranking only stops when it can be guaranteed that all efficient solutions of the MOIO problem have been found. It is not clear a priori what value of k needs to be chosen to ensure all efficient solutions are generated.

Clímaco and Martins (1982) choose a lexicographic objective, e.g. $lex(1,2)$, which gives a first (and optimal) solution $x^0 = x_{lex(1,2)}$. Solutions are then ranked (with respect to the lexicographic objective $lex(1,2)$) until a solution with objective value equal to that of the other lexicographically optimal solution $x_{lex(2,1)}$ is obtained. Figure 1.7 shows how points $z(x^i)$ are found by the ranking algorithm. The numbers $i = 0, \ldots, 9$ next to the points indicate the order in which the points are obtained. In the depicted example ranking stops after point $z(x^9) = z(x_{lex(2,1)})$ is reached. During or after the ranking process, efficient solutions can be selected among all solutions found.

On the other hand, solutions could also be ranked according to a weighted sum objective. Weighting factors can be derived from the $lex(1,2)$- and $lex(2,1)$-

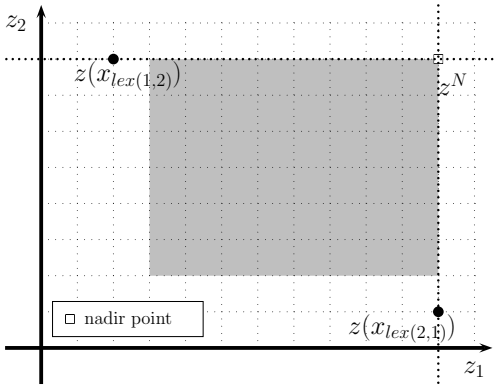


Figure 1.9. Bounds on z_1 and z_2.

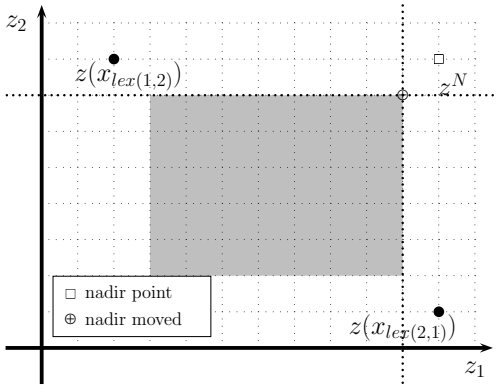


Figure 1.10. Improved bounds on z_1 and z_2.

best solutions, $x_{lex(1,2)}$ and $x_{lex(2,1)}$, by

$$\lambda_1 = z_2(x_{lex(1,2)}) - z_2(x_{lex(2,1)}) \text{ and } \lambda_2 = z_1(x_{lex(2,1)}) - z_1(x_{lex(1,2)}). \quad (1.10)$$

Now the ranking algorithm is applied to the problem with weighted sum objective $c^\lambda(x) = \lambda_1 z_1(x) + \lambda_2 z_2(x)$. The illustration of the ranking process yields the same points, but obtained in a different order, see Figure 1.8.

Upper bounds derived from $x_{lex(1,2)}$ and $x_{lex(2,1)}$ can be used to eliminate solutions $\hat{x}$ that cannot be efficient. Every solution $\hat{x}$ must satisfy

$$z_1(\hat{x}) \leqq z_1(x_{lex(2,1)}) \text{ and } z_2(\hat{x}) \leqq z_2(x_{lex(1,2)}), \quad (1.11)$$

otherwise it can be deleted. The point $z^N = (z_1(x_{lex(2,1)}), z_2(x_{lex(1,2)}))$ is called the *nadir point* of the BSP problem, the situation is illustrated in Figure 1.9.

The bounds (1.11) can be further improved by the fact that we are dealing with integer problems. The objective vector of efficient solutions, which are not equivalent to solutions obtained previously, can only be situated one unit below and one unit to the left of the nadir point z^N as indicated in Figure 1.10. Assuming all objective values are integer, we get the following improved bounds:

$$z_1(\hat{x}) \leqq z_1(x_{lex(2,1)}) - 1 \text{ and } z_2(\hat{x}) \leqq z_2(x_{lex(1,2)}) - 1. \quad (1.12)$$

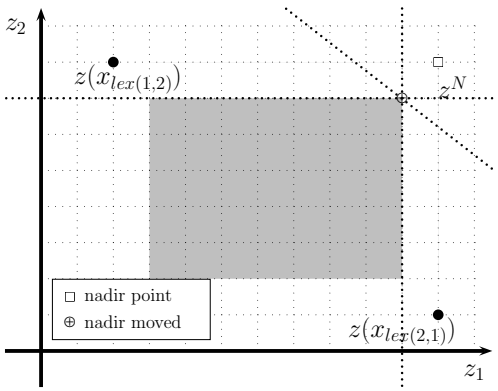


Figure 1.11. Initial weighted sum bound.

An upper bound δ of the weighted sum value of any efficient solution of the problem is the weighted sum value of $\big(z_1(x_{lex(2,1)}) - 1, z_2(x_{lex(1,2)}) - 1\big)$. It is calculated as $\delta = \lambda_1 \big(z_1(x_{lex(2,1)}) - 1\big) + \lambda_2 \big(z_2(x_{lex(1,2)}) - 1\big)$. The resulting bound is:

$$\lambda_1 z_1(\hat{x}) + \lambda_2 z_2(\hat{x}) \leqq \delta. \tag{1.13}$$

This weighted sum bound is indicated as the dotted line parallel to the straight line connecting $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ in Figure 1.11. Now the ranking algorithm can be stopped as soon as the first solution with weighted sum value exceeding δ is found.

Whenever the ranking algorithm returns a new solution that is not dominated by any of the solutions obtained prior to it, this solution is efficient. It can only be guaranteed that all efficient solutions have been found once the algorithm terminates. It is, however, possible to exploit the efficient solutions already found in order to improve the upper bound δ. We take advantage of the fact that every computed non-dominated point excludes a certain area of the objective space by domination.

Firstly, the *local nadir point* of two points $z^k = (z_1^k, z_2^k)$ and $z^l = (z_1^l, z_2^l)$ with $z_1^k < z_1^l$ and $z_2^k > z_2^l$ is defined to be $z^{LN} = (z_1^l, z_2^k)$. We consider straight lines parallel to the line connecting the points $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ through the local nadir point of any two consecutive currently available non-dominated points and $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ as indicated in Figure 1.12. The upper bound corresponds to the line through the point that has maximal

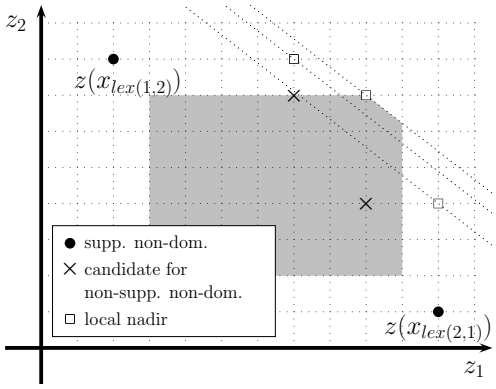


Figure 1.12. Weighted sum bounds (two candidate points).

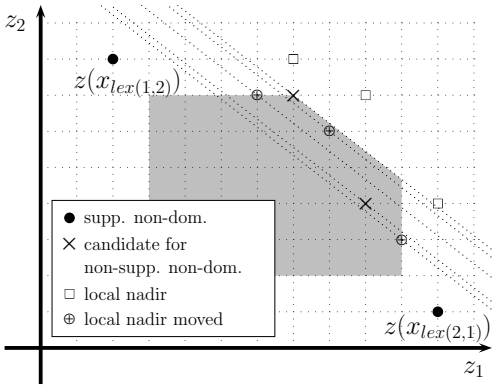


Figure 1.13. Improved weighted sum bounds (two candidate points).

distance from the straight line connecting $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$. Let $z^j = (z_1(x^j), z_2(x^j))$ with $j \in \{0, \ldots, r\}$ be the currently available non-dominated points ordered by increasing z_1, we also include the points $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ separately. This yields the upper bound Δ':

$$\begin{aligned}
\gamma'_1 &= \max\{\lambda_1 z_1(x^0) + \lambda_2 z_2(x_{lex(1,2)}), \lambda_1 z_1(x_{lex(2,1)}) + \lambda_2 z_2(x^r)\},\\
\gamma'_2 &= \max\{\lambda_1 z_1(x^{j+1}) + \lambda_2 z_2(x^j); j = 0, \ldots, r-1\},\\
\Delta' &= \max\{\gamma'_1, \gamma'_2\}.
\end{aligned}$$

Again, the upper bound can be improved by considering the point one unit below and one unit to the left of the local nadir point between each pair of consecutive points. Additionally, to ensure that *all* non-dominated solutions are found, we also have to include the currently available non-dominated points themselves as illustrated in Figure 1.13. This gives component γ_3 below. The improved upper bound Δ is:

$$\begin{aligned}
\gamma_1 &= \max\{\lambda_1(z_1(x^0) - 1) + \lambda_2(z_2(x_{lex(1,2)}) - 1),\\
&\qquad \lambda_1(z_1(x_{lex(2,1)}) - 1) + \lambda_2(z_2(x^r) - 1)\},\\
\gamma_2 &= \max\{\lambda_1(z_1(x^{j+1}) - 1) + \lambda_2(z_2(x^j) - 1), j = 0, \ldots, r-1\},\\
\gamma_3 &= \max\{\lambda_1 z_1(x^j) + \lambda_2 z_2(x^j), j = 0, \ldots, r\},\\
\Delta &= \max\{\gamma_1, \gamma_2, \gamma_3\}. \qquad (1.14)
\end{aligned}$$

Algorithm 3 Solving Bi-objective Integer Problems by Ranking with c^λ

1: **input**: Solutions $x_{lex(1,2)}$ and $x_{lex(2,1)}$.
2: $\mathcal{E} = \{\}$ /* Initialise set of candidate solutions */
3: $l = 1$ /* Initialise counter for non-dominated solutions */
4: Compute λ_1, λ_2 and $c^\lambda = \lambda_1 c^1 + \lambda_2 c^2$. /* See (1.10) for λ_1, λ_2 */
5: $\Delta = \delta$ /* Initial value for Δ; see (1.13) for δ */
6: $k = 0$ /* Initialise counter for k-best algorithm */
7: Compute optimal solution x^0 of problem $\min_{x\in\mathcal{X}} c^\lambda$.
8: **while** $c^\lambda(x^k) \leqq \Delta$ **do**
9: **if** $c(x^k)$ in feasible region (1.12) and not dominated by the objective vector of any element of $\mathcal{E}$ and x^k not equivalent to any $x \in \mathcal{E}$ **then**
10: Insert x^k into $\mathcal{E}$.
11: Update Δ. /* See (1.15) for Δ */
12: **end if**
13: $k = k + 1$
14: Compute k^{th}-best solution $x^k \in \mathcal{X}$ with $c^\lambda(x^{k-1}) \leqq c^\lambda(x^k)$.
15: **end while**
16: **output**: Complete set $\mathcal{E}$ of efficient solutions.

Note that in order to obtain a complete set of efficient solutions, it is sufficient to choose

$$\Delta = \max\{\gamma_1, \gamma_2\}. \tag{1.15}$$

We refer to Przybylski et al. (2008) for a more detailed presentation of the upper bounds (1.12) and (1.14).

Combining the ideas above yields a general solution method for bi-objective integer problems based on ranking with weighted sum objective c^λ, see Algorithm 3. In order to compute all efficient solutions, the upper bound Δ should be calculated as in (1.14) and all efficient solutions should be saved in the set $\mathcal{E}$ in Step 9 of Algorithm 3, rather than only those that are not equivalent to any other efficient solution.

Theorem 1.3.1 *The set $\mathcal{E}$ generated by Algorithm 3 is a complete set of efficient solutions of the bi-objective integer problem* (1.9).

Proof *All solutions contained in $\mathcal{E}$ obtained by Algorithm 3 are efficient*: whenever a solution x^k is first inserted into $\mathcal{E}$, its objective vector lies within the region described by constraints (1.12) and is not dominated by the objective

vector of any $x \in \mathcal{E}$, and x^k is not equivalent to any $x \in \mathcal{E}$. Suppose there exists $x \in \mathcal{X}$ dominating x^k, which implies $z(x) \leq z(x^k)$ and therefore $c^\lambda(x) < c^\lambda(x^k)$. Thus x would have been found before x^k by the ranking procedure.

When Algorithm 3 stops ranking solutions (i.e. when the upper bound Δ is exceeded), the set $\mathcal{E}$ contains a complete set of efficient solutions: the ranking procedure enumerates all solutions x with $c^\lambda(x) \leqq \Delta$. Among all enumerated solutions, a complete set of efficient solutions $\mathcal{E}$ is identified.

Assume that, after the ranking procedure stops, there exists an efficient solution $x^e \notin \mathcal{E}$ that is not equivalent to some $x \in \mathcal{E}$. Ranking stops as soon as $c^\lambda(x^k) > \Delta$. As the solution $x^e \notin \mathcal{E}$ was not obtained during the ranking process before it was stopped (otherwise x^e or an equivalent solution would be in $\mathcal{E}$), it follows that $c^\lambda(x^e) \geqq c^\lambda(x^k) > \Delta$.

We always have $|\mathcal{E}_i| \geqq 2$, as at least solutions $x_{lex(1,2)}$ and $x_{lex(2,1)}$, or solutions equivalent to them, exist. Assume the set $\mathcal{E} = \{x^1, x^2, \ldots, x^r\}$ is sorted by increasing z_1-values. Therefore $z_1(x^i) < z_1(x^e) < z_1(x^{i+1})$ and $z_2(x^i) > z_2(x^e) > z_2(x^{i+1})$ for some $i \in \{j : j = 0, \ldots, r-1$ and $z_1(x^{j+1}) - z_1(x^j) \geqq 2$ and $z_2(x^j) - z_2(x^{j+1}) \geqq 2\}$. In particular, $z_1(x^e) \leqq z_1(x^{i+1}) - 1$ and $z_2(x^e) \leqq z_2(x^i) - 1$. We can derive the following contradiction:

$$\lambda_1 z_1(x^e) + \lambda_2 z_2(x^e) \leqq \lambda_1(z_1(x^{i+1}) - 1) + \lambda_2(z_2(x^i) - 1) \text{ or } c^\lambda(x^e) \leqq \Delta.$$

This contradicts the existence of an efficient solution $x^e \notin \mathcal{E}$ within bounds (1.12) that is not equivalent to some solution in $\mathcal{E}$ and that was not obtained during the ranking procedure. □

The details of a ranking algorithm depend on the problem considered. For example, a so-called k-best flow ranking algorithm was proposed by Hamacher (1995) for integer minimum cost flow problems as explained in Chapter 3. In the same chapter, the ranking algorithm is used to find all solutions of a bi-objective integer minimum cost flow problem. In Chapter 2 on the other hand, a so-called *near shortest path* algorithm (Carlyle and Wood 2005) is used. It is related to a ranking algorithm, but does not obtain solutions with increasing objective value. Instead, all solutions within a certain range around the optimal solution are computed in no particular order.

Remark 1.3.1 A ranking-based algorithm to solve the multi-objective shortest path problem was proposed by Azevedo and Martins (1991). Here, shortest paths are ranked according to a lexicographic objective. Ranking continues until *all* s-t paths are ranked, and a set of efficient paths can be selected among them. The bounds discussed in this section cannot be extended to the multi-objective case as it is impossible to find the Nadir point a priori (without actually knowing all efficient solutions) when $p > 2$. For $p > 2$ one cannot simply identify the Nadir point by combining the p different lexicographic optima. For example, Korhonen et al. (1997) present a linear programme with three objectives for which the Nadir point cannot be derived from lexicographic optima.

1.3.2 Two Phase Method

An alternative to the ranking approach for solving bi-objective integer problems is the Two Phase Method (Ulungu and Teghem 1995). The two phase method is taking advantage of the problem structure by computing supported and non-supported solutions separately. In Phase 1, only supported efficient solutions are computed as indicated in Figure 1.14 for a bi-objective integer problem. It may even be sufficient to compute extreme supported efficient solutions, i.e. efficient solutions which define extreme points of the convex hull of the set of feasible objective vectors. Supported solutions can for example be computed as solutions to weighted sum problems (1.1). For many bi-objective problems, Phase 1 can be solved by solving several instances of a single-objective problem derived from the bi-objective problem. This has the advantage that algorithms for solving those single-objective problems which are often well-known, efficient, and fast, can be used. In Phase 2 the remaining efficient solutions need to be computed. Here, enumerative methods may have to be used. It is expected that methods to solve Phase 2 can be accelerated significantly by restricting the search area to relevant regions of the objective space by exploiting information gained in Phase 1. For a bi-objective problem, the search space in Phase 2 can be restricted to triangles given by two consecutive supported non-dominated points as indicated in Figure 1.15. The advantage of the Two Phase Method is that this restriction of the search space may lead to a much quicker execution of Phase 2 than a purely enumerative

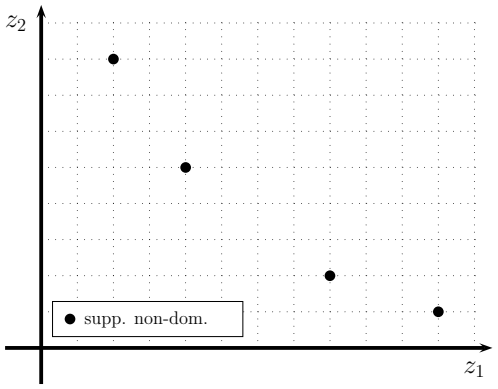


Figure 1.14. Supported non-dominated points.

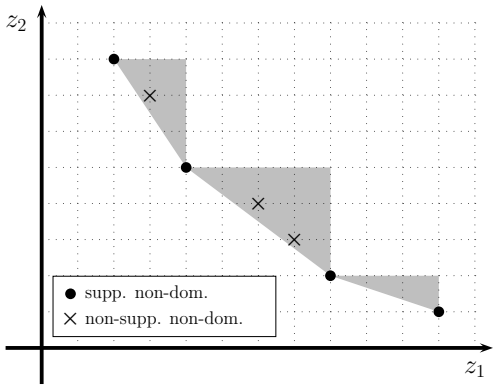


Figure 1.15. All non-dominated points.

approach on the entire feasible set would yield.

Remark 1.3.2 Although, in the literature, the Two Phase Method is mostly used to solve bi-objective problems, the general principle is not restricted to problems with two objectives. For multi-objective problems with three or more criteria, it is not trivial to keep track of the adjacency of supported solutions, the faces of $conv(\mathcal{Z})$ they define, and corresponding weights. Phase 1 algorithms for the multi-objective problem are discussed in Przybylski et al. (2009). Similarly, the derivation of correct bounds in Phase 2 is more involved than for the bi-objective problem. In Przybylski et al. (2007), the Two Phase Method is detailed for multi-objective problems and it is applied to the assignment problem with three objectives.

We now outline two different approaches for the calculation of supported solutions in Phase 1, a dichotomic approach for the general bi-objective integer problem (1.9) and a parametric network simplex based approach for the integer minimum cost flow problem. A similar simplex-based approach can be formulated for any integer linear bi-objective programme that is not based on a network structure by using the standard parametric simplex algorithm. The dichotomic approach, on the other hand, iteratively solves a single-objective (weighted sum) version of the bi-objective problem. Hence, standard solution algorithms for the single-objective problem can be used, whenever such algorithms exist. Subsequently, a Phase 2 approach based on ranking is presented.

Phase 1 – Dichotomic Approach

Phase 1 is dedicated to the computation of supported efficient solutions. This is achieved by solving several single objective problems in weighted sum formulation (1.1), within a *dichotomic* approach (first proposed by Cohon 1978; Aneja and Nair 1979; Dial 1979).

In a dichotomic approach, weights are chosen to obtain a supported non-dominated point that has the maximal distance to the straight line connecting the two initial points $z(x_{lex(1,2)})$ and $z(x_{lex(2,1)})$ as illustrated in Figure 1.16 for an example problem. The efficient solution $\hat{x}$ thus obtained leads to two new weighted sum problems: one between $z(x_{lex(1,2)})$ and $z(\hat{x})$ (yielding no new solution in the example), and one between $z(\hat{x})$ and $z(x_{lex(2,1)})$ (yielding one more solution in the example), see Figure 1.17. If the image of the obtained efficient solution of such a problem does not lie on the line connecting the images of the two supported solutions, $z(\overline{x})$ and $z(\tilde{x})$, defining it, two new sub-problems can be formulated. Otherwise, there are no more extreme supported points between $z(\overline{x})$ and $z(\tilde{x})$, so the current sub-problem does not have to be split up further. The dichotomic method iterates until all weighted sum problems and arising sub-problems have been solved and a complete set of the extreme supported efficient solutions is obtained.

The dichotomic method might not allow us to find all supported non-dominated points on $conv(\mathcal{Z})$ in case there are more than two solutions on the same face of $conv(\mathcal{Z})$. However, all extreme points will be computed. Missing non-extreme supported solutions are computed in Phase 2.

Phase 1 – Parametric Approach

An alternative to the dichotomic approach is the parametric approach, which is based on the (network) simplex method. Hence, this approach is only suitable to solve *bi-objective integer minimum cost network flow (BIMCF) problems*.

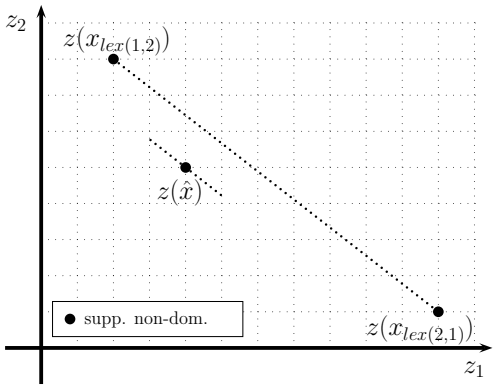


Figure 1.16. Dichotomic method, first iteration.

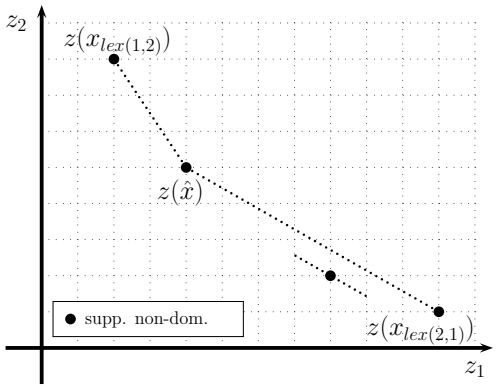


Figure 1.17. Dichotomic method, second iteration.

Using the symbols from Model (1.6), the BIMCF problem is defined as

$$\begin{aligned} \min \quad & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} \\ \text{s.t.} \quad & A^G x = b \\ & u \geqq x \geqq 0 \\ & x \text{ integer}, \end{aligned} \tag{1.16}$$

with objectives $z_1(x) = (c^1)^\top x$ and $z_2(x) = (c^2)^\top x$. We assume that $c^1, c^2 \in \mathbb{R}^m$, $b \in \mathbb{Z}^m$ and $u \in \mathbb{N}^m$. The matrix A^G is *totally unimodular*, as every square sub-matrix of A^G has determinant $+1, -1$, or 0. With the total unimodularity of the constraint matrix A^G and integrality of b and u it follows that every extreme efficient solution of BCMCF, the relaxation of BIMCF, is an integer solution, see e.g. Wolsey (1998). Therefore, any method to obtain all extreme points of BCMCF can be used to solve Phase 1 of BIMCF. We use the parametric network simplex approach here.

Note that in general Phase 1 can be solved by an approach based on the parametric simplex whenever a linear integer problem with totally unimodular constraint matrix is considered. We explain only the network simplex version here, as we are only dealing with network flow problems in subsequent chapters.

There exist several simplex-based approaches that obtain extreme efficient solutions of BCMCF, and hence of BIMCF, e.g. Lee and Pulat (1991); Pulat

et al. (1992); Sedeño-Noda and González-Martín (2000).

Phase 2 – Ranking

In Phase 2, all missing supported solutions as well as all non-supported solutions must be found. We adapt the pure ranking approach from Section 1.3.1 to solve Phase 2 . The advantage is that knowledge about dominated areas of the objective space can be exploited to obtain bounds that are more effective in restricting the ranking process. The bounds are derived similar to those obtained for the pure ranking approach.

Let $z^1, \ldots, z^s$, where $z^i = (z_1(x^i), z_2(x^i))$ and z^i are sorted by increasing z_1, be the non-dominated extreme points obtained in Phase 1. Now each pair of consecutive extreme efficient solutions x^i and x^{i+1} defines a triangle T_i given by the three points $(z_1(x^i), z_2(x^i))$, $(z_1(x^{i+1}), z_2(x^{i+1}))$, and $(z_1(x^{i+1}), z_2(x^i))$. The triangles T_i are the highlighted areas in Figure 1.15. For any solution x that lies within the triangle T_i, $z_1(x^i) \leqq z_1(x) \leqq z_1(x^{i+1})$ and $z_2(x^{i+1}) \leqq z_2(x) \leqq z_2(x^i)$ holds. Areas of the objective space that lie outside those triangles cannot contain non-supported points. We derive different weighting factors similar to those in (1.10) for each triangle T_i separately, rather than for the whole area contained between $x_{lex(1,2)}$ and $x_{lex(2,1)}$.

For each pair of neighbouring extreme points z^i and z^{i+1}, we define weighting factors

$$\lambda_1^i = z_1(x^{i+1}) - z_1(x^i) \text{ and } \lambda_2^i = z_2(x^i) - z_2(x^{i+1}).$$

Using λ_1 and λ_2 in a weighted sum problem (1.1), we obtain a single-objective MCF problem which has optimal solutions x^i, x^{i+1} (all convex combinations of x^i and x^{i+1} are optimal as well, but they are not necessarily all integer solutions, of course). We denote the weighted sum objective by $c^{\lambda^i} = \lambda_1^i z_1(x) + \lambda_2^i z_2(x)$.

Applying a ranking algorithm to the single objective problem $\min_{x \in \mathcal{X}} c^{\lambda^i}$, we can generate feasible solutions in order of their cost. The ranking algorithm is used to generate all feasible integer solutions in the current triangle until it can be guaranteed that all non-dominated points have been found. For this purpose, lower and upper bounds are derived analogously to Section 1.3.1.

Solutions are ranked as long as

$$c^{\lambda^i}(x) \leqq \delta_i \text{ with } \delta_i = \lambda_1^i(z_1(x^{i+1}) - 1) + \lambda_2^i(z_2(x^i) - 1). \tag{1.17}$$

Again, the upper bound δ_i can be improved with every efficient solution found that lies within the current triangle T_i. Let $\mathcal{E}_i = \{x^{i,0}, x^{i,1}, \ldots, x^{i,r}, x^{i,r+1}\}$ be a set of feasible non-equivalent solutions whose objective vectors are not dominating each other, and with image in the triangle T_i defined by the supported efficient solutions $x^{i,0} = x^i$ and $x^{i,r+1} = x^{i+1}$. Furthermore, let the elements of $\mathcal{E}_i$ be ordered by increasing z_1-value, so that $z_1(x^{i,0}) < z_1(x^{i,1}) < \ldots < z_1(x^{i,r}) < z_1(x^{i,r+1})$. This yields the upper bound Δ_i such that $c^{\lambda^i}(x) \leqq \Delta_i$ for all efficient solutions x with image in T_i:

$$\begin{aligned} \gamma_1^i &= \max\{\lambda_1^i(z_1(x^{i,j+1}) - 1) + \lambda_2^i(z_2(x^{i,j}) - 1), j = 0, \ldots, r\}, \\ \gamma_2^i &= \max\{\lambda_1^i(z_1(x^{i,j})) + \lambda_2^i(z_2(x^{i,j})), j = 0, \ldots, r\}, \\ \Delta_i &= \max\{\gamma_1^i, \gamma_2^i\}. \end{aligned} \tag{1.18}$$

Δ_i is derived similarly to Δ in (1.14) with x^i and x^{i+1} instead of $x_{lex(1,2)}$ and $x_{lex(2,1)}$. If it is sufficient to find a complete set of efficient solutions, Δ_i is chosen as

$$\Delta_i = \gamma_1^i. \tag{1.19}$$

A Phase 2 ranking algorithm basically follows the bi-objective ranking algorithm from Section 1.3.1. This algorithm must be run for every triangle T_i with different c^{λ^i} separately. To adapt Algorithm 3, $x_{lex(1,2)}$ and $x_{lex(2,1)}$ are replaced by x^i and x^{i+1}. Then, the algorithm can be used to obtain a complete set of efficient solutions $\mathcal{E}_i$ for each of the triangles T_i. Correctness of the Phase 2 ranking approach, i.e. that $\bigcup_{i=1,\ldots,s-1} \mathcal{E}_i$ is a complete set of efficient solutions, follows similar to the proof of Theorem 1.3.1.

Apart from ranking algorithms, problem specific algorithms can be applied in Phase 2. An example is the bi-objective shortest path problem (see Chapter 2) which can be solved with bi-objective labelling algorithms. Bounds can again reduce computation time significantly when compared to running the labelling algorithms without the preceding Phase 1. The bounds restricting solutions to triangles T_i can be incorporated into most Phase 2 algorithms,

such as the bi-objective labelling algorithms, for example. Depending on the specific solution algorithm at hand, it may also possible to take advantage of the weighted sum bounds $c^{\lambda^i}(x) \leqq \Delta_i$.

Chapter 2

Bi-objective Shortest Path Problems

The single objective shortest path problem is extensively studied in the literature (e.g. Gallo and Pallotino 1988; Cherkassy et al. 1996). Examples of applications of shortest path problems with more than one objective include transportation problems (Pallottino and Scutellà 1998), routing in railway networks (Müller-Hannemann and Weihe 2006), and problems in satellite scheduling (Gabrel and Vanderpooten 2002).

We consider the bi-objective shortest path (BSP) problem as the natural extension of the single objective case. BSP belongs to the class of MOCO problems. The BSP problem is an $\mathcal{NP}$-hard problem (Serafini 1986) and it also is intractable, i.e. the number of efficient solutions may be exponential in the number of nodes (Hansen 1980). Despite this fact, Müller-Hannemann and Weihe (2006) suggest that in practical applications with certain characteristics we can expect to find a reasonably small number of efficient solutions.

There are two main approaches to solving BSP problems. The first type includes enumerative approaches such as label correcting (e.g. Skriver and Andersen 2000; Brumbaugh-Smith and Shier 1989), label setting (e.g. Martins 1984; Tung and Chew 1988, 1992), and ranking methods (e.g. Martins and Clímaco 1981; Clímaco and Martins 1982). The second main approach is the Two Phase Method (Mote et al. 1991; Ulungu and Teghem 1995), which exploits the problem structure.

Labelling methods work similarly to their single objective (e.g. Bertsekas 1998) counterparts. For single-objective shortest path problems, every node has a single label that represents the distance from the start (source) node to this node. While the algorithm runs, the distance labels may change. At termination of the algorithm, however, they represent correct shortest path distances from the source node to every other node in the network. In BSP problems a node can have several labels, where each label represents a (different) path from the source node. The labels at the same node do not dominate one another. The set of efficient solutions of the BSP problem corresponds to all labels at the target node after a labelling algorithm finishes. In label correcting and label setting methods, either one label at a certain node is extended by all arcs out of that node (label-selection) or all labels at a node are extended simultaneously (node-selection).

Ranking methods may utilise single objective k-shortest path methods. According to the literature (Huarng et al. 1996; Skriver 2000) k-shortest path methods are not competitive with label correcting methods. Therefore, we investigate the application of a near-shortest path method by Carlyle and Wood (2005), which the authors successfully apply to the k-shortest path problem.

Next to enumerative approaches, a second type of solution approach is the Two Phase Method. In Phase 1, the extreme supported efficient solutions are computed. In Phase 2, the remaining efficient solutions are computed with one of the enumerative approaches mentioned before. The enumerative methods can be employed in a very effective way as enumeration can be restricted to small areas of the objective space, see also Chapter 1.

We present well-known strategies to solve the BSP problem and introduce the two Phase Method with near-shortest path ranking by adapting a near-shortest path approach in Phase 2. This is the first computational study in which all known solution approaches for BSP are compared on a large set of different test networks. In particular, the Two Phase Method is intensively studied by comparing different computational methods for each of its components. Our aim is to compare the performance of the different solution approaches. We investigate performance on two different artificial network structures and also on road networks, to include some real world network structures into our considerations as is done by Zhan and Noon (1998) for single objective shortest

path problems. This comparison is in contrast to some earlier studies, where only a single network type has been used. We are able to show that contrary to previous research results, the Two Phase Method is competitive with, if not better than, the traditional bi-objective labelling approaches in many cases.

Furthermore, we propose a technique to improve the run-time of labelling algorithms, called *bounded labelling*. Here, a standard labelling technique is modified to discard labels corresponding to unfinished paths, once it is known they can never lead to efficient paths anyway. We test the effects our modification has on run-time, which shows that the modification achieves significant improvements for many problem instances.

We also describe a novel application of BSP problems to modelling route choices of commuter cyclists. It is assumed that cyclists aim to minimise the distance travelled but also to maximise the attractiveness of a path. Attractiveness includes factors important to cyclist route choice, for example road gradient and safety.

In this chapter, BSP problems are introduced in Section 2.1 followed by a discussion of recent literature in Section 2.2. In Section 2.3 we present the different algorithms currently used to solve the BSP problem, namely label correcting, label setting, near-shortest path, and the Two Phase Method. Finally, numerical results are presented in Section 2.4. We introduce bounded labelling algorithms in Section 2.5. Section 2.6 is dedicated to cyclist route choice. All tables containing numerical results appear at the end of this chapter in Section 2.8.

The content of this chapter is based on Raith and Ehrgott (2009a), Raith (2008b), and Raith (2006).

2.1 Problem Formulation

In this section, terminology and basic theory of bi-objective shortest path problems are introduced.

Let $G = (\mathcal{V}, \mathcal{A})$ be a *directed graph* with a set of nodes (vertices) $\mathcal{V} = \{1, \ldots, n\}$ and a set of arcs $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$. Two positive costs $c_a = (c_a^1, c_a^2) \in \mathbb{N} \times \mathbb{N}$ are associated with each arc $a \in \mathcal{A}$. In a road network, for example, the costs c_a^1

and c_a^2 could represent time and distance for traversing arc a, respectively.

A *path* in G from node $i \in \mathcal{V}$ to node $j \in \mathcal{V}$ is a sequence $\{a_1, a_2, \ldots, a_l\}$ of arcs in $\mathcal{A}$ with tail node $t(a_1) = i$, head node $h(a_n) = j$, and $h(a_k) = t(a_{k+1}), i = 1, 2, \ldots, l-1$. The *bi-objective shortest path problem* (BSP) with *source node* $s \in \mathcal{V}$ and *target node* $t \in \mathcal{V}$ can be formulated as a network flow problem:

$$
\begin{aligned}
\min \quad & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} \\
\text{s.t.} \quad & \sum_{a \in \mathcal{A}, t(a)=i} x_a - \sum_{a \in A, h(a)=i} x_a = \begin{cases} 1 & \text{if } i = s \\ 0 & \text{if } i \neq s, t \\ -1 & \text{if } i = t \end{cases} \quad \text{for all } i \in \mathcal{V} \\
& x_a \in \{0, 1\} \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{for all } a \in \mathcal{A},
\end{aligned}
\tag{2.1}
$$

with $z_1(x) = \sum_{a \in \mathcal{A}} c_a^1 x_a$ and $z_2(x) = \sum_{a \in \mathcal{A}} c_a^2 x_a$. Here x is a vector of flow on the arcs and the first set of constraints represent flow balance at the different nodes. A balance of 1, -1, and 0 indicates that there exists a surplus of one unit of flow, a demand of one unit of flow, or neither of the two, respectively. The first set of constraints are flow conservation constraints, which ensure that one unit of flow is transported through the network from s to t. The second set of constraints restricts flow on an arc to be either 0 or 1. The arcs with flow value 1 form a path from s to t.

2.2 Literature on BSP Problems

Skriver (2000) contains a survey on BSP problems. The surveys on MOCO problems by Ehrgott and Gandibleux (2000) and Ehrgott and Gandibleux (2002) both include a section on shortest path problems. A brief survey on MSP problems is also contained in Tarapata (2007), the part on solution methods to compute a (complete) efficient set for BSP and MSP is identical with the two previous references.

In the following we discuss literature not yet covered in a survey. We focus on exact methods here, that is methods to obtain a complete set of efficient solutions. There are heuristic approaches to solve the BSP / MSP problem

(Stewart and White 1989) and methods to find an approximation of the set of efficient solutions (Warburton 1978; Hansen 1980; Tsaggouris and Zaroliagis 2006; Diakonikolas and Yannakakis 2007). Other methods take advantage of a utility function according to which an "optimal" solution can be obtained or help guide a decision maker to a particular solution (Henig 1985; Carraway et al. 1990; Murthy and Olson 1994; Modesti and Sciomachen 1998; Tarapata 2007). We do not include literature on multi-objective shortest path problems with other types of objectives than those of the BSP model (2.1). We comment only on recent literature on identifying a complete set of efficient solutions of BSP.

Martins and Santos (2000) discuss labelling algorithms for the multi-objective shortest path (MSP) problem with arbitrary arc costs. They prove boundedness and finiteness results for the MSP problem and also correctness of the label setting and label correcting approach. They present a generic labelling algorithm with label selection. They also propose a label setting algorithm based on node selection for acyclic networks, taking advantage of the fact that acyclic networks can be put in topological order.

Guerriero and Musmanno (2001) investigate label correcting and label setting methods for the multi-objective shortest path tree problem. They propose several strategies for label-selection and node-selection. Computational results are presented for two different classes of test problems. There are problem instances where label-selection is superior and others where node-selection is superior. Furthermore, label setting is superior for some instances, and label correcting is superior for others.

Sastry et al. (2003) propose several algorithms for multi-objective shortest path problems with positive and negative arc costs. First, the networks are checked for negative cycles which are detected by a repeated application (at most once for every objective) of some single objective shortest path algorithm that can detect negative cycles. If there is no negative cycle, Sastry et al. suggest to use a label correcting multi-objective shortest path algorithm with node-selection similar to the one presented by Brumbaugh-Smith and Shier (1989). Sastry et al. also propose two other label correcting approaches. They are both variations to the approach by Corley and Moon (1985). In each iteration of the algorithm, the labels at each node are updated from all predecessor nodes. The

algorithm stops when either none of the label sets is modified in an iteration or when after n iterations the existence of a negative cycle is asserted. In each iteration nodes are chosen randomly by Sastry et al. whereas Corley and Moon choose nodes in order of their indices $1, 2, \ldots, n$. The other variation by Sastry et al. is to change the manner in which label sets are updated, the approach is similar to Yen (1970). Now, each iteration is split into two phases. In the first phase, nodes are updated by labels at nodes with smaller index than the current node only, in the second phase nodes are updated by labels at nodes with bigger index. Sastry et al. remark that the first algorithm, label correcting similar to Brumbaugh-Smith and Shier (1989), performs best in practical tests.

Müller-Hannemann and Weihe (2006) investigate the cardinality of the set of efficient solutions that arises in practical applications. They examine the characteristics of shortest path problems in train networks with two and three objectives. They relate network and problem characteristics to the actual number of efficient solutions. They find that this number is very low despite the fact that bi-objective shortest path problems are in general intractable.

Paixão and Santos (2007) compare different strategies for labelling algorithms to solve multi-objective shortest path algorithms. Only labelling algorithms with label selection are considered. Label setting and label correcting are implemented with different data structures to investigate how labels are best stored and selected. Label correcting is implemented with two different methods of selecting the next label: FIFO, where labels are inserted at the back of a list and selected from its front, and what they call a DEQue, where labels are inserted in front whenever they are considered better (according to some function) than the current first label and at the back otherwise. For label setting, the next label is extracted from an ordered list, a so-called Dial data structure and a binary heap. The different implementations of labelling algorithms are compared for a large set of test instances with between two and 20 objectives. The study shows that label correcting with FIFO list is the overall best approach for smaller data sets, for label setting the Dial data structure performs best. For large data sets, again both label correcting approaches outperform the label setting ones.

Martins et al. (2007) propose a special shortest path ranking algorithm based

on what they call "shortest deviation paths". This algorithm is then used to solve the MSP problem by first ranking all paths and then choosing the efficient paths among all ranked paths. Computational studies with between two and ten objectives show that this improved ranking algorithm solves MSP slower than label correcting with label selection and FIFO list (Paixão and Santos 2007).

Paixão and Santos (2008) again solve the MSP problem with a ranking algorithm. Here, they introduce bounds on the ranking procedure which should drastically reduce computation time as not *all* paths need to be ranked any more. For a problem with p objectives $z_1, z_2, \ldots, z_p$, they propose to run the ranking algorithm p times. Each time, a different lexicographic objective is considered, so that each objective $i = 1, 2, \ldots, k$ is the preferred component once (it does not matter which order the remaining objectives appear in). Initially, a shortest path p^* with respect to objective $\sum_{i=1}^{p} z_i$ is computed. Clearly, for every non-dominated solution q of the problem, $z_i(q) \leqq z_i(p^*)$ holds for at least one $i \in \{1, 2, \ldots, p\}$. Therefore, ranking of paths with preferred objective i can be stopped as soon as the i^{th} objective value of the ranked path exceeds $z_i(p^*)$. Apart from initially computing p^* as above, they also propose to construct and update such an upper bound vector while the algorithm runs. They find this approach inferior to initially computing p^*. Using the ranking algorithm by Martins et al. (2007), they show that MSP is solved quicker by ranking with their new bounds than by label correcting and label setting algorithms for two out of three network types. For grid networks, however, their new ranking approach performs significantly worse than label correcting with label selection and a FIFO list.

Table 2.1 gives a summary of the literature related to the BSP and MSP problem.

2.3 Solution Methods for BSP Problems

Different methods to find a complete set of efficient solutions of BSP are investigated here. Three main approaches are identified and then compared in Section 2.4. One is bi-objective labelling, where we distinguish two different basic strategies. Label *correcting* with node-selection is identified as the most

Table 2.1. Literature on the exact solution of BSP/MSP problems.

Reference	Problem	Solution approach
Hansen (1980)	BSP	Label setting, label-selection
Martins and Clímaco (1981)	BSP	Ranking
Clímaco and Martins (1982)	BSP	Ranking
Martins (1984)	MSP	Label setting, label-selection
Corley and Moon (1985)	MSP	Label correcting node-selection
Hartley (1985)	MSP	Label correcting node-selection
Henig (1985)	BSP	Label correcting node-selection
Tung and Chew (1988)	BSP	Label setting, label-selection
Brumbaugh-Smith and Shier (1989)	BSP	Label correcting, node-selection
Azevedo and Martins (1991)	MSP	Ranking
Mote et al. (1991)	BSP	Two Phase Method
Tung and Chew (1992)	MSP	Label setting, label-selection
Huarng et al. (1996)	BSP	Computational comparison
Skriver and Andersen (2000)	BSP	Label correcting, node-selection
Martins and Santos (2000)	MSP	Label setting and correcting node- and label-selection
Guerriero and Musmanno (2001)	MSP	Label setting and correcting node- and label-selection
Sastry et al. (2003)	MSP	Label correcting, node-selection
Paixão and Santos (2007)	MSP	Label setting and correcting label-selection
Martins et al. (2007)	MSP	Ranking
Paixão and Santos (2008)	MSP	Ranking

successful approach to solve BSP problems by Skriver and Andersen (2000) whereas label *setting* was found superior by Guerriero and Musmanno (2001), for example. The second solution method is the adaptation of a near-shortest path procedure by Carlyle and Wood (2005) to BSP in lieu of a shortest path ranking algorithm. Finally, the Two Phase Method for BSP by Mote et al. (1991) is investigated. We compare several solution strategies that can be used to solve the different components of the Two Phase Method.

2.3.1 Bi-objective Label Correcting

A bi-objective label correcting method is a straightforward extension of the single objective version. The main difference for two or more objectives is that there may be several labels at a node, each corresponding to one path, which do not dominate one another.

Approaches to label correcting differ in whether they employ label-selection or node-selection. Label-selection means that all labels are treated separately. A label l at some node i is extended along all arcs a with tail node $t(a) = i$. The extended label $l + c_a$ is inserted into the label set at node $j = h(a)$ if it is not dominated. The new label may dominate other labels at node j which are deleted. Node-selection means that a node i is selected and *all* its labels are extended via all outgoing arcs. If the label set at node j changes, for example due to the addition of a new label $l + c_a$, it has to be reconsidered in a later iteration. This means that the node has to be scanned again and all its labels have to be extended via all outgoing arcs to enable the change of the label set at node j to propagate through the network.

Algorithm 4 summarises the bi-objective label correcting algorithm with node-selection. Initially, the only labelled node is the source node s with label set $Labels(s) = \{(0, 0)\}$. All labels at a particular node i are extended along all outgoing arcs a with $t(a) = i$. Dominated labels are eliminated from the labels extended from node i and the labels already present at the end node $j = h(a)$. The remaining labels form the new label set at node j. Whenever the label set of a node changes, the node has to be marked for reconsideration. At reconsideration, the mark of the node is deleted. When no nodes are marked for reconsideration any more, the algorithm terminates. When traversing an

Algorithm 4 Bi-objective Label Correcting

input: Graph $(\mathcal{V}, \mathcal{A})$, cost function $c = (c^1, c^2)$, and source node s.
$modNodes = \{s\}$: list of nodes with modified labels that have not yet been reconsidered, treated in FIFO order.
$Labels(s) = \{(0,0)\}$ and $Labels(i) = \emptyset, i \in \mathcal{V}\backslash\{s\}$: $Labels(i)$ is the list of labels at a particular node i.
while $modNodes$ is nonempty **do**
  Remove first node i from $modNodes$. /* FIFO */
  for all $a \in \mathcal{A}$ with $t(a) = i$ **do**
    $j = h(a)$
    $merge(Labels(i) + c_a, Labels(j))$ /* extend all labels at i by c_a and merge with labels at j, eliminating all dominated labels */
    if the label set of j has changed and $j \notin modNodes$ **then**
      Append j to $modNodes$. /* FIFO */
    end if
  end for
end while
output: Efficient path length from source node s to all other nodes, paths can be backtracked using labels.

outgoing arc from a node with multiple labels, every label has to be extended along this arc and tested for dominance with the labels of the end node of the arc, this operation is called *merging*. Merging is the most expensive component of a bi-objective label correcting algorithm. The label sets are ordered so that the first component is increasing to reduce computational effort of the merge operation, which in our case is $\mathcal{O}(|\mathcal{L}| + |\mathcal{M}|)$ when the sets $\mathcal{L}$ and $\mathcal{M}$ are merged (Brumbaugh-Smith and Shier 1989). We also implement the condition to detect dominance of the whole label set by Skriver and Andersen (2000).

Once the label correcting algorithm terminates, the set $Labels(t)$ contains all non-dominated path costs at the target node t. The corresponding efficient solutions (the paths) can be obtained by backtracking the appropriate labels. To facilitate this, we store the in-going arc with every label.

Despite the results of Guerriero and Musmanno (2001), we opt for node-selection, the approach also chosen by Skriver and Andersen (2000) (see also Brumbaugh-Smith and Shier 1989), which is described in Algorithm 4 above. We believe that it is more efficient from a computational point of view to compare whole label sets rather than just individual labels within a label correcting algorithm. We were able to confirm this presumption by implementing

label correcting with node-selection and with label-selection. Our numerical tests show that node-selection is the faster approach. The additional computational effort of maintaining individual labels, extending them via outgoing arcs a and merging each individual extended label with the label set at node $h(a)$ outweighs the reduction in run-time achieved by not having to consider all labels of a node even though only some of the labels in the set may have changed.

Label correcting with node-selection is implemented as follows. Label lists at each node are represented by linked lists. Linked lists have the advantage that dominated labels can easily be removed from the list. Labels are ordered according to $\leqq_{lex(1,2)}$ which ensures that the label sets can be compared efficiently. The list $modNodes$, which keeps track of which nodes to consider next (in FIFO order), is implemented as a circular list similar to the FIFO list of the single-objective label correcting algorithm in Pape (1974).

2.3.2 Bi-objective Label Setting

While one can choose between node-selection and label-selection when implementing a label correcting algorithm, label setting schemes only work with label-selection as it has to be guaranteed that the label of a non-dominated path is selected. To explain the label setting algorithm, only the main differences to the label correcting algorithm described in Section 2.3.1 are highlighted.

Out of all labels one that is guaranteed to *remain* non-dominated has to be selected. There are different rules of selection to guarantee this, see Tung and Chew (1988) and Paixão and Santos (2007). The selection of a lexicographically smallest label, for instance, complies with this requirement. All newly generated labels are tentative. In our implementation, in each iteration a lexicographically smallest label is selected among all tentative labels. This label is guaranteed to belong to an efficient path from source node s to the node i the label l belongs to. The label l at node i is extended via all outgoing arcs a with tail node $t(a) = i$ similar to the procedure described in Section 2.3.1 for label correcting. Only one label is extended at a time and compared with all labels at the head node $j = h(a)$. Dominated labels are deleted from the label

Algorithm 5 Bi-objective Label Setting

input: Graph $(\mathcal{V}, \mathcal{A})$, cost function $c = (c^1, c^2)$, and source node s.
$Labels(s) = \{(0,0)\}$ and $Labels(i) = \emptyset, i \in \mathcal{V}\backslash\{s\}$: $Labels(i)$ is the list of labels at a particular node i.
$TentativeLabels = \{(0,0) \triangleright s\}$: contains all tentative labels and the node i the label is associated with indicated by $\triangleright\, i$.
while $TentativeLabels \neq \emptyset$ **do**
  Remove a lex(1,2)-best label $(l_1, l_2) \triangleright i$ from $TentativeLabels$.
  for all a with $t(a) = i$ **do**
    $j = h(a)$
    $merge((l_1, l_2) + c_a, Labels(j))$ /* extend label (l_1, l_2) at i by c_a and merge with labels at j, eliminating all dominated labels */
    if the label set of j has changed **then**
      Insert new label $(l_1, l_2) + c_a \triangleright j$ into $TentativeLabels$.
      Remove all deleted labels at j from $TentativeLabels$.
    end if
  end for
end while
output: Efficient path length from source node s to all other nodes, paths can be backtracked using labels.

list at node j and also from the set of tentative labels.

A $lex(1,2)$-best element needs to be extracted from $TentativeLabels$ in every iteration. We initially implemented the set $TentativeLabels$ as an ordered list, where the first element is always a $lex(1,2)$-best label. However, this list implementation is inferior to a binary heap implementation. The bi-objective label setting algorithm LSET is summarised in Algorithm 5

We implement $TentativeLabels$ as a binary heap to facilitate the extraction of a $lex(1,2)$-best label from $TentativeLabels$. Among other data structures, heaps are used to efficiently store labels in single-objective label setting algorithms (Ahuja et al. 1993). Our binary heap is implemented based on a binary heap by Weiss (last visited 02/2009). Removing an element from the heap is a costly operation. We tried deleting labels from the heap in case they are dominated, or leaving them in the heap and only marking them as dominated. We found it more efficient to delete labels as the large number of labels can lead to a memory shortage that eventually slows down the computer. Also, the resulting heap structure becomes very large making insertion and extraction operations all the more time consuming.

2.3.3 Ranking – Near-Shortest Path

Methods such as the k-shortest path algorithm generate one path after the other, with increasing objective function values. When used to solve a bi-objective problem, among all generated paths the non-dominated ones are selected as discussed in Section 1.3.1.

Computing paths ordered by their length comes with a large computational effort, but for the solution of the BSP problem, the order in which solutions are computed does not matter, as long as all efficient solutions are obtained. According to the literature, k-shortest path approaches cannot be successfully applied to BSP problems as the cost of finding paths in order of their lengths is quite high (Huarng et al. 1996; Skriver 2000). Instead of a k-shortest path procedure, we use the near-shortest path method by Carlyle and Wood (2005), which aims at finding all paths the length of which is within a certain deviation ε from the optimal path length ω, thus having a maximal path length of $\delta = \omega + \varepsilon$. The near-shortest path algorithm does find all paths within a certain deviation of the optimal path value, but they are not computed in any particular order. On the basis of computational tests, Carlyle and Wood conclude that their near-shortest path routine even solves the k-shortest path problem faster than other algorithms dedicated to solving the k-shortest path problem. We use the implementation of the method `ANSPRO` by Carlyle and Wood, which the authors identify as best approach, and carry out some slight modifications.

In order to use the near-shortest path (NSP) procedure, a weighted sum of the two objectives in BSP, see (1.1), is considered. To this end weighting factors $\lambda_1 > 0$ and $\lambda_2 > 0$ are defined as in (1.10) in Section 1.3.1. The required $lex(1,2)$- and $lex(2,1)$-best solutions are determined in an initialisation phase. We investigate the usage of different algorithms in initialisation, see Section 2.3.4.

Upper bounds originating from the two lexicographically best solutions $x_{lex(1,2)}$ and $x_{lex(2,1)}$ can be used to restrict enumeration, refer to the bounds (1.12).

Algorithm 6 gives a description of the NSP algorithm for a directed graph $G = (\mathcal{V}, \mathcal{A})$ with source node s and target node t. A cost $c_a^\lambda > 0$ is associated with each arc $a \in A$, where $c_a^\lambda = \lambda_1 c_a^1 + \lambda_2 c_a^2$. The worst possible weighted

Algorithm 6 NSP

input: Graph $(\mathcal{V}, \mathcal{A})$, cost function $c = (c^1, c^2)$, source node s, and target node t.
$L(i)$: the weighted sum path length at i.
$d^1(i), d^2(i)$: length of the path at i for the first and second objective.
for all $i \in N$ **do**
  $d(i)$ = weighted shortest path distance from i to t
end for
$stack = s$
$L(s) = 0$ and $d^k(s) = 0; k = 1, 2$
while the stack is not empty **do**
  i = top node of *stack*
  if $nextArcOutOf(i) \neq \emptyset$ **then**
    a = next arc out of i; $j = h(a)$
    if $(L(i) + c_a^\lambda + d(j) \leqq \delta)$ and $(d^1(i) + c_a^1 \leqq z^1(x_{lex(2,1)}) - 1)$ and $(d^2(i) + c_a^2 \leqq z^2(x_{lex(1,2)}) - 1)$ **then**
      $L(j) = L(i) + c_a^\lambda$ and $d^k(j) = d^k(i) + c_a^k; k = 1, 2$
      if j is target node t **then**
        Save current candidate solution. /* possibly eliminating previous candidate solutions that are now dominated */
        Pop j from *stack*.
      else
        Put j on top of *stack*.
      end if
    end if
  else
    Pop i from *stack*. /* no more outgoing arcs */
  end if
end while
output: Efficient paths from node s to node t and their lengths.

sum value of a feasible efficient path is the weighted sum value of the improved nadir point $\delta = \lambda_1(z_1(x_{lex(1,2)}) - 1) + \lambda_2(z_2(x_{lex(2,1)}) - 1)$, see also Equation (1.13). We modify NSP slightly to integrate the bounds given in (1.12) on the respective objectives. We simply add two label sets d^1 and d^2 to keep track of the current value of the two objectives and thus allow for comparison with the respective upper bounds. See Algorithm 6 which includes our changes to the original NSP.

The NSP algorithm repeatedly computes candidate solutions $\hat{x}$ satisfying the bounds in (1.12) and $c^\lambda(\hat{x}) \leqq \delta$. Only after the algorithm terminates, we know that the remaining candidate solutions are indeed efficient. It is possible to

exploit candidate solutions in order to improve the upper bound δ by the upper bound on the weighted sum value, refer to the computation of Δ in Equation (1.14). We again take advantage of the fact that every computed candidate excludes a certain area of the objective space by domination, even though it may not represent a non-dominated point. In Algorithm 6, we can insert an additional step where δ is updated by Δ whenever a new candidate solution is computed. We insert the following step between steps 16 and 17:

$$\text{Compute } \Delta \text{ and update } \delta = \Delta.$$

The shortest path distances from all nodes i to t in Steps 4-6 in Algorithm 6 are computed with a single-objective label correcting algorithm in the implementation of NSP by Carlyle and Wood (2005). In addition to their implementation, we replace their label correcting algorithm by Dijkstra's algorithm and denote this approach by NSPD. For details on the two single-objective shortest path algorithms refer to the section on initialisation in Section 2.3.4.

2.3.4 Two Phase Method

The general idea of the Two Phase Method is discussed in Section 1.3.2. Here, it is applied to solve BSP problems. In Phase 1 extreme supported efficient solutions are computed. In Phase 2 the remaining supported and non-supported efficient solutions are computed with an enumerative approach. An initialisation phase to compute one or two initial solutions is necessary to start the Two Phase Method off. Different solution methods for initialisation, Phase 1, and Phase 2 are implemented and compared.

In Phase 1 two main approaches are pursued. On the one hand single objective label setting and label correcting shortest path methods are used to solve the single-objective problems arising in the dichotomic approach from Section 1.3.2. On the other hand a network simplex algorithm solves BSP in a parametric approach. In Phase 2, the ranking and bi-objective labelling approaches discussed in the previous sections are employed with the additional benefit of bounds derived in Phase 1.

Mote et al. (1991) propose a Two Phase Approach to solve the BSP problem

with a parametric network simplex approach in Phase 1 and a label correcting algorithm in Phase 2. Here, we investigate the usage of several different algorithms for initialisation, Phase 1, and Phase 2 in order to identify a good combination.

Initialisation

In the initialisation phase a $lex(1,2)$-best or $lex(2,1)$-best solution or both need to be computed, depending on which approach is chosen in Phase 1. Here, single objective shortest path problems are solved with appropriate objective functions: the relations $\leqq$ and $\geqq$ in the standard single-objective shortest path algorithms are replaced by $\leqq_{lex(1,2)}$ and $\geqq_{lex(1,2)}$ to compute $x_{lex(1,2)}$ (or by $\leqq_{lex(2,1)}$ and $\geqq_{lex(2,1)}$ to compute $x_{lex(2,1)}$). The following shortest path algorithms are investigated:

- The single objective label correcting algorithm is, for example, discussed in Bertsekas (1998). In the context of their near-shortest path algorithm, Carlyle and Wood (2005) implement such a label correcting algorithm where nodes are treated in FIFO order and the list to determine which node is considered next is circular (as proposed in Pape 1974). We modify the original implementation to incorporate the lexicographic objective. This approach will be denoted by "L".
- The single objective label setting algorithm is also discussed in Bertsekas (1998). A study by Cherkassy et al. (1996) compares several implementations of such label setting algorithms. One of the most successful algorithms is `DIKBD`, a double bucket implementation of Dijkstra's label setting algorithm (as proposed in Denardo and Fox 1979). We select this algorithm and modify the original implementation to adjust it to the lexicographic objective. This approach will be denoted by "D" in the following.

We also tried to solve the initialisation problem with a single-objective network simplex algorithm. We observe that all but the smallest problem instances are solved much faster with any of the two above single-objective labelling algorithms. Therefore, the simplex method will be omitted from the computational results section.

Phase 1

Both Phase 1 approaches are discussed in Section 1.3.2. We employ a dichotomic approach as well as the parametric network simplex approach.

Two different solution strategies are distinguished for the dichotomic approach.

- The first dichotomic approach uses the label correcting method (L) discussed above among the initialisation approaches to solve the single objective weighted sum problems that arise from the dichotomic approach. We denote the dichotomic label correcting approach by "LDIC".
- The second dichotomic approach uses the label setting algorithm (D) from the initialisation section above to solve the single objective weighted sum problems that arise from the dichotomic approach. We denote the dichotomic label setting approach by "DDIC".

When solving the BSP problem with a parametric network simplex approach, as described in Section 1.3.2, the BSP formulation (2.1) above has some disadvantages. Problems arise as the network simplex method performs many basis exchanges without an actual flow change because the flow on all basic arcs that are not part of the actual path from s to t is zero. If a basis exchange involves only those arcs, there is no flow change at all. To avoid this situation we use another formulation, the *bi-objective shortest path tree* (BSPT) problem. This formulation is also used by Mote et al. (1991):

$$
\begin{array}{lll}
\min & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} & \\
\text{s.t.} & \displaystyle\sum_{a \in \mathcal{A}, t(a)=i} x_a - \sum_{a \in \mathcal{A}, h(a)=i} x_a = \begin{cases} n-1 & \text{if } i = s \\ -1 & \text{if } i \neq s \end{cases} & \text{for all } a \in \mathcal{A} \\
& x_a \geqq 0 & \text{for all } a \in \mathcal{A} \\
& x \text{ integer} & \text{for all } a \in \mathcal{A}.
\end{array}
$$

By modifying the constraint set of BSP, we now state the problem of finding the shortest path from source node s to all other nodes, resulting in nonzero flow on all basic arcs. Although not every basis exchange leads to a change

of the shortest s-t path, it does lead to some change in the shortest path tree rooted at s. This approach ensures a flow change whenever the basis changes and degeneracy of the problem is avoided.

Whenever the shortest path from s to t changes, another efficient solution is found. This yields the third and final Phase 1 approach.

- The parametric network simplex approach is obtained by modifying the existing implementation of a network simplex algorithm, called `MCF` (Löbel 2004). The parametric network simplex algorithm is denoted by "SPAR" in the following.

Phase 2

In Phase 2 it is possible to benefit from the work already done in Phase 1 to significantly reduce computation time of the enumerative methods used. For each of the triangles defined for two consecutive adjacent extreme non-dominated points, bounds are derived as described in Section 1.3.2. In each triangle an enumerative shortest path method is used to obtain non-supported solutions (if there are any).

We investigate a label correcting method (LCOR), a label setting method (LSET) and a ranking method (NSP/NSPD). In Phase 2 the following different approaches are studied.

- Bi-objective label correcting (LCOR) as described in Section 2.3.1. The LCOR algorithm can be run for every pair of consecutive supported non-dominated points. Labels are discarded as soon as they violate any bounds. We find that a lot of effort is put into the enumeration of paths that are discarded at a very late stage of the algorithm. In particular, many paths are enumerated for every pair of consecutive solutions that do not end up within the bounds for any of them.

  Therefore, in Phase 2 we run LCOR just once (instead of once for every triangle), and discard labels that are not in any of the areas defined by two consecutive supported non-dominated points or that cannot be extended to end up within any of them. In contrast to using LCOR in Phase 2, the original LCOR algorithm can only delete labels that

are dominated by other labels at some node, but there are no "global" bounds on objective values.

- Bi-objective label setting (LSET) as described in Section 2.3.2 is employed in a similar fashion as LCOR above.

- Near-shortest path (NSP and NSPD) as described in Section 2.3.3, is executed for every pair of consecutive supported non-dominated points $z(x^i)$ and $z(x^{i+1})$. The upper bounds are calculated in terms of x^i and x^{i+1} instead of $x_{lex(1,2)}$ and $x_{lex(2,1)}$ as explained in Section 1.3.2 on solving Phase 2 with a ranking algorithm. Paths are only expanded if they do not violate any bounds. Due to the lower bounds considered in NSP/NSPD, paths can often be discarded early during computations.

2.4 Numerical Results

We investigate the performance of the different solution methods on three different kinds of networks. Firstly, we introduce the types of networks considered and then present computational results.

2.4.1 Test Instances

We investigate three different network types: grid networks, random NetMaker networks and road networks. We also experiment with networks generated by NETGEN (Klingman et al. 1974), which we modify to incorporate two costs for each arc. The networks thus generated have very few efficient paths, often only between one and three. Therefore NETGEN networks were not included in our computational experiments.

Grid Networks

Nodes are arranged in a rectangular grid with given height h and width w. Every node has at most four outgoing arcs (up, down, left and right), to its immediate neighbours. Only nodes on the boundary of the grid have less outgoing arcs. There are two distinct nodes beyond the grid structure: a source

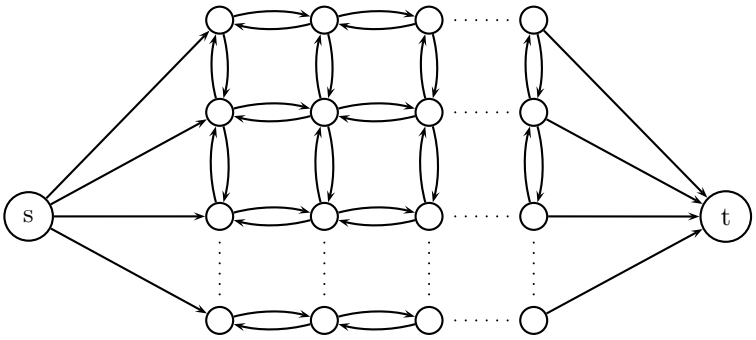


Figure 2.1. Structure of grid networks.

node s and a target node t. There is an arc from s to every node on the left margin of the grid and an arc from every node of the right margin of the grid to the target node t. Figure 2.1 shows the structure of grid networks. The costs (c_a^1, c_a^2) for arc a are chosen randomly from a discrete uniform distribution with $c_a^k \in \{1, 2, \ldots, 10\}, k = 1, 2$. Carlyle and Wood (2005) use grid networks for numerical tests on NSP algorithms. Refer to Table 2.2 for a listing of problem instances. Instances G15-G33 are grid networks with approximately the same number of nodes, but varying in width and height.

NetMaker

Skriver and Andersen (2000) propose an alternative, NetMaker, to using a pure random network generator such as NETGEN. They state that NETGEN generates networks containing very few efficient paths, an observation we agree with. Here, nodes are numbered from 1 to n, where node 1 is the source node, node n is the target node. We use a random number generator that generates discrete uniformly distributed random numbers. NetMaker networks are constructed by first generating a random Hamiltonian cycle to ensure that the network is connected. Then a random number of arcs out of every node is generated, in between a minimum and maximum number of outgoing arcs. An arc out of node i can only reach nodes j with $j \in [i - \lceil \frac{I_{node}}{2} \rceil, i + \lceil \frac{I_{node}}{2} \rceil]$, where I_{node} denotes the *node interval*, the maximum allowed range for an arc. Arc costs are determined randomly. It is randomly chosen whether $c_a^1 \in$

Table 2.2. Grid network test problems.

Name	$h \times w$	Nodes	Arcs	$\lvert\mathcal{Z}_N\rvert$	Name	$h \times w$	Nodes	Arcs	$\lvert\mathcal{Z}_N\rvert$
G1	30×40	1202	4720	37	G15	2450×2	4902	19596	6
G2	20×80	1602	6240	80	G16	1225×4	4902	19592	6
G3	50×90	4502	17820	124	G17	612×8	4898	19586	10
G4	90×50	4502	17900	46	G18	288×17	4898	19550	15
G5	50×200	10002	39600	290	G19	196×25	4902	19550	18
G6	200×50	10002	39900	12	G20	140×35	4902	19530	32
G7	100×150	15002	59700	149	G21	111×44	4886	19448	54
G8	150×100	15002	59800	122	G22	92×53	4878	19398	53
G9	100×200	20002	79600	247	G23	79×62	4900	19468	77
G10	200×100	20002	79800	132	G24	70×70	4902	19460	93
G11	200×150	30002	79800	204	G25	62×79	4900	19343	95
G12	50×50	10002	39600	52	G26	53×92	4878	19320	93
G13	100×100	10002	39800	113	G27	44×111	4886	19314	137
G14	200×200	40002	159600	309	G28	35×140	4902	19320	209
					G29	25×196	4902	19208	244
					G30	17×288	4898	19008	371
					G31	8×612	4898	18360	819
					G32	4×1225	4902	17150	1383
					G33	2×2450	4902	19596	1594

$\{1, 2, \ldots, 33\}$ or $c_a^1 \in \{67, 68, \ldots, 100\}$ and a number in the chosen interval is randomly allocated as cost. The cost c_a^2 is then randomly chosen from the other interval. We include three modifications to the structure of NetMaker:

a) Penalise the cycle: arc weights c_a^k, $k = 1, 2$ as above but for all arcs in the Hamiltonian cycle, choose c_a^k, $k = 1, 2$ randomly in $\{1, 2, \ldots, 10000\}$.

b) Balance outgoing arcs: to make NetMaker networks more comparable to grid networks, we enforce that roughly half of the arcs out of a node go to nodes with higher node numbers and half of them to nodes with lower numbers. Arc weights are chosen like in a) for all arcs of the Hamiltonian cycle, for all other arcs choose $c_a^k \in \{1, 2, \ldots, 10\}, k = 1, 2$.

c) More penalty on cycle: for all arcs that are part of the Hamiltonian cycle $c_a^k = 10000$, $k = 1, 2$. Everything else is the same as in b).

The structure of NetMaker networks is illustrated in Figure 2.2 and problem instances are described in Table 2.3. The instances were constructed to wrap around, so that arcs from nodes with low numbers that reach backwards may connect to a node with very high number, i.e. close to the target node. Note

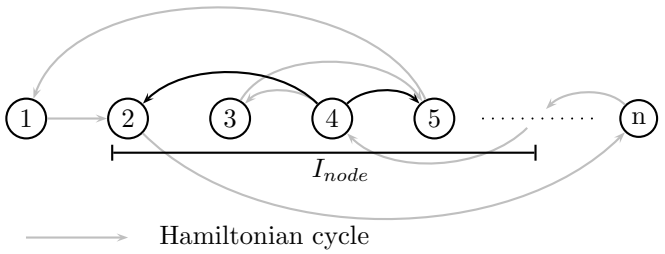


Figure 2.2. Structure of NetMaker networks.

that this may not be exactly the network structure the authors of Skriver and Andersen (2000) had in mind originally.

Road Networks

The road networks of the states of the US were extracted by Schultes (2005) from US Census (2000). We use road networks to test our methods on real world data. In the original data, networks are undirected and we convert them into directed networks by duplicating arcs. We also add a Hamiltonian cycle with high arc costs to ensure connectedness of the networks. In the original data, there is not always a path from a node to every other node. This happens for example for the Rhode Island data, as there are a few islands that are not connected to the mainland via roads. Each arc a is equipped with arc costs where c_a^1 is the time needed to travel the arc and c_a^2 is the travel distance in metres. Travel time is determined by multiplying the travel distance of an arc by one of four different road quality factors. Source and target node are chosen randomly from a discrete uniform distribution. Figure 2.3 shows the road network of Washington DC, the smallest of the road network instances. In the figure, roads of type "primary highway with limited access" and "primary road without limited access" were combined into the group "primary road".

We run tests with three different kinds of road networks. We use the networks of the states Washington DC, Rhode Island, and New Jersey, of which the network sizes are listed in Table 2.4. For each road network we test nine instances with different (randomly chosen) source and target nodes.

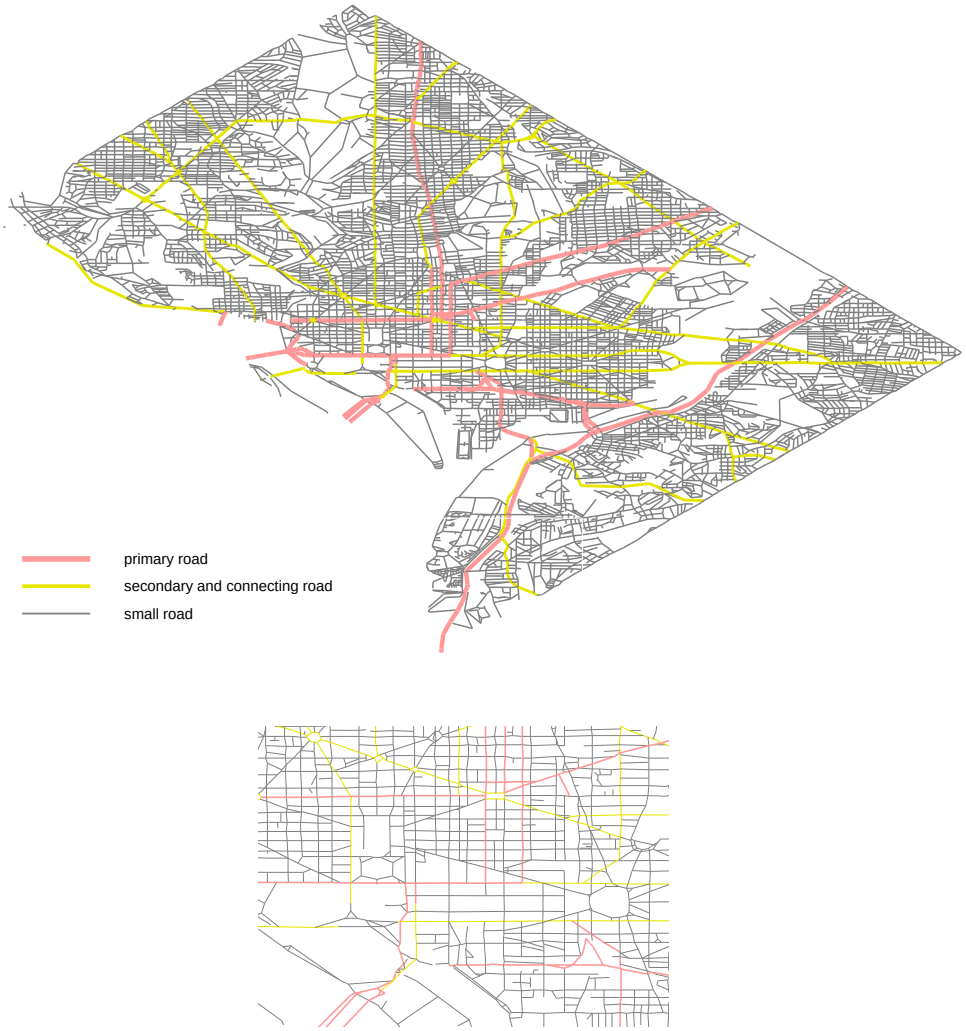


Figure 2.3. Road network of Washington DC and close-up of the area around the White House.

Table 2.3. NetMaker network test problems.

			Outgoing arcs		Var a)		Var b)		Var c)	
Name	Nodes	I_{node}	min	max	Arcs	$\lvert\mathcal{Z}_N\rvert$	Arcs	$\lvert\mathcal{Z}_N\rvert$	Arcs	$\lvert\mathcal{Z}_N\rvert$
NM1	3000	20	5	15	31559	6	31502	1	31646	3
NM2	3000	20	1	20	33224	8	33122	1	33229	4
NM3	3000	50	5	15	31345	9	31548	2	31775	2
NM4	3000	50	1	20	33536	15	32641	3	32963	4
NM5	3000	50	10	40	76095	6	76924	3	77388	3
NM6	7000	20	5	15	73524	6	73940	1	73575	2
NM7	7000	20	1	20	77024	5	76775	3	76547	3
NM8	7000	50	5	15	73676	3	73282	2	73369	3
NM9	7000	50	1	20	76821	7	77518	1	76658	3
NM10	7000	50	10	40	178476	6	178292	6	180611	4
NM11	14000	20	5	15	146598	6	147388	2	146979	2
NM12	14000	20	1	20	154159	6	154115	4	154252	1
NM13	14000	50	5	15	146919	2	146900	2	147187	1
NM14	14000	50	1	20	153742	17	154213	2	153068	4
NM15	14000	50	10	40	357866	7	358264	3	356367	3
NM16	21000	20	5	15	220313	5	220685	3	220794	3
NM17	21000	20	1	20	231402	4	230403	1	230432	1
NM18	21000	50	5	15	220687	7	219606	3	219931	1
NM19	21000	50	1	20	230497	4	231876	2	232465	1
NM20	21000	50	10	40	534288	5	536151	3	533980	3

Table 2.4. Road network test problems.

Name	State	Nodes	Arcs	$\lvert\mathcal{Z}_N\rvert$: average	Min	Max
DC1-DC9	Washington DC	9559	39377	3.33	1	7
RI1-RI9	Rhode Island	53658	192084	9.44	2	22
NJ1-NJ9	New Jersey	330386	1202458	10.44	2	21

2.4.2 Results

All numerical tests are performed on a Linux (Fedora Core 6, kernel 2.6.20) computer with 2.40GHz Intel® Core™2 Duo processor and 2GB RAM. We use the gcc compiler (version 4.1.1) with compile option -O3. The methods are implemented in C. We adapt program code from Carlyle and Wood (2005) for NSP, NSPD and L. For D we adapt program code for the `DIKBD` algorithm presented in Cherkassy et al. (1996). The network simplex is a modification of `MCF` (by Löbel 2004). When measuring run-time, we disregard the time it takes to read the problem from a problem file. Run-time does include the generation of all non-dominated path labels together with the actual paths. The only exception are LCOR and LSET, where the paths can be obtained by backtracking the labels at each node, here the time for the backtracking

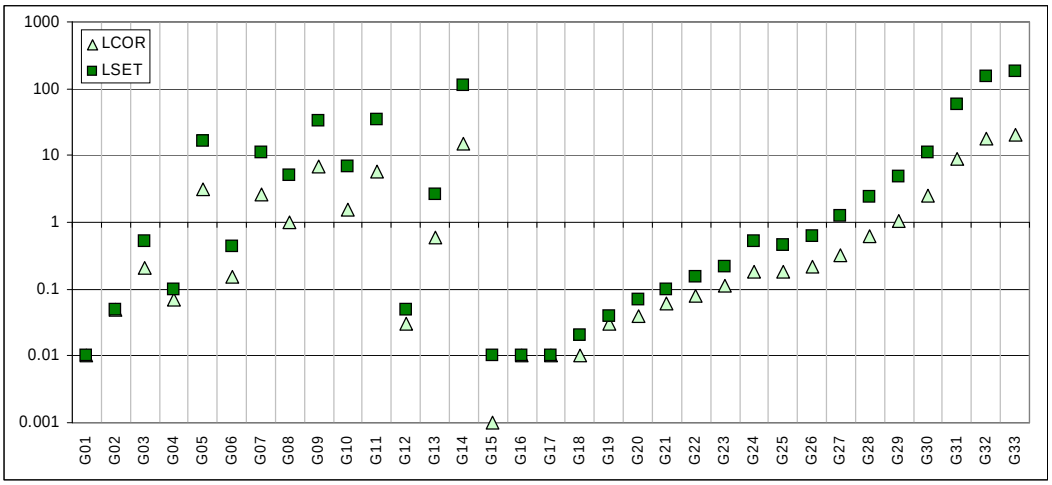


Figure 2.4. LCOR vs LSET - grid networks; logarithmic scale; 0.001 represents run-time < 0.01.

process is not included in the run-time. Run-time is measured with a precision of 0.01 seconds, any run-time less than 0.01 seconds will appear in the tables as 0.00. All figures are in logarithmic scale, in order to display all run-times in the same figure. We also modify run-times that appear as 0.00 in tables to 0.001 in figures. Whenever the run-time exceeds 3600 seconds, the computation is stopped, this is indicated in tables by "-". We present run-times in figures in this section. The corresponding tables can be found in Section 2.8 at the end of this chapter.

We first compare the performance of LCOR and LSET. We then comment on the performance of NSP and NSPD. We also identify the best approaches to use in initialisation, Phase 1 and Phase 2 of the Two Phase Method. Finally, we compare the Two Phase Method with the best bi-objective labelling approach and the best near-shortest path approach.

Best Bi-objective Labelling Approach

A comparison of run-times of LCOR and LSET is shown in Figures 2.4 - 2.6 and Table 2.10 in Section 2.8. LSET performs better than LCOR only for road networks. LCOR performs better than LSET for grid and most NetMaker networks. Despite the superior performance of single-objective label setting (D) this is an expected result. In a BSP problem, there are significantly more

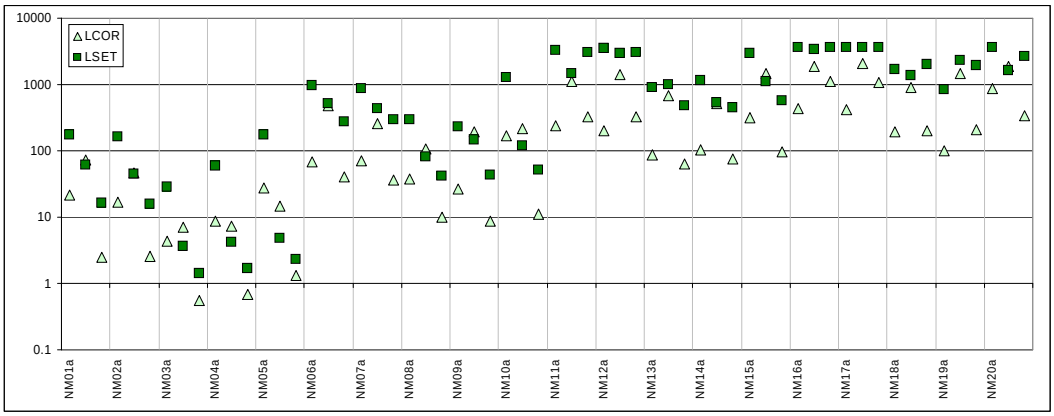


Figure 2.5. LCOR vs LSET - NetMaker networks.

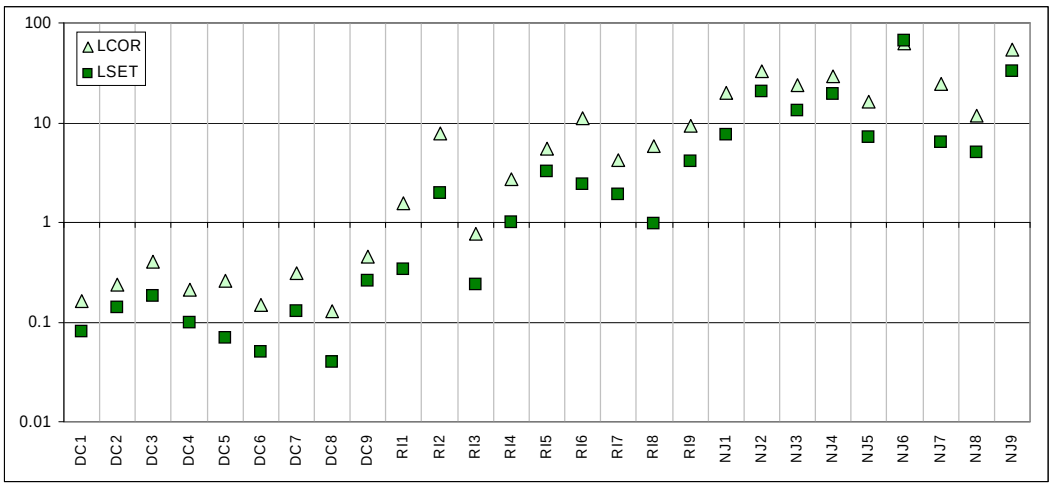


Figure 2.6. LCOR vs LSET - road networks.

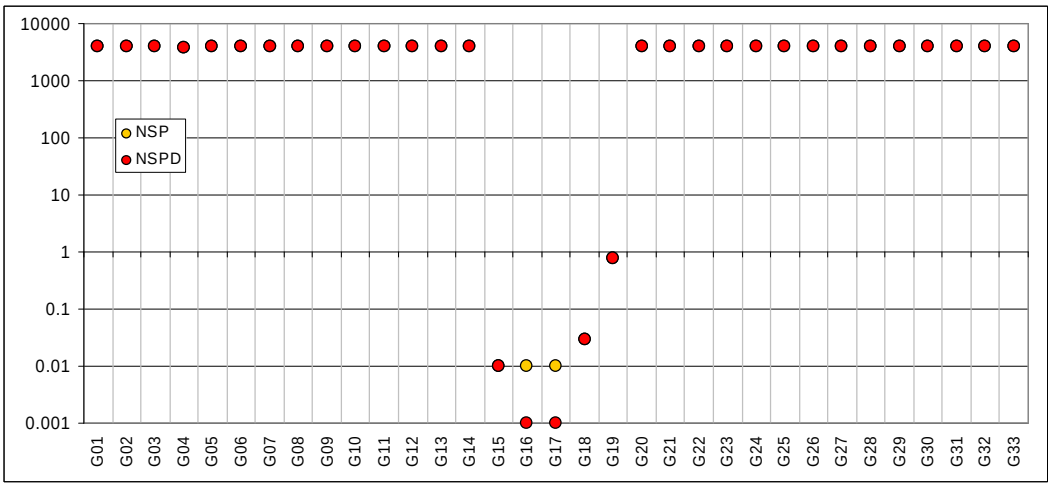


Figure 2.7. NSP vs NSPD - grid networks; run-times of NSP and NSPD are identical where only the red marker is visible; logarithmic scale; 0.001 represents run-time < 0.01.

labels than in a single-objective problem, as there may be several labels at every node. A label setting algorithm entails the additional effort of extracting in every iteration a lexicographically minimal label among all the tentative labels that currently exist at all nodes. Another advantage of our implementation of LCOR is that all labels at a node i are extended along arc a with $t(a) = i$ and compared to all labels at node $h(a)$. A label setting algorithm only allows to extend and compare one label at a time again presenting an advantage in run-time for LCOR.

Guerriero and Musmanno (2001) report that bi-objective label setting outperforms label correcting for random network instances with between 500 and 1000 nodes and high density, i.e. big ratio m/n. In our results, however, problems with high density such as NetMaker instances are solved particularly badly by the label setting algorithm. In Guerriero and Musmanno (2001), LSET and LCOR also perform similar for grid networks (which are different to ours as they contain additional random arcs). Since the authors do not give details about their implementation of LSET, we are unable to explain why our results differ.

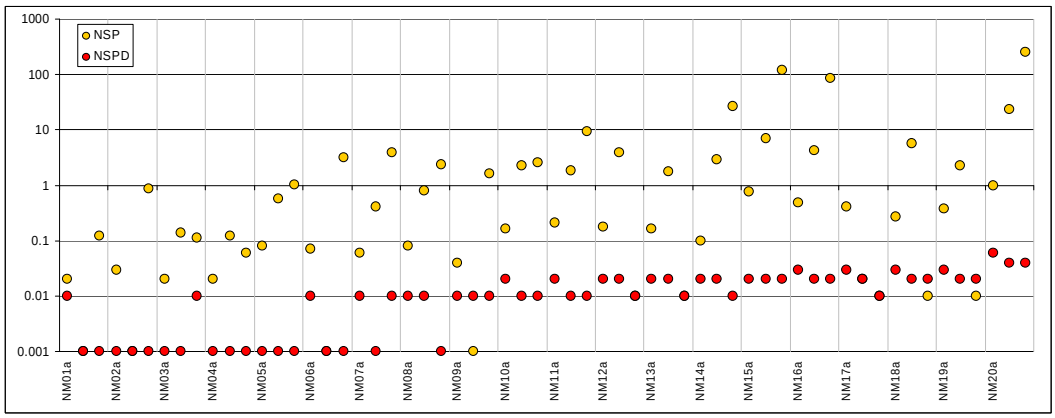


Figure 2.8. NSP vs NSPD - NetMaker networks; logarithmic scale; 0.001 represents run-time < 0.01.

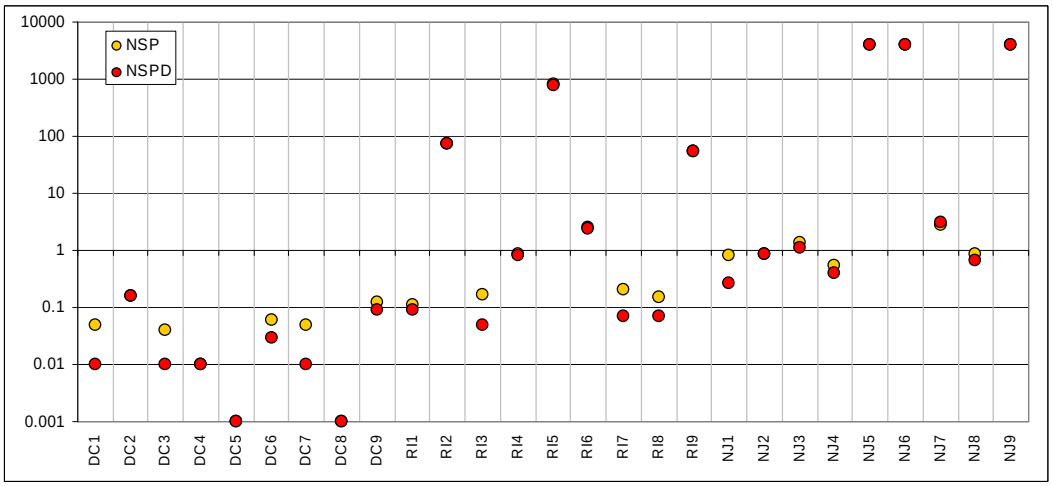


Figure 2.9. NSP vs NSPD - road networks; run-times of NSP and NSPD are similar or identical where only the red marker is visible; logarithmic scale; 0.001 represents run-time < 0.01.

Best Near-Shortest Path Approach

To run NSP/NSPD the $lex(1,2)$-best and the $lex(2,1)$-best solution are needed to set up initial bounds and find a weighting factor. We use the best initialisation approach as identified in the next section.

Results are presented in Figures 2.7 - 2.9 and Table 2.11 in Section 2.8. Clearly, NSPD performs better than NSP for NetMaker and road networks. The reason is that the near-shortest path algorithm computes single-objective shortest path distances from every node i to t before enumerating paths. As D is the best approach to solve the single-objective problem for both network types (refer to Section 2.4.2), we expect NSPD to perform best here.

The pure near-shortest path algorithm is badly suited to solve grid network problems which is due to the structure of the solution space. In objective space many feasible points are very close to the non-dominated points. Therefore, it can only be determined that they are dominated at a late stage of the algorithm. Their corresponding labels have to be considered throughout the algorithm before they are deleted at the very end of it, which leads to large computational effort and long run-times. The near-shortest path algorithm benefits from an upper bound on the weighted sum objective value to restrict enumeration. For grid networks this bound is quite far from the optimal weighted sum objective value, hence very many feasible paths are enumerated. Most grid network instances were stopped after running NSP and NSPD for an hour.

NSP was tested by Carlyle and Wood (2005) on grid networks of size 40×25 and 100×50 but with the deviation from the optimal path varying from $\varepsilon = 1$ to $\varepsilon = 6$. These values of ε are significantly smaller than the values that arise when solving our BSP problems instances. For grid networks, we get values of ε that mostly exceed 10000 initially (with an *average* initial value of 5351394). The deviation ε can never be improved to a value better than the weighted sum value of $x_{lex(1,2)}$ and $x_{lex(2,1)}$. In our grid instances the smallest possible deviation, the difference between the weighted sum value $c^{\lambda}(x_{lex(1,2)})$ and the length of the optimal weighted path ω, is at most 0.27 times the original value of ε. We observe that problem instances with an initial value of $\varepsilon \leq 10000$ can be solved within not even a second, whereas the algorithm does not terminate within 3600 seconds for instances with larger initial ε.

Paixão and Santos (2008) also apply a ranking algorithm to different networks and find it performs better than a labelling algorithm for two different network types. However, for their third network type, namely grid networks, they report that their ranking approach yields much longer run-times than labelling algorithms.

Best Approaches for Two Phase Method

We present several different approaches for each phase of the Two Phase Method. They are

- Initialisation with L/D,
- Phase 1 with LDIC/DDIC/SPAR, and
- Phase 2 with LCOR/LSET/NSP/NSPD.

Initialisation results are shown in Figures 2.10 - 2.12 and Table 2.12 in Section 2.8. Our experiments show that Dijkstra's algorithm is clearly the best approach for NetMaker and road networks. For grid networks both Dijkstra's algorithm (D) and the label correcting algorithm (L) show similar performance. There is no clear winner for grid networks, as the problems are fairly small single-objective shortest path problems so that the running times never exceed 0.01 seconds. Results by Cherkassy et al. (1996) indicate that performance of label setting and label correcting algorithms is fairly similar for simple grid networks like we use them in our tests. We choose to use Dijkstra's algorithm for initialisation for all three network types.

Whenever there is only one efficient solution to a problem, this is detected after initialisation for any dichotomic approach (LDIC or DDIC) as the $lex(1,2)$-best and the $lex(2,1)$-best solution are identical. In this case there is no need to run Phases 1 and 2 which is indicated by "NA" in Tables 2.13 and 2.14 in Section 2.8. In Figures 2.13 - 2.18 the corresponding results do not appear.

In Phase 1 we investigate the dichotomic approach where the arising single-objective problems are solved by the label correcting algorithm (LDIC) or Dijkstra's algorithm (DDIC). Another approach is the parametric simplex ap-

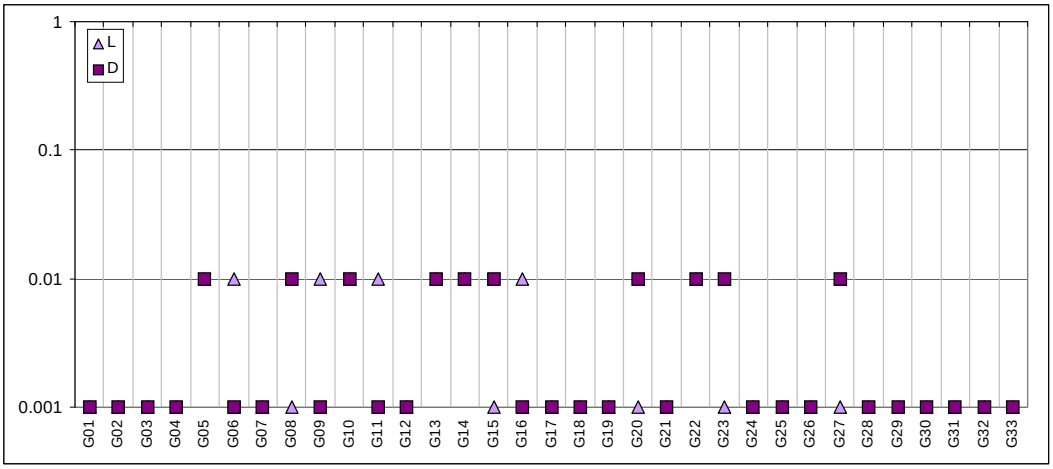


Figure 2.10. Initialisation - grid networks; run-times of L and D are identical where only the square marker is visible; logarithmic scale; 0.001 represents run-time < 0.01.

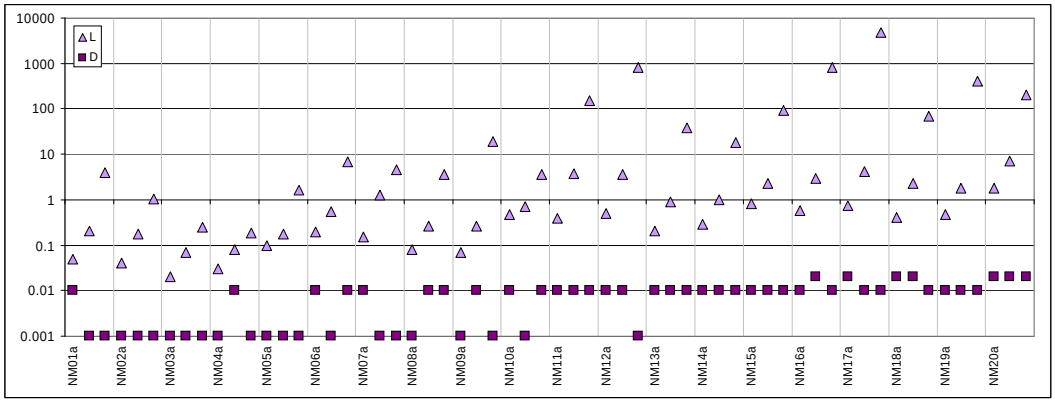


Figure 2.11. Initialisation - NetMaker networks; logarithmic scale; 0.001 represents run-time < 0.01.

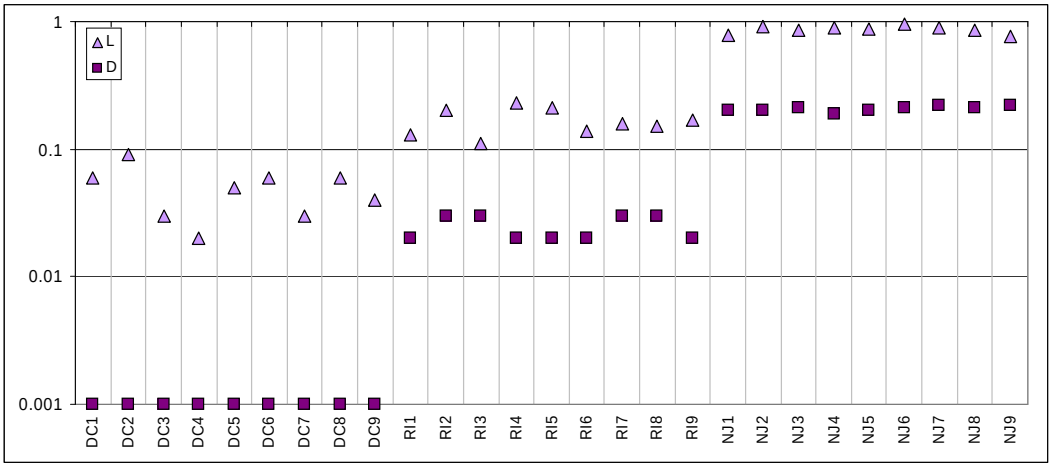


Figure 2.12. Initialisation - road networks; logarithmic scale; 0.001 represents run-time < 0.01.

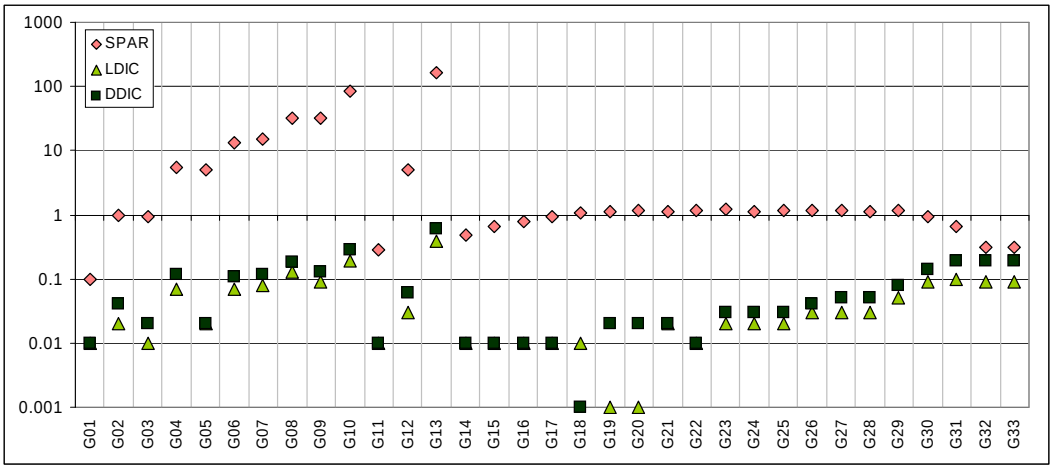


Figure 2.13. Phase 1 - grid networks; run-times of LDIC and DDIC are identical where only the square marker is visible; logarithmic scale.

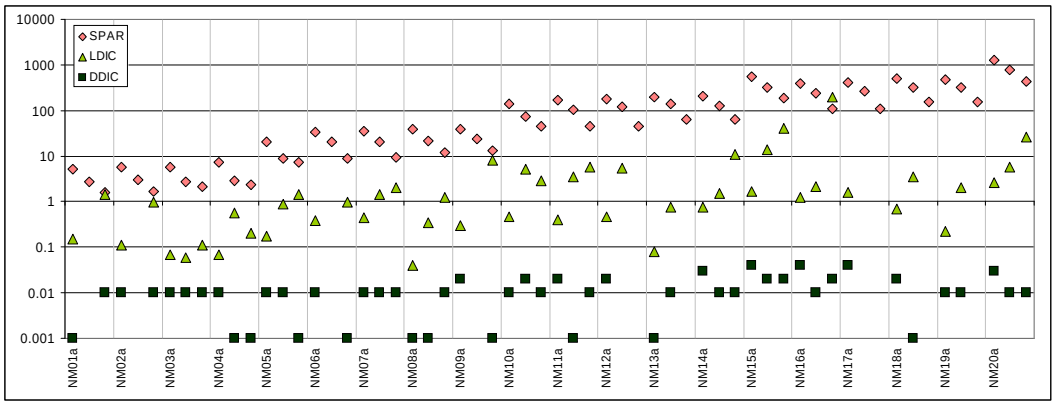


Figure 2.14. Phase 1 - NetMaker networks; logarithmic scale.

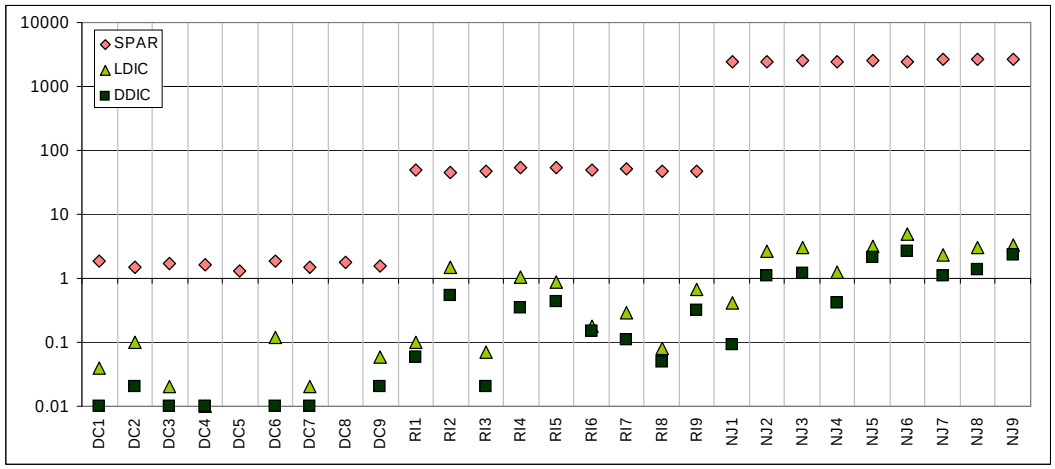


Figure 2.15. Phase 1 - road networks; logarithmic scale.

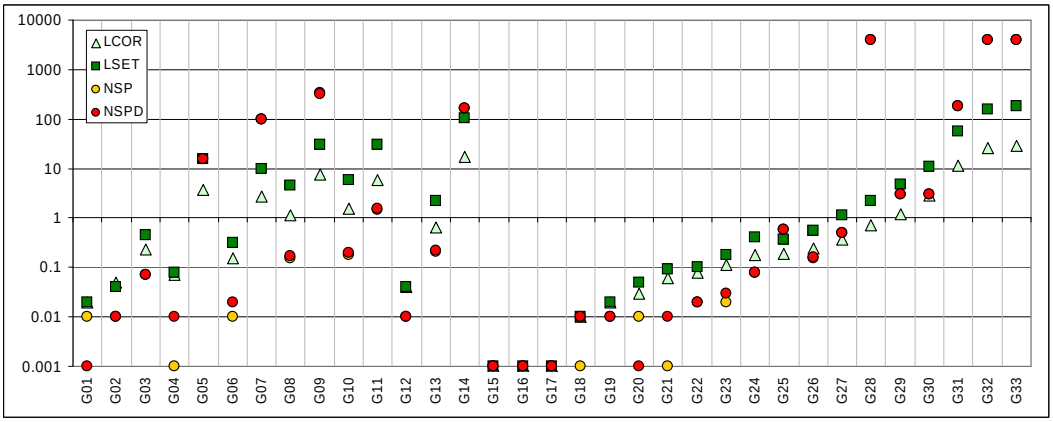


Figure 2.16. Phase 2 - grid networks; run-times of NSP and NSPD are similar or identical where only the red marker is visible; logarithmic scale; 0.001 represents run-time < 0.01.

proach (SPAR). For detailed results, refer to Figures 2.13 - 2.15 and Table 2.13 in Section 2.8.

Our experiments show that the parametric approach is not competitive with the others. DDIC is clearly the best approach for NetMaker and road networks. For grid networks LDIC performs slightly better than DDIC, the reason being that for grid networks the single-objective approach L is slightly better than D, and this slight advantage adds up when solving several single-objective problems in the dichotomic approach. For grid networks Phase 1 is solved in less than one second by LDIC and DDIC and their run-times are fairly similar. We choose DDIC as best Phase 1 approach for all network types.

For Phase 2 we investigate bi-objective label correcting (LCOR) and label setting (LSET) and also the usage of a near-shortest path algorithm, with one version using single-objective label correcting to initialise shortest paths (NSP) and the other using Dijkstra's algorithm (NSPD). For results, refer to Figures 2.16 - 2.18 and Table 2.14 in Section 2.8.

The run-times in Phase 2 for grid networks show that LCOR performs better than LSET. For road networks, however, LSET is better than LCOR. For NetMaker instances, all run-times of both LCOR and LSET are less than 0.01 seconds. The advantage of the Two Phase Method becomes apparent here, as LSET by itself performs very poorly for almost all NetMaker networks (see

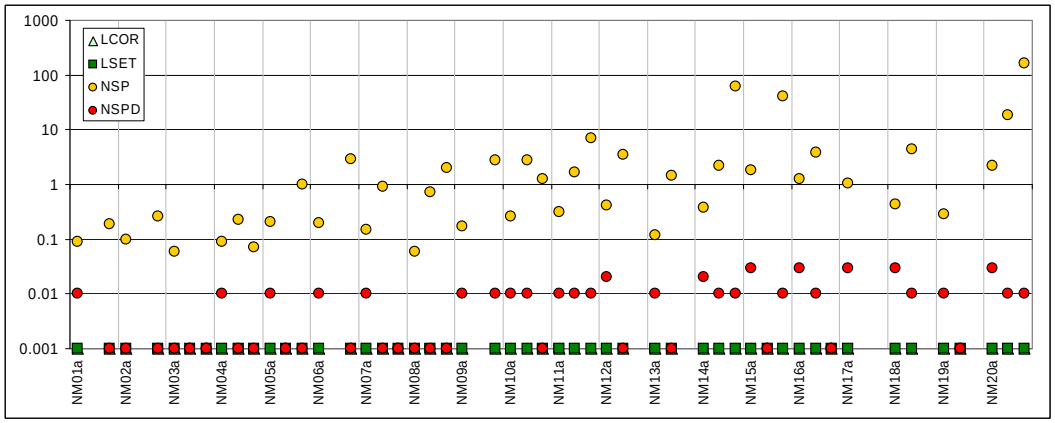


Figure 2.17. Phase 2 - NetMaker networks; run-times of LSET and LCOR are identical where the triangle marker is not visible; logarithmic scale; 0.001 represents run-time < 0.01.

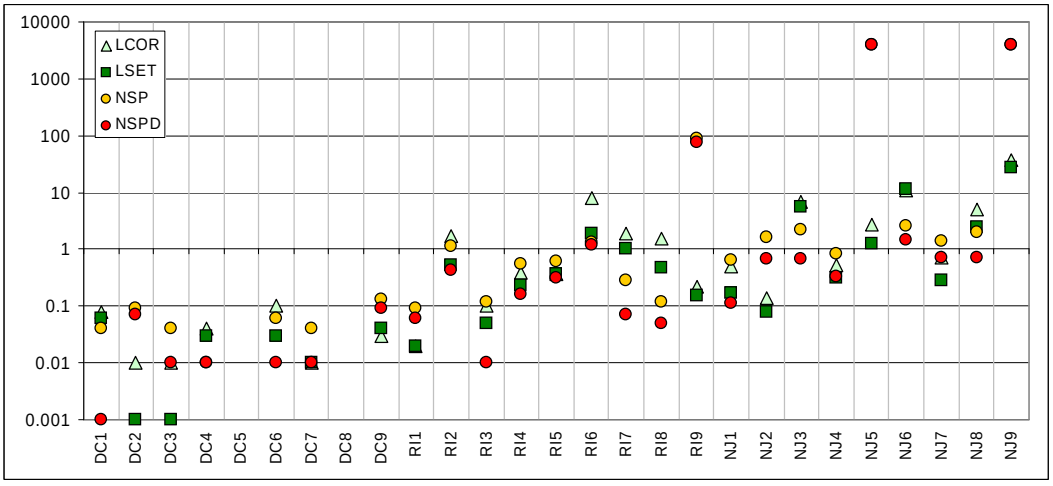


Figure 2.18. Phase 2 - road networks; run-times of NSP and NSPD are identical where only the red marker is visible; run-times of LCOR and LSET are similar or identical where the triangle marker is not visible; logarithmic scale; 0.001 represents run-time < 0.01.

Figure 2.5). Utilised in Phase 2, however, LSET is a very successful approach.

Similar to Section 2.4.2, NSPD outperforms NSP. We can also see the benefit of the Two Phase Method again – despite the bad performance of NSP and NSPD for grid networks in Section 2.4.2, the Two Phase Method with NSP and NSPD in Phase 2 finishes quickly for most grid instances, with only few timeouts. Here, NSP is slightly better than NSPD.

We choose LCOR and LSET as best Phase 2 approaches. For road and grid networks, LCOR and LSET are not always the best approaches but reasonably fast and more reliable as there are no extreme run-times as happens for NSP and NSPD for instances with a large number of efficient solutions such as G28, G32, G33, NJ5, and NJ9.

Numerical Results – Comparing Best Approaches

We compare the best approaches as discussed in the previous sections. More precisely, we compare the following:

LCOR: Bi-objective label correcting.

LSET: Bi-objective label setting.

NSPD: Near-shortest path.

2LCOR: The Two Phase Method with initialisation D, Phase 1 DDIC, and Phase 2 LCOR.

2LSET: The Two Phase Method with initialisation D, Phase 1 DDIC, and Phase 2 LSET.

We do not include NSPD results for grid networks as they performed badly as shown in the previous Section.

Grid networks (Figure 2.19 and Table 2.15 in Section 2.8):
Both LCOR and 2LCOR perform quite well, and LCOR is the best solution approach for grid networks. With increasing number of efficient solutions the run-time of the different approaches increases in a similar way, this can be observed for problems G15-G33. The network of instance G15 is very high

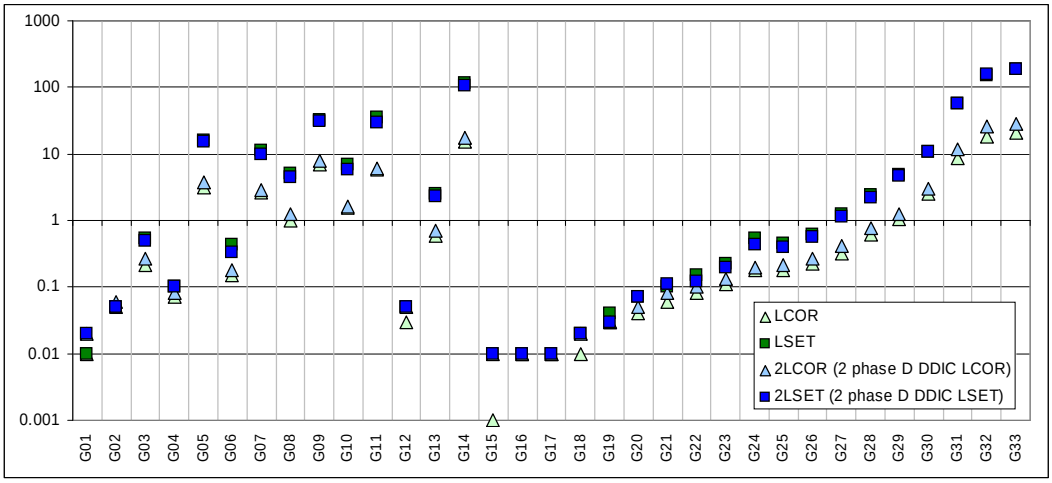


Figure 2.19. Best approach - grid networks; for run-times where markers are invisible, see Table 2.15; logarithmic scale; 0.001 represents run-time < 0.01.

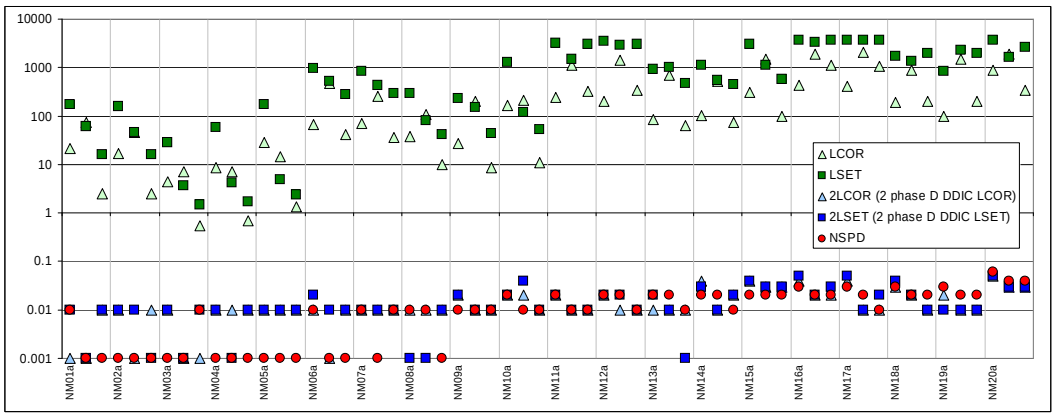


Figure 2.20. Best approach - NetMaker networks; for run-times where markers are invisible, see Table 2.16; logarithmic scale; 0.001 represents run-time $<$ 0.01.

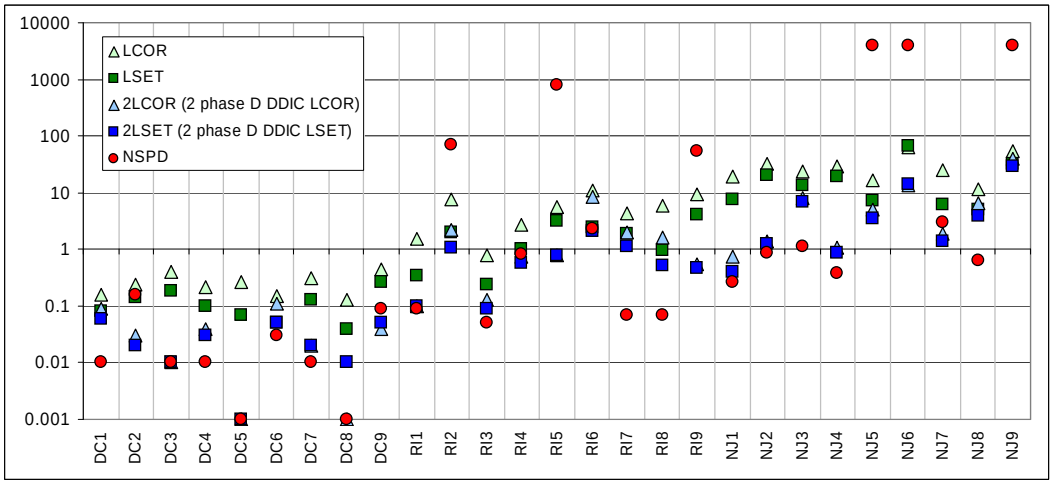


Figure 2.21. Best approach - road networks; for run-times where markers are invisible, see Table 2.17; logarithmic scale; 0.001 represents run-time < 0.01.

and thin, then the grid network instances decrease in height but increase in width. As the instances grow in width, the problems contain more and more efficient solutions. Other examples of instances with many efficient solutions are large instances such as G9, G11, and G14. As mentioned above, the run-time of NSPD for grid networks is not included in figures and tables as most instances were stopped after 3600 seconds.

NetMaker networks (Figure 2.20 and Table 2.16 in Section 2.8): NSPD and both Two Phase approaches (2LCOR and 2LSET) show similar very good performance with run-times well under 1 second. LCOR and LSET on the other hand are significantly worse with an average run-time of 368 and 1205 seconds and a minimum run-time of 0.55 and 1.44 seconds. It should be noted that this is the only network type where the NSPD approach does not have a high variance in run-time (on the instances we tested). We cannot relate the number of solutions to the run-time of an approach, compared to grid networks there are not many efficient solutions at all. It appears that increasing problem size does not significantly increase the run-time of 2LCOR, 2LSET, or NSPD, run-times of NM16-NM20 are not much longer than of NM11-NM15 for example.

Road networks (Figure 2.21 and Table 2.17 in Section 2.8): The Two Phase Method (2LCOR and 2LSET) is clearly the best approach, with run-

Table 2.5. Best approaches for different network types.

type	best	second best
grid	LCOR	2LCOR
NetMaker	2LCOR, 2LSET, and NSPD	-
road	2LSET	2LCOR

times less than LCOR and LSET. This shows again the strength of the Two Phase Method as Phase 1 restricts the area to be enumerated by LCOR and LSET in Phase 2. Among the Two Phase approaches, 2LSET performs best. NSPD performs very well for some instances (in fact better than 2LCOR and 2LSET), but its performance is very bad for others such as RI2, RI5, RI9, NJ5, NJ6, and NJ9. This illustrates that small variations in the problem such as the selection of source and target node may significantly complicate a problem.

Again, there are not many efficient solutions compared to grid networks. The number of efficient solutions and the problem size do not influence the run-time in an obvious way – other factors, however, such as distance between source and target node and how branched the roads between source and target are, play an important role. For example, run-times for 2LSET and the biggest road network, NJ, vary from 0.41 to 29.00 with an average of 6.83 for the nine instances that only differ in their respective location of source and target node.

In Table 2.5 the best approaches for the different network types are summarised. It appears that out of all approaches 2LCOR is the overall winner, as it performs consistently well for all network types.

2.5 Bounded Labelling: Improving Labelling Algorithms for BSP

The efficiency of bi-objective and multi-objective labelling algorithms can be improved by exploiting the fact that the cost vector of every enumerated path from source node s to target node t dominates other paths in the network. It may not always be necessary to extend a path to the target node to confirm that it is dominated. All paths that connect s to t, obtained at any time

while the algorithm runs, may dominate other paths at t but also at any other node of the network (Note that this works as long as arc costs are assumed to be positive). Therefore, it may be possible to delete "incomplete" s-i-paths, with $i \neq t$, at an early stage of the algorithm rather than extending them to the target node t and then deleting them. This is similar to the way bounds derived from supported solutions found in Phase 1 of the Two Phase Method can significantly speed up labelling algorithms in Phase 2 as demonstrated in Section 2.4.2. No such bounded labelling approach is described in the relevant literature. The A* algorithms in Stewart and White (1989) may describe a similar approach.

2.5.1 Bounded Labelling Algorithm

A bi-objective label correcting algorithm stops when there are no marked nodes any more, whereas a bi-objective label setting algorithm stops once there are no more tentative labels. In both cases, all efficient paths from s to all other nodes are obtained, including of course the ones to the target node t.

While a bi-objective labelling algorithm runs, every label at t corresponds to the cost vector of a path from s to t that is currently not dominated. Therefore, this label dominates parts of the objective space as no label that is dominated by it can represent an efficient path. Labels at *any* node of the network that are dominated by a label at t can be deleted as path costs are non-negative so that once a label anywhere in the network is dominated by a label at t it remains dominated. It is therefore not necessary to extend this label until its path reaches t, it can be deleted as soon as dominance is detected.

The proposed bounded labelling algorithm is obtained by modifying any of the labelling algorithms as follows: the algorithm runs as described in Sections 2.3.1 and 2.3.2 while there is no label at node t. Once there is at least one label at t, one can start checking bounds. For each newly generated label, one checks whether it is dominated by at least one label at target node t. The labels at the target node, $Labels(t) = \{l^1 = (z_1^1, z_2^1), \ldots, l^m = (z_1^m, z_2^m)\}$, are sorted by increasing first objective value, i.e. $z_1^1 < z_1^2 < \ldots < z_1^m$ and $z_2^1 > z_2^2 > \ldots > z_2^m$. Procedure 3 shows how to check whether the newly generated label $l = (z_1, z_2)$ is dominated.

Procedure 3 `bounded_labelling_dominance_check`

1: **input**: Newly generated label $l = (z_1, z_2)$ and sorted label set $Labels(t) = \{l^1 = (z_1^1, z_2^1), \ldots, l^m = (z_1^m, z_2^m)\}$.
2: Set *dominated* = FALSE and $i = 1$.
3: **while** (*dominated* == FALSE) and $(i \leqq m)$ and $(z_1^i \leqq z_1)$ **do**
4: **if** $z_2^i \leqq z_2$ **then**
5: Set *dominated* = TRUE.
6: **else**
7: $i = i + 1$
8: **end if**
9: **end while**
10: **output:** *dominated*

Only labels that are not dominated by any of the labels at t are retained, all others are deleted. It is easy to implement this check into any labelling algorithm. Step 8 in Algorithms 4 and 5 is modified by simply including the dominance check from Procedure 3 when "eliminating all dominated labels". The resulting algorithms are called *bounded labelling* algorithms and denoted by bLCOR and bLSET. Although the dominance check is formulated for the bi-objective problem here, our idea is immediately applicable to multi-objective problems.

Remark 2.5.1 It should be noted that bi-objective labelling algorithms actually yield the efficient paths from s to all other nodes, not just to the target node t. With our improvement the algorithm is restricted to finding the shortest paths from s to t. If the aim was obtaining shortest paths from s to a small number of target nodes $t_1, \ldots, t_l$, the bound check could be modified to only deleting a label if it is dominated by at least one label at *each* target node $t_1, \ldots, t_l$. Too many target nodes might diminish the effectiveness of the bounds.

In some cases it can be determined that a label is not dominated by any of the labels at t without actively comparing all of them. Let the labels at the target node, $l^1, \ldots, l^m$, be sorted by increasing first objective value as above. A label $l = (z_1, z_2)$ at any node will clearly not be dominated by any of the labels $l^1, \ldots, l^m$ if $z_1 < z_1^1$ or $z_2 < z_2^m$. Note that checking $z_1 < z_1^1$ is included as part of the while loop of the dominance check. If $z_2 < z_2^m$, it is not necessary to

check a label l against all labels $l^1, \ldots, l^m$ as it is clear a priori that l cannot be dominated. This additional condition was implemented, but achieved no further run-time improvements, which is why the corresponding results are not reproduced here.

2.5.2 Numerical Experiments

Table 2.6. Grid network test problems.

	run-time (sec)		ratio	run-time (sec)		ratio
name	LCOR	bLCOR	bLCOR over LCOR	LSET	bLSET	bLSET over LSET
G01	0.01	0.01	-	0.01	0.01	-
G02	0.05	0.05	-	0.05	0.05	-
G03	0.21	0.21	1.00	0.53	0.53	1.00
G04	0.07	0.06	-	0.10	0.09	-
G05	3.14	3.77	1.20	16.19	19.03	1.18
G06	0.15	0.14	0.93	0.44	0.34	0.77
G07	2.62	2.80	1.07	11.21	10.81	0.96
G08	1.00	1.06	1.06	5.04	4.74	0.94
G09	6.93	7.74	1.12	32.86	35.52	1.08
G10	1.54	1.51	0.98	6.99	6.20	0.89
G11	5.82	5.90	1.01	35.01	33.81	0.97
G12	0.03	0.04	-	0.05	0.05	-
G13	0.59	0.60	1.02	2.57	2.39	0.93
G14	15.14	17.39	1.15	114.56	121.10	1.06
G15	0.00	0.00	-	0.01	0.01	-
G16	0.01	0.01	-	0.01	0.01	-
G17	0.01	0.01	-	0.01	0.01	-
G18	0.01	0.01	-	0.02	0.02	-
G19	0.03	0.02	-	0.04	0.03	-
G20	0.04	0.04	-	0.07	0.06	-
G21	0.06	0.06	-	0.10	0.10	-
G22	0.08	0.08	-	0.15	0.13	0.87
G23	0.11	0.09	0.82	0.22	0.20	0.91
G24	0.18	0.18	1.00	0.53	0.50	0.94
G25	0.18	0.18	1.00	0.46	0.42	0.91
G26	0.22	0.23	1.05	0.62	0.62	1.00
G27	0.32	0.36	1.12	1.25	1.31	1.05
G28	0.62	0.70	1.13	2.41	2.59	1.07
G29	1.03	1.17	1.14	4.89	5.57	1.14
G30	2.54	2.80	1.10	10.98	13.11	1.19
G31	8.72	12.73	1.46	57.33	68.26	1.19
G32	18.10	32.27	1.78	153.97	179.14	1.16
G33	20.85	32.91	1.58	183.93	199.38	1.08
avg			1.13			1.01
min			0.82			0.77
max			1.78			1.19

The main aspect of the numerical experiments is comparing the run-time of the

proposed bi-objective bounded labelling algorithms to that of the standard bi-objective labelling algorithms. Furthermore, the bounded labelling algorithm is briefly compared to the most successful algorithms for the BSP problem as identified in Section 2.4.2. The test instances from Section 2.4.1 are used again to evaluate the performance of the bounded labelling algorithms. For the computational experiments, the setup is the same as for the BSP experiments in Section 2.4.2. We calculate the ratio $\frac{\text{bLCOR}}{\text{LCOR}}$ and $\frac{\text{bLSET}}{\text{LSET}}$, respectively. A ratio < 1 indicates that the bounded labelling algorithm performs better, whereas a ratio > 1 indicates the original labelling algorithm is preferable.

Table 2.7. NetMaker network test problems.

	run-time (sec)		ratio	run-time (sec)		ratio
name	LCOR	bLCOR	bLCOR over LCOR	LSET	bLSET	bLSET over LSET
NM01a	21.64	0.00	0.00	173.16	0.00	0.00
NM01b	72.80	0.00	0.00	62.22	0.00	0.00
NM01c	2.45	0.00	0.00	16.28	0.00	0.00
NM02a	16.76	0.00	0.00	160.79	0.00	0.00
NM02b	46.61	0.00	0.00	45.24	0.00	0.00
NM02c	2.55	0.01	0.00	15.71	0.00	0.00
NM03a	4.33	0.00	0.00	28.00	0.00	0.00
NM03b	7.10	0.00	0.00	3.65	0.00	0.00
NM03c	0.55	0.00	0.00	1.44	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
NM17a	413.13	0.00	0.00	NA	0.00	0.00
NM17b	2080.91	0.00	0.00	NA	0.00	0.00
NM17c	1085.21	0.00	0.00	NA	0.00	0.00
NM18a	193.88	0.00	0.00	1697.10	0.00	0.00
NM18b	892.60	0.00	0.00	1374.62	0.00	0.00
NM18c	204.35	0.00	0.00	2005.44	0.00	0.00
NM19a	99.37	0.00	0.00	854.51	0.00	0.00
NM19b	1478.49	0.00	0.00	2278.00	0.00	0.00
NM19c	204.50	0.00	0.00	1958.59	0.00	0.00
NM20a	881.79	0.00	0.00	NA	0.00	0.00
NM20b	1842.54	0.00	0.00	1647.66	0.00	0.00
NM20c	341.01	0.00	0.00	2614.00	0.00	0.00
avg			0.00			0.00
min			0.00			0.00
max			0.00			0.00

Grid networks have a network structure that does not permit (average) improvement of run-time through bounded label correcting, exhibited by the average ratio $1.13 > 1$, see Table 2.6. Here, the additional effort of checking for every newly created label whether it is dominated by any of the (often many!) labels at t seems larger than what is saved by discarding labels occasionally. For the label setting algorithm, we observe similar run-times of the

Table 2.8. Road network test problems.

	run-time (sec)		ratio	run-time (sec)		ratio
name	LCOR	bLCOR	bLCOR over LCOR	LSET	bLSET	bLSET over LSET
DC1	0.16	0.16	1.00	0.08	0.08	-
DC2	0.24	0.01	0.04	0.14	0.00	0.00
DC3	0.40	0.01	0.02	0.18	0.00	0.00
DC4	0.21	0.10	0.48	0.10	0.06	-
DC5	0.26	0.00	0.00	0.07	0.00	-
DC6	0.15	0.10	0.67	0.05	0.05	-
DC7	0.31	0.02	0.06	0.13	0.01	0.08
DC8	0.13	0.08	0.62	0.04	0.03	-
DC9	0.45	0.04	0.09	0.26	0.04	0.15
RI1	1.54	0.06	0.04	0.34	0.04	0.12
RI2	7.74	2.75	0.36	1.97	0.68	0.35
RI3	0.77	0.31	0.40	0.24	0.11	0.46
RI4	2.68	0.66	0.25	0.99	0.27	0.27
RI5	5.48	1.06	0.19	3.22	0.41	0.13
RI6	11.03	9.63	0.87	2.44	2.33	0.95
RI7	4.27	2.38	0.56	1.89	1.17	0.62
RI8	5.83	1.85	0.32	0.97	0.53	0.55
RI9	9.43	0.28	0.03	4.10	0.15	0.04
NJ1	19.77	1.29	0.07	7.53	0.30	0.04
NJ2	32.47	0.58	0.02	20.61	0.14	0.01
NJ3	23.70	16.51	0.70	13.13	7.65	0.58
NJ4	29.48	1.40	0.05	19.20	0.47	0.02
NJ5	16.42	5.37	0.33	7.18	1.73	0.24
NJ6	62.41	16.32	0.26	66.33	13.33	0.20
NJ7	24.66	1.45	0.06	6.30	0.33	0.05
NJ8	11.76	7.72	0.66	5.07	2.76	0.54
NJ9	53.44	41.44	0.78	32.35	26.87	0.83
avg			0.33			0.28
min			0.00			0.00
max			1.00			0.95

bounded algorithm compared to the original one with an average ratio of 1.01.

The results for the other two network types are more promising:

The most striking results appear for the NetMaker instances as displayed in Table 2.7. Here, run-time is always reduced to $\leqq 0.01$ although the run-time of LCOR and LSET is very high in most cases leading to an average ratio of 0.00 in all four cases. We reduce the table by only showing complete entries for NM1a-NM3c and NM17a-NM20c as all results look similar – instances grow bigger and with them the run-time of the original labelling approaches, but not that of the bounded approaches. This can be explained via the structure of the networks. As the instances were constructed to "wrap around", there are often arcs from nodes with low numbers that reach backwards and connect

to a node with very high number, i.e. close to the target node. In many instances, only few efficient paths exist which are also very short as they reach "backwards" from the nodes with low numbers to those with high numbers. For any labelling algorithm to finish, however, it is necessary to generate the paths from the source to all other nodes, whereby many long paths are enumerated that can never be efficient once they reach t. The bounds are very effective here, because efficient s-t paths are found quickly and have fairly low costs in both components, so that many labels can be discarded.

Finally, the results for road networks are listed in Table 2.8. For this problem type, the bounded algorithms bLCOR and bLSET both improve the run-time significantly when compared with the original algorithms LCOR and LSET. On average we observe a ratio of 0.33 for bounded label correcting and 0.28 for bounded label setting. These ratios indicate that the achieved run-time improvements of the bounded labelling algorithms are very significant.

This shows that bounded labelling does significantly improve the run-time of labelling algorithms. To complete the discussion of numerical results, the best approaches identified in Section 2.4.2 are compared with the bounded labelling algorithm. In Section 2.4.2 the labelling algorithms, LCOR and LSET, are compared with the two other algorithm types, Two Phase Method (2LCOR and 2LSET) and the near-shortest path algorithm (NSPD). Hence, the run-times reported for 2LCOR, 2LSET and NSPD in Tables 2.16 and 2.17 are compared to those of bLCOR and bLSET in Tables 2.7 and 2.8. We generate new figures by including bLCOR and bLSET in Figures 2.20 and 2.21 depicting final results. As bounded labelling does not have any positive effect for grid networks, a further comparison for grid networks is omitted here.

In case of the NetMaker networks, the bounded labelling algorithms clearly are competitive with the other solution algorithms (the Two Phase Method and the near-shortest path algorithm) or even better, see Figure 2.22 for a visual display of this result. The bounded labelling algorithms exceed the performance of the other algorithms with all run-times being less than 0.01 seconds. Comparisons between the approaches are omitted here as neither the Two Phase Method nor the near-shortest path algorithm take more than 0.06 seconds to solve any of the instances. The experiments show that the Two Phase Method, near-shortest path, and the bounded labelling algorithms are

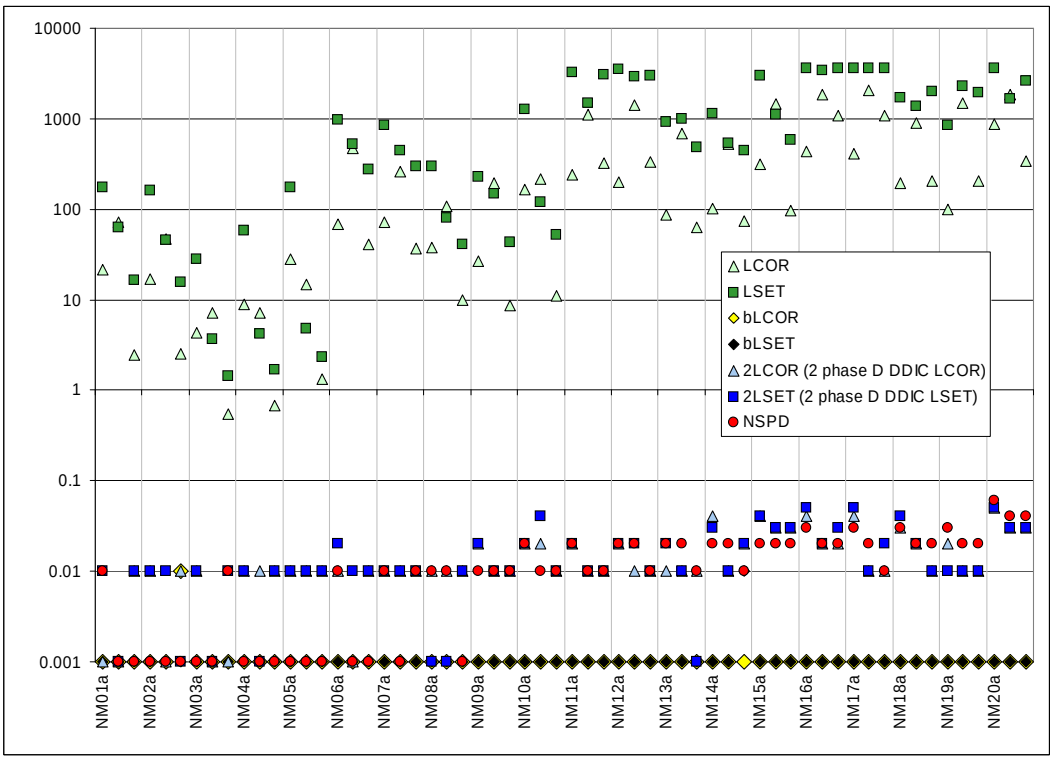


Figure 2.22. Comparing best BSP approaches and bounded labelling - NetMaker networks; for run-times where markers are invisible, see Tables 2.7 and 2.16; logarithmic scale; 0.001 represents run-time < 0.01.

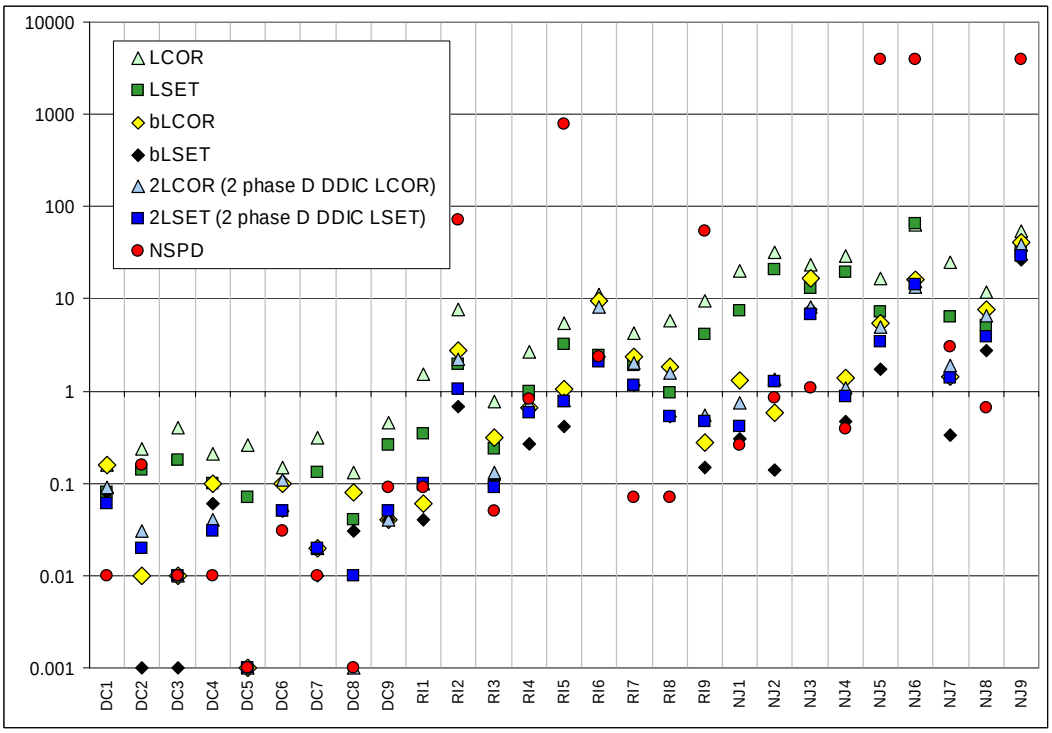


Figure 2.23. Comparing best BSP approaches and bounded labelling - road networks; for run-times where markers are invisible, see Tables 2.8 and 2.17; logarithmic scale; 0.001 represents run-time < 0.01.

equally well suited to solve NetMaker instances as all instances can be solved in less than 0.1 seconds by any of the algorithms.

When solving road networks the Two Phase Method with label setting in phase 2 (2LSET) was found to be the superior approach in Section 2.4.2. In Figure 2.23, we can see that bLSET has shorter running time than LSET, and bLCOR achieves a shorter running time than LCOR. Also, bLSET often performs better than the Two Phase Approach 2LSET.

Additionally to the figure, we compare 2LSET and the new bounded labelling algorithms by calculating the ratios of run-time bLCOR/2LSET and bLSET/2LSET. In case of label correcting, 2LSET remains the superior approach as the average ratio bLCOR/2LSET is 1.97 with a minimum value of 0.45 and a maximum of 4.61. The bounded label setting approach, bLSET, outperforms 2LSET for many instances with an average ratio bLSET/2LSET of 0.71, a minimum of 0.11 and a maximum of 1.22.

2.6 Modelling Cyclist Route Choice – an Application of BSP

Modelling traffic and route choice of motorised vehicles in particular is discussed in detail in Chapter 4. This modelling process, the so-called four-stage model, is briefly summarised here. In the first two steps of the four-stage model, it is determined how much travel demand originates from different zones in the network and where it is heading. Then the total travel demand is split between different possible transport modes, such as private car, tram, bus, cycling, walking, to name a few. In the final stage of the four-stage model the actual route choice is determined, often separately for each mode. It is usually assumed that each trip maker aims at minimising their individual travel time or some generalised cost function consisting of time and other route choice factors.

Modelling route choice of drivers of motorised vehicles and cyclists are two significantly different problems. Car travel is subject to network congestion which implies that the travel time is not proportional to road length. Traffic volumes influence travel time on a route for private motor vehicles such as cars. The equilibrium problem solved to model route choice of motor vehicles within a road network is discussed in Chapter 4.

While route choice of motorised vehicles is modelled, so-called *active modes* such as walking and cycling are often disregarded as they are not subject to congestion, and they also do not contribute to it. Active modes are acknowledged to be effective options for commuting trips, as they do not create emissions, they do not contribute to traffic congestion, and they hardly endanger other traffic participants. Furthermore, active modes entail an additional health benefit due to exercising. Unfortunately cycling can also be dangerous. In Auckland, New Zealand, this is due to high traffic volumes, narrow lanes on roads, a lack of cycle paths, but also careless car drivers.

In order to encourage more commuters to choose the bicycle for their trip to work or other destinations, it is necessary to improve dedicated cycle path infrastructure and thus safety. It is important to correctly assess the impact of this new piece of infrastructure at a planning stage, to answer questions such as: will cyclists use the cycle path? When a piece of road is added to a

road network, the traffic assignment process for motorised vehicles described above is used to determine the expected traffic volume on the road and also the impact it has on the surrounding network, such as effects on congestion in the area.

At present, actual cycle route choice is not modelled at all and it is therefore impossible to evaluate the impact of an improvement of the cycle network. No model exists to predict how many cyclists benefit from an improvement. Especially when different alternatives for improvement of a cycle network are proposed, knowing how each one affects cyclist route choice could help select the best alternative.

The problem of modelling cyclist route choice poses a different challenge compared to modelling route choice of motorised vehicles. The problem is a simpler one in terms of travel time estimation as cyclists are generally not subject to traffic congestion – due to their small size, they are able to pass queues. Therefore, it can be assumed that their travel time is proportional to road length. What complicates the cyclist route choice problem is that cyclists have more than one objective influencing their route choice, a very important one being safety. This appears to be particularly true in Auckland, where only little infrastructure is dedicated solely to cyclists, instead cyclists share roads with cars or they share bus lanes.

We identify the cyclist route choice problem to be at the core of every transport planning tool that deals with cycling facilities. As a first step, cyclist route choice is modelled here, which could later be incorporated into the four-stage model of transport planning. We study cyclist route choice on the basis of the Auckland road network, where trips between two particular zones of Auckland are considered during the morning peak period. In particular trips from the green zone to the red zone in Figure 2.24 are studied. The green zone is Point Chevalier, a residential area in Auckland, whereas the red zone lies within Auckland's Central Business District. The selected origin and destination represent a typical commute to work in Auckland.

We discuss the two objectives identified as main influencing factors of cyclist route choice in the subsequent section. Then, a solution algorithm for the cyclist route choice problem is proposed followed by a discussion of possible areas of application. The results presented in this section are also contained

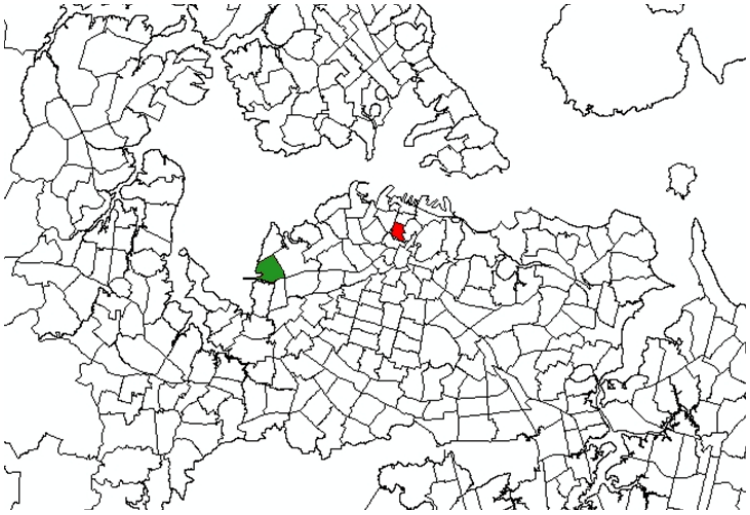


Figure 2.24. Zones of the Auckland road network; the green zone represents Point Chevalier and the red zone lies in the Central Business District; source: Auckland Regional Council.

in Raith et al. (2009).

2.6.1 Objectives in Cyclist Route Choice

The identification of main factors that influence cyclist route choice is the subject of numerous studies. Travel time is identified as the most important objective according to which cyclists choose their route (e.g. Aultman-Hall et al. 1997; Stinson and Bhat 2003). Aultman-Hall et al. (1997), for example, compare the actual routes taken by commuter cyclists to their shortest-distance paths. They find that 58% of all commuters actually follow their shortest-distance path. This clearly indicates that travel distance (or travel time, which should be highly correlated with distance) is one of the most important factors of route choice. However, there are other influencing factors that lead cyclists to divert from their shortest-distance route.

Other factors affecting cyclist route choice are studied in Dill and Carr (2003); Stinson and Bhat (2003); Aultman-Hall et al. (1997). Dill and Carr (2003) are concerned with the general willingness to cycle. Here the correlation of percentage of cyclists in a city and factors such as rainy days per year, cycling facilities, and household income is investigated. Aultman-Hall et al. (1997);

Stinson and Bhat (2003) are more concerned with the actual route choice. Factors that influence route choice are road traffic, road gradient, presence of dedicated cycling facilities, to name just a few.

Therefore, it is reasonable to formulate cyclist route choice as a bi-objective problem with travel time as one objective, whereas all other route choice factors are combined into a second objective that we call *attractiveness*.

Travel Time Objective

We denote by r a path that connects a cyclist's origin and destination. The travel time along an arc a is t_a, which we assume independent of any factors other than length of the arc – in particular it does not depend on the amount of traffic on the arc. For simplicity we assume that distance equals time, a correct factor for the proportionality of the two can easily be introduced later.

The travel time t_r on a path r is then obtained as the sum of travel times on each of its arcs:

$$t_r = \sum_{a \in r} t_a. \tag{2.2}$$

A significant portion of travel time along an inner-city route is spent waiting at intersections with traffic lights (signalised intersections). The original road network of Auckland, the Auckland Regional Transport Planning Model (known as ART model) provided by the Auckland Regional Council, needs to be adapted to explicitly account for delay at signalised intersections. The original model consists of nodes that represent large intersections and arcs that represent the roads connecting large intersections, see Figure 2.25 for an intersection in the original network.

The network is modified by replacing every node in the study area that represents a major signalised intersection by dummy nodes and corresponding dummy arcs as shown in Figure 2.26. The figure shows an intersection where two roads intersect. This allows to add the average time spent waiting at an intersection to the path travel time. We assume that cyclists do not have to queue at the signal, as they are able to pass the queue and get to its front relatively easily. Hence their average waiting time is influenced only by the total time of one signal cycle T_S and the time the signal is red T_{red}. Whether a

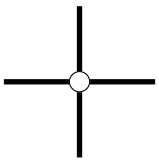


Figure 2.25. Original representation of a signalised intersection.

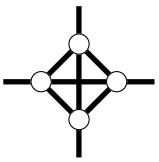


Figure 2.26. New representation of a signalised intersection.

cyclist needs to stop at a red light is measured by the ratio $\frac{T_{red}}{T_S}$. If the cyclists needs to wait, the average delay at the light is $\frac{T_{red}}{2}$. Multiplying those two ratios yields the average delay $\frac{T_{red}^2}{2T_S}$. The corresponding average delay values represent the travel time associated with the dummy intersection arcs.

Apart from modifying the network at intersections, additional arcs are introduced into the model corresponding to infrastructure that can be used only by cyclists, but not four-wheeled motor vehicles. In particular, an off-road bicycle way in the considered area (between the two highlighted zones in Figure 2.24) was introduced into the model.

Attractiveness Objective

To model cyclist route choice, one needs to evaluate the attractiveness of a route and each road. Several articles deal with the identification of factors that influence the perception of road attractiveness and also propose how to measure it. A report by the Land Transport Safety Authority (2004) proposes to assess the so-called *level of service (LOS)* which, in the context of cycling, can be interpreted as bicycle compatibility or cyclability.

Procedures have been proposed to allow the conversion of the qualitative measure attractiveness into a quantitative one. There are different studies that link road characteristics to the attractiveness of the road for cyclists. Harkey et al. (1998); Florida Department of Transportation (2002) present formulae to obtain a quantitative measure of LOS based on factors such as lane width, traffic volume, amount of heavy vehicles, pavement condition, and parking (both suddenly opening car doors and vehicles reversing into parking spaces pose great danger to cyclists). Both measures do not include factors such as road gradient that seem important for Auckland. In Palmer et al. (1998)

Table 2.9. Scoring system for attractiveness objective.

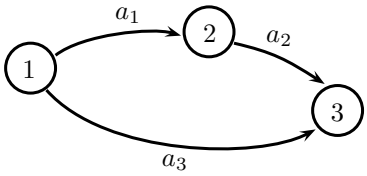
Attractiveness rating	Score	Integer value
A	81-100	6
B	61-80	5
C	41-60	4
D	21-40	3
E	1-20	2
F	$\leqq 0$	1

many more factors are included in the attractiveness rating such as motor traffic speed, volume, lane width, presence of on-street parking, road gradient, provided cycle facilities, and pavement condition. A scoring system is used to convert these factors into a simple A-F score.

We collected road characteristics data for the study area, which is converted into a score following the Palmer et al. (1998) model. The obtained score between 0 and 100 corresponds to an A-F grade as shown in Table 2.9, with A being the best possible rating and F the worst. We represent this grade by an integer value with the highest grade A corresponding to a value of 6. We denote by α_a the integer value corresponding to the attractiveness rating of arc a.

We also need an attractiveness rating for every dummy intersection arc. A measure of intersection safety for through movements is presented in Landis et al. (2003). Attractiveness is rated according to lane width, crossing distance, traffic volume, and number of through lanes. Every intersection in the study area of the Auckland network was analysed and an A-F rating was derived. As the literature does not describe how to measure attractiveness in turning movements, they were rated similar to through movements and according to personal judgement.

Attractiveness of a path cannot be obtained as the sum of attractiveness values on each arc of the path, unlike travel time which is obtained by summing individual arc travel times (2.2). This can be seen based on a small example:



Assume that the travel time and safety associated with arcs a_1, a_2, a_3 are $t_{a_1} = t_{a_2} = 1, t_{a_3} = 2$ and $\alpha_{a_1} = \alpha_{a_2} = 4, \alpha_{a_3} = 6$. If both costs and times are assumed additive, the path consisting of arcs a_1 and a_2 has objective vector $(2, 8)$, whereas the path consisting of arc a_3 has objective vector $(2, 6)$. Both paths connect node 1 to node 3. Path a_1, a_2 dominates path a_3 as the corresponding attractiveness value of the first path is higher. Clearly, path a_3 should be the better path as both path travel times are identical, but path a_3 is much safer than the other path. A similar example can be constructed when small attractiveness values represent the most attractive path and we aim at minimising total attractiveness.

It appears that the attractiveness measure for a path should be calculated by taking path safety into account, but also the time a cyclist travels on roads with a certain safety factor. Thus, we measure the attractiveness along a path r as a time-weighted average value of attractiveness along a route:

$$\alpha_r = \frac{\sum_{a \in r} t_a \alpha_a}{\sum_{a \in r} t_a} = \frac{\sum_{a \in r} t_a \alpha_a}{t_r}.$$

2.6.2 Solving the Cyclist Route Choice Problem

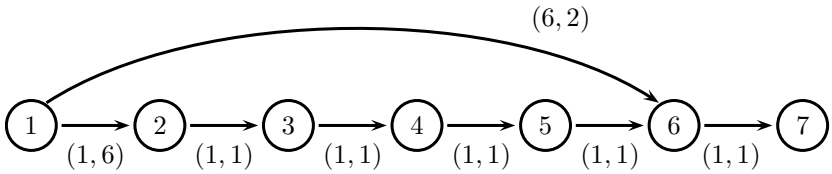
Cyclists choose their route according to the two criteria travel time t and path attractiveness α, where the aim is to minimise time while maximising path attractiveness. This is a bi-objective shortest path problem with a minimisation and a maximisation objective:

$$\begin{array}{ll} \min & \sum_{a \in r} t_a \\ \max & \frac{\sum_{a \in r} t_a \alpha_a}{\sum_{a \in r} t_a} \\ \text{s.t.} & r \in \mathcal{R}, \end{array} \tag{2.3}$$

where $\mathcal{R}$ denotes the set of all paths cyclists can choose to follow from their origin to their destination.

Unfortunately, the problem with attractiveness objective α_r cannot be solved with a BSP algorithm as adding new arcs to a path may decrease the value of the attractiveness objective. On the basis of the following example we show that any algorithm fails that is based on the assumption that only efficient paths from origin to an intermediate node can lead to efficient paths to the destination node.

Example 2.6.1 *We consider the following network with travel time and attractiveness* (t_a, α_a) *right next to each arc.*



We consider paths from node 1 *to node* 7*. Both possible paths through nodes* $1, 2, 3, 4, 5, 6, 7$ *and through nodes* $1, 6, 7$ *are efficient. Their respective path cost vectors are* $\left(7, \frac{13}{7}\right)$ *and* $\left(6, \frac{11}{6}\right)$*. However, out of the two possible paths from node* 1 *to node* 6*, the path following the arc from* 1 *to* 6 *is dominated by the path through nodes* $1, 2, 3, 4, 5, 6$ *as their respective path cost vectors are*

$$\begin{pmatrix} 6 \\ 2 \end{pmatrix} \text{ and } \begin{pmatrix} 5 \\ 2 \end{pmatrix}.$$

Therefore, a shortest path labelling algorithm cannot find the path $1, 6, 7$*.*

Remark 2.6.1 Apart from bi-objective labelling approaches, NSP and SPAR are discussed as solution algorithms for BSP problems earlier in this chapter. NSP is an enumerative algorithm that we apply to a weighted sum version of the BSP problem. A single-objective weighted sum problem derived from (2.3) would also not be guaranteed to converge and present a correct answer. The simplex based approach is not applicable as arc costs cannot be considered fixed as the value of the second objective component may change when the same arc is added to two different paths.

We can show that efficient solutions of problem (2.3) are always efficient so-

lutions of an auxiliary problem obtained by dropping the denominator of the attractiveness objective

$$\begin{array}{ll} \min & \sum_{a\in r} t_a \\ \max & \sum_{a\in r} t_a\alpha_a \\ \text{s.t.} & r \in \mathcal{R}. \end{array} \tag{2.4}$$

Problem (2.4) satisfies the assumption that the path cost vector increases as an arc is added to the path. We modify a bi-objective label correcting algorithm to maximise the second objective component, and also to ensure that no nodes repeat within one path to avoid the generation of paths that contain cycles. This algorithm can be used to find all efficient solutions of (2.4). Among those solutions, all efficient paths for (2.3) are selected, which solves the cyclist route choice problem (2.3). We must therefore verify the following Proposition.

Proposition 2.6.1 *An efficient path of* (2.3) *is always an efficient path of* (2.4).

Proof Assume the contrary, i.e. that there exists an efficient path r of (2.3) that is not efficient for (2.4). Then there exists a path r' that dominates r for (2.4), which means

$$\sum_{a\in r'} t_a \leqq \sum_{a\in r} t_a \text{ and } \sum_{a\in r'} t_a\alpha_a \geqq \sum_{a\in r} t_a\alpha_a,$$

with at least one strict inequality. This implies that

$$\frac{\sum_{a\in r'} t_a\alpha_a}{\sum_{a\in r'} t_a} > \frac{\sum_{a\in r} t_a\alpha_a}{\sum_{a\in r} t_a}.$$

Hence, r' dominates r for (2.3), a contradiction. □

The problem is solved using a bi-objective label correcting algorithm modified as discussed above. Figure 2.27 shows the obtained paths and Figure 2.28 the corresponding path cost vectors (note that the second objective is maximised). Four out of the five obtained paths use an off-road cycle corridor called the "Northwestern Cycleway". However, this is not the most direct route, the purple route is shorter.

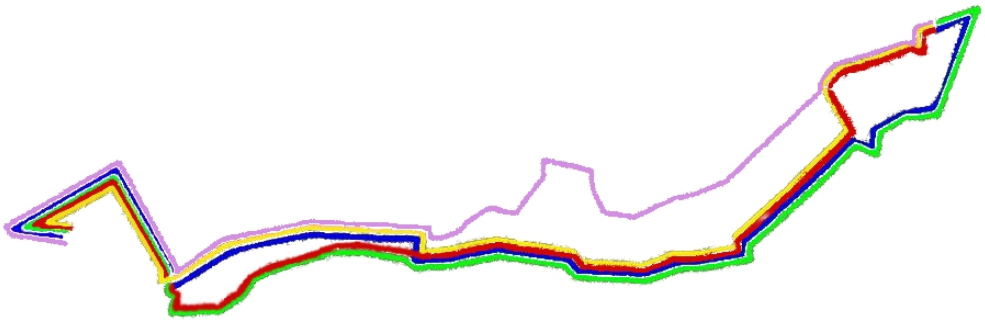


Figure 2.27. Efficient paths for cyclist trips from Point Chevalier to Auckland CBD.

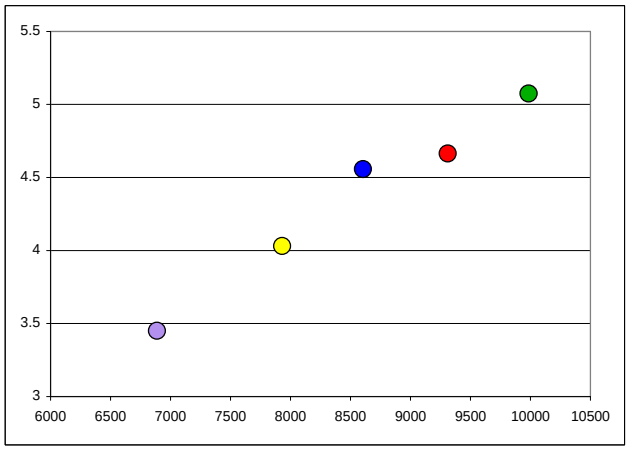


Figure 2.28. Costs associated with efficient paths for cyclist trips from Point Chevalier to Auckland CBD. Path length is displayed on the horizontal axis, attractiveness on the vertical axis.

2.6.3 Discussion

The method above allows to identify possible routes for cyclists without making the behavioural assumption that all cyclists choose their route according to a single objective, such as the shortest distance. The range of efficient paths obtained in the above example shows how valuable the Northwestern Cycleway may be on a commuting trip to the city, as four out of five paths use this cycleway. Also, the resulting paths indicate where the shortest path and the more attractive paths deviate, which may give insight into where cyclist infrastructure needs improvement.

A possible application of the cyclist route choice model (2.3) is the generation of a cycle map. This map could promote cycling as a viable alternative to commuting by car. The map should highlight the best compromise of travel time and attractiveness for an average commuter cyclist. In order to do this, the fastest path with a minimum level of attractiveness could be chosen. If, for example, a minimal attractiveness of 4.5 is required, the blue path in Figure (2.27) is selected. Of course the best routes for all major origin-destination pairs within the Auckland region would have to be obtained for a complete map.

Transport planning decisions require that different options for infrastructure investment are evaluated and compared. With respect to infrastructure for motor vehicles, this can be achieved by a comprehensive traffic modelling process whereby a "do-nothing" and a "do-something" approach are modelled. Comparing the two highlights the benefits for network users and possible shifts in traffic flows, and also allows to compare the impact of different investment strategies. With cycling, there exist no computerised models for route choice and also no cyclist traffic assignment process to determine which of all possible routes cyclists choose. The lack of such a cyclist assignment makes it difficult to evaluate the impact of changes made to the cycle network.

Assuming a certain number of cyclist trips between an origin and destination (the demand), such an assignment model needs to assign portions of trips to different paths connecting origin and destination. It is widely accepted that cyclists do not base this choice purely on travel time as discussed above. The bi-objective problem with objectives travel time and attractiveness returns efficient paths that are all reasonable route choices for a cyclist. It can be

assumed that most cyclists choose one of those routes. Now, portions of the total demand need to be assigned to each efficient path. Experienced commuter cyclists might not worry about safety and gradient of a route and therefore choose the shortest one. As mentioned earlier, Aultman-Hall et al. (1997) find that 58% of the cyclists in their study of cyclist commuting choose this shortest route. The portions of demand on the other paths could be determined through cyclist surveys, depending on how cyclists value their safety compared to travel time. More details on this follow in Chapter 4 on bi-objective traffic assignment for motor vehicles. In particular, different methods to split travel demand between several efficient paths are discussed in Section 4.5, which can be adapted to cyclist route choice.

2.7 Concluding Remarks on Bi- and Multi-objective Shortest Path Algorithms

We are able to show that the Two Phase Method is competitive with other commonly applied approaches to solve the BSP problem. The Two Phase Method works well with both a ranking, a label correcting, and a label setting approach in Phase 2, but the label correcting and setting approaches appear to be preferable as their run-times are more reliable, they do not differ between different problem instances, as sometimes occurs for the near-shortest path approach: the purely enumerative near-shortest path approach is a very successful approach to solve some problem instances, but the run-time on others is very long.

We illustrate all this on various test instances. It also becomes clear that the best performing approach depends on the network type, and even small variations on the network may have a high impact on performance.

An area of future research is the generation of test instances. It is difficult to randomly generate networks with a high number of efficient solutions; we are unable to obtain high numbers of efficient solutions in networks without a grid structure.

Furthermore, a speed-up technique, called bounded labelling, is presented here. For two of our three network types, namely NetMaker and road networks,

bounded labelling performs better than its counterpart standard bi-objective labelling. In comparison with the Two Phase Method, the proposed technique yields an algorithm that is better than the Two Phase Method for many problem instances of the NetMaker and road network types.

The bounded labelling technique can easily be incorporated into multi-objective shortest path labelling algorithms. It is a subject of future research to assess whether bounded labelling also improves on run-times of multi-objective labelling algorithms.

Other speedups of bi- and multi-objective label correcting and label setting algorithms should be investigated in the future. For single-objective shortest path problems, the efficient utilisation of data structures does provide significant speedups and similar improvements may be obtained for BSP and MSP problems.

We also present a novel area of application for BSP problems, namely the modelling of cyclist route choice. We were able to show how the cyclist problem can be modelled by considering a time and attractiveness objective, and also how it can be solved. We point out different areas where the ability to model cyclist route choice is valuable. The integration of the cyclist route choice model into a traffic assignment procedure for cyclists is an important area for future research.

2.8 Tables from Section 2.4

For completeness, all tables to support statements on the outcome of computational experiments from Section 2.4 are given on the following pages.

Table 2.10: Bi-objective labelling with LCOR and LSET; run-time in seconds.

	LCOR	LSET		LCOR	LSET		LCOR	LSET
G01	0.01	0.01	DC8	0.13	0.04	NM07c	36.79	298.43
G02	0.05	0.05	DC9	0.45	0.26	NM08a	37.39	294.95

Continued on Next Page...

Table 2.10 – Continued

	LCOR	LSET		LCOR	LSET		LCOR	LSET
G03	0.21	0.53	RI1	1.54	0.34	NM08b	107.48	80.70
G04	0.07	0.10	RI2	7.74	1.97	NM08c	9.85	41.12
G05	3.14	16.19	RI3	0.77	0.24	NM09a	26.91	230.66
G06	0.15	0.44	RI4	2.68	0.99	NM09b	196.27	148.19
G07	2.62	11.21	RI5	5.48	3.22	NM09c	8.55	43.45
G08	1.00	5.04	RI6	11.03	2.44	NM10a	165.92	1262.47
G09	6.93	32.86	RI7	4.27	1.89	NM10b	213.73	120.94
G10	1.54	6.99	RI8	5.83	0.97	NM10c	11.01	51.73
G11	5.82	35.01	RI9	9.43	4.10	NM11a	240.73	3260.00
G12	0.03	0.05	NJ1	19.77	7.53	NM11b	1103.90	1486.70
G13	0.59	2.57	NJ2	32.47	20.61	NM11c	325.31	3090.00
G14	15.14	114.56	NJ3	23.70	13.13	NM12a	199.80	3519.00
G15	0.00	0.01	NJ4	29.48	19.20	NM12b	1424.13	2935.00
G16	0.01	0.01	NJ5	16.42	7.18	NM12c	330.86	3019.00
G17	0.01	0.01	NJ6	62.41	66.33	NM13a	86.36	907.54
G18	0.01	0.02	NJ7	24.66	6.30	NM13b	684.42	995.65
G19	0.03	0.04	NJ8	11.76	5.07	NM13c	63.74	477.96
G20	0.04	0.07	NJ9	53.44	32.35	NM14a	102.97	1137.95
G21	0.06	0.10	NM01a	21.64	173.16	NM14b	517.02	541.96
G22	0.08	0.15	NM01b	72.80	62.22	NM14c	74.44	444.82
G23	0.11	0.22	NM01c	2.45	16.28	NM15a	312.65	2985.00
G24	0.18	0.53	NM02a	16.76	160.79	NM15b	1467.20	1098.58
G25	0.18	0.46	NM02b	46.61	45.24	NM15c	96.33	580.16
G26	0.22	0.62	NM02c	2.55	15.71	NM16a	432.66	-
G27	0.32	1.25	NM03a	4.33	28.00	NM16b	1857.96	3407.00
G28	0.62	2.41	NM03b	7.10	3.65	NM16c	1091.88	-
G29	1.03	4.89	NM03c	0.55	1.44	NM17a	413.13	-
G30	2.54	10.98	NM04a	8.81	58.77	NM17b	2080.91	-
G31	8.72	57.33	NM04b	7.18	4.19	NM17c	1085.21	-
G32	18.10	153.97	NM04c	0.68	1.70	NM18a	193.88	1697.10
G33	20.85	183.93	NM05a	27.93	174.34	NM18b	892.60	1374.62
DC1	0.16	0.08	NM05b	14.68	4.80	NM18c	204.35	2005.44
DC2	0.24	0.14	NM05c	1.34	2.34	NM19a	99.37	854.51
DC3	0.40	0.18	NM06a	67.91	972.42	NM19b	1478.49	2278.00
DC4	0.21	0.10	NM06b	475.19	519.49	NM19c	204.50	1958.59
DC5	0.26	0.07	NM06c	40.83	278.42	NM20a	881.79	-
DC6	0.15	0.05	NM07a	71.74	857.47	NM20b	1842.54	1647.66
DC7	0.31	0.13	NM07b	257.42	439.81	NM20c	341.01	2614.00

dash (-): run-time exceeds 3600 seconds

Table 2.11: Enumeration with NSP and NSPD; run-time in seconds.

	NSP	NSPD		NSP	NSPD		NSP	NSPD
G01	-	-	DC8	0.00	0.00	NM07c	0.01	3.82
G02	-	-	DC9	0.09	0.12	NM08a	0.01	0.08
G03	-	-	RI1	0.09	0.11	NM08b	0.01	0.80
G04	-	-	RI2	71.80	72.25	NM08c	0.00	2.34
G05	-	-	RI3	0.05	0.17	NM09a	0.01	0.04
G06	-	-	RI4	0.82	0.86	NM09b	0.01	0.00
G07	-	-	RI5	794.01	806.67	NM09c	0.01	1.59
G08	-	-	RI6	2.36	2.45	NM10a	0.02	0.16
G09	-	-	RI7	0.07	0.20	NM10b	0.01	2.24
G10	-	-	RI8	0.07	0.15	NM10c	0.01	2.54
G11	-	-	RI9	53.89	55.20	NM11a	0.02	0.21
G12	-	-	NJ1	0.26	0.82	NM11b	0.01	1.84
G13	-	-	NJ2	0.85	0.87	NM11c	0.01	9.38
G14	-	-	NJ3	1.10	1.39	NM12a	0.02	0.18
G15	0.01	0.01	NJ4	0.39	0.55	NM12b	0.02	3.89
G16	0.00	0.01	NJ5	-	-	NM12c	0.01	0.01
G17	0.00	0.01	NJ6	-	-	NM13a	0.02	0.16
G18	0.03	0.03	NJ7	3.02	2.84	NM13b	0.02	1.74
G19	0.79	0.79	NJ8	0.65	0.84	NM13c	0.01	0.01
G20	-	-	NJ9	-	-	NM14a	0.02	0.10
G21	-	-	NM01a	0.01	0.02	NM14b	0.02	2.89
G22	-	-	NM01b	0.00	0.00	NM14c	0.01	26.61
G23	-	-	NM01c	0.00	0.12	NM15a	0.02	0.75
G24	-	-	NM02a	0.00	0.03	NM15b	0.02	6.85
G25	-	-	NM02b	0.00	0.00	NM15c	0.02	119.67
G26	-	-	NM02c	0.00	0.85	NM16a	0.03	0.49
G27	-	-	NM03a	0.00	0.02	NM16b	0.02	4.19
G28	-	-	NM03b	0.00	0.14	NM16c	0.02	86.09
G29	-	-	NM03c	0.01	0.11	NM17a	0.03	0.40
G30	-	-	NM04a	0.00	0.02	NM17b	0.02	0.02
G31	-	-	NM04b	0.00	0.12	NM17c	0.01	0.01
G32	-	-	NM04c	0.00	0.06	NM18a	0.03	0.27
G33	-	-	NM05a	0.00	0.08	NM18b	0.02	5.62
DC1	0.01	0.05	NM05b	0.00	0.56	NM18c	0.02	0.01
DC2	0.16	0.16	NM05c	0.00	1.04	NM19a	0.03	0.37
DC3	0.01	0.04	NM06a	0.01	0.07	NM19b	0.02	2.23
DC4	0.01	0.01	NM06b	0.00	0.00	NM19c	0.02	0.01
DC5	0.00	0.00	NM06c	0.00	3.19	NM20a	0.06	0.96
DC6	0.03	0.06	NM07a	0.01	0.06	NM20b	0.04	23.45
DC7	0.01	0.05	NM07b	0.00	0.40	NM20c	0.04	255.35

dash (-): run-time exceeds 3600 seconds

Table 2.12: Initialisation; run-time in seconds.

	L	D		L	D		L	D
G1	0	0	DC8	0.06	0	NM7c	4.58	0.00
G2	0	0	DC9	0.04	0	NM8a	0.08	0.00
G3	0	0	RI1	0.13	0.02	NM8b	0.26	0.01
G4	0	0	RI2	0.2	0.03	NM8c	3.58	0.01
G5	0.01	0.01	RI3	0.11	0.03	NM9a	0.07	0.00
G6	0.01	0	RI4	0.23	0.02	NM9b	0.26	0.01
G7	0	0	RI5	0.21	0.02	NM9c	19.38	0.00
G8	0	0.01	RI6	0.14	0.02	NM10a	0.47	0.01
G9	0.01	0	RI7	0.16	0.03	NM10b	0.71	0.00
G10	0.01	0.01	RI8	0.15	0.03	NM10c	3.6	0.01
G11	0.01	0	RI9	0.17	0.02	NM11a	0.38	0.01
G12	0	0	NJ1	0.78	0.2	NM11b	3.77	0.01
G13	0.01	0.01	NJ2	0.92	0.2	NM11c	153.64	0.01
G14	0.01	0.01	NJ3	0.85	0.21	NM12a	0.49	0.01
G15	0	0.01	NJ4	0.9	0.19	NM12b	3.5	0.01
G16	0.01	0	NJ5	0.87	0.2	NM12c	799.07	0.00
G17	0	0	NJ6	0.95	0.21	NM13a	0.21	0.01
G18	0	0	NJ7	0.89	0.22	NM13b	0.88	0.01
G19	0	0	NJ8	0.85	0.21	NM13c	38.43	0.01
G20	0	0.01	NJ9	0.77	0.22	NM14a	0.29	0.01
G21	0	0	NM1a	0.05	0.01	NM14b	0.97	0.01
G22	0.01	0.01	NM1b	0.21	0	NM14c	18.04	0.01
G23	0	0.01	NM1c	3.95	0	NM15a	0.81	0.01
G24	0	0	NM2a	0.04	0	NM15b	2.3	0.01
G25	0	0	NM2b	0.18	0	NM15c	91.25	0.01
G26	0	0	NM2c	1.02	0	NM16a	0.57	0.01
G27	0	0.01	NM3a	0.02	0	NM16b	2.89	0.02
G28	0	0	NM3b	0.07	0	NM16c	790.97	0.01
G29	0	0	NM3c	0.25	0	NM17a	0.75	0.02
G30	0	0	NM4a	0.03	0	NM17b	4.05	0.01
G31	0	0	NM4b	0.08	0.01	NM17c	-	0.01
G32	0	0	NM4c	0.19	0	NM18a	0.41	0.02
G33	0	0	NM5a	0.1	0	NM18b	2.33	0.02
DC1	0.06	0	NM5b	0.18	0	NM18c	69.75	0.01
DC2	0.09	0	NM5c	1.66	0	NM19a	0.48	0.01
DC3	0.03	0	NM6a	0.2	0.01	NM19b	1.83	0.01
DC4	0.02	0	NM6b	0.55	0	NM19c	407.42	0.01
DC5	0.05	0	NM6c	6.79	0.01	NM20a	1.79	0.02
DC6	0.06	0	NM7a	0.15	0.01	NM20b	7.03	0.02
DC7	0.03	0	NM7b	1.3	0	NM20c	204.62	0.02

dash (-): run-time exceeds 3600 seconds

Table 2.13: Phase 1; run-time in seconds.

	SPAR	LDIC	DDIC		SPAR	LDIC	DDIC
G1	0.1	0.01	0.01	NM1a	5.18	0.15	0.00
G2	0.98	0.02	0.04	NM1b	2.72	NA	NA
G3	0.92	0.01	0.02	NM1c	1.61	1.42	0.01
G4	5.46	0.07	0.12	NM2a	5.56	0.11	0.01
G5	4.98	0.02	0.02	NM2b	3.08	NA	NA
G6	13.37	0.07	0.11	NM2c	1.64	0.98	0.00
G7	15.26	0.08	0.12	NM3a	5.72	0.07	0.00
G8	32.35	0.13	0.18	NM3b	2.72	0.06	0.00
G9	32.64	0.09	0.13	NM3c	2.1	0.11	0.00
G10	82.93	0.19	0.29	NM4a	7.2	0.06	0.01
G11	0.28	0.01	0.01	NM4b	2.86	0.56	0.00
G12	5.05	0.03	0.06	NM4c	2.34	0.2	0.00
G13	164.62	0.39	0.6	NM5a	20.16	0.17	0.00
G14	0.49	0.01	0.01	NM5b	9.05	0.88	0.00
G15	0.65	0.01	0.01	NM5c	7.4	1.4	0.00
G16	0.77	0.01	0.01	NM6a	32.84	0.38	0.01
G17	0.92	0.01	0.01	NM6b	20.28	NA	NA
G18	1.05	0.01	0	NM6c	8.95	0.98	0.00
G19	1.12	0	0.02	NM7a	34.3	0.44	0.01
G20	1.15	0	0.02	NM7b	20.9	1.44	0.00
G21	1.13	0.02	0.02	NM7c	9.34	2.01	0.01
G22	1.15	0.01	0.01	NM8a	39.22	0.04	0.00
G23	1.24	0.02	0.03	NM8b	21.54	0.34	0.00
G24	1.13	0.02	0.03	NM8c	12.14	1.25	0.00
G25	1.19	0.02	0.03	NM9a	37.9	0.3	0.02
G26	1.18	0.03	0.04	NM9b	23.43	NA	NA
G27	1.19	0.03	0.05	NM9c	13.05	8.03	0.00
G28	1.12	0.03	0.05	NM10a	139.21	0.46	0.01
G29	1.15	0.05	0.08	NM10b	73.85	5.24	0.02
G30	0.92	0.09	0.14	NM10c	45.56	2.8	0.01
G31	0.67	0.1	0.19	NM11a	170.45	0.41	0.02
G32	0.31	0.09	0.19	NM11b	103.98	3.5	0.00
G33	0.31	0.09	0.19	NM11c	44.62	5.66	0.00
DC1	1.86	0.04	0.01	NM12a	174	0.47	0.02
DC2	1.52	0.1	0.02	NM12b	119.01	5.32	0.00
DC3	1.72	0.02	0	NM12c	45.7	NA	NA
DC4	1.64	0.01	0.01	NM13a	193.63	0.08	0.00
DC5	1.33	NA	NA	NM13b	138.57	0.76	0.00
DC6	1.82	0.12	0.01	NM13c	62.62	NA	NA
DC7	1.5	0.02	0	NM14a	202.46	0.75	0.03

Continued on Next Page...

Table 2.13 – Continued

	SPAR	LDIC	DDIC		SPAR	LDIC	DDIC
DC8	1.76	NA	NA	NM14b	124.53	1.54	0.00
DC9	1.55	0.06	0.02	NM14c	62.5	11	0.01
RI1	48.88	0.1	0.06	NM15a	541.43	1.69	0.04
RI2	44.46	1.52	0.52	NM15b	328.52	13.8	0.02
RI3	48.03	0.07	0.02	NM15c	190.96	41.24	0.02
RI4	54.6	1.03	0.34	NM16a	398.4	1.26	0.04
RI5	53.34	0.89	0.42	NM16b	243.5	2.15	0.01
RI6	49.7	0.18	0.15	NM16c	107.48	192.37	0.02
RI7	51.27	0.28	0.11	NM17a	408.82	1.58	0.04
RI8	47.44	0.08	0.05	NM17b	264.65	NA	NA
RI9	47.88	0.66	0.31	NM17c	108.63	NA	NA
NJ1	2439	0.42	0.09	NM18a	490.24	0.68	0.02
NJ2	2443.5	2.66	1.08	NM18b	324.76	3.48	0.00
NJ3	2491	3	1.17	NM18c	151.72	NA	NA
NJ4	2450.5	1.24	0.42	NM19a	484.05	0.22	0.00
NJ5	2583.5	3.16	2.13	NM19b	325.62	2.06	0.00
NJ6	2389	4.88	2.64	NM19c	157.09	NA	NA
NJ7	2617	2.26	1.06	NM20a	1283.35	2.62	0.03
NJ8	2686	3.08	1.36	NM20b	778.32	5.58	0.01
NJ9	2602.5	3.28	2.3	NM20c	434.38	26.24	0.01

NA: no need to run Phase 1, as there is only one efficient solution, which is detected after initialisation.

Table 2.14: Phase 2; run-time in seconds.

	LCOR	LSET	NSPD	NSP		LCOR	LSET	NSPD	NSP
G01	0.02	0.02	0.00	0.01	NM01a	0.00	0.00	0.01	0.09
G02	0.05	0.04	0.01	0.01	NM01b	NA	NA	NA	NA
G03	0.23	0.45	0.07	0.07	NM01c	0.00	0.00	0.00	0.19
G04	0.07	0.08	0.01	0.00	NM02a	0.00	0.00	0.00	0.10
G05	3.67	15.37	15.52	15.57	NM02b	NA	NA	NA	NA
G06	0.15	0.31	0.02	0.01	NM02c	0.00	0.00	0.00	0.26
G07	2.72	9.71	99.98	99.46	NM03a	0.00	0.00	0.00	0.06
G08	1.11	4.42	0.17	0.15	NM03b	0.00	0.00	0.00	0.00
G09	7.59	30.34	328.95	330.37	NM03c	0.00	0.00	0.00	0.00
G10	1.52	5.73	0.20	0.18	NM04a	0.00	0.00	0.01	0.09
G11	5.91	29.99	1.52	1.43	NM04b	0.00	0.00	0.00	0.23
G12	0.04	0.04	0.01	0.01	NM04c	0.00	0.00	0.00	0.07
G13	0.63	2.21	0.22	0.21	NM05a	0.00	0.00	0.01	0.21
G14	16.85	104.03	163.46	164.91	NM05b	0.00	0.00	0.00	0.00

Continued on Next Page...

Table 2.14 – Continued

	LCOR	LSET	NSPD	NSP		LCOR	LSET	NSPD	NSP
G15	0.00	0.00	0.00	0.00	NM05c	0.00	0.00	0.00	1.00
G16	0.00	0.00	0.00	0.00	NM06a	0.00	0.00	0.01	0.20
G17	0.00	0.00	0.00	0.00	NM06b	NA	NA	NA	NA
G18	0.01	0.01	0.01	0.00	NM06c	0.00	0.00	0.00	2.84
G19	0.02	0.02	0.01	0.01	NM07a	0.00	0.00	0.01	0.15
G20	0.03	0.05	0.00	0.01	NM07b	0.00	0.00	0.00	0.90
G21	0.06	0.09	0.01	0.00	NM07c	0.00	0.00	0.00	0.00
G22	0.08	0.10	0.02	0.02	NM08a	0.00	0.00	0.00	0.06
G23	0.11	0.18	0.03	0.02	NM08b	0.00	0.00	0.00	0.73
G24	0.18	0.41	0.08	0.08	NM08c	0.00	0.00	0.00	2.01
G25	0.19	0.37	0.59	0.58	NM09a	0.00	0.00	0.01	0.17
G26	0.24	0.54	0.16	0.15	NM09b	NA	NA	NA	NA
G27	0.37	1.12	0.50	0.49	NM09c	0.00	0.00	0.01	2.83
G28	0.71	2.18	-	-	NM10a	0.00	0.00	0.01	0.26
G29	1.19	4.66	3.06	3.04	NM10b	0.00	0.00	0.01	2.71
G30	2.88	10.79	2.97	2.97	NM10c	0.00	0.00	0.00	1.25
G31	11.53	57.39	185.81	186.44	NM11a	0.00	0.00	0.01	0.32
G32	26.03	154.16	-	-	NM11b	0.00	0.00	0.01	1.64
G33	28.08	183.98	-	-	NM11c	0.00	0.00	0.01	7.09
DC1	0.08	0.06	0.00	0.04	NM12a	0.00	0.00	0.02	0.42
DC2	0.01	0.00	0.07	0.09	NM12b	0.00	0.00	0.00	3.43
DC3	0.01	0.00	0.01	0.04	NM12c	NA	NA	NA	NA
DC4	0.04	0.03	0.01	0.01	NM13a	0.00	0.00	0.01	0.12
DC5	NA	NA	NA	NA	NM13b	0.00	0.00	0.00	1.46
DC6	0.10	0.03	0.01	0.06	NM13c	NA	NA	NA	NA
DC7	0.01	0.01	0.01	0.04	NM14a	0.00	0.00	0.02	0.38
DC8	NA	NA	NA	NA	NM14b	0.00	0.00	0.01	2.19
DC9	0.03	0.04	0.09	0.13	NM14c	0.00	0.00	0.01	63.27
RI1	0.02	0.02	0.06	0.09	NM15a	0.00	0.00	0.03	1.80
RI2	1.67	0.53	0.43	1.13	NM15b	0.00	0.00	0.00	0.00
RI3	0.10	0.05	0.01	0.12	NM15c	0.00	0.00	0.01	40.80
RI4	0.38	0.23	0.16	0.56	NM16a	0.00	0.00	0.03	1.29
RI5	0.37	0.36	0.32	0.60	NM16b	0.00	0.00	0.01	3.81
RI6	8.12	1.93	1.21	1.32	NM16c	0.00	0.00	0.00	0.00
RI7	1.86	1.02	0.07	0.29	NM17a	0.00	0.00	0.03	1.07
RI8	1.53	0.47	0.05	0.12	NM17b	NA	NA	NA	NA
RI9	0.22	0.15	74.38	87.16	NM17c	NA	NA	NA	NA
NJ1	0.50	0.17	0.11	0.66	NM18a	0.00	0.00	0.03	0.43
NJ2	0.14	0.08	0.67	1.59	NM18b	0.00	0.00	0.01	4.42
NJ3	6.73	5.49	0.69	2.25	NM18c	NA	NA	NA	NA
NJ4	0.52	0.32	0.33	0.82	NM19a	0.00	0.00	0.01	0.28
NJ5	2.67	1.23	-	-	NM19b	0.00	0.00	0.00	0.00

Continued on Next Page...

Table 2.14 – Continued

	LCOR	LSET	NSPD	NSP		LCOR	LSET	NSPD	NSP
NJ6	10.78	11.70	1.46	2.60	NM19c	NA	NA	NA	NA
NJ7	0.73	0.28	0.71	1.36	NM20a	0.00	0.00	0.03	2.19
NJ8	5.09	2.44	0.71	2.04	NM20b	0.00	0.00	0.01	18.71
NJ9	36.71	26.69	-	-	NM20c	0.00	0.00	0.01	166.87

dash (-): run-time exceeds 3600 seconds;
NA: no need to run Phase 2, as there is only one efficient solution, which is detected after initialisation.

Table 2.15: Final results for grid networks; run-time in seconds.

	LCOR	LSET	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$		LCOR	LSET	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$
G01	0.01	0.01	0.02	0.02	37	G15	0.00	0.01	0.01	0.01	6
G02	0.05	0.05	0.06	0.05	80	G16	0.01	0.01	0.01	0.01	6
G03	0.21	0.53	0.27	0.49	124	G17	0.01	0.01	0.01	0.01	10
G04	0.07	0.10	0.08	0.10	46	G18	0.01	0.02	0.02	0.02	15
G05	3.14	16.19	3.78	15.49	290	G19	0.03	0.04	0.03	0.03	18
G06	0.15	0.44	0.18	0.34	44	G20	0.04	0.07	0.05	0.07	32
G07	2.62	11.21	2.83	9.83	149	G21	0.06	0.10	0.08	0.11	54
G08	1.00	5.04	1.22	4.53	122	G22	0.08	0.15	0.10	0.12	53
G09	6.93	32.86	7.77	30.52	247	G23	0.11	0.22	0.13	0.20	77
G10	1.54	6.99	1.63	5.85	132	G24	0.18	0.53	0.20	0.44	93
G11	5.82	35.01	6.17	30.24	204	G25	0.18	0.46	0.21	0.39	95
G12	0.03	0.05	0.05	0.05	52	G26	0.22	0.62	0.27	0.57	93
G13	0.59	2.57	0.69	2.27	113	G27	0.32	1.25	0.41	1.16	137
G14	15.14	114.56	17.42	104.59	309	G28	0.62	2.41	0.76	2.23	209
						G29	1.03	4.89	1.25	4.71	244
						G30	2.54	10.98	2.97	10.87	371
						G31	8.72	57.33	11.67	57.53	819
						G32	18.10	153.97	26.22	154.35	1383
						G33	20.85	183.93	28.27	184.17	1594

Table 2.16: Final results for NetMaker networks; run-time in seconds.

	LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$		LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$
NM01a	21.64	173.16	0.01	0.00	0.01	6	NM11a	240.73	3260.00	0.02	0.02	0.02	6
NM01b	72.80	62.22	0.00	0.00	0.00	1	NM11b	1103.90	1486.70	0.01	0.01	0.01	2
NM01c	2.45	16.28	0.00	0.01	0.01	3	NM11c	325.31	3090.00	0.01	0.01	0.01	2
NM02a	16.76	160.79	0.00	0.01	0.01	8	NM12a	199.80	3519.00	0.02	0.02	0.02	6

Continued on Next Page...

Table 2.16 – Continued

	LCOR	LSET	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$			LCOR	LSET	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$	
NM02b	46.61	45.24	0.00	0.00	0.01	1	NM12b	1424.13	2935.00	0.02	0.01	0.02	4
NM02c	2.55	15.71	0.00	0.01	0.00	4	NM12c	330.86	3019.00	0.01	0.01	0.01	1
NM03a	4.33	28.00	0.00	0.01	0.01	9	NM13a	86.36	907.54	0.02	0.01	0.02	2
NM03b	7.10	3.65	0.00	0.00	0.00	2	NM13b	684.42	995.65	0.02	0.01	0.01	2
NM03c	0.55	1.44	0.01	0.00	0.01	2	NM13c	63.74	477.96	0.01	0.01	0.00	1
NM04a	8.81	58.77	0.00	0.01	0.01	15	NM14a	102.97	1137.95	0.02	0.04	0.03	17
NM04b	7.18	4.19	0.00	0.01	0.00	3	NM14b	517.02	541.96	0.02	0.01	0.01	2
NM04c	0.68	1.70	0.00	0.01	0.01	4	NM14c	74.44	444.82	0.01	0.02	0.02	4
NM05a	27.93	174.34	0.00	0.01	0.01	6	NM15a	312.65	2985.00	0.02	0.04	0.04	7
NM05b	14.68	4.80	0.00	0.01	0.01	3	NM15b	1467.20	1098.58	0.02	0.03	0.03	3
NM05c	1.34	2.34	0.00	0.01	0.01	3	NM15c	96.33	580.16	0.02	0.03	0.03	3
NM06a	67.91	972.42	0.01	0.01	0.02	6	NM16a	432.66	-	0.03	0.04	0.05	5
NM06b	475.19	519.49	0.00	0.00	0.01	1	NM16b	1857.96	3407.00	0.02	0.02	0.02	3
NM06c	40.83	278.42	0.00	0.01	0.01	2	NM16c	1091.88	-	0.02	0.02	0.03	3
NM07a	71.74	857.47	0.01	0.01	0.01	5	NM17a	413.13	-	0.03	0.04	0.05	4
NM07b	257.42	439.81	0.00	0.01	0.01	3	NM17b	2080.91	-	0.02	0.01	0.01	1
NM07c	36.79	298.43	0.01	0.01	0.01	3	NM17c	1085.21	-	0.01	0.01	0.02	1
NM08a	37.39	294.95	0.01	0.01	0.00	3	NM18a	193.88	1697.10	0.03	0.03	0.04	7
NM08b	107.48	80.70	0.01	0.01	0.00	2	NM18b	892.60	1374.62	0.02	0.02	0.02	3
NM08c	9.85	41.12	0.00	0.01	0.01	3	NM18c	204.35	2005.44	0.02	0.01	0.01	1
NM09a	26.91	230.66	0.01	0.02	0.02	7	NM19a	99.37	854.51	0.03	0.02	0.01	4
NM09b	196.27	148.19	0.01	0.01	0.01	1	NM19b	1478.49	2278.00	0.02	0.01	0.01	2
NM09c	8.55	43.45	0.01	0.01	0.01	3	NM19c	204.50	1958.59	0.02	0.01	0.01	1
NM10a	165.92	1262.47	0.02	0.02	0.02	6	NM20a	881.79	-	0.06	0.05	0.05	5
NM10b	213.73	120.94	0.01	0.02	0.04	6	NM20b	1842.54	1647.66	0.04	0.03	0.03	3
NM10c	11.01	51.73	0.01	0.01	0.01	4	NM20c	341.01	2614.00	0.04	0.03	0.03	3

Table 2.17: Final results for road networks; run-time in seconds.

	LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$		LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$
DC1	0.16	0.08	0.01	0.09	0.06	2	RI6	11.03	2.44	2.36	8.28	2.09	3
DC2	0.24	0.14	0.16	0.03	0.02	6	RI7	4.27	1.89	0.07	1.99	1.15	3
DC3	0.40	0.18	0.01	0.01	0.01	3	RI8	5.83	0.97	0.07	1.59	0.53	4
DC4	0.21	0.10	0.01	0.04	0.03	2	RI9	9.43	4.10	53.89	0.54	0.47	22
DC5	0.26	0.07	0.00	0.00	0.00	1	NJ1	19.77	7.53	0.26	0.74	0.41	2
DC6	0.15	0.05	0.03	0.11	0.05	7	NJ2	32.47	20.61	0.85	1.36	1.28	6
DC7	0.31	0.13	0.01	0.02	0.02	2	NJ3	23.70	13.13	1.10	8.03	6.77	21
DC8	0.13	0.04	0.00	0.00	0.01	1	NJ4	29.48	19.20	0.39	1.07	0.88	5
DC9	0.45	0.26	0.09	0.04	0.05	6	NJ5	16.42	7.18	-	4.91	3.45	7
RI1	1.54	0.34	0.09	0.10	0.10	3	NJ6	62.41	66.33	-	13.52	14.43	12

Continued on Next Page...

Table 2.17 – Continued

	LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$		LCOR	LSET	NSPD	2LCOR	2LSET	$\lvert\mathcal{Z}_N\rvert$
RI2	7.74	1.97	71.80	2.21	1.06	15	NJ7	24.66	6.30	3.02	1.88	1.39	6
RI3	0.77	0.24	0.05	0.13	0.09	2	NJ8	11.76	5.07	0.65	6.50	3.84	13
RI4	2.68	0.99	0.82	0.73	0.58	17	NJ9	53.44	32.35	-	39.07	29.00	24
RI5	5.48	3.22	794.01	0.80	0.77	16							

dash (-): run-time exceeds 3600 seconds

Chapter 3

Bi-objective Integer Minimum Cost Flow Problems

Single-objective integer minimum cost flow problems have received a lot of attention in the literature as they have various applications (see for example Ahuja et al. 1993). As with most real-world optimisation problems, there is usually more than one objective that has to be taken into account, thus leading to multi-objective integer minimum cost network flow problems (MIMCF). We restrict our considerations to the bi-objective case (BIMCF). The aim in BIMCF is to find efficient solutions. The problem of finding all efficient solutions of BIMCF is intractable, Ruhe (1988) presents an example problem with exponentially many efficient solutions. BIMCF is an $\mathcal{NP}$-hard problem, as the bi-objective shortest path problem, a special case of BIMCF, was shown to be $\mathcal{NP}$-hard by Serafini (1986).

We propose to solve the BIMCF problem using the Two Phase Approach. In Phase 1, extreme efficient solutions are computed with a parametric network simplex algorithm (Sedeño-Noda and González-Martín 2000). Other efficient solutions are computed in Phase 2 using a ranking algorithm (Hamacher 1995) on restricted areas of the objective space. Our work constitutes one of the first correct published algorithm to solve BIMCF, see Raith and Ehrgott (2009b), and a preliminary version appeared in Raith and Ehrgott (2007).

We test our algorithm on different problem instances generated with the well known network generator NETGEN and also on networks with a grid structure.

The chapter is organised as follows: in Section 3.1 basic concepts of BIMCF problems are introduced. Recent literature is discussed in Section 3.2. In Section 3.4 we present an algorithm to solve BIMCF. Finally, numerical results are shown in Section 3.5.

3.1 Problem Formulation

In this section terminology and basic theory of bi-objective integer minimum cost flow (BIMCF) problems are introduced. The model is that of an MCF problem with two objectives and integer variables. The BIMCF problem also appears as Model (1.16) in Section 1.3.2. We repeat the formulation of BIMCF here.

Let $N = (G, c, l, u)$ be a *directed network* where the graph $G = (\mathcal{V}, \mathcal{A})$ consists of a set of nodes or vertices $\mathcal{V} = \{1, \ldots, n\}$ and a set of arcs $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$ with $|\mathcal{A}| = m$. Two costs $c_a = (c_a^1, c_a^2) \in \mathbb{Z} \times \mathbb{Z}$ are associated with each arc $a \in \mathcal{A}$. An integer numerical value b_i, the *balance*, is associated with each node $i \in \mathcal{V}$. A value $b_i > 0$, $b_i < 0$, or $b_i = 0$ indicates that, at node i, there exists a *supply* of flow, a *demand* of flow, or neither of the two (i is then called *trans-shipment* node). The BIMCF problem is defined by the following mathematical programme:

$$
\begin{array}{lll}
\min & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} & \\
\text{s.t.} & \sum\limits_{\{a \in \mathcal{A}: t(a) = i\}} x_a - \sum\limits_{\{a \in \mathcal{A}: h(a) = i\}} x_a = b_i & \text{for all } i \in \mathcal{V} \\
& u_a \geqq x_a \geqq l_a & \text{for all } a \in \mathcal{A} \\
& x_a \text{ integer} & \text{for all } a \in \mathcal{A},
\end{array}
\tag{3.1}
$$

with $z_1(x) = \sum_{a \in \mathcal{A}} c_a^1 x_a$ and $z_2(x) = \sum_{a \in \mathcal{A}} c_a^2 x_a$. Here x is the vector of flow on the arcs, the first set of constraints represents flow conservation at the different nodes, and we assume that $\sum_{i \in \mathcal{V}} b_i = 0$ since otherwise the problem is infeasible. The second set of constraints ensures that for each arc a flow remains between lower bound l_a and upper bound u_a. We assume $l_a = 0$ and $u_a \geqq l_a$. The third set of constraints ensures integer arc flow.

3.2 Literature on BIMCF Problems

An excellent and very recent review on multi-objective minimum cost flow problems is given by Hamacher et al. (2007). We will therefore only briefly mention relevant literature. There is little published work on MIMCF, so the following is mainly dedicated to BIMCF. Most exact solution approaches to find a (complete) set of efficient solutions for BIMCF, i.e. supported and non-supported efficient solutions, consist of two phases, i.e. they are following the Two Phase Method.

In case all capacities, supplies, and demands are integer, which we assume in this chapter, any approach to solve the bi-objective continuous network flow problem can be used in Phase 1 of BIMCF to find a complete set of (extreme) supported efficient solutions, e.g. Lee and Pulat (1991); Pulat et al. (1992); Sedeño-Noda and González-Martín (2000, 2003). We discussed this property of BIMCF problems when introducing the parametric network simplex approach for Phase 1 in Section 1.3.2. The algorithms presented by Lee and Pulat (1991); Pulat et al. (1992) may generate some non-extreme supported efficient solutions, whereas the algorithms by Sedeño-Noda and González-Martín (2000, 2003) generate extreme efficient solutions only.

Lee and Pulat (1991) claim that their procedure can be extended to generate all integer efficient solutions with image on the edges of $conv(\mathcal{Z})$, i.e. all supported efficient solutions. Their procedure works as follows: every efficient solution found by their algorithm corresponds to a basic feasible solution of (1.1), represented by a spanning tree in N. Two solutions x^1 and x^2 are called *adjacent* if the two corresponding trees have $n-2$ arcs in common. For adjacent solutions x^1 and x^2, the arc that is in the basis of x^2, but not in the basis of x^1, can be introduced into the basic tree of x^1 resulting in a cycle. As much additional flow as possible is sent along that cycle until the flow on one of the arcs in the cycle reaches its upper or lower bound, this determines the flow change σ along the cycle. The arc with flow at upper or lower bound leaves the tree, resulting in the tree of x^1's adjacent solution, x^2. Whenever the flow changes by σ along the cycle, when moving from one efficient solution to an adjacent one, the authors propose to increase the flow stepwise by $1, 2, \ldots, \sigma - 1$ to obtain *intermediate* solutions and claim to obtain all supported efficient solutions this way. This claim is incorrect, as not all non-extreme supported

efficient solutions can be obtained as intermediate solutions of two adjacent basic efficient solutions, an example is given by Eusébio and Figueira (2006).

Several papers are dedicated to the computation of non-supported efficient solutions of BIMCF, assuming all non-dominated extreme points are known. Lee and Pulat (1993) perform an explicit search of the solution space, by using intermediate solutions between adjacent basic solutions (which is not sufficient, see remark above) and modifying upper and lower bounds of arcs. They assume non-degeneracy of the problem. Huarng et al. (1992) extend this algorithm to allow degeneracy in the problems.

Sedeño-Noda and González-Martín (2001) argue that these two papers are incorrect and present an approach that is based on the basic tree structure of solutions. Having found a complete set of extreme efficient solutions in Phase 1, the algorithm by Sedeño-Noda and González-Martín (2001) moves from one efficient solution to adjacent solutions, in order to identify efficient ones among them. Przybylski et al. (2006) give an example of a network where one efficient solution is not adjacent to any of the other efficient solutions, hence showing that the approach by Sedeño-Noda and González-Martín (2001) cannot generate a complete efficient set. We would like to point out that the same example can also be used to show that the approach by Lee and Pulat (1993) is incorrect, see Section 3.3.

Eusébio and Figueira (2009b) illustrate and give proof that supported efficient solutions are indeed connected via chains of zero-cost cycles in the incremental graph constructed from basic feasible solutions corresponding to extreme efficient solutions. They use this relationship to characterise all supported efficient solutions to a MIMCF problem and present an algorithm for their computation.
The same result can be obtained by considering a weighted sum formulation (1.1) of MIMCF for which two extreme efficient solutions corresponding to two consecutive non-dominated extreme points are optimal. The non-dominated points on the edge of $conv(\mathcal{Z})$ connecting the two extreme non-dominated points can be obtained by applying the k best flow algorithm by Hamacher (1995) to the problem with weighted sum objective. The k best flow algorithm is also based on cycles in the incremental graph. We explain how to apply the k best flow algorithm to find all supported and non-supported solutions of

BIMCF in Section 3.4.2.

Eusébio and Figueira (2009a) present an approach where ε-constraint problems, $\min\{z_1(x) : x \in X, z_2(x) \leqq \varepsilon\}$, are repeatedly solved to obtain all efficient solutions (supported and non-supported). It is crucial that the arising ε-constraint problems are solved efficiently. The (possibly fractional) solution of the ε-constraint problem is obtained by sending a fractional amount of flow along the cycle connecting two adjacent basic efficient solutions x^1 and x^2, one with $z_2(x^1) \leqq \varepsilon$ and the other with $z_2(x^2) > \varepsilon$. If the solution is fractional, an integer solution can be obtained by branching on arcs with fractional flow. This can be done efficiently as only arcs in the cycle can have non-integer flow and it is therefore sufficient to branch on arcs in the cycle.

Eusébio and Figueira (2006) give examples of networks, where for a supported extreme and supported non-extreme non-dominated point, both basic and non-basic supported efficient solutions exist. It is known from linear programming that there is always a basic feasible solution for every extreme non-dominated point, but the authors show that there may be other non-basic efficient solutions that lead to the same point. The network simplex method can be used to identify all basic efficient solutions at that point. The non-basic solutions can be obtained as convex combinations of the basic ones, but not by the network simplex algorithm itself. Eusébio and Figueira (2006) also give a network in which supported efficient solutions exist that cannot be obtained as intermediate solutions between two extreme efficient solutions.

3.3 Incorrectness of the approach by Lee and Pulat (1993)

Using the network by Przybylski et al. (2006), we can show that the approach by Lee and Pulat (1993) will not find all efficient solutions. First, we repeat the example from Przybylski et al. (2006).

Example 3.3.1 *The network is given in Figure 3.1. The lower bound on each arc is* 0 *and there is no upper bound restriction on flow, furthermore all flow values are required to be integer. Node* 1 *has balance* $b_1 = 1$ *and node* 13 *has*

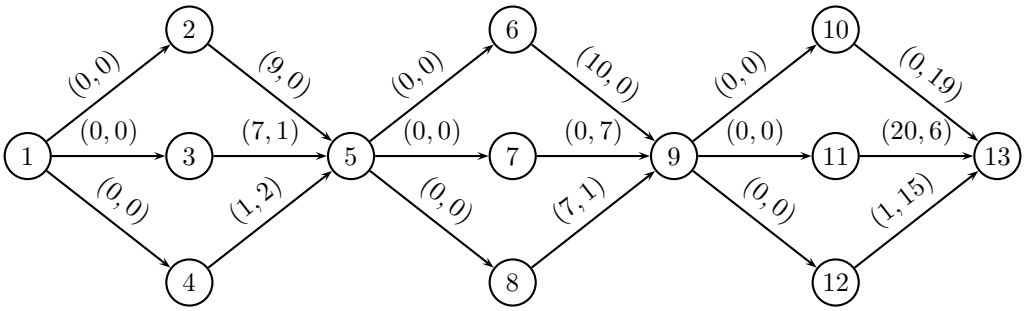


Figure 3.1. Network of Example 3.3.1.

balance $b_{13} = -1$*, all other nodes have balance* $b_i = 0, i = 2, \ldots, 12$*. Arc costs are given above the arcs in the figure. This is a BSP problem, as one unit of flow is sent from node* 1 *to node* 13*. Every feasible solution has a flow of* 1 *on every arc that is part of the path, and zero flow on all other arcs.*

Lee and Pulat (1993) first formulate what they call a "brute force" version of their algorithm, which is then refined. We show here that the first version of the algorithm, summarised in Algorithm 7, already fails to generate all efficient solutions.

It should be noted that within this algorithm the lower bound on only one arc increases at a time. It is not clear why this should yield all non-supported efficient solutions or all non-dominated points. Sedeño-Noda and González-Martín (2001) also point this out but do not give an example for which the algorithm actually fails to identify all solutions. In the following we show that the non-supported efficient solution from Figure 3.2 with objective vector $(27, 14)$ cannot be obtained by Algorithm 7.

To show this, we simply look at the set of feasible solutions of the network when lower bounds on certain arcs of N are increased to 1. The highlighted solution with objective vector $(27, 14)$ will only be feasible when increasing the lower bound on any of the horizontal (red) arcs. We plot all obtained feasible solutions when increasing the lower bound on arcs $1 \to 3$ or $3 \to 5$ (Figure 3.3), on arcs $5 \to 7$ or $7 \to 9$ (Figure 3.4), or on arcs $9 \to 11$ or $11 \to 13$ (Figure

Algorithm 7 "Brute force" Algorithm by Lee and Pulat (1993)

input: Network (G, c, l, u), and $\mathcal{E}_{supp}$, the set of all extreme supported efficient solutions (BFSs), $x^1, x^2 \ldots, x^r$ of BIMCF and all other supported solutions.
for all $t = 1, \ldots, r$ **do**
  $\mathcal{T}_t$ is the basis associated with x_t.
  for all non-basic arcs $s \in \overline{\mathcal{T}}_t$, with $s \notin \mathcal{T}_{t+1}$ **do**
    Derive new lower bounds l' as $\begin{cases} l'_a = l_a + 1 & \text{if } a = s \\ l'_a = l_a & \text{otherwise} \end{cases}$
    while $l'_s \leqq u_s$ **do**
      E = the set of all (integer) supported solutions of (G, c, l', u).
      $\mathcal{E}_{t,s} = \mathcal{E}_{t,s} \cup E$ /* $\mathcal{E}_{t,s}$ contains all feasible solutions computed for t by increasing the lower bound on arc s. */
      $l'_s = l'_s + 1$
    end while
  end for
end for
$\mathcal{E}$ = all efficient solutions among $\left(\bigcup_{t,s} \mathcal{E}_{t,s}\right) \cup \mathcal{E}_{supp}$
output: Efficient non-supported solutions $\mathcal{E}$.

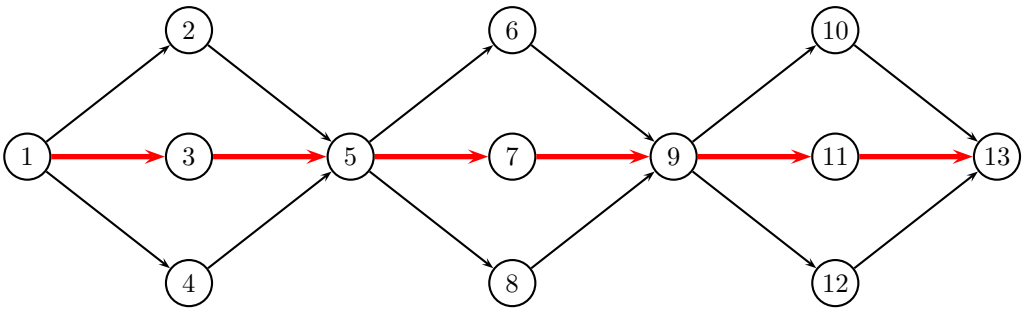


Figure 3.2. One feasible solution of network of Example 3.3.1 with a flow of one on arcs $1 \to 3$, $3 \to 5$, $5 \to 7$, $7 \to 9$, $9 \to 11$, $11 \to 13$, and zero flow on all other arcs. The objective vector of the solution is $(27, 14)$.

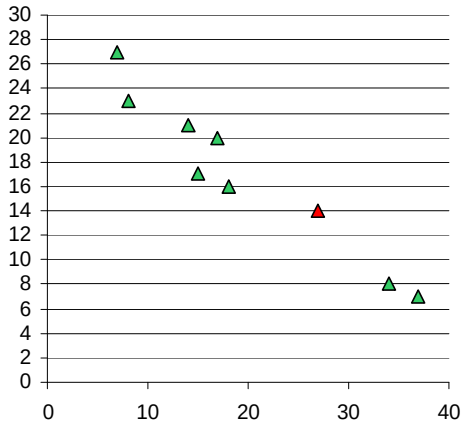


Figure 3.3. All feasible solutions with increased lower bound on $1 \rightarrow 3$ or $3 \rightarrow 5$. The solution from Figure 3.2 is red.

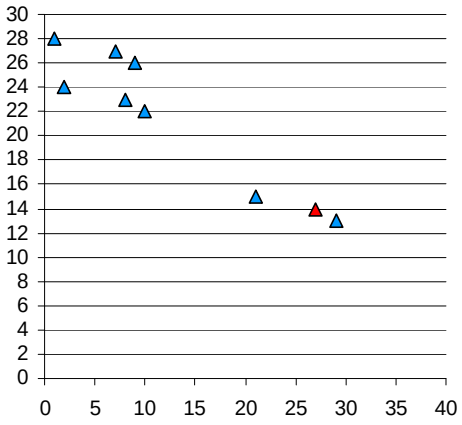


Figure 3.4. All feasible solutions with increased lower bound on $5 \rightarrow 7$ or $7 \rightarrow 9$. The solution from Figure 3.2 is red.

3.5). In each of the figures, the solution from Figure 3.2 (the red triangle) is non-supported, and can therefore not be identified by Algorithm 7.

Note that there is no degeneracy in the network from Example 3.3.1, which is an assumption in Lee and Pulat (1993). This can be seen as arcs with positive flow are always part of the basic tree and all other arcs in the basic tree have flow at their lower bound. Whenever a non-basic arc is added to the tree, it is always possible to send one unit of flow along the resulting cycle.

3.4 A Two Phase Algorithm to Solve BIMCF

We solve the BIMCF problem with the Two Phase Method as introduced in Section 1.3.2

In Phase 1 extreme efficient solutions are computed. There are two main approaches in the context of BIMCF: Sedeño-Noda and González-Martín (2003) follow a dichotomic approach in repeatedly computing solutions to a problem in which a Tchebycheff norm is minimised using Lagrangian relaxation. Whenever a new extreme point is obtained, the problem is split into two sub problems each with their own Tchebycheff norm derived from the local nadir point. The other main approach is based on the network simplex method where extreme

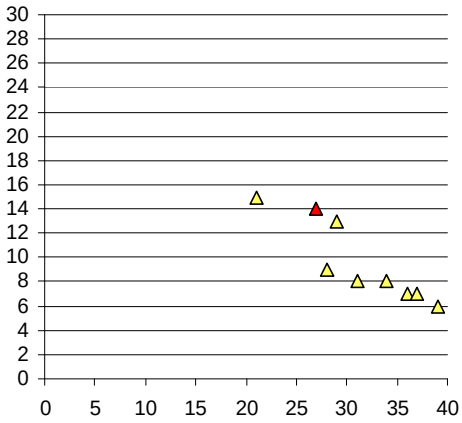


Figure 3.5. All feasible solutions with increased lower bound on 9 → 11 or 11 → 13. The solution from Figure 3.2 is red.

efficient solutions are generated in a right-to-left (or left-to-right) fashion, e.g. Sedeño-Noda and González-Martín (2000). We use the latter approach here. In Phase 2 the remaining supported and non-supported non-dominated points are computed with an enumerative approach.

3.4.1 Phase 1 – Parametric Simplex

Phase 1 is solved with the parametric network simplex algorithm as detailed in Section 1.2.2 for continuous BCMCF problems. As stated in Section 1.3.2, the parametric network simplex algorithm will find integer extreme efficient solutions given all bounds, the costs, and the balances of the BIMCF problem are integer. We can thus simply use Algorithm 2, the parametric network simplex algorithm, to solve Phase 1.

3.4.2 Phase 2 – Ranking k Best Flows

In Phase 2, (at least) a complete set of the remaining supported non-extreme efficient solutions and non-supported efficient solutions is computed. As discussed in Section 1.3.2, the objective vectors of those solutions can only be

situated in the triangle defined by two consecutive extreme points as indicated in Figure 1.15. We make the same assumption as for the Phase 2 ranking approach from Section 1.3.2: let $z^1, \ldots, z^s$, where $z^i = (z_1(x^i), z_2(x^i))$ and z^i are sorted by increasing z_1, be the non-dominated extreme points obtained in Phase 1. For each pair of neighbouring extreme points z^i and z^{i+1}, weighting factors are defined as in (1.17). Using λ_1 and λ_2 in (1.1), we obtain a single-objective flow problem which has optimal solutions x^i, x^{i+1}. We denote the weighted sum objective by $c^\lambda = \lambda_1 z_1(x) + \lambda_2 z_2(x)$.

Applying the k best flow algorithm by Hamacher (1995) to the single objective problem $\min_{x \in X} c^\lambda$, we can generate feasible network flows in order of their cost. The k best flow algorithm is used to generate all feasible integer flows in the current triangle until it can be guaranteed that all non-dominated points have been found, see Section 1.3.1. Before we continue with the algorithm for Phase 2, we explain the k best flow algorithm.

The k Best Flow Algorithm

We give a summary of the k best flow algorithm here, the reader is referred to Hamacher (1995) for a more detailed description and proofs. First, we outline the k best flow algorithm for the single-objective minimum cost flow problem.

Starting with an optimal solution x in the network N, a so-called *incremental graph* G_x is constructed in which every arc represents an arc in N on which flow may be increased or decreased. Any cycle in G_x represents a change of flow that leads from x to another feasible flow. Identifying a minimal cycle in the incremental graph leads to a second best flow solution in N. Then, the problem is partitioned by modifying one lower and upper bound on an arc of N so that in one partition the original solution is optimal and the second best solution is infeasible and vice versa. By iterating this process, a ranking of the k best solutions can be obtained.

In Hamacher (1995) the algorithm is designed to solve problems in networks with the property that there cannot be more than one arc connecting nodes i and j, no matter if they have the same or opposite directions. When solving BIMCF problems, randomly generated networks generally do not satisfy this property. Also, real-world networks will most likely not satisfy this prop-

erty (e.g. road networks). We outline a generalisation of the algorithm in the following. The only difficulty with multiple arcs between a pair of nodes is keeping track of the correspondence of arcs in the incremental graph and arcs in the original network.

First, construct the incremental graph, a directed graph $G_x = (\mathcal{V}, \mathcal{A}_x)$ with arc costs c^x, corresponding to an *optimal flow* x in $N = (G, c, l, u)$ with

$$\begin{aligned} a^+ &\in \mathcal{A}_x^+ \qquad \text{for all } a \in \mathcal{A} \text{ and } x_a < u_a, \\ a^- &\in \mathcal{A}_x^- \qquad \text{for all } a \in \mathcal{A} \text{ and } x_a > l_a, \end{aligned}$$

and let $\mathcal{A}_x = \mathcal{A}_x^+ \cup \mathcal{A}_x^-$. The arcs in $\mathcal{A}_x$ have the following relationship to arcs in $\mathcal{A}$: we define each $a^+ \in \mathcal{A}_x$ as arc with $t(a^+) = t(a)$, $h(a^+) = h(a)$, and $c^x_{a^+} = c_a$. For each $a^- \in \mathcal{A}_x$ we define $t(a^-) = h(a)$, $h(a^-) = t(a)$, and $c^x_{a^-} = -c_a$. Note that in the following G_x is always constructed from the network N in which x is a best solution.

If for an arc $a \in \mathcal{A}$ both $a^+ \in \mathcal{A}_x$ and $a^- \in \mathcal{A}_x$ (which is the case only if $l_a < x_a < u_a$), we call a^+ and a^- a pair of *symmetric* arcs, otherwise we call an arc *non-symmetric*. A *proper minimum cost cycle* is a minimum cost cycle in G_x, excluding all cycles that consist of one or multiple pairs of symmetric arcs a^+, a^-. The set of all cycles to be considered is

$$\begin{aligned} \mathcal{C}(G_x) = \{C : C \text{ cycle in } G_x \text{ with at least one arc } a^+ \in C \text{ and } a^- \notin C \\ \text{or } C \text{ cycle in } G_x \text{ with at least one arc } a^- \in C \text{ and } a^+ \notin C\}. \end{aligned} \tag{3.2}$$

From $\mathcal{C}(G_x)$ a proper minimum cost cycle $C \in \operatorname{argmin}\{c^x(C) : C \in \mathcal{C}(G_x)\}$ is obtained. By increasing the flow by one unit along C, a second-best flow $\hat{x}$ is obtained in the original network N: in N, increasing the flow on arc $a^+ \in \mathcal{A}_x$ corresponds to increasing the flow on arc $a \in \mathcal{A}$, and increasing the flow on $a^- \in \mathcal{A}_x$ corresponds to decreasing the flow on arc $a \in \mathcal{A}$, see Procedure 4. This yields a second best flow $\hat{x}$ with $c(x) \leqq c(\hat{x})$, where $c(\hat{x}) = c(x) + c^x(C)$.

Now the network N is modified, by adjusting one lower and upper bound of N, so that x remains optimal in the network (G, c, l, u') with modified upper bound u' and $\hat{x}$ is infeasible in this network. Also, new lower bounds l'' are derived, so that $\hat{x}$ becomes optimal and x infeasible in (G, c, l'', u). In order to do this, the bounds of only one of the arcs a where flow was increased by

Procedure 4 `compute_second_best_flow`$[N, x, C]$

1: **input**: Network $N = (G, c, l, u)$, best solution x, and proper minimum cost cycle C.

2: $\hat{x}_a = \begin{cases} x_a + 1 & \text{if } a^+ \in C \cap \mathcal{A}_x^+ \\ x_a - 1 & \text{if } a^- \in C \cap \mathcal{A}_x^- \\ x_a & \text{otherwise} \end{cases}$

3: **output**: Second best flow $\hat{x}$.

one unit are modified as in Procedure 5. Note that this arc always exists as $c^x(C) \geqq 0$ implies $C \cap \mathcal{A}^+ \neq \emptyset$.

Procedure 5 `derive_partition`$[N, x, C]$

1: **input**: Network $N = (G, c, l, u)$, best solution x, and proper minimum cost cycle C.

2: Select one arc $\tilde{a}$ with $\tilde{a}^+ \in C \cap \mathcal{A}_x^+$

3: $u' = \begin{cases} u'_a = x_a & \text{if } a = \tilde{a} \\ u'_a = u_a & \text{otherwise} \end{cases}$ and $l'' = \begin{cases} l''_a = x_a + 1 & \text{if } a = \tilde{a} \\ l'_a = l_a & \text{otherwise} \end{cases}$

4: **output**: Network $N' = (G, c, l, u')$ and network $N'' = (G, c, l'', u)$.

In each of the two networks with modified bounds l'' and u', respectively, we can again compute a second best flow. Out of the two second best solutions, the flow with smaller cost is selected, this is the third best solution in the original network N. The partition in which the third best flow was obtained is again partitioned and both new partitions resolved. This process continues iteratively. Pseudo-code for the k best flow algorithm is shown in Algorithm 8.

Adaptation of the k Best Flow Algorithm in Phase 2

When solving Phase 2, we cannot specify a value of k a priori. Instead, we continue until it is guaranteed that all non-dominated points in each triangle T^i, given by the three points $(z_1(x^i), z_2(x^i))$, $(z_1(x^{i+1}), z_2(x^{i+1}))$, and $(z_1(x^{i+1}), z_2(x^i))$, have been found as detailed in Section 1.3.2.

Whenever a solution with cost vector within the triangle is found that is not dominated by a solution found previously (and also not equivalent to a solution found previously), it is saved and the upper bound can be improved, as the new point dominates parts of the triangle. Ranking flows continues until the

Algorithm 8 k best flows

input: Network $N = (G, c, l, u)$, best solution x, and k.
/* For a network N with best solution x, we denote the incremental graph by $G_x = (\mathcal{V}, \mathcal{A}_x)$ and it has costs c^x */
$C \in \operatorname{argmin}\{c^x(C) : C \in \mathcal{C}(G_x)\}$ /* See (3.2) for $\mathcal{C}(G_x)$ */
$\hat{x} =$ `compute_second_best_flow`$[N, x, C]$
$\mathcal{P} = \{(x, \hat{x}, N, C)\}$ /* Initialise set of partitions. */
$l = 2$
while $\mathcal{P} \neq \emptyset$ and $l < k$ **do**
  Choose $(x^p, \hat{x}^p, N^p, C^p)$ with $c(\hat{x}^p) = \min\{c(\hat{x}^q) : (x^q, \hat{x}^q, N^q, C^q) \in \mathcal{P}\}$.
  $\mathcal{P} = \mathcal{P} \backslash (x^p, \hat{x}^p, N^p, C^p)$
  $\{N', N''\} =$ `derive_partition`$[N^p, x^p, C^p]$ /* $x^p, \hat{x}^p$ is best solution in N', N'', respectively */
  if $\mathcal{C}(G_{x^p}) \neq \emptyset$ **then**
    Identify $C' \in \operatorname{argmin}\{c^{x^p}(C) : C \in \mathcal{C}(G_{x^p})\}$. /* (3.2) for $\mathcal{C}(G_{x_p})$ */
    $x' =$ `compute_second_best_flow`$[N', x^p, C']$
    $\mathcal{P} = \mathcal{P} \cup \{(x^p, x', N', C')\}$
  end if
  if $\mathcal{C}(G_{\hat{x}^p}) \neq \emptyset$ **then**
    Identify $C'' \in \operatorname{argmin}\{c^{\hat{x}^p}(C) : C \in \mathcal{C}(G_{\hat{x}^p})\}$. /* (3.2) for $\mathcal{C}(G_{\hat{x}^p})$ */
    $x'' =$ `compute_second_best_flow`$[N'', \hat{x}^p, C'']$
    $\mathcal{P} = \mathcal{P} \cup \{(\hat{x}^p, x'', N'', C'')\}$
  end if
  Save l^{th} best flow $\hat{x}^p$.
  $l = l + 1$
end while
output: $2^{\text{nd}}, 3^{\text{rd}}, \ldots, l^{\text{th}}$ best flow and $l \leqq k$.

weighted sum value c^{λ^i} of the flow solutions exceeds upper bound Δ_i from (1.19). Thus we aim at computing a complete set of efficient solutions.

The Phase 2 algorithm incorporating the upper bound and parts of the k best flow algorithm is described in Algorithm 9.

Let $\mathcal{E}_i = \{x^{i,0}, x^{i,1}, \ldots, x^{i,r}, x^{i,r+1}\}$ be a set of feasible non-equivalent solutions whose objective vectors are not dominating each other, and with image in triangle T_i defined by the supported efficient solutions $x^{i,0} = x^i$ and $x^{i,r+1} = x^{i+1}$. Then, the set $\mathcal{E} = \bigcup_{i=1,\ldots,s-1} \mathcal{E}_i$ generated by Algorithm 9 is a complete set of efficient solutions of BIMCF. This can be seen by applying the proof of Theorem 1.3.1 to every triangle T_i.

Remark 3.4.1 If there are multiple equivalent efficient solutions the rank-

Algorithm 9 Phase 2 BIMCF

1: **input**: Network (G, c, l, u) with $c = (c^1, c^2)$ and list of extreme efficient solutions $x^1, \ldots, x^s$.
2: $i = 1$
3: **while** $i < s$ **do**
4: $\mathcal{E}_i = \{x^i, x^{i+1}\}$ /* $x^{i,0} = x^i$ and $x^{i,r+1} = x^{i+1}$ */
5: Compute λ_1^i, λ_2^i and $c^{\lambda^i} = \lambda_1^i c^1 + \lambda_2^i c^2$. /* See (1.17) for λ_1^i, λ_2^i */
6: $\Delta_i = \delta_i$ /* Initial value for Δ_i; see (1.17) for δ_i */
7: $C = \text{argmin}\{c^x(C) : C \in \mathcal{C}(G_x)\}$ /* See (3.2) for $\mathcal{C}(G_x)$ */
8: $\hat{x} =$ `compute_second_best_flow`$[N, x^{i+1}, C]$
9: $\mathcal{P} = \{(x^{i+1}, \hat{x}, N, C)\}$
10: **while** $\mathcal{P} \neq \emptyset$ and $\min\{c^{\lambda^i}(\hat{x}^p) : (x^p, \hat{x}^p, N^p, C^p) \in \mathcal{P}\} \leqq \Delta_i$ **do**
11: Steps 7-19 in Algorithm 8. /* Execute one iteration of k best flow */
12: **if** $z(\hat{x}^p)$ in current triangle and not dominated by the objective vector of any element of $\mathcal{E}_i$ and $\hat{x}^p$ not equivalent to any $x \in \mathcal{E}_i$ **then**
13: Insert $\hat{x}^p$ into $\mathcal{E}_i$.
14: Update Δ_i. /* See (1.19) for Δ_i */
15: **end if**
16: **end while**
17: $i = i + 1$
18: **end while**
19: **output**: Complete set $\mathcal{E} = \bigcup_{i=1,\ldots,s-1} \mathcal{E}_i$ of efficient solutions.

ing algorithm can generate them all. In the Phase 2 approach described in Algorithm 9, however, only one of them is inserted into the set $\mathcal{E}_i$. *All* efficient solutions can be found by slightly altering the Phase 2 approach to keep equivalent solutions. Also, another component needs to be considered when calculating the upper bound to ensure that ranking is not terminated before all equivalent solutions have been obtained. The upper bound Δ_i can be chosen as in Equation (1.18) in this case. In addition, Step 12 of Algorithm 9 is modified to keep every efficient solution $\hat{x}^p$ including those that are equivalent to a previously obtained efficient solution.

As we aim at obtaining a *complete* set of efficient solutions, in our implementation we only keep one of the equivalent solutions for each non-dominated point in objective space.

Remark 3.4.2 Unfortunately the k best flow algorithm generates solutions with objective vector outside the current triangle T_i which cannot be removed from $\mathcal{P}$ as those solutions might later lead to other solutions within the trian-

gle. Whenever a solution x^* with cost outside the current triangle lies within another triangle T_j, we could save this solution and use it to compute a better upper bound Δ^* in T_j. This will, however, not speed up the algorithm, as flows in T_j still have to be ranked starting from the least cost flow. When ranking flows in T_j, one of the following two cases occurs:

- Ranking flows and updating the upper bound in T_j stops the algorithm before the solution x^* is enumerated, or
- Ranking flows in T_j generates the solution x^* again, now the bound is updated to Δ^* (or a better upper bound $\Delta_j < \Delta^*$).

Thus, saving solutions in other triangles cannot improve the run-time of Phase 2.

Remark 3.4.3 We can use smaller triangles than those given by extreme efficient solutions: We can consider intermediate solutions whenever the flow between two adjacent solutions obtained in Phase 1 changes by $\sigma > 1$ along the cycle connecting the two solutions. Intermediate solutions can be used to construct $\sigma - 1$ smaller triangles. Due to the nature of the Phase 2 algorithm, including those smaller triangles instead of the one defined by the two extreme efficient solutions does not present an advantage. The ranking algorithm would generate the same rankings $\sigma - 1$ times as we cannot restrict the ranking to the current triangle. There is also no advantage in a better upper bound, as the ranking algorithm will first generate all alternative optimal solutions (i.e. the non-extreme supported efficient solutions including the intermediate solutions), and after that the upper bound will be as good as it would be in the smaller triangles.

Remark 3.4.4 We briefly comment on the implementation of this algorithm. As mentioned before, the network simplex algorithm in Phase 1 is an adaptation of the `MCF` algorithm by Löbel (2004). In Phase 2, we implemented the k-best flow algorithm as outlined in Algorithm 9. The most time-consuming part of this algorithm is finding a proper minimum cost cycle in Steps 2, 11, and 16 of Algorithm 8 and Step 7 of Algorithm 9. This is implemented as suggested in Hamacher (1995): the set $\mathcal{C}(G_x)$ is obtained by combining each

non-symmetric arc $a \in \mathcal{A}$ with a shortest path from $h(a)$ to $t(a)$ in the network. Symmetric arcs a^+ or a^- are combined with the shortest path from their head node to their tail node in the network with arc set $\mathcal{A}_x \setminus \{a^-\}$ or $\mathcal{A}_x \setminus \{a^+\}$, respectively. Altogether the shortest paths between every pair of nodes connected by an arc needs to be calculated in a network with negative arc costs c^x. To be able to use the efficient implementation of Dijkstra's shortest path algorithm despite the negative costs, initially the shortest paths from one node to all other nodes are calculated using a shortest path algorithm suitable for negative arc costs such as Floyd's algorithm (e.g. Ahuja et al. 1993). Using the obtained shortest path distances, the costs c^x can be adjusted to positive costs which enables us to use Dijkstra's algorithm. Nevertheless, more than 99% of the runtime is spent in Phase 2, mostly calculating shortest paths for the proper minimum cost cycle.

3.5 Numerical Results

We investigate the performance of the proposed Two Phase Method with parametric network simplex in Phase 1 and flow ranking in Phase 2. In order to do so, networks are generated by NETGEN (Klingman et al. 1974), which is slightly modified to include a second objective function. We generate two sets of test instances, with the following parameters fixed for all problems: $mincost = 0$, $maxcost = 100$, $\%highcost = 0$, $\%capacitated = 100$, $mincap = 0$, and $maxcap = 50$. Furthermore, we vary parameters as in Table 3.1. The table shows in each column which values of n, m, number of sources and sinks, etc. are selected. We generate 30 problems for each set of parameters. We generate problems N01-N12 with varying sum of supply ($\sum_{i \in \mathcal{V}: b_i > 0} b_i$) and problems F01-F12 with fixed total sum of supply, as we observe that increasing the sum of supply with the network size significantly complicates the problem.

We also generate networks with a grid structure. Nodes are arranged in a rectangular grid with given height and width. Every node has at most four outgoing arcs (up, down, left, and right), to its immediate neighbours. Grid networks have a structure identical to that of grid networks in the previous chapter, see Figure 2.1 in Section 2.4.1. A grid is defined by the parameters

Table 3.1. Test Instances: NETGEN.

Name	n	m	sources	sinks	$\sum_{i\in\mathcal{V}:b_i>0} b_i$	transhipment sources	trans-shipment sinks
N01 / F01	20	60	9	7	90 / 100	4	3
N02 / F02	20	80	9	7	90 / 100	4	3
N03 / F03	20	100	9	7	90 / 100	4	3
N04 / F04	40	120	18	14	180 / 100	9	7
N05 / F05	40	160	18	14	180 / 100	9	7
N06 / F06	40	200	18	14	180 / 100	9	7
N07 / F07	60	180	27	21	270 / 100	14	10
N08 / F08	60	240	27	21	270 / 100	14	10
N09 / F09	60	300	27	21	270 / 100	14	10
N10 / F10	80	240	35	38	350 / 100	17	14
N11 / F11	80	320	35	38	350 / 100	17	14
N12 / F12	80	400	35	38	350 / 100	17	14

Table 3.2. Test Instances: grid.

Name	h	w	n	m	c_{max}	u_{max}	$\sum_{i\in\mathcal{V}:b_i>0} b_i$
G01	4	5	20	62	100	50	100
G02	5	8	40	134	100	50	100
G03	6	10	60	208	100	50	100
G04	8	10	80	284	100	50	100
G05	6	10	60	208	100	75	100
G06	6	10	60	208	100	100	100
G07	6	10	60	208	25	50	100
G08	6	10	60	208	50	50	100
G09	8	10	80	284	100	75	100
G10	8	10	80	284	100	100	100
G11	8	10	80	284	25	50	100
G12	8	10	80	284	50	50	100

height h, width w, maximum cost c_{max}, maximum capacity u_{max}, and sum of supply $\sum_{i\in\mathcal{V}:b_i>0} b_i$. Nodes are randomly chosen to be demand nodes, supply nodes, or trans-shipment nodes with probabilities 0.4, 0.4, and 0.2, respectively. It is, however, possible that some demand- or supply-nodes are assigned a balance of 0. Instances G01-G04 are created with the same number of nodes as instances N01-N12 and the same $\sum_{i\in\mathcal{V}:b_i>0} b_i$. In instances G05/G06 and G09/G10 we increase u_{max} of G03 and G04, respectively. In instances G07/G08 and G11/G12 we decrease c_{max} of G03 and G04, respectively. Again, we generate 30 problems for each choice of parameters h, w, n, m, c_{max}, and u_{max} as listed in Table 3.2. Integer costs and capacity of arcs are randomly selected in the interval $[0; c_{max}]$ and $[0; u_{max}]$, respectively.

All numerical tests are performed on a Linux (Ubuntu 7.04) computer with

Table 3.3. Results for problems N01 – N12.

	$\lvert\mathcal{Z}_N\rvert$			$\lvert\mathcal{Z}_{SN}\rvert/\lvert\mathcal{Z}_{NN}\rvert$	time in seconds		
Name	average	min	max	average	average	min	max
N01	168.13	15	392	0.28	0.40	0.01	1.45
N02	271.13	66	852	0.22	0.76	0.09	3.17
N03	375.43	126	702	0.18	1.40	0.27	3.78
N04	455.10	137	879	0.15	7.09	1.67	26.36
N05	660.63	252	1801	0.14	11.84	3.16	36.95
N06	948.30	266	2280	0.12	22.58	5.05	74.91
N07	867.80	410	1399	0.11	42.21	11.48	94.32
N08	1510.37	531	2834	0.09	90.88	27.11	245.20
N09	1553.47	808	2448	0.09	112.62	32.77	238.82
N10	1138.77	552	1901	0.10	125.42	46.44	372.95
N11	2036.20	989	4109	0.08	289.05	69.97	559.34
N12	2480.70	1287	3921	0.07	397.94	138.38	813.76

Table 3.4. Results for problems F01 – F12.

	$\lvert\mathcal{Z}_N\rvert$			$\lvert\mathcal{Z}_{SN}\rvert/\lvert\mathcal{Z}_{NN}\rvert$	time in seconds		
Name	average	min	max	average	average	min	max
F01	181.13	24	491	0.27	0.52	0.04	2.81
F02	260.53	15	685	0.24	0.99	0.02	4.58
F03	353.77	158	788	0.20	1.54	0.28	6.41
F04	213.87	65	380	0.20	2.44	0.58	5.58
F05	354.10	144	701	0.15	5.19	1.86	11.77
F06	478.87	176	714	0.13	9.20	2.53	33.65
F07	203.97	48	410	0.16	7.17	0.87	22.40
F08	343.23	165	860	0.14	13.48	5.31	41.27
F09	454.17	230	950	0.12	21.35	8.18	47.9
F10	146.43	72	277	0.18	8.80	2.75	17.27
F11	277.90	131	680	0.15	19.64	8.38	54.04
F12	414.50	234	693	0.12	34.03	12.57	66.47

2.80GHz Intel Pentium D processor and 1GB RAM. We use the gcc compiler (version 4.1) with compile option -O3. The methods are implemented in C. When measuring run-time, we disregard the time it takes to read the problem from a problem file. Run-time does include the generation of all non-dominated points together with the efficient flows and it is measured with a precision of 0.01 seconds.

In Table 3.3 - Table 3.5 we show the average, minimum, and maximum $|\mathcal{Z}_N|$, average $|\mathcal{Z}_{SN}|/|\mathcal{Z}_{NN}|$, and average, minimum, and maximum CPU time for the two different NETGEN instances (N and F), and the grid instances (G), respectively. We make the following observations (see Tables 3.3 to 3.5):

Table 3.5. Results for problems G01 – G12.

	$\lvert\mathcal{Z}_N\rvert$			$\lvert\mathcal{Z}_{SN}\rvert/\lvert\mathcal{Z}_{NN}\rvert$	time in seconds		
Name	average	min	max	average	average	min	max
G01	74.13	5	276	0.52	0.11	0.00	0.79
G02	211.23	37	817	0.27	1.99	0.09	10.54
G03	256.07	86	592	0.22	8.72	2.22	33.23
G04	354.20	58	1092	0.20	21.20	2.40	99.01
G05	319.67	64	1034	0.21	8.90	1.45	23.48
G06	420.60	106	955	0.19	12.17	2.66	37.72
G07	194.63	39	433	0.30	6.78	0.44	25.18
G08	235.33	25	477	0.27	8.00	0.61	40.42
G09	477.33	176	1094	0.17	34.38	6.00	293.53
G10	397.77	113	1069	0.19	21.54	2.04	65.64
G11	265.93	35	541	0.27	23.61	1.33	55.53
G12	326.80	109	645	0.20	21.27	5.62	70.89

- When fixing the number of nodes n in a network but increasing the number of arcs m the number of non-dominated points $|\mathcal{Z}_N|$ increases. This is illustrated by instances N01-N12 and F01-F12 when comparing three consecutive rows with the same number of nodes, i.e. rows N01-N03, N04-N06, and so on.

- For all presented instances, we observe that the more non-dominated points there are in a problem, the longer the run-time of the algorithm. Despite the instances being fairly small, they have a lot of non-dominated points.

- The sum of supply ($\sum_{i\in\mathcal{V}:b_i>0} b_i$) significantly increases the number of non-dominated points, which can be seen by comparing the results for problems F01-F12 with the corresponding results of problems N01-N12. It is, however, more realistic to increase $\sum_{i\in\mathcal{V}:b_i>0} b_i$ while increasing the network size.

- We generate grid network instances G01-G04 with similar numbers of nodes and arcs as instances F01-F12 and N01-N12 generated by NETGEN. Comparing the number of non-dominated points of G01-G04 to those of (corresponding similar sized networks) N01-N12 we observe that there are (on average) fewer non-dominated points in the grid networks. This is not the case when comparing the average number of non-dominated points of G03 and G04 to those of F07 and F10/F11, respectively.

- When decreasing c_{max} in grid instances G07/G08 and G11/G12, we observe that smaller c_{max} leads to fewer non-dominated points, but not necessarily to a faster run-time. When increasing u_{max} in G05/G06, the number of solutions increases and so does the run-time. But increasing u_{max} to 100 in G10 leads to less solutions than increasing u_{max} to 75 in G09, so we are unable to draw conclusions about the relation of the number of non-dominated points and u_{max}.

- $|\mathcal{Z}_{SN}|/|\mathcal{Z}_{NN}|$, the ratio of supported and non-supported non-dominated points, is decreasing when the total number of solutions is increasing for NETGEN instances, on average, an observation also made by Sedeño-Noda and González-Martín (2001). For grid instances there seems to be the same trend, but the total number of solutions does not increase as much. In most NETGEN and grid instances, less than 20% and 30% of all solutions are supported, respectively. Thus, the majority of non-dominated points are non-supported.

- In Figures 3.6 - 3.8, the non-dominated points of one instance of each of the classes F01, N01, and G01 are shown. This illustrates that most non-supported points are in fact very close to the boundary of $conv(\mathcal{Z} + \mathbb{R}^2_{\leqq})$. The given figures are just three examples, but we observe a similar behaviour in most of the problem instances. By obtaining only the supported non-dominated points of a problem, a fairly good approximation of the set of non-dominated points can be obtained. There are, however, exceptions such as the example in Figure 3.9 showing an instance of G02, where there are a lot of non-supported points far from the boundary of $conv(\mathcal{Z} + \mathbb{R}^2_{\leqq})$.

3.6 Adaptation of the Two Phase Method for the Bi-objective Transportation Problem

The transportation problem is a special case of the MCF problem. The set of nodes is split into a set of n_1 supply nodes $\mathcal{V}_S$ and n_2 demand nodes $\mathcal{V}_D$. All arcs are directed from supply nodes to demand nodes, so $t(a) \in \mathcal{V}_S$ and $h(a) \in \mathcal{V}_D$ for all $a \in \mathcal{A}$. A non-negative cost vector $c_a = (c_a^1, c_a^2) \in \mathbb{N} \times \mathbb{N}$ is associated

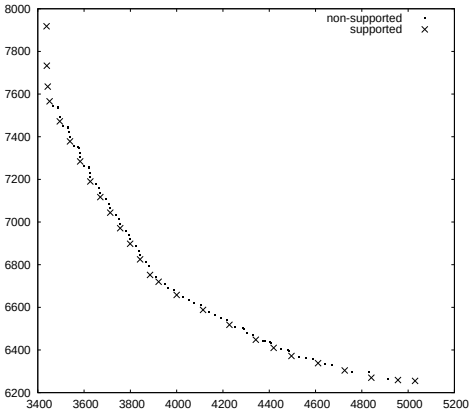


Figure 3.6. All non-dominated points of one instance of class F01.

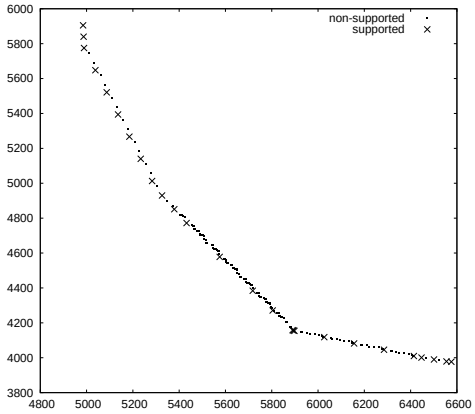


Figure 3.7. All non-dominated points of one instance of class N01.

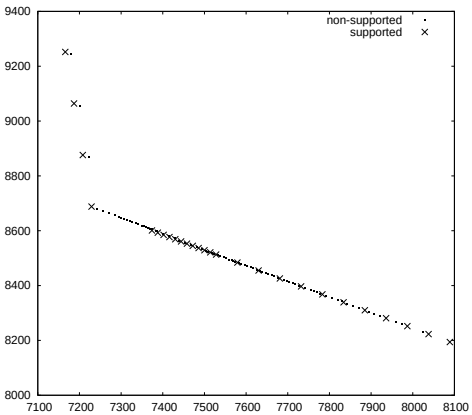


Figure 3.8. All non-dominated points of one instance of class G01.

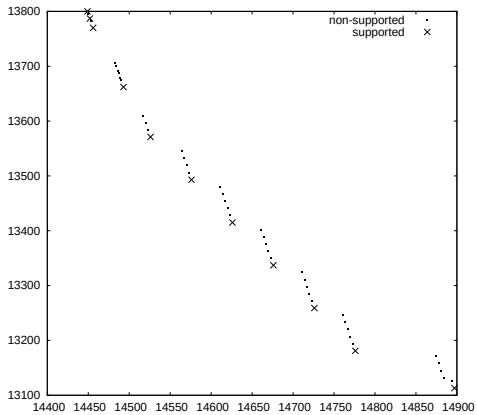


Figure 3.9. All non-dominated points of one instance of class G02.

with each arc a. Furthermore every demand node $i \in \mathcal{V}_D$ has demand $b_i^D \geqq 0$ and every supply node $i \in \mathcal{V}_S$ has supply $b_i^S \geqq 0$. A feasible solution only exists when $\sum_{i\in\mathcal{V}_D} b_i^D = \sum_{i\in\mathcal{V}_S} b_i^S$. The *bi-objective integer transportation (BIT) problem* can be stated as

$$
\begin{aligned}
\min \quad & z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} \\
\text{s.t.} \quad & \sum_{\{a\in\mathcal{A}:t(a)=i\}} x_a = b_i^S && \text{for all } i \in \mathcal{V}_S \\
& \sum_{\{a\in\mathcal{A}:h(a)=i\}} x_a = b_i^D && \text{for all } i \in \mathcal{V}_D \\
& x_a \geqq 0 && \text{for all } a \in \mathcal{A} \\
& x_a \text{ integer} && \text{for all } a \in \mathcal{A},
\end{aligned}
$$

with $z_1(x) = \sum_{a\in\mathcal{A}} c_a^1 x_a$ and $z_2(x) = \sum_{a\in\mathcal{A}} c_a^2 x_a$. BIT is a special case of BIMCF. As such, it can of course be solved with the same methods as BIMCF. As explained in Remark 3.4.4 the implementation of the k best flow algorithm in Phase 2 involves many shortest path calculations that take up a large portion of the algorithm's run-time. Therefore, this is where most potential for run-time improvement exists.

A more efficient solution method can be obtained, however, by adapting the k best flow algorithm to the transportation problem. Philip (2008) shows that for BIT, any proper minimum cost cycle $\mathcal{C}(G_x)$ in G_x for a BFS x contains an arc with zero flow. This observation can be used to efficiently identify this proper minimum cost cycle: as a proper minimum cost cycle must contain an arc with zero flow, only those arcs $a \in \mathcal{A}_x$ with $x_a = 0$ have to be considered when calculating shortest paths.

The numerical results in Philip (2008) indicate that this simple modification of the Phase 2 algorithm yields an average run-time improvement of 73% over the original BIMCF implementation for a set of 12 BIT test instances.

3.7 Concluding Remarks on Bi-objective Integer Minimum Cost Flow Problems

The presented Two Phase Algorithm solves the BIMCF problem, but the problems solved within reasonable run-time are fairly small. It is therefore worth investigating how to increase the performance of the presented algorithm to make it possible to solve bigger problems. Future research could address the extension of the Two Phase Algorithm for BIMCF to the MIMCF problem with more than two objectives. This can be done along the lines of Przybylski et al. (2009, 2007), where a Two Phase Method for multi-objective integer programming is presented together with an example of the application to the assignment problem with three objectives.

Chapter 4

Bi-objective Traffic Assignment

The traffic assignment problem models route choice of users of a road network assuming known and fixed travel demand between all origin and destination points within the network. The aim of traffic assignment (TA) is to determine how many users choose certain routes and/or how much traffic runs on each section of the roads in the network. TA can give valuable insights into the current usage of a road network, for instance highlight congestion-prone areas and which parts of the network are under-used. TA is often used as a planning tool as it helps predicting the impact of a change to the road network, such as building or widening a road.

A topic that is getting more important in today's world with ever increasing numbers of cars on the road and thus increasing traffic congestion is road tolling as a tool for controlling congestion. Many cities (Singapore, London, Melbourne, etc.) have introduced road tolling to cope with congested city centres or to finance construction of new roads or public transport infrastructure. Again, TA can be used to predict usage of roads and therefore toll revenues collected. Of course, charging a toll may alter the users' travel behaviour – some users might not want to pay the toll at all and therefore take a detour on the way to their destination whereas others might happily pay to arrive at their destination quickly (others may choose another mode of transport or not to travel at all). Our aim is to model travel behaviour taking the effect of tolls (or other monetary costs) into account. It has been observed that from a car driver's point of view, the monetary costs associated with running a vehicle (petrol, insurance, repairs) do not influence route choice very much. Toll costs,

however, are perceived very differently. Even low tolls provoke opposition from drivers that otherwise happily pay for the petrol to fill up their cars.

TA models the route choice of travellers within a road network. A frequent basic assumption is that travellers choose their route to minimise travel time or to minimise a linear combination of time and other objectives such as monetary cost. We highlight why we believe our bi-objective approach of considering the time and toll objectives independently is a more realistic approach to modelling traffic.

We first explain the key role single-objective TA plays in transportation planning in Section 4.1. We then formally introduce the single-objective TA problem in Section 4.2 and discuss under which assumptions it is equivalent to an optimisation problem and variational inequalities. We show how this relationship is exploited when solving TA. In Section 4.3 we discuss approaches to TA from the literature that deal with two or more objectives, which are usually combined into a single generalised cost objective. Then, in Section 4.4, we introduce the bi-objective and multi-objective TA problem where two (or more) objectives are not combined via a generalised cost function, but treated as individual objectives. We discuss why bi-objective TA and related optimisation and variational inequality problems are not equivalent, which is in contrast to single-objective TA. In Section 4.5 we propose new heuristic solution approaches dedicated to the bi-objective problem.

Preliminary results on bi-objective traffic assignment appeared in Wang et al. (2008) and Raith (2008a).

4.1 Traffic Assignment within the Transportation Planning Process

Traffic assignment is one of the key components of *transportation planning*. One major component of the transportation planning process is the modelling step, which is generally split into four different components, also known as *four-stage strategic transport planning model (FSM)*. TA is part of the final stage in the FSM. A comprehensive book on transport modelling by Ortúzar and Willumsen (2002) details the FSM and how the separate steps are solved.

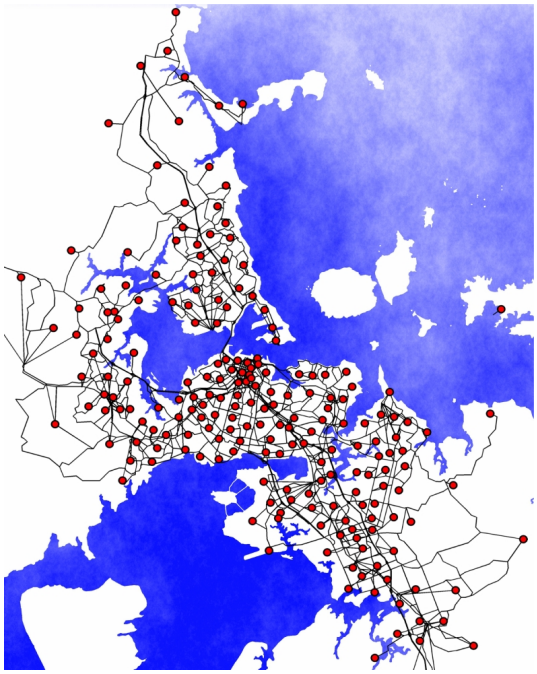


Figure 4.1. Simplified road network of the Auckland region. The data for generating this figure is kindly provided by the Auckland Regional Council.

McNally (2000) skeptically recapitulates the FSM and presents a typical example application in the US.

The FSM is based on an available network representation of the transportation infrastructure. All roads, or at least all major roads, are included in the road network as arcs, and nodes represent intersections. The public transport network also contains other transportation links such as railroad tracks or ferry connections. The network basis for TA and FSM is formally introduced in Section 4.2.1. There is a function associated with every transportation link in the network that represents the travel time on the link, which, in case of roads, is often dependent on the amount of traffic using the link or even the amount of traffic throughout the network. Instead of considering traffic originating from every node, the network may be split into zones. Each zone is associated with a so-called *centroid*, an artificial node at which all traffic into the zone ends or traffic out of the zone originates. Centroids are typically connected to the network via artificial arcs that may only be used as first or last arcs of a trip.

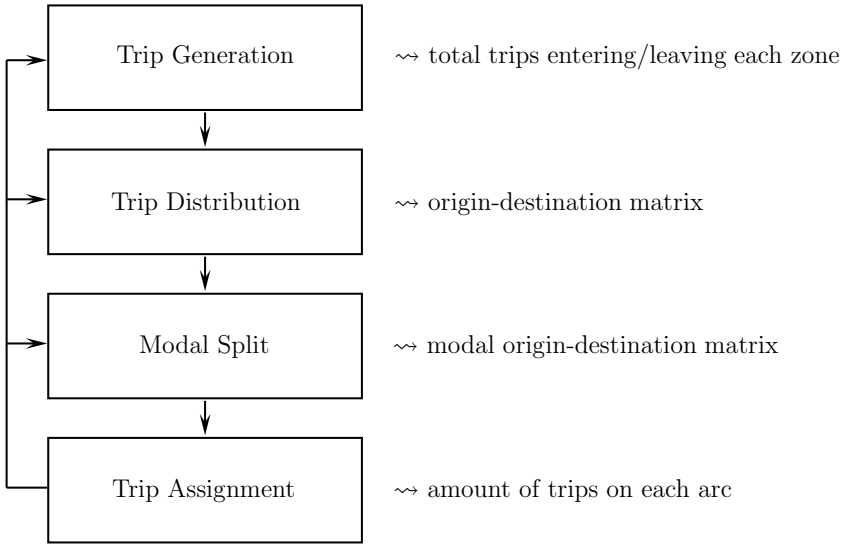


Figure 4.2. Illustration of the four-stage strategic transport planning model.

Figure 4.1 depicts the network of the Auckland Regional Transport (ART) Planning Model. It shows a simplified road network of the Auckland region, which contains only those arcs corresponding to major roads. Centroids are the red circles in the figure.

Figure 4.2 shows the four different stages of FSM, which are usually executed in sequence from top to bottom. Initially, it is determined how many trips start and end in every zone of the network, this is the *trip generation* step. Next, *trip distribution* splits the total trips originating at a zone into different target zones, and also splits the total trips arriving in a zone between different origin zones. At the end of trip distribution, an origin-destination matrix is obtained, which details the travel demand from each zone in the network to each other zone. Once it is determined how many travellers there are between the zones the next question is how they choose to travel. They might for example choose a *mode of transport* such as walking, cycling, using public transport, or driving a car. It is determined which portion of travellers chooses each mode in the *modal split* step. Finally, *trip assignment* determines the actual route choice of travellers for their mode of transport. This may be done separately for the different modes. The component of trip assignment that is studied in this thesis is *traffic assignment (TA)* where the routes of vehicular traffic are determined. It is normally assumed that travellers aim to minimise their own generalised

cost when making route choice decisions. Trip assignment yields important information on the traffic load on individual roads, or the occupancy level of public transport.

Even though the FSM is solved sequentially, the different stages have been linked together in the literature to enable feedback to earlier stages. For example, given the volume of traffic on each transportation link as output of trip assignment, trip generation, trip distribution, or modal split might produce different results. This reflects that, for example, people choose their workplace also with respect to its accessibility. Also, the mode choice obviously depends on the amount of traffic on roads as public transport may be more attractive when alternative routes for vehicular traffic are congested. This feedback is indicated in Figure 4.2

After running the FSM the resulting load on each link can be used to evaluate the performance of the network and identify its weaknesses. TA determines the load on different links which can indicate sections of the network where congestion is high and single out bottlenecks in the network. For public transport it is important to be able to derive expected revenue or determine how much capacity is needed. TA can also be used for predictions such as how the addition of new roads affects traffic throughout the system.

Prior to running the FSM and TA, the network needs to be calibrated. It is crucial that the functions of travel time on each link portray the actual situation correctly. It must be correctly modelled how the amount of traffic (also known as *traffic volume*) on a road influences travel time. The arising functions for car travel will be non-linear in general as travel speed decreases with traffic volume and traffic flow may even come to a stand-still if there are too many vehicles on a road. The same is true for modes that share infrastructure with other users, for example buses mostly share roads with cars, and are therefore susceptible to the same congestion effects. Subways, on the other hand, do not share their tracks and therefore their travel time may be assumed constant for links. The same is true for pedestrians, they are not affected by congestion as pedestrian walkways and sidewalks rarely reach maximum capacity. In the following we make the general assumption that arc cost functions are non-linear functions as the focus here is on car travel. In practice the travel time cost function is a function of passenger car units. Light

goods vehicles and heavy goods vehicles are accounted for by converting them to equivalent passenger car units.

We assume that the first three transportation modelling steps are completed, i.e. an origin-destination matrix is given and we also perform TA for only one transportation mode, namely car travel. In the following, we discuss how TA is modelled, and how it can be solved.

4.2 Single-objective Traffic Assignment

An extensive discussion of TA can be found in Ortúzar and Willumsen (2002), a summary of the main points is contained in Willumsen (2000). The book by Patriksson (1994) is dedicated to the many different algorithms available for solving TA.

We facilitate notation by restricting our considerations to car traffic. Car travel is most relevant for us as our aim is to model road tolls as opposed to other monetary transportation costs which mostly affect private motor vehicles. We first introduce the basis for traffic assignment, the traffic network.

4.2.1 Model

We continue to refer to network components as nodes and arcs, although in the traffic literature it is more common to refer to arcs as *links*. We denote by $\mathcal{V}$ the set of nodes, which can be understood as intersections, or points where one can change between different arcs. The set of arcs is $\mathcal{A} \subseteqq \mathcal{V} \times \mathcal{V}$, where an arc represents a directed road section. There are $|\mathcal{A}| = m$ arcs and $|\mathcal{V}| = n$ nodes in the network. The road network $N = (G, \overline{c})$ consists of a directed graph $G = (\mathcal{V}, \mathcal{A})$ and a *travel time function* $\overline{c}$ that assigns to every arc a the time $\overline{c}_a$ it takes to traverse the arc. We assume that the finite demand of travel between pairs of nodes $w = (n_1, n_2) \in \mathcal{W} \subseteqq \mathcal{V} \times \mathcal{V}$ is $d_w \geqq 0$, where $\mathcal{W}$ is the set of all *origin-destination (OD) pairs*.

For every OD pair w there is a set of paths $\mathcal{R}_w$ connecting the two nodes. The set of paths for all OD pairs is $\mathcal{R} = \bigcup_{w \in \mathcal{W}} \mathcal{R}_w$. The flow on each path $r \in \mathcal{R}$

is denoted by f_r. Arc flow $\overline{f}_a$ is related to path flow by

$$\overline{f}_a = \sum_{r \in \mathcal{R}} f_r \phi_{ar} \quad \text{for all } a \in \mathcal{A}, \tag{4.1}$$

where ϕ_{ar} has value 1 if path r contains arc a, and 0 otherwise. The vector of path flow is $f \in \mathbb{R}^\rho, \rho = |\mathcal{R}|$, and the vector of arc flow is $\overline{f} \in \mathbb{R}^m$. We denote by $\Phi \in \mathbb{R}^{m \times \rho}$ the *arc-path incidence matrix* with elements ϕ_{ar}. Then (4.1) can be re-written as $\overline{f} = \Phi f$. Travel demand is satisfied if

$$d_w = \sum_{r \in \mathcal{R}_w} f_r \quad \text{for all } w \in \mathcal{W}. \tag{4.2}$$

The set $\mathcal{K}$ of feasible path flow vectors is given by

$$\mathcal{K} = \{f \in \mathbb{R}^\rho : f \geqq 0 \text{ satisfies (4.2)}\}.$$

The set $\mathcal{K}_\mathcal{A}$ of feasible arc flow vectors is given by

$$\mathcal{K}_\mathcal{A} = \left\{\overline{f} \in \mathbb{R}^m : \exists f \in \mathcal{K} \text{ such that } \overline{f} = \Phi f\right\}.$$

The sets $\mathcal{K}$ and $\mathcal{K}_\mathcal{A}$ are both convex, closed and bounded subsets of $\mathbb{R}^\rho$ and $\mathbb{R}^m$, respectively. This can be easily seen:

- *convex:* For $f, g \in \mathcal{K}$ it follows that $\lambda f + (1-\lambda)g \in \mathcal{K}$ with $\lambda \in [0;1]$ as $\lambda f + (1-\lambda)g \geqq 0$ and demand is still satisfied:

$$\sum_{r \in \mathcal{R}_w} \lambda f_r + (1-\lambda) g_r = \lambda \sum_{r \in \mathcal{R}_w} f_r + (1-\lambda) \sum_{r \in \mathcal{R}_w} g_r = d_w.$$

- *closed:* We have to show that the limit of every converging sequence with limit $f = \lim_{k\to\infty} f^k$ and $f^k \in \mathcal{K}$ lies in $\mathcal{K}$. For all $w \in \mathcal{W}$ and for all f^k, we have

$$\sum_{r \in \mathcal{R}_w} f_r^k = d_w.$$

With $f_r = \lim_{k\to\infty} f_r^k$ we conclude

$$\sum_{r \in \mathcal{R}_w} f_r = \sum_{r \in \mathcal{R}_w} \lim_{k\to\infty} f_r^k = \lim_{k\to\infty} \sum_{r \in \mathcal{R}_w} f_r^k = d_w,$$

which shows that f satisfies (4.2). Also, assume that f does not satisfy $f \geqq 0$, i.e. there exists $r \in \mathcal{R}$ with $f_r < 0$. Choosing $\epsilon = |f_r| > 0$, there exists $K \in \mathbb{N}$ such that for all $k \geqq K$

$$|f_r^k - f_r| = |\underbrace{\underbrace{f_r^k}_{\geqq 0} + \underbrace{(-f_r)}_{>0}}_{\geqq |f_r|}| < |f_r|,$$

a contradiction. Therefore $f \in \mathcal{K}$.

- *bounded:* The set $\mathcal{K}$ is bounded as every component f_r of $f \in \mathcal{K}$ is bounded by $0 \leqq f_r \leqq d_w$ for $r \in \mathcal{R}_w$.

The same results hold for $\mathcal{K}_\mathcal{A}$, where proofs are obtained similar to those above by replacing $\overline{f} = \Phi f$.

Most importantly it is assumed that there is a single objective function which all users aim to minimise: the *travel time* associated with an arc a is denoted by $\overline{c}_a(\overline{f})$, which may be a non-linear function depending on the flow of the *whole* network. It is a reasonable assumption that travel time on an arc is positive and continuous. The travel time along a path r is denoted by $c_r(f)$.

Definition 4.2.1 We call the path travel time function $c_r(f), r \in \mathcal{R}$ *additive* if it can be obtained by summing up the appropriate arc time functions $\overline{c}_a(\overline{f})$:

$$c_r(f) = \sum_{a \in \mathcal{A}} \overline{c}_a(\overline{f}) \phi_{ar}.$$

The arc time vector is denoted by $\overline{c}(\overline{f}) \in \mathbb{R}^m$ and the path time vector is $c(f) \in \mathbb{R}^\rho$. Although we refer to $\overline{c}$ as travel time here, other objectives can be included into $\overline{c}$: the function $\overline{c}$ often is a combination of time, cost and other factors relevant for route choice. Again, Φ can be used to derive path cost from arc cost by $c(f) = \Phi^\top \overline{c}(\Phi f)$. The following are important properties of the functions $\overline{c}_a$.

Definition 4.2.2 If $\overline{c}_a(\overline{f}) = \overline{c}_a(\overline{f}_a)$ for all arcs $a \in \mathcal{A}$, i.e. arc travel time is only a function of $\overline{f}_a$, the flow on a, the cost function is called *separable*. If for any $a, b \in \mathcal{A}$, $\partial \overline{c}_a / \partial \overline{f}_b = \partial \overline{c}_b / \partial \overline{f}_a$ the cost function is called *symmetric*. A

separable cost function is always symmetric as for separable $\overline{c}_a(\overline{f}_a)$ and $\overline{c}_b(\overline{f}_b)$ we get $\partial\overline{c}_a/\partial\overline{f}_b = 0 = \partial\overline{c}_b/\partial\overline{f}_a$ whenever $a \neq b$.

It is commonly assumed that travellers all have the same objective to minimise travel time, that they have perfect knowledge of the travel time on each alternative path, and that they behave according to *Wardrop's first principle* (Wardrop 1952, p.345):

> The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route.

This means that no individual driver can decrease their travel time by unilaterally choosing a different path, therefore all travellers between the same OD pair experience the same travel time, even when they travel on different paths. If there were two paths $r, s \in \mathcal{R}_w$ with $c_r(f) < c_s(f)$ and $f_s > 0$, at least some of the travellers on s would switch from s to r. The stable solution described by Wardrop is obtained when all travellers for an OD pair experience the same minimal travel time (and the unused paths have travel time greater or equal to that of used paths), as no one has an incentive to change to another route – we call this stable solution a *solution of the TA problem*. This problem is an *equilibrium problem*, and we refer to it as SEQ from now on. We denote this formulation by SEQ, as it is a *scalar* equilibrium problem, not to be confused with the vector equilibria that will be discussed later. The problem is also known as *User Equilibrium (UE)*. Wardrop's description of SEQ is cast into mathematical terms as follows.

Definition 4.2.3 The feasible vector $f^* \in \mathcal{K}$ is called *equilibrium flow* if and only if

$$\text{(SEQ)} \quad \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad c_r(f^*) < c_s(f^*) \Rightarrow f_s^* = 0. \tag{4.3}$$

It is typically assumed that car drivers in a road network can be modelled by taking Wardrop's first principle as behavioural basis. Drivers behave in a selfish way in finding a route that is fastest among all their alternative choices.

The solution of the TA equilibrium problem is obtained as a feasible solution satisfying SEQ. Finding such a solution is not trivial, as a solution of SEQ can only be derived analytically for very simple problems. It can be shown that SEQ is equivalent to other problems, for which solution algorithms are known, see Sections 4.2.2 and 4.2.3.

When a solution satisfies SEQ, it follows that on all used paths for an OD pair w, drivers experience the same minimal travel time η_w, whereas the travel time on all unused paths is at least as high. When f^* satisfies SEQ, there exist minimal driving times η_w for each $w \in \mathcal{W}$ such that the following holds for all OD pairs $w \in \mathcal{W}$ and any path $r \in \mathcal{R}_w$:

$$\begin{aligned} c_r(f^*) &= \eta_w \quad \text{if } f_r^* > 0, \\ c_r(f^*) &\geqq \eta_w \quad \text{if } f_r^* = 0. \end{aligned} \tag{4.4}$$

For completeness, we mention the *capacitated* version of SEQ (some remarks on capacitated SEQ can be found in Patriksson 1994). It is assumed that arc flow must lie between some lower and upper limit $\overline{l}, \overline{u} \in \mathbb{R}^m$. The new set of feasible arc flow vectors becomes $\hat{\mathcal{K}}_{\mathcal{A}} = \{\overline{f} \in \mathbb{R}^m : \exists \overline{f} \in \mathcal{K}_{\mathcal{A}}$ such that $\overline{l} \leqq \overline{f} \leqq \overline{u}\}$, and the set of feasible path flow vectors becomes $\hat{\mathcal{K}} = \{f \in \mathbb{R}^\rho : \exists \overline{f} \in \hat{\mathcal{K}}_{\mathcal{A}}$ such that $\overline{f} = \Phi f\}$.

Definition 4.2.4 The feasible vector $f^* \in \hat{\mathcal{K}}$ is called *capacitated equilibrium flow* if and only if

$$\text{(SEQC)} \quad \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad c_r(f^*) < c_s(f^*) \Rightarrow f_s^* = l_s \text{ or } f_r^* = u_r.$$

4.2.2 Optimisation Problem Formulation

It was first published by Beckmann et al. (1956) that the SEQ problem (4.3) can, under certain assumptions, be equivalently solved via the optimisation problem

$$\begin{aligned} \min \quad & \sum_{a \in \mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a(v) dv \\ \text{s.t.} \quad & \overline{f} \in \mathcal{K}_{\mathcal{A}}, \end{aligned} \tag{4.5}$$

given additive path cost functions. This equivalence as well as existence of a solution of (4.5) is guaranteed by making the assumptions that for all OD pairs $w \in \mathcal{W}$ it holds that $|\mathcal{R}_w| \geqq 1$, $d_w \geqq 0$, and that the arc travel time function $\overline{c}_a$ is positive, continuous, and separable (Patriksson 1994). Instead of assuming separable functions, it suffices to assume that the functions $\overline{c}_a$ are symmetric, i.e. that the Jacobian matrix $\nabla \overline{c}$ is symmetric, see Nagurney (1993). Uniqueness of the (arc flow) solution follows from the additional assumptions that $\forall w \in \mathcal{W}$ $d_w > 0$ and that $\overline{c}_a(\overline{f}_a)$ are strictly increasing (for example Patriksson 1994). It is a reasonable assumption that travel time increases with increasing flow on an arc, as more travellers mean more congestion and thus a longer time to traverse the arc.

Remark 4.2.1 Only uniqueness of arc flows can be guaranteed, but not uniqueness of path flows (Patriksson 1994).

4.2.3 Variational Inequality Formulation

Other equivalent formulations of (4.3) are based on *variational inequality (VI)* formulations. Nagurney (1993) shows the applicability of variational inequalities within many different subject areas, and there is a chapter on TA and its equilibrium formulation SEQ. A VI that represents SEQ can be formulated in terms of path flow and in terms of arc flow.

Definition 4.2.5 The *path-based variational inequality problem* is the problem of finding flow $f^* \in \mathcal{K}$ that satisfies

$$\text{(VIp)} \quad c(f^*)^\top (h - f^*) \geqq 0 \quad \forall h \in \mathcal{K}.$$

There is also an arc-based VI formulation. The flow $\overline{f}^* \in \mathcal{K}_\mathcal{A}$ is a solution to the *arc-based variational inequality problem* if and only if

$$\text{(VIa)} \quad \overline{c}(\overline{f}^*)^\top (\overline{h} - \overline{f}^*) \geqq 0 \quad \forall \overline{h} \in \mathcal{K}_\mathcal{A}.$$

Smith (1979) shows that VIp and VIa are equivalent if the path costs are additive (then the total cost $c(f^*) = \overline{c}(\overline{f}^*)$ remains unchanged as it does not

matter whether it is calculated via path flows or via arc flows). The arc-based formulation VIa, however, may be advantageous as it only requires one variable per arc, rather than one variable per path – and there can be many more paths than arcs in a network. SEQ and VIp / VIa are equivalent as stated in the following theorem:

Theorem 4.2.1 *(Smith 1979) A vector $f \in \mathcal{K}$ satisfies SEQ if and only if it satisfies VIp. When cost functions are additive, a vector $\overline{f} \in \mathcal{K}_{\mathcal{A}}$ satisfies SEQ if and only if it satisfies VIa. Existence of a solution is guaranteed by the additional assumption that the respective cost function c or $\overline{c}$ is continuous.*

In Patriksson (1994) stronger assumptions for the equivalence of SEQ and VIp (and VIa) are made, namely the assumption that $\forall w \in \mathcal{W}$ it holds that $|\mathcal{R}_w| \geqq 1$, $d_w \geqq 0$, and that $\overline{c}_a(\overline{f})$ is a positive, continuous function. The stronger assumptions are necessary as the relation is proved via the corresponding optimisation formulation, which requires stronger assumptions.

Both VI formulations are more general than the optimisation formulation (4.5) as there are less underlying assumptions on arc cost functions: for VI the functions $\overline{c}(\overline{f}), c(f)$ may depend on the flow in the *entire* network. As such a VI formulation is very general (and equivalent to SEQ), it is often given as *definition* of the traffic assignment equilibrium problem (e.g. Oettli 2001) instead of the definition of SEQ (Definition (4.3)).

4.2.4 Solving Single-Objective Traffic Assignment

There is a multitude of algorithms dedicated to solving TA, by solving one of its re-formulations SEQ, VIp/VIa, or the optimisation problem (4.5). We give a quick introduction to some of the most common basic algorithms here. Patriksson (1994) introduces many algorithms to solve TA and gives the historic context of their development.

Algorithm 10 All-or-Nothing Assignment

1: **input**: Graph $(\mathcal{V}, \mathcal{A})$, constant arc cost vector $\overline{c}$, set of OD pairs $\mathcal{W}$, and demand d.
2: Set $f = 0$.
3: **for all** $w \in \mathcal{W}$ **do**
4: Identify shortest path $r \in \mathcal{R}_w$.
5: $f_r = d_w$ /* Assign all flow d_w to this path */
6: **end for**
7: $\overline{f} = \Phi f$
8: **output**: Arc flow vector $\overline{f}$ and path flow vector f.

All-or-Nothing Assignment

The most basic TA problem arises when all travel time functions $\overline{c}_a$ have constant values, which corresponds to a road network in which there is no congestion. In this case, all travellers for an OD pair w simply travel along the shortest path(s) connecting their origin to their destination. To solve this simple TA problem, one assigns the whole demand d_w for each $w \in \mathcal{W}$ to the shortest path(s) in $\mathcal{R}_w$. This process is known as *All-or-nothing (AON) Assignment*, summarised in Algorithm 10. AON assignment is a common building block for many TA algorithms, as explained below.

General Iterative Solution Approach

When travel times $\overline{c}_a$ are not constant, however, every additional traveller on a path (or arc) may affect the travel time on this path (or arc) and also on other paths. Generally solution algorithms for TA are of iterative nature. An initial AON assignment of arc flows $\overline{f}^0$ is created by assigning all demand for OD pair w to the shortest path(s) $r \in \mathcal{R}_w$ based on fixed travel time derived from zero flow on all arcs. Travel times are updated using the new arc flows and another AON assignment is performed giving arc flow $\overline{h}$. Next, the new solution $\overline{f}^{i+1}$ is obtained as a convex combination of the current solution, $\overline{h}$, and the previous one, $\overline{f}^i$. This process continues iteratively until a convergence criterion is satisfied. The general iterative approach is described in Algorithm 11.

The general iterative scheme in Algorithm 11 is based on arc flow. Although shortest paths are calculated for every AON assignment, it is not necessary to

Algorithm 11 General Iterative Approach for Solving SEQ

1: **input**: Graph $(\mathcal{V}, \mathcal{A})$, arc cost functions $\overline{c}$, set of OD pairs $\mathcal{W}$, and demand d.
2: Calculate fixed arc travel times $\overline{c}(0)$.
3: Perform AON assignment yielding arc flow vector $\overline{f}^0$.
4: $i = 0$
5: **while** Convergence criterion not satisfied **do**
6: Calculate fixed times $\overline{c}(f^i)$.
7: Perform AON assignment yielding arc flow vector $\overline{h}$.
8: Find convex combination $\overline{f}^{i+1} = (1-\lambda)\overline{f}^i + \lambda\overline{h}$ with $\lambda \in [0;1]$.
9: $i = i + 1$
10: **end while**
11: **output**: Arc flow vector $\overline{f}$.

store path flow, it is sufficient to keep track of the cumulative arc flow. This has the advantage that only m variables, one for each arc, are involved in the solution algorithm, rather than one for every path.

There are different strategies to obtain a convex combination in Step 8 of Algorithm 11. Two of them are mentioned here and they give rise to different well-known solution methods for TA.

Method of Successive Averages

A basic scheme for solving TA, called *method of successive averages (MSA)*, is obtained by choosing $\lambda = \frac{1}{i+1}$. MSA can be visualised as initially assigning all travellers for w to one path and then allowing some of them to switch to another path according to the new path costs. With each iteration less travellers have an incentive to switch paths and the solution slowly approaches equilibrium. Powell and Sheffi (1982) give proof of the convergence of the MSA approach.

Frank-Wolfe Algorithm

Throughout the literature the most popular approach to solving the optimisation formulation (4.5) of the TA problem seems to be the *Frank-Wolfe (FW)* algorithm. Patriksson (1994) gives a description of the algorithm and its history. It is assumed that the cost function $\overline{c}_a(\overline{f})$ is additive. In the FW al-

gorithm, the convex combination in Step 8 is determined by choosing λ so that $\sum_{a\in\mathcal{A}} \int_0^{(1-\lambda)\overline{f}_a^i+\lambda\overline{h}_a} \overline{c}_a(v)dv$, i.e. the objective of (4.5), attains its minimum. This means the convex combination of $\overline{f}^i$ and $\overline{h}$ is chosen to optimise the objective of the optimisation problem.

Path Equilibration Algorithm

Path equilibration is a different algorithmic approach tackling path flow rather than arc flow. The idea is that, in every iteration, the two paths that are the *least* (as discussed below) in equilibrium are selected and flow is shifted between them.

An initial AON assignment is performed based on arc travel times $\overline{c}(0)$, after which times are updated according to the new path flow values. Iteratively, the shortest path as well as the longest path with *positive* flow are determined for every OD pair and *equilibrated* by shifting parts (or all) of the flow from the longest path to the shortest one with the aim of making their travel times equal. Then times are updated and the process repeats until a convergence criterion is satisfied. Although this is a path-based approach to solving SEQ, it is not necessary to enumerate all paths: it is sufficient to keep track of all paths with *positive* flow from which the longest path is selected, whereas a shortest path algorithm can be used to identify the current shortest path in the network. We outline this equilibration algorithm in Algorithm 12. The algorithm is first described by Dafermos and Sparrow (1969). The idea of generating new paths (and path variables) as they are needed in a column generation scheme can be found in Leventhal et al. (1973).

The path equilibration approach allows to monitor convergence by simply comparing the highest and lowest cost on paths connecting each OD pair. Once this difference is sufficiently small, the iterative procedure can be stopped.

Algorithm 12 Iterative Equilibration Approach for Solving SEQ

input: Graph $(\mathcal{V}, \mathcal{A})$, arc cost functions $\bar{c}$, set of OD pairs $\mathcal{W}$, and demand d.
Calculate fixed times $\bar{c}(0)$.
Perform AON assignment to obtain path flow vector f.
for all $w \in \mathcal{W}$ **do**
  $\mathcal{R}_w^> = \{r \in \mathcal{R}_w : f_r > 0\}$ /* Initialise set of paths with positive flow */
end for
while Convergence criterion not satisfied **do**
  for all $w \in \mathcal{W}$ **do**
    Calculate constant path costs $c(f)$.
    Find longest path $r \in \mathcal{R}_w^>$. /* Longest path with positive flow. */
    Find shortest path $s \in \mathcal{R}_w$.
    Determine σ such that (with $h_u = f_u, u \neq r, s$)
      $c_r(h) = c_s(h)$ with $h_s = f_s + \sigma$, $h_r = f_r - \sigma$, and $h_r \geqq 0$, or
      $c_r(h) \geqq c_s(h)$, with $h_s = f_s + \sigma$, $h_r = f_r - \sigma = 0$.
    $f_r = f_r - \sigma$, $f_s = f_s + \sigma$, and $f_t, t \in \mathcal{R}_w \setminus \{r, s\}$ remain unchanged.
    $\mathcal{R}_w^> = \mathcal{R}_w^> \cup \{s\}$.
    if $f_r = 0$ **then**
      $\mathcal{R}_w^> = \mathcal{R}_w^> \setminus \{r\}$
    end if
    Update path flow vector f.
  end for
end while
output: Path flow vector f and arc flow vector $\overline{f} = \Phi f$.

4.3 TA with Explicit Distinction of Two or More Objectives

In the following, literature on TA where two or more objectives are explicitly distinguished is discussed. A straight-forward approach is forming the weighted sum of the two (or more) objectives. More sophisticated models distinguish user classes with different weighting factors in each class, or assume a distribution of weighting factors. There also exist approaches where variable weighting factors are obtained by a non-linear function. We introduce the different approaches and discuss their properties. In the literature, travel time and other components of the objectives involved in route choice are often distinguished. Most frequently, the two objectives travel time and monetary travel cost are studied, and we denote those particular objectives by $\bar{t}_a(\overline{f})$ and $\overline{m}_a(\overline{f})$, respectively.

4.3.1 Conventional TA with Two or More Objectives

Our aim is modelling the behaviour of network users when faced with route choice taking into account time as well as a possible (toll) cost on some arcs. Traditionally, this is modelled by assuming users aim to minimise a linear combination of time and cost (and often other route choice criteria as well), e.g. Sheffi (1985). The resulting objective function has the form $\overline{g}_a^c(\overline{f}) = \alpha\overline{t}_a(\overline{f}) + \overline{m}_a(\overline{f})$, where $\overline{t}_a$ represents the arc travel time component and $\overline{m}_a$ represents the monetary component associated with each arc. The function $g_a^c(\overline{f})$ is called the *generalised cost function* and $\alpha > 0$ *value of time (VOT)*. Of course it is also possible to convert the cost component into time, which leads to a *generalised time function* $\overline{g}_a^t(\overline{f}) = \overline{t}_a(\overline{f}) + \frac{\overline{m}_a(\overline{f})}{\alpha}$, again with VOT $\alpha > 0$. Similarly, weighting factors for both objectives, $\overline{g}_a(\overline{f}) = \omega_1\overline{t}_a(\overline{f}) + \omega_2\overline{m}_a(\overline{f})$ with $\omega_1, \omega_2 > 0$, are sometimes used.

It should be noted that, even though the objectives are referred to as time and toll here, other interpretations are possible. Nagurney et al. (2002b) for instance include an environmental objective representing emissions. We use time and cost as representative objectives as they are the main objectives distinguished throughout the literature. When an article assumes more than two objectives, they may also be denoted by $\overline{c}_a^1(\overline{f}), \ldots, \overline{c}_a^p(\overline{f})$.

Clearly, assuming every traveller has the same VOT value, and thus the same valuation of travel time, is not very realistic. Different studies such as Eliasson (2000); Leurent (2001) confirm this. More advanced models allow several classes of users with different VOT values or even assume a VOT distribution. The resulting models are similar to the single-objective TA discussed above and are solved with similar algorithms. When considering different user classes, travel demand d_w^i is given for each user class i and the whole demand is assigned to all paths that are shortest with respect to the generalised cost function with VOT class value α_i (Nagurney 2000, for example). When a distribution of α is given, all paths that are optimal for some range of α are determined. Each of those paths is assigned a share of the total demand d_w according to the distribution of α (Dial 1999a,b; Leurent 1998, for example).

4.3.2 Literature

Literature on the TA problem with two or more objectives is presented here. We first discuss literature in which a finite number of user classes is assumed, then literature that assumes a distribution of weighting factors (or VOT values). Finally, approaches that assume a non-linear valuation of time or cost are discussed. The section concludes with a table that summarises the presented literature.

In the following, if the functions are denoted by $\bar{t}_a(\overline{f})$, it is assumed that arc travel time depends on flow of the whole network, whereas $\bar{t}_a(\overline{f}_a)$ indicates that the arc travel time functions are separable. If not stated otherwise, path cost functions are assumed to be additive.

Finite Number of User Classes

We denote by $i \in \mathcal{I}$ a particular *user class*. The travel demand for class i is d_w^i. *Class arc flow* $\overline{f}_a^i$ and *class path flow* f_r^i are distinguished. In case of multiple user classes, arc flow $\overline{f}_a$ is obtained as $\sum_{i\in\mathcal{I}} \overline{f}_a^i$, and path cost is derived accordingly.

The approach to bi-objective traffic assignment problems taken in Nagurney (2000) is assuming a finite number of user classes each with their own weighting factor ω_1^i, ω_2^i for the travel time and cost objectives $\bar{t}_a(\overline{f}), \overline{m}_a(\overline{f})$. A known demand for each OD pair and each user class is assumed. The arising traffic assignment problem can be cast into a VI. Existence of a solution is shown, whereas uniqueness of the arising arc flow pattern can be guaranteed under certain assumptions. The proposed solution algorithm is the modified projection method (e.g. Nagurney 1993), which can be used to solve any VI with certain assumptions on the governing functions. The approach is illustrated using some small numerical examples.

Even though the application studied in Nagurney et al. (2001) is an equilibrium problem concerned with (tele-)shopping by considering a finite number of users making the decision whether to shop at different physical locations or online, the basic network formulation and behavioural principle is that of traffic assignment. It is assumed that users repeatedly purchase products and want

to minimise their generalised cost for doing so taking into account several objectives such as time, cost, security, convenience, denoted by $\overline{c}_a^1(\overline{f}), \ldots, \overline{c}_a^p(\overline{f})$. Users are divided into classes with known demand and weighting factors for each objective, the weightings may differ between user classes and also for every arc in the network.

In Nagurney and Dong (2002) the traffic equilibrium problem with two flow dependent objectives, $\overline{t}_a(\overline{f})$ and $\overline{m}_a(\overline{f})$, is considered. The model from Nagurney (2000) is extended to allow each user class to have different weighting factors on each arc as well as incorporating elastic travel demands. A formulation of equilibrium conditions as well as a VI formulation is given followed by conditions for existence of a solution and its uniqueness. The problem is solved with the modified projection method and illustrated by means of small examples. The paper also contains a table of related literature.

Nagurney et al. (2002a) model decision making between commuting and telecommuting, possible applications are working from home compared to commuting to work or shopping on-line compared to driving to a shop. To do this, they consider p different objectives $\overline{c}_a^1(\overline{f}), \ldots, \overline{c}_a^p(\overline{f})$ which may include cost and time, but also safety and opportunity cost. Demand may be elastic or fix. Their former models (Nagurney and Dong 2002; Nagurney et al. 2002a) are extended to allow weights for each class not only to differ for each arc, but also to depend on the objective values of this arc. Therefore, the weighting factors are $\alpha_a^i(\overline{c}_a^i)$. VI formulations are given, existence and conditions for uniqueness of solutions are discussed. The proposed solution approach is the modified projection method to solve the given VIs.

Nagurney et al. (2002b) study a traffic equilibrium problem with three objectives $\overline{t}_a(\overline{f}), \overline{m}_a(\overline{f}), \overline{e}_a(\overline{f})$, interpreted as time, cost, and environmental (emission) cost combined into a generalised cost function via class and arc-dependent weighting factors. Demand is fixed. Equilibrium conditions and a VI formulation are given, a proof of existence is given, as well as one of uniqueness in a special case. Furthermore, the problem with only two objectives, cost $\overline{m}_a(\overline{f})$ and the environmental component $\overline{e}_a$ given by a fixed emission factor, is studied. The resulting model is related to an emission pricing scheme and it is shown how the weighting factor associated with the environmental component is a measure of how environmentally conscious a user is. A solution algorithm

(modified projection method) is given.

Huang and Li (2007) study the TA with $\overline{g}_a^t(\overline{f}) = \overline{t}_a(\overline{f}) + \frac{\overline{m}_a(\overline{f})}{\alpha}$, where some vehicles are equipped with an "advanced traveller information system" (ATIS). The VOT factor α is given through a distribution from which a finite number of user classes, each with fixed value α_i, is derived. A logit-based model is proposed, where route choice depends stochastically on generalised cost with an additional error term. They distinguish cars with ATIS, that provide their user with a more accurate idea of the true value of $\overline{g}^t$ represented by an error term with smaller variance. The authors propose to solve the problem with MSA. A small numerical example is used to show the results of this model compared to three other ones (multiple user classes with $\overline{g}^t$; multiple user classes with $\overline{g}^c$; single user class with $\overline{g}^t$).

User Classes – Criticism

We present an example to illustrate that the assumption that network users can be divided into different user classes, each with identical weighting factors, may be a strong behavioural assumption that implicitly excludes viable choices.

Example 4.3.1 *Assuming there is only one OD pair we plot the two components t_r and m_r separately for all paths $r \in \mathcal{R}_w$, see Figure 4.3. In the figure we can distinguish dominated and non-dominated objective vectors. In particular, the non-dominated point z^2 is non-supported.*

Let path $r_i \in \mathcal{R}_w$ correspond to point z^i in the figure. A single VOT value of β_0 for instance would result in all demand for OD pair w being assigned to path r_3 because r_3 is the single minimal path for VOT β_0 as indicated in the figure. When several user classes with different VOT values are considered, flow can only be assigned to paths r_1, r_3, r_4, r_5, r_6, but no flow will be assigned to path r_2 as it is never an optimal solution for any generalised cost function. Clearly, the path associated with point z^2 should attract some flow as it is not dominated *by any other path, i.e. there is no other point in the figure that is better in both components than z^2. There is no reason why network users should not choose this path. The assumption that route choice behaviour can be based on generalised cost, generalised time, or weighted sum is a very strong behavioural assumption that prohibits the choice of paths that are clearly*

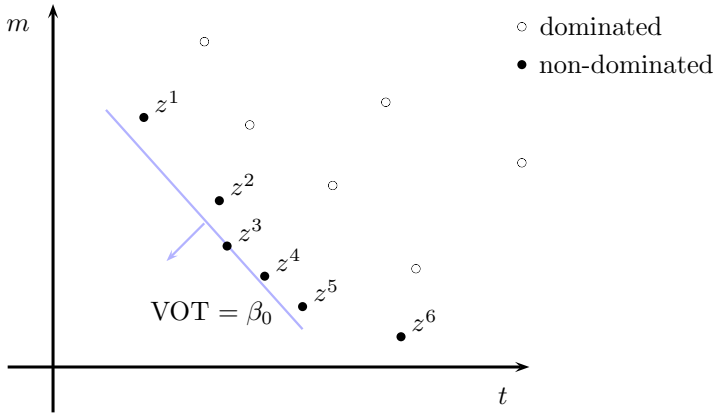


Figure 4.3. All points $(t_r(f), m_r(f)), r \in \mathcal{R}_w$ for some $f \in \mathcal{K}$ and one OD pair w.

reasonable alternatives.

Distribution of VOT or Weighting Factors

Another common approach is to assume a distribution of the VOT value α in $\overline{g}^t$ or $\overline{g}^c$, or the weighting factors in $\overline{g}$. Now, paths optimal for different ranges of values of α or of the weighting factors need to be identified. An appropriate portion of the total demand is assigned to the identified paths.

Quandt (1967) aims at modelling mode choice. He assumes that the (fixed) time and cost is known for every transportation mode, and that the mode choice is based on a probabilistic utility function. The probability that each mode is chosen is then determined. Quandt explicitly studies the problem with two modes and two objectives yielding at most two non-dominated points. Furthermore problems with three and four modes and two objectives are studied, as well as two modes with three objectives.

Schneider (1968) is one of the first to explicitly distinguish (fixed) time $\overline{t}$ and cost $\overline{m}$ as two main components of route choice. The two objectives $\overline{t}$ and $\overline{m}$ are combined into a weighted sum $\overline{g} = \omega_1\overline{t} + \omega_2\overline{m}$. Assuming a distribution of the two weighting factors, he characterises all relevant paths, namely those with optimal generalised cost function for a range of ω_1, ω_2. He further suggests to derive the portion of all trips using those paths from a distribution of the

weighting factors.

Dial (1979) also considers the two objectives independently and proposes an algorithm. Fixed travel time $\bar{t}$ and cost $\overline{m}$ are assumed. Firstly, a dichotomic algorithm (see also Section 1.3.2) to obtain extreme supported solutions of this BSP problem is introduced. Secondly, Dial proposes how to assign a share of all trip makers for each OD pair to efficient paths connecting this OD pair, now assuming a weighted sum $\overline{g} = \omega_1\bar{t} + \omega_2\overline{m}$. For a given value of ω_1, ω_2, $\overline{g}$ becomes optimal for at least one of the extreme efficient paths. In fact, there is a range of values of ω_1, ω_2 for which an efficient path is the optimal choice. Now a probability density function of the ratio ω_1/ω_2 is used to assign a portion of demand to each path.

Dafermos (1983) is the first to consider the problem with flow dependent objectives $\bar{t}_a(f_a)$ and $\overline{m}_a(f_a)$, thus taking congestion effects into account. The generalised cost function is assumed to be of the form $\overline{g}_a^c = \alpha\bar{t}_a(f_a)+(1-\alpha)\overline{m}_a(f_a)$ with $\alpha \in [0;1]$. The problem can be interpreted as traffic assignment with infinitely many user classes. From a corresponding equilibrium formulation, an equivalent infinite-dimensional VI formulation is derived along with a proof of the existence of a solution. Uniqueness is established for certain functions $\bar{t}$ and $\overline{m}$. The approach is demonstrated for a small example.

Leurent (1993) investigates the traffic assignment problem with travel time $\bar{t}_a(f_a)$, fixed travel cost $\overline{m}_a$, and elastic demand. A generalised time function $\overline{g}_a^t = \bar{t}_a(\overline{f}_a)+\frac{\overline{m}_a}{\alpha}$ is used as behavioural basis together with a VOT distribution according to which demand is split between different efficient paths for each OD pair. Then, a definition of the equilibrium problem with varying travel time is given, shown to be equivalent to a convex optimisation problem. Existence of a solution is shown, whereas some assumptions on travel time and demand functions guarantee uniqueness. Finally, an adaptation of the MSA algorithm for the solution of this traffic assignment problem is presented together with a proof of convergence. The paper concludes with a small example.

Marcotte (1995) discusses the usefulness of VIs in four different equilibrium problems including single-objective TA and TA with two objectives. For the latter, fixed demand is assumed as well as variable time $\bar{t}_a(\overline{f})$, fixed cost $\overline{m}_a$, and a generalised time formulation $\overline{g}^t(\overline{f})$ with VOT distribution. Previous formulations (Dafermos 1983; Leurent 1993) are repeated. Marcotte shows how

the optimisation formulation can be solved with the Frank-Wolfe algorithm, where the efficient shortest paths are obtained with a parametric shortest path algorithm (see also Section 2.3.4). Another possible reduction to a convex optimisation problem is given, which does not require a separable function $\bar{t}$, it suffices that the Jacobian of $\bar{t}$ is symmetric. It is also shown that, if two paths between the same OD pair never have the same cost $\overline{m}_a$, the VI can still be solved with a slightly modified Frank-Wolfe algorithm.

Leurent (1995) presents another possible solution algorithm for the problem from Leurent (1993). The procedure is called "Equalisation-by-Transfer" in one of his later articles. Similar to the path equilibration algorithm (Section 4.2.4) the longest paths with positive flow and shortest paths are identified for each OD pair, then flow is re-allocated from the longest to the shortest path with the aim of making their generalised cost equal. An equalisation algorithm for models with elastic demand appeared in Schittenhelm (1990). Leurent adapts this algorithm to account for the VOT distribution. Sensitivity analysis is discussed and a small example is presented.

Leurent (1996) generalises his previous work. Again, time $\bar{t}_a(\overline{f}_a)$ is flow dependent, while cost $\overline{m}_a$ is not. Demand is elastic and upper bound constraints on arc flows are included in the model. For each OD pair paths with equal cost are collected in the same class. An equilibrium principle is derived: only classes with some minimal "impedance"-value can receive a share of flow (which is of course assigned to the minimum time path within each class). An equivalent VI is introduced. In the case of symmetric $\bar{t}_a$, previously introduced algorithms (MSA, MSA with path enumeration, and path equilibration) are compared according to different convergence measures. Path equilibration converges the fastest. A new algorithm is proposed for TA with capacity constraints. Furthermore, it is investigated how errors in input data are propagated through the model.

Extending his work from 1979, Dial (1996) presents an algorithm for bi-objective traffic assignment where both objectives, $\bar{t}_a(\overline{f}_a)$ and $\overline{m}_a(\overline{f}_a)$, may vary with arc flow. Again, users are assumed to choose their route according to the best generalised cost function value $\overline{g}^c_a(f) = \alpha\bar{t}_a(\overline{f}_a) + \overline{m}_a(\overline{f}_a)$, where VOT values α follow some known distribution. The solution algorithm is similar to the Franke-Wolfe approach. Iteratively, the bi-objective traffic

assignment problem for fixed objective values is solved in a sub problem as described in Dial (1979). A convex combination of the previous solution and the one obtained by the sub problem forms the new solution. There is no proof of convergence of this algorithm, but a proof for a simpler version of the algorithm with flow independent monetary component $\overline{m}$ is given. The author illustrates his algorithm with a small 2-arc example and also shows some computational results for a network with 9 nodes.

Dial (1997) improves his previous algorithms. The sub problem is now solved by finding efficient shortest path trees for each origin node and loading trips according to the distribution of α in a more efficient manner. Where the master problem from Dial (1996) constructs a combination of the current and previous solution, the new solution is now obtained as convex combination of all previously obtained solutions. The author finds his improved algorithms to be more efficient. To show different possible applications of bi-objective traffic assignment, the author also discusses different pairs of objectives that model for instance time versus toll cost, expected time versus awareness of delay, travel time versus risk of being delayed, and time spent on preferred roads versus time spent on non-preferred roads.

Marcotte and Zhu (1997) consider equilibrium problems with multiple (but finitely many) user classes as well as a VOT distribution yielding an infinite dimensional VI formulation. The latter is the main concern of the paper. Existence of a solution and uniqueness of arc and class arc flow is proved under certain conditions. A variation of the Frank-Wolfe algorithm is proposed: the so-called linearisation algorithm to minimise a gap function derived from the equilibrium problem is given together with different step size rules. Finally, an application to the traffic assignment problem is given together with two small test instances.

Leurent (1998) studies the multi-objective traffic assignment problem with objectives $\overline{c}^1(\overline{f}), \ldots, \overline{c}^p(\overline{f})$ and fixed or variable demand. It is assumed that the first objective is cost, while the $p-1$ remaining ones represent other non-monetary objectives. The objectives are combined in a generalised cost function using weights $\omega_1, \ldots, \omega_{p-1}$ to transform the $p-1$ non-cost objectives into their cost equivalent. The weights are given by a distribution. Uniqueness and existence of solutions are discussed and a VI formulation is given. A

bi-objective example with a two-arc network is used to illustrate the theory. Existing literature and solution algorithms are also discussed.

VOT Distribution – Criticism

Again, we consider Example 4.3.1. In the following we will discuss the generalised cost function $\overline{g}^c$ only, as $\overline{g}^c = \alpha\overline{g}^t$ implies that $\overline{g}^c$ and $\overline{g}^t$ have the same solutions. Furthermore, the weighted sum function $\overline{g} = \omega_1\overline{t} + \omega_2\overline{m}$ is equivalent to the two previous functions which can be seen by dividing by ω_1 and choosing $\alpha = \frac{\omega_2}{\omega_1}$. If a VOT distribution is given, every non-dominated point in Figure 4.3 except for z^2 is optimal for at least one value of α. There exist values $\beta_1 > \beta_2 > \beta_3$ such that in Figure 4.3 point z^1 is optimal for $\alpha \geqq \beta_1$, z^3 is optimal for $\beta_1 \leqq \alpha \leqq \beta_2$, z^4 is optimal for β_2 only, z^5 is optimal for $\beta_2 \leqq \alpha \leqq \beta_3$, and z^6 is optimal for $\alpha \leqq \beta_3$.

Again, point z^2 is not optimal for any value of α, and path r_2 will therefore not be assigned any flow. Furthermore, most solution algorithms will also not assign any flow to path r_4, as its objective vector z^4 is only optimal for exactly one value of α. Again there is no reason why no network user should choose the paths corresponding to points r_2 and r_4 as they are not dominated.

Non-linear VOT

The following papers assume a non-linear VOT function. It is assumed that the valuation of time is not actually linear. In practice this means that a small delay, for example, may not be conceived as grave whereas a long delay may be very severe – even for someone with a normally low VOT. Non-linear VOT functions are not additive in general, and the valuation function can only be applied to the final path travel time. This entails the necessity for models that differ from those discussed earlier in this section and, accordingly, different solution approaches.

Gabriel and Bernstein (1997) argue that the assumption of a linear VOT is not realistic when computing the monetary equivalent of time. The authors point out that while a travel time of a few minutes more or less does not make a big difference, a travel time of, for example, 30 minutes can be perceived as very troublesome (e.g. when commuting to work). Hence, Gabriel and

Bernstein (1997) assume that the value of time is a non-linear function of path travel time $t_r(f)$. Even though path travel time is assumed to be additive, the corresponding valuation of time $v(t_r(f))$ is not. It is also assumed that there is a path-dependent fixed monetary component to take into account costs such as tolls, which is also not additive (as often the toll from A to C is not equal to the toll from A to B plus that from B to C). Thus, the studied problems have a generalised path cost function $g_r^c(f) = v(t_r(f)) + \lambda t_r(f) + m_r$ with nonlinear VOT function v, costs $\lambda t_r(f)$ proportional to travel time, and path-dependent monetary costs m_r such as tolls. This TA problem must have a formulation based on path flow as two main components of the cost function, $v(t_r(f))$ and m_r, can only be evaluated on a path-level. It is shown under which conditions existence and uniqueness of the arising elastic TA problem follow. A solution algorithm using simplicial decomposition is described in which only a subset of all paths is generated (using column generation). A solution is obtained as convex combination of previously generated path flow solutions (master problem) until it can be guaranteed that the equilibrium solution is obtained. This is checked within the column generation sub problem: do better solutions exist in the sub problem, i.e. can paths r with better $g_r^c(f)$ be found? It is not necessary to keep track of all path variables within the solution algorithm, only those with positive flow need to be stored. Only one path is generated at a time, before the master problem is solved again.

Bernstein and Gabriel (1997) extend their previous simplicial decomposition approach to allow a column generation scheme that may introduce more than a single path into the problem. By doing so, the master problem is solved less frequently. They also allow the nonlinear VOT function to vary for each path. Numerical tests are presented for a sample network.

Larsson et al. (2002) study a problem similar to the one studied by the previous authors. Instead of applying a nonlinear VOT function to express time in terms of cost, however, they transform cost into time, again using a nonlinear valuation function $\tilde{v}(m_r)$. They study the problem with generalised time function $g_r^t(f) = t_r(f) + \tilde{v}_w(m_r)$, where the valuation function $\tilde{v}_w$ may vary between OD pairs. Time t_r is assumed additive as well as separable, and path costs m_r are fixed. An equivalent optimisation model can be formulated by combining the objective (4.5) with the fixed cost component $\sum_{w \in \mathcal{W}} \sum_{r \in \mathcal{R}_w} f_r \tilde{v}_w(m_r)$. Thus, applying a generalised time function here, the problem is greatly simpli-

fied compared to previous approaches (Gabriel and Bernstein 1997; Bernstein and Gabriel 1997) as the fixed toll cost component in the objective function does not influence the solution of the minimisation problem. The proposed algorithms are both simplicial decomposition algorithms. In the master problem an optimal solution based on a restricted set of shortest paths, $\overline{\mathcal{R}} \subset \mathcal{R}$, is obtained. In one approach the master selects an optimal flow solution for the current set of routes $\overline{\mathcal{R}}$, whereas in the other one the master selects a new arc-flow solution as convex combination of previous arc-flow solutions. Due to the non-additive objective $\tilde{v}(m_r)$, the column generation sub problem is solved by a bi-objective label setting algorithm. Among all generated efficient paths only a single path is selected and added to the set $\overline{\mathcal{R}}$, namely the one with minimal cost $g_r^t(f) = t_r(f) + \tilde{v}_w(m_r)$.

It should be noted that the problems with generalised cost function $g^c = v(t_r(f)) + m_r$ (similar to Gabriel and Bernstein 1997) and $g^t(f) = t_r(f) + \tilde{v}_w(m_r)$ (Larsson et al. 2002) with $\tilde{v} = v^{-1}$ are not equivalent. Other than in the case of VOT factors α, it is not possible to simply calculate the "inverse" valuation as $\tilde{v} = v^{-1}$.

Literature Table

We summarise the literature discussed in this section in Table 4.1. The main features of every article are condensed here. The 'objectives' studied in the TA problem follow the same notation as in this whole section. Column 'how' stands for the type of generalised cost function, such as VOT (VOT value), W (weighting factors) combined with distr. (distribution), nonl. (non-linear), and uc (user classes). Column 'dem.' states whether demand is elastic (el.), fix, or both. Column 'cap.' indicates whether a capacitated TA is studied (yes) or not (no). Column 'aim' briefly summarises the main purpose of the paper and 'algorithm' describes the solution algorithm used, if applicable. Finally column 'numerical test' describes problem instances used when numerical tests were made ($a|b|c$ stands for the number of centroids a, the number of nodes b, and the number of arcs c), the number of user classes is also given, if applicable.

Table 4.1: Literature on Traffic Assignment taking into account more than one objective.

Reference	objectives	how	dem.	cap.	aim	algorithm	numerical test
Quandt (1967)	$\bar{t}$, $\overline{m}$, $\bar{c}$	U	fix	no	Probability that every solution (out of 2 or 3 non-dom. points) is chosen	-	-
Schneider (1968)	$\bar{t}$, $\overline{m}$	W distr.	fix	no	Characterisation of "minimal" paths as those with optimal generalised cost for a range of weighting factors		
Dial (1979)	$\bar{t}$, $\overline{m}$	VOT distr.	fix	no	Un-congested traffic assignment, with fixed time / cost.	Dichotomic BSP algorithm; assignment of demand to each extreme efficient path according to VOT distribution	38|268|688
Dafermos (1983)	$\bar{t}_a(\overline{f}_a)$, $\overline{m}_a(\overline{f}_a)$	W distr.	fix	no	Equilibrium formulation; equivalent infinite-dim VI; existence and uniqueness	-	2|2|N
Leurent (1993)	$\bar{t}_a(\overline{f}_a)$, $\overline{m}_a$	VOT distr.	el.	no	Introduce equilibrium problem; equivalent optimisation problem; existence and uniqueness	MSA	2|2|2
Marcotte (1995)	$\bar{t}_a(\overline{f})$, $\bar{t}_a(\overline{f}_a)$, $\overline{m}_a$	VOT distr.	fix	no	VI and optimisation problem from Dafermos (1983) / Leurent (1993). Proposes solution algorithms: optimisation problem solved via F-W, VI solved via modified F-W	F-W	-
Leurent (1995)	$\bar{t}_a(\overline{f}_a)$, $\overline{m}_a$	VOT distr.	el.	no	Another algorithm to solve problem from Leurent (1993); run-time comparison of this one and MSA	pathEq	141|?|2000

Continued on Next Page...

Table 4.1 – Continued

Reference	objectives	how	dem.	cap.	aim	algorithm	numerical test
Leurent (1996)	$\bar{t}_a(\overline{f})$, $\overline{m}_a$	VOT distr.	both	both	Theory of bi-objective traffic assignment with generalised cost function and side constraints; for problem without capacity constraints and symmetric $t_a(f)$ comparison of: MSA, MSA path based, F-W (fixed demand), Equalisation-by-Transfer	For problem with capacity constraints: Augmented Lagrangian Method	141|?|2000
Dial (1996)	$\bar{t}_a(\overline{f}_a)$, $\overline{m}_a(\overline{f}_a)$, $\overline{m}_a$	VOT distr.	fix	no	Problem formulation; VI; algorithms	Generalisation of F-W (iterative solution: combination of current and last) with sub problem from Dial (1979)	2|2|2 9|9|12
Dial (1997)	$\bar{t}_a(\overline{f}_a)$, $\overline{m}_a(\overline{f}_a)$	VOT distr.	fix	no	Improving algorithms from Dial (1996)	Iterative solution method: combination of all previous solutions; improved sub problem algorithm	9|9|12 36|121|440 ?|441|1680 to ?|5929|23408
Marcotte and Zhu (1997)	$\bar{t}_a(\overline{f})$, $\overline{m}_a(\overline{f})$, $\bar{t}_a(\overline{f}_a)$, $\overline{m}_a$	VOT distr. and VOT	fix	no	problem formulation via VI; existence, uniqueness; propose algorithm	Several step size variations for F-W type algorithm (minimising gap function)	4|4|5, 24|24|76
Gabriel and Bernstein (1997)	$t_r(f)$, $m_r(f)$, m_r	VOT nonl.	el.	no	Motivation for non-linear VOT; formulation as nonlinear complementarity problem; existence, uniqueness; path-based algorithm	solution as combination of all previously generated paths for each OD pair in master; paths are obtained via column generation	-
Bernstein and Gabriel (1997)	$t_r(f)$, $m_r(f)$, m_r	VOT nonl.	el.	no	path-based algorithm; interpretation of results	solution as combination of all previously generated paths for each OD pair in master; paths are obtained via column generation; adding several paths per iteration	9|9|28

Continued on Next Page...

Table 4.1 – Continued

Reference	objectives	how	dem.	cap.	aim	algorithm	numerical test
Leurent (1998)	$\overline{m}_a(\overline{f})$, $\overline{c}_a^1(\overline{f})$, ..., $\overline{c}_a^{p-1}(\overline{f})$	W distr.	both	no	Overview; multi-objective formulation; VI	Discussion of existing solution algorithms	2|2|2
Nagurney (2000)	$\overline{t}_a(\overline{f})$, $\overline{m}_a(\overline{f})$	W uc	fix	no	Model biobjective network equilibrium with finite user classes; equivalent VI formulation; uniqueness and existence of solutions	Mod. Proj.	4|4|5 2uc, 10|10|13 2uc
Nagurney et al. (2001)	$\overline{c}_a^1(\overline{f})$, ..., $\overline{c}_a^p(\overline{f})$	W uc	fix	no	Model multiobjective network equilibrium with finite user classes and different weighting for each arc; equivalent VI formulation; uniqueness and existence of solutions	Mod. Proj.	12|12|20 2uc
Nagurney and Dong (2002)	$\overline{t}_a(\overline{f})$, $\overline{m}_a(\overline{f})$	W uc	both	no	Model biobjective network equilibrium with finite user classes, different weighting for each arc, and elastic demand; equivalent VI formulation; uniqueness and existence of solutions	Mod. Proj.	2|2|2 2uc, 10|10|13 2uc
Nagurney et al. (2002a)	$\overline{c}_a^1(\overline{f})$, ..., $\overline{c}_a^p(\overline{f})$	W uc	both	no	Model multiobjective network equilibrium with finite user classes, different weighting for each arc *depending* on arc objectiv values, and elastic demand; equivalent VI formulation; uniqueness and existence of solutions	Mod. Proj.	-
Nagurney et al. (2002b)	$\overline{c}_a(\overline{f})$, $\overline{t}_a(\overline{f})$, $\overline{e}_a(\overline{f})$	W uc	fix	no	Different weighting factors for each arc and each class; equilibrium conditions; equivalent VI formulation; existence and uniqueness; interpretation of emission component e_a	Mod. Proj.	10|10|15 2uc

Continued on Next Page...

Table 4.1 – Continued

Reference	objectives	how	dem.	cap.	aim	algorithm	numerical test
Larsson et al. (2002)	$\overline{t}_a(\overline{f}_a)$, m_r	VOT nonl.	fix	no	Formulation of problem with non-linear VOT based on generalised time; equivalent optimisation formulation; path-based algorithm	Combination of all previously generated paths for each OD pair in master or combination of aggregate arc flows. Paths are obtained via a bi-objective shortest path column generation algorithm	9|9|28, 24|24|76
Huang and Li (2007)	$\overline{t}_a(\overline{f})$, $\overline{c}_a(\overline{f})$	VOT uc	fix	no	stochastic model with two different error terms, i.e lower variance for users equipped with ATIS	MSA	4|13|19

objectives: $\overline{t}_a$ time objective, $\overline{m}_a$ cost objective, $\overline{c}^1, \ldots, \overline{c}^p$ unspecified objectives; if $\overline{t}_a(\overline{f}_a)$: objective is arc-flow-dependent; if $\overline{t}_a(\overline{f})$: objective flow-dependent – *how*: assuming VOT (VOT), weighting factors for the objectives (W); utility function (U), finite number of user classes (uc), VOT distribution (distr.) – *dem.*: demand is either elastic (el), fixed (fix), or both (both) – *cap.* the network is capacitated if there are upper bound capacities on arc-flows (yes, no, both) – *algorithm*: Method of Successive Averages (MSA), Frank-Wolfe (F-W), Modified Projection Method (Mod. Proj.), equilibration algorithm (pathEq) – *numerical test* $a|b|c$ indicates network with a centroids, b nodes, and c arcs.

4.4 Bi-objective Traffic Assignment

Before introducing the concept of bi-objective traffic assignment, we briefly summarise our criticism of the conventional approach.

As mentioned earlier, the conventional approaches may not consider paths "attractive" that have non-supported or non-extreme objective vectors. All practical papers that present actual solution algorithms are of the generalised cost type, some considering several user classes, VOT distributions or non-linear VOT functions. More flexible approaches have been presented that allow a different weighting for each user class on each arc, or even varying weighting factors according to the actual arc costs. It may be difficult to correctly estimate the different weighting factors for all user classes on the different arcs, so practical tests will have to show the viability of these approaches.

In this thesis, we propose another approach that enables modelling of TA with two (or more) objectives, namely bi-objective (or multi-objective) TA which allows to consider the objectives separately. With the application to a TA problem with travel time and toll cost objective in mind, it appears that those two objectives are perceived very differently and should not simply be combined by using weighting factors. People often see travel time spent on the road as well as monetary costs such as petrol and car maintenance costs as a necessity. Tolls on the other hand are perceived as a nuisance as they are imposed by an authority. Especially in TA with toll costs it seems that assuming there exists a generalised cost function is a very strong assumption. Therefore, we propose to replace the generalised cost function assumption with a bi-objective approach.

We are able to show that the approach by Larsson et al. (2002), with non-linear valuation function in the generalised cost function, can be used to solve BEQ in Section 4.5.1.

Apart from TA with toll costs, we also see other possible areas of application of bi-objective TA. In the literature, it has been observed that conventional TA often does not represent traveller behaviour very well, for example during off-peak time. During off-peak time there is little congestion in a network and therefore TA assigns the network users for each OD pair to very similar paths. Observations of the actual traffic during off-peak times show that the

modelled path choice does not agree with that observed in practice. McNally (2000, p.49f) states that at off-peak times stochastic TA produces more realistic results as traffic demand (volume) is spread out across more paths:

> [...] for off peak assignments, stochastic assignment is often used, which tends to assign trips across more paths and thus better reflects the observed traffic volumes in un-congested periods [...]

Stochastic assignment models that a user's perception of the same path can differ, which is achieved by adding a stochastic error term to the actual path cost. Therefore, each user may choose a different path that they perceive as the shortest one, which achieves that the trips associated with each OD pair w spread out over more of the available paths $\mathcal{R}_w$ even though the actual path cost (without perception error) of some of the paths is non-optimal.

Spreading out trips across paths could also be achieved by a bi-objective (or multi-objective) TA as this recognises that there is more than one objective in route choice and that drivers spread out on all available efficient (trade-off) solutions. While one objective remains travel time, the other objective(s) would have to be determined to model those other factors that determine route choice.

In contrast to most literature discussed in the previous section, we give a general characterisation of *bi-objective* traffic assignment that is based on weaker behavioural assumptions than assuming a linear choice function as discussed in Section 4.3.2. This definition generalises to the multi-objective case. We believe that our definitions of bi-objective TA directly extends the idea of Wardrop's equilibrium principle to two (or multiple) objectives. According to Wardrop's first principle, traffic arranges itself so that all drivers between one OD pair experience the same minimal travel time. We believe that a natural extension of this principle means that every driver for an OD pair travels on minimal paths according to the two (or more) objectives, i.e. on efficient paths:

Definition 4.4.1 Under bi-objective user equilibrium (BUE) conditions traffic arranges itself in such a way that no individual trip maker can improve either their toll or travel time or both without worsening the other component by unilaterally switching to another route.

Similarly, under multi-objective user equilibrium (MUE) conditions traffic arranges itself in such a way that no individual trip maker can improve at least one of their p objectives without worsening any of the others by unilaterally switching to another route.

Note that, similar to Wardrop's principle (see quote on p. 143), it is again assumed that travellers all have the same objectives, which are minimised, and that they have perfect knowledge of the objective vector associated with each alternative route.

The first introduction of this definition of BUE or MUE is attributed to Chen and Yen (1993).

In order to formalise the BUE/MUE concept, we need some more notation. We combine the two objective vectors $\overline{t}, \overline{m} : \mathbb{R}^m \mapsto \mathbb{R}$ into the objective matrix $\overline{C} : \mathbb{R}^m \mapsto \mathbb{R}^{2\times m}$ where the first row of $\overline{C}$ is $\overline{C}_{1\cdot} = \overline{t}$ and the second row of $\overline{C}$ is $\overline{C}_{2\cdot} = \overline{m}$. Similarly, if there are p objective vectors $\overline{c}^1, \ldots, \overline{c}^p$ they are combined into the objective matrix $\overline{C} : \mathbb{R}^m \mapsto \mathbb{R}^{p\times m}$ with rows $\overline{C}_{i\cdot} = \overline{c}^i$. The matrix $C : \mathbb{R}^\rho \mapsto \mathbb{R}^{p\times\rho}$ represents the corresponding matrix of path cost functions. As we have $c^i = \Phi^\top \overline{c}^i(\overline{f}), i = 1, \ldots, p$, the path and arc cost matrices are related by $C = \Phi^\top \overline{C}$.

4.4.1 Vector Equilibrium, Vector Optimisation, and Vector Variational Inequality

In this section, we define three different multi-objective problems that are multi-objective extensions of the corresponding problems in the scalar case discussed in Section 4.2 (SEQ, the optimisation formulation (4.5), and VI).

Vector Equilibrium Problem

A solution vector $h \in \mathcal{K}$ satisfies MUE, if all dominated paths have zero flow and only non-dominated paths may have positive flow. The *vector equilibrium principle* that formalises MUE is given by the following definition. The function $C_{\cdot r} : \mathbb{R} \mapsto \mathbb{R}^p$ is the p-dimensional path cost function for path r.

Definition 4.4.2 A flow vector $f^* \in \mathcal{K}$ is said to be in *vector equilibrium* if and only if

$$\textit{(VEQ)} \quad \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad C_{\cdot s}(f^*) \geq C_{\cdot r}(f^*) \Rightarrow f_s^* = 0.$$

We also denote the corresponding bi-objective problem with $p = 2$ as BEQ. The corresponding vector of arc flow is $\overline{f}^* = \Phi f^*$. The *weak vector equilibrium (WVEQ)* problem is closely related to VEQ. WVEQ is obtained by replacing $\geq$ in VEQ by $>$, i.e. a strict dominance relation between path cost vectors is considered.

The vector equilibrium problem VEQ, i.e. an equilibrium problem in which two or more objectives are distinguished, made its first appearance in Chen and Yen (1993) and is later reconsidered in many other contributions (Yang and Goh 1997; Goh and Yang 1999; Chen et al. 1999; Yang and Goh 2000; Khan and Raciti 2005; Yang and Yu 2005; Li et al. 2007, 2008).

Vector Optimisation Problem

A first approach to solving VEQ is to investigate whether VEQ relates to a multi-objective optimisation problem similar to the single-objective optimisation formulation (4.5). We call this multi-objective optimisation problem *vector optimisation problem (VOP)* here, as this is the term used in the related literature. It is assumed that path cost functions are additive.

Definition 4.4.3 By solutions of VOP we mean the set of efficient solutions of the following multi-objective problem with objective vector z.

$$\begin{aligned} \min \quad z(\overline{f}) &= \begin{pmatrix} z_1(\overline{f}) = \sum_{a \in \mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a^1(v)dv \\ \vdots \\ z_p(\overline{f}) = \sum_{a \in \mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a^p(v)dv \end{pmatrix} \\ \text{s.t.} \quad \overline{f} &\in \mathcal{K}_{\mathcal{A}}. \end{aligned} \tag{4.6}$$

The bi-objective version of VOP is denoted BOP.

In the literature, the weak version of VOP, denoted by WVOP, has also received considerable attention. Here, efficient solutions are replaced by weakly efficient solutions of the optimisation problem.

Vector Variational Inequality Problem

The multi-objective extension of a VI is known as vector variational inequality (VVI), first introduced by Gianessi (1980). In the single-objective case, SEQ can be expressed through a VI, and attempts are made in the related literature to establish a similar connection between the two problems VEQ and VVI (with less success).

Definition 4.4.4 Assume $\mathcal{K} \subset \mathbb{R}^\rho$ closed and convex, $C : \mathcal{K} \mapsto \mathbb{R}^{p\times\rho}$ is a matrix-valued function. The *vector variational inequality problem* is the problem of finding flow $f \in \mathcal{K}$ that satisfies

$$\textit{(VVI)} \quad C(f)(h-f) \nleq 0, \forall h \in \mathcal{K}.$$

Or equivalently,

$$C(f)(h-f) \notin -\mathbb{R}^p_{\geq}, \forall h \in \mathcal{K}.$$

A *weak vector variational inequality (WVVI)* is obtained by replacing $\nleq$ in the definition of VVI with $\not<$, which gives the condition $C(f)(h-f) \notin -\mathbb{R}^p_{>}$. This definition of VVI and WVVI appears for example in Yang and Goh (1997). The existence of solutions of VVI is, for example, discussed in Yang and Goh (1997); Chen et al. (2005).

In the context of TA, the set $\mathcal{K}$ is interpreted as the set of feasible path flow vectors, ρ is the number of paths in $\mathcal{R}$, and the path cost functions $C_{\cdot r}(f)$ have p different components for each path r.

Similar to the scalar TA problem, we can define a VVI based on arc flow, which is the problem of finding $\overline{f} \in \mathcal{K}_\mathcal{A}$ that satisfies

$$\textit{(VVIa)} \quad \overline{C}(\overline{f})(\overline{h}-\overline{f}) \nleq 0, \forall \overline{h} \in \mathcal{K}_\mathcal{A}.$$

VVI and VVIa are equivalent problems provided that the path cost function

is additive. This can be shown similar to Smith (1979). We have

$$\begin{aligned} C(f)h &= \sum_{r\in\mathcal{R}} C_r(f)h_r = \sum_{r\in\mathcal{R}} \left(\sum_{a\in\mathcal{A}} \overline{C}_a(\Phi f)\phi_{ar} \right) h_r \\ &= \sum_{a\in\mathcal{A}} \overline{C}_a(\overline{f}) \left(\sum_{r\in\mathcal{R}} h_r\phi_{ar} \right) = \sum_{a\in\mathcal{A}} \overline{C}_a(\overline{f})\overline{h}_a \\ &= \overline{C}(\overline{f})\overline{h}. \end{aligned}$$

It follows that $C(f)(h-f) = \overline{C}(\overline{f})(\overline{h}-\overline{f})$ and therefore VVI and VVIa are equivalent for additive path cost functions.

Lee et al. (1998) introduce a *scalarised variational inequality problem*. For $\xi \in \mathbb{R}^p_{\geqq}$, the flow $f \in \mathcal{K}$ is a solution to the *scalarised variational inequality problem* if and only if:

$$(VI_\xi) \quad \left(\sum_{i=1}^{p} \xi_i C_{i\cdot}(f) \right) (h-f) \geqq 0, \forall h \in \mathcal{K}. \tag{4.7}$$

4.4.2 Literature

The literature on VEQ and related problems is of mainly theoretical nature. Many articles discuss existence of solutions and attempt to relate VEQ and WVEQ to other problems such as (W)VVI and (W)VOP. Some articles mention that a possible application of VEQ is in TA, but no actual TA problems have been solved and no solution algorithms have been proposed.

In the literature, some incorrect results can be found. Some of them have been reported before, whereas we discuss some new ones. We provide counter examples to false claims and report on known incorrect results.

Before we discuss the literature on VEQ, VVI, and VOP, we need to introduce more notation. Given a feasible flow solution $f \in \mathcal{K}$, the following denote the set of (non-dominated) path cost vectors and the set of efficient paths.

Definition 4.4.5 For a feasible $f \in \mathcal{K}$, the set of all path cost vectors for OD

pair w is given by

$$\mathcal{Z}^w(f) = \{z \in \mathbb{R}^p : \exists r \in \mathcal{R}_w \text{ with } z = C_{\cdot r}(f)\}.$$

We define the set of non-dominated points for an OD pair $w \in \mathcal{W}$ by

$$\mathcal{Z}_N^w(f) = \{z \in \mathcal{Z}^w(f) : \ z \text{ is non-dominated in } \mathcal{Z}^w(f)\},$$

while the corresponding set of efficient paths is

$$\mathcal{X}_E^w(f) = \{r \in \mathcal{R}_w : \exists z \in \mathcal{Z}_N^w(f) \text{ such that } z = C_{\cdot r}(f)\}.$$

Chen and Yen (1993)

The concept of VEQ was first introduced in a report by Chen and Yen (1993). This report is described in several other articles, but we were unfortunately unable to obtain a copy of it. After generalising Wardrop's first principle to VEQ, existence of solutions and a sufficient condition for VEQ solutions are established. Furthermore, Theorem 4.4.5 (see also Section 4.4.4) is presented, where a special VVI and VEQ are shown to be equivalent problems given that there is only a single efficient solution for each OD pair.

Yang and Goh (1997)

Yang and Goh (1997) apply the VEQ principle to the TA problem with multiple objectives. They also introduce WVEQ, VVI, WVVI, VOP, and WVOP based on a closed and convex subset $\mathcal{K}$ of $\mathbb{R}^n$. It is assumed that $F : \mathcal{K} \mapsto \mathbb{R}^{p \times n}$ is a continuously differentiable function. It is shown that a solution of VVI is a solution of VEQ (as in Theorem 4.4.4), and also that a solution of WVVI is a solution of WVEQ. They repeat Theorem 4.4.5 (Chen and Yen 1993). In the remainder of the paper, VVI, WVVI, VOP, and WVOP are related to each other. A VOP with objective f is derived from the VVI

$$\text{find } x \in \mathcal{K}, \text{ s.t. } F(x)(y - x) \not\leq 0, \forall y \in \mathcal{K},$$

given that each component $F_{i\cdot}, i = 1, \ldots, p$ of F is separable, by integrating each of the component functions $f_i = \int^x F_{i\cdot}(v)dv$. It is shown that a solution of VVI is an efficient solution of the related VOP if the objective function f of VOP is $\mathbb{R}^n_{\leqq}$-convex, and an example showing that the reverse does not hold is given. They repeat a theorem from Chen and Yen (1993) which states that a weakly efficient solution of WVOP is a solution of WVVI, and the converse is true if the objective function f of WVOP is $\mathbb{R}^n_{\leqq}$-convex. Strict $\mathbb{R}^n_{\leqq}$-convexity is necessary to conclude that a solution of WVOP also solves VOP. Strict $\mathbb{R}^n_{\leqq}$-concavity, on the other hand implies that a solution of WVOP also solves VVI. Therefore, under the same assumption, a solution of VOP is a solution of VVI (as every solution of VOP is a solution of WVOP). This also seems to indicate that there can be no equivalence relationship between VOP and VVI, as convexity and strict concavity cannot be satisfied simultaneously.

Goh and Yang (1999)

Goh and Yang (1999) study VEQ and aim at establishing a link with VOP. Given a scalarised VI_ξ with $\xi \in \mathbb{R}^p_{>}$, they show that the obtained solution of VI_ξ satisfies VEQ. They also propose the re-formulation of VI_ξ as a parametric complementarity problem. They show under which conditions a properly efficient solution of VOP solves VEQ, see also Remark 4.4.1. Furthermore, the same problems are studied in the case of separable, affine, and monotone arc cost functions. As shown by Li et al. (2006), only parts of the results in Goh and Yang (1999) are correct, which we demonstrate in the following.

Goh and Yang (1999) introduce a *parametric equilibrium principle*:

Definition 4.4.6 (Goh and Yang 1999)
$\Lambda := \{\lambda \in \mathbb{R}^p : \lambda_i \geqq 0, \sum_{i=1}^p \lambda_i = 1\}$. Let a $\lambda \in \Lambda$ be given. A path flow vector f is in λ-*equilibrium* if for all $w \in \mathcal{W}$ and for all $s \in \mathcal{R}_w$

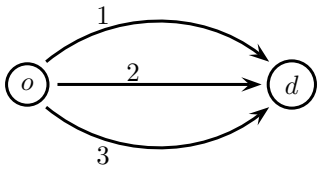
$$f_s = 0 \text{ whenever } \exists z \in \mathcal{Z}^w_N(f) \text{ such that } \lambda^\top C_{\cdot s}(f) > \lambda^\top z.$$

Goh and Yang propose that if f is a solution of VEQ and if all non-dominated points lie on the boundary of their convex hull, $\mathcal{Z}^w_N(f) \subset bd(conv(\mathcal{Z}^w_N(f)))$, then there exists $\lambda \in \Lambda$ such that f is in λ-equilibrium. Note that this does

not require that all non-dominated points are supported, as a non-supported point may also lie on the boundary. A restriction to supported points only can be achieved by considering $conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq})$.

The proposed theorem by Goh and Yang (1999) is incorrect, as shown by Li et al. (2006). This counterexample contains a non-supported point.

Example 4.4.1 Example by Li et al. (2006) *A network with the following structure is used.*



The cost matrix is given as

$$C(f) = \begin{pmatrix} 6f_1 & 5f_2 & 7f_3 \\ 3f_1 & 4f_2 & f_3 \end{pmatrix}.$$

With the choice of

$$f^* = (2,2,2)^\top \text{ and } C(f^*) = \begin{pmatrix} 12 & 10 & 14 \\ 6 & 8 & 2 \end{pmatrix},$$

solutions $C_{\cdot 2}(f^*)$ and $C_{\cdot 3}(f^*)$ are supported, whereas $C_{\cdot 1}(f^*)$ is non-supported. Li et al. (2006) show that that f^* is not in λ-equilibrium, a contradiction to Theorem 2.1.(i) by Goh and Yang (1999).

We show that the proposed theorem by Goh and Yang (1999) is still incorrect when assuming that all points are supported, i.e.$\mathcal{Z}_N^w(f) \subset bd(conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq}))$.

Example 4.4.2 *A network with the same structure as Example 4.4.1 is used. The total demand from o to d is 1000 vehicles, so that the set of feasible path*

flows is given by $\mathcal{K} = \left\{ f \in \mathbb{R}^3_{\geqq} : f_1 + f_2 + f_3 = 1000 \right\}$. *Arc cost functions are:*

$$C(f) = \begin{pmatrix} 10\left[1 + 0.15\left(\frac{f_1}{200}\right)^4\right] & 20\left[1 + 0.15\left(\frac{f_2}{400}\right)^4\right] & 25\left[1 + 0.15\left(\frac{f_3}{300}\right)^4\right] \\ 20 & 15 & 0 \end{pmatrix}.$$

The first cost component $C_{1\cdot}$ *represents travel time, whereas the second one,* $C_{2\cdot}$*, represents a toll. Thus, Route* 1 *is the fastest route with the highest toll while Route* 3 *is toll free and the slowest.*

The vector f^*, defined as follows, is a feasible path flow solution that satisfies VEQ:

$$f^* = (300, 300, 400)^\top \text{ with } C(f^*) = \begin{pmatrix} 17.59 & 20.95 & 36.85 \\ 20 & 15 & 0 \end{pmatrix}.$$

The three objective vectors $C_{\cdot i}(f^*), i = 1, 2, 3$ are non-dominated and supported as they satisfy $\mathcal{Z}^w_N(f^*) \subset bd(conv(\mathcal{Z}^w_N(f^*) + \mathbb{R}^p_{\geqq}))$. As all three objective vectors are supported, we can choose $\lambda_i \in \Lambda$ such that each one of $\lambda_i^\top C_{\cdot i}(f^*), i = 1, 2, 3$ solves the problem $\min_{z \in \mathcal{Z}^w_N(f^*)} \lambda_i^\top z$. For any fixed choice of $\lambda \in \Lambda$, at most two of the three objective vectors become optimal. Therefore, for fixed $\lambda \in \Lambda$, the above solution f^* is never in λ-equilibrium, a contradiction to Theorem 2.1.(i) by Goh and Yang (1999).

Li et al. (2006) also provide a correction of Theorem 2.1.(i) by Goh and Yang (1999) described in the previous section. In order to do so, they give a different definition of *weakened parametric equilibrium*:

Definition 4.4.7 A path flow vector f is in *weakened parametric equilibrium* if for all $w \in \mathcal{W}$ and for all $s \in \mathcal{R}_w$

$$h_s = 0 \text{ whenever } \forall \lambda \in \Lambda, \exists z \in \mathcal{Z}^w_N(f) \text{ such that } \lambda^\top C_{\cdot s}(f) > \lambda^\top z.$$

Now, Theorem 2.4 in Li et al. (2006) states that whenever f solves VEQ, f is in weakened parametric equilibrium. Note that they omit the assumption $\mathcal{Z}^w_N(f) \subset bd(conv(\mathcal{Z}^w_N(f) + \mathbb{R}^p_{\geqq}))$. In this case their own Example 4.4.1 serves

as a counterexample as the non-supported solution $C_{\cdot 1}(f^*)$ does never solve a problem of the form $\min_{z \in \mathcal{Z}_N^w(f^*)} \lambda_i^\top z$, therefore a solution with $f_1^* > 0$ is never in weakened parametric equilibrium. The theorem can be corrected by including the assumption $\mathcal{Z}_N^w(f) \subset bd(conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq}))$.

Theorem 4.4.1 *If a flow $f \in \mathcal{K}$ solves VEQ and $\mathcal{Z}_N^w(f) \subset bd(conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq}))$, then f is in weakened parametric equilibrium.*

It should be noted that incorrectness of Theorem 2.1.(i) in Goh and Yang (1999) entails incorrectness of Theorems 3.2.(i) and 3.3 (Goh and Yang 1999). The corresponding Theorems in Yang and Goh (2000, Theorems 4(i),7,9), Chen et al. (2005, Chapter 6), and Yang and Yu (2005, Proposition 5.2) are also incorrect.

Chen et al. (1999)

Chen et al. (1999) apply a non-linear scalarisation function, ξ_{ea}, to a VVI and to the VEQ principle. They relate the scalarised VEQ to a scalarised VVI. They also show that solutions satisfying this (non-linear) scalarised VEQ principle also satisfy VEQ, but unfortunately the converse claim is not true as shown by Li et al. (2006). We repeat the main points here:

Let $K \subsetneqq \mathbb{R}^p$ a closed, convex, and pointed cone, such as $K = \mathbb{R}^p_{\leqq}$. Chen et al. (1999) propose to link VEQ and VVI via the non-linear scalarisation function $\xi_{ea} : K \mapsto \mathbb{R}$, which for given $e \in int(K)$ and $a \in \mathbb{R}^p$ is defined as

$$\xi_{ea}(y) = \min \{t \in \mathbb{R} : y \in a + te - K\}.$$

When $K = \mathbb{R}^p_{\leqq}$ with $e = (e_1, e_2, \ldots, e_p)$, ξ_{ea} can be re-written as

$$\xi_{ea}(y) = \max \left\{ \frac{y_i - a_i}{e_i}, 1 \leqq i \leqq p \right\}.$$

The notion of ξ_{ea}-equilibrium is introduced as follows:

Definition 4.4.8 The path flow vector $f \in \mathcal{K}$ is said to be in ξ_{ea}-*equilibrium*

if there exist $e \in int(\mathbb{R}^p_{\leqq})$ and $a \in \mathbb{R}^p$ such that

$$\forall w \in \mathcal{W}, \forall s,t \in \mathcal{R}_w \quad \xi_{ea} \circ C_{\cdot s}(f) > \xi_{ea} \circ C_{\cdot t}(f) \Longrightarrow f_s = 0.$$

Chen et al. (1999) propose that a path flow f satisfies (WVEQ) if and only if f is in ξ_{ea}-equilibrium for some $e \in int(\mathbb{R}^p_{\leqq})$ and $a \in \mathbb{R}^p$. This is incorrect, as explained in Li et al. (2006) on the basis of Example 4.4.1. Li et al. also give an alternative formulation for weak ξ_{ea}-equilibrium:

Definition 4.4.9 The path flow vector $f \in \mathcal{K}$ is said to be in *weak* ξ_{ea}*-equilibrium* if for any $w \in \mathcal{W}$ we have

$$\forall s,t \in \mathcal{R}_w \quad \xi_{eC_{\cdot t}(f)} \circ C_{\cdot s}(f) < 0 \Longrightarrow f_t = 0.$$

With this definition of weak ξ_{ea}-equilibrium, they obtain the following:

Theorem 4.4.2 *A flow $f \in \mathcal{K}$ satisfies WVEQ if and only if f is in weak ξ_{ea}-equilibrium.*

Unfortunately, Theorem 4.4.1 as well as 4.4.2 do not seem to give rise to a solution algorithm, whereas the incorrect versions of the theorems by Goh and Yang (1999) and Chen et al. (1999) reduced the problem to easily-solvable scalar equilibrium problems, which can be solved via their related variational inequality problems.

Yang and Goh (2000)

Yang and Goh (2000) develop the relationship between VVI, VOP, VEQ, their weak versions, and the scalarised versions based on orderings defined by a closed convex cone (rather than simply $\mathbb{R}^p_{\leqq}$ as in previous papers). Some results from their previous paper (Goh and Yang 1999) are extended, which are incorrect as demonstrated earlier. This affects Yang and Goh (2000, Theorems 4(i),7,9).

Oettli (2001)

Oettli (2001) studies TA in a very general setting that allows for interpretation as capacitated equilibrium with user classes. He aims to derive a vector equilibrium principle from a vector variational inequality and to establish equivalence. The author takes a different approach to establishing an equivalence relation between vector variational inequalities and vector equilibria by simply modifying the equilibrium conditions to yield the desired equivalence. Oettli (2001, p.224) states

> But rather the variational equilibrium, which remains invariant if one passes to the vectorial setting, should be the basic notion, and the definition of a Wardrop equilibrium must be adapted in each case.

Unfortunately, it remains unclear to what extent the variational inequalities (which variational equilibria are defined on) are invariant.

Oettli (2001) introduces an equilibrium problem with multiple classes and objectives as $f^* \in \mathcal{K}$ satisfying

$$\sum_{r\in\mathcal{R}} C_r(f^*)\,(h_r - f_r^*) \in \mathbb{R}_{\geqq}, \quad \forall h \in \mathcal{K}, \tag{4.8}$$

where the vectors h_r and f_r are column vectors in $\mathbb{R}^q$ and $C_r(f^*)$ is a $p \times q$ matrix. For $q = 1$ this is VI_ξ (4.7) with weighting factors $\xi_i = 1$. Oettli (2001) attempts to derive equilibrium conditions that are equivalent to (4.8).

The obtained equilibrium conditions model the problem with capacitated path flow. The aim is to find $f \in \mathcal{K}$ that, for any fixed cost vector $\tilde{C} = C(f)$, satisfies

$$\begin{gathered} \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \\ \tilde{C}_r - \tilde{C}_s \in -\mathbb{R}_{\geq} \Rightarrow f_s - l_s \notin \mathbb{R}_{>} \text{ or } f_r - u_r \notin -\mathbb{R}_{>}. \end{gathered} \tag{4.9}$$

Oettli (2001) derives necessary and sufficient conditions for a solution $f \in \hat{\mathcal{K}}$ of equilibrium conditions (4.9) also satisfying (4.8) with fixed cost vector $\tilde{C}$. While the necessary condition is straight forward, the sufficient condition has strong assumptions. In Oettli (2001, Proposition 4.3) both necessary and sufficient conditions are given. Unfortunately, they appear to be contradicting

when $p > 1$. In the setting of the paper $\mathcal{Y}$ is the space of linear mappings between $\mathcal{X}$ (space of path flow) and $\mathcal{Z}$ (objective space). Each cost vector $C_r(f)$ is such a linear mapping. The convex ordering cone $P_\mathcal{Y}$ of $\mathcal{Y}$ is assumed to be pointed so that $P_\mathcal{Y} \cap (-P_\mathcal{Y}) = \{0\}$. On the other hand, the proposition assumes that $P_\mathcal{Y} \cup (-P_\mathcal{Y}) = \mathcal{Y}$. The two assumptions are trivially true for the single-objective problem with $\mathcal{Z} = \mathbb{R}$ which makes the space of linear mappings the set of scalars $\mathcal{Y} = \mathbb{R}$, and the ordering cone $P_\mathcal{Y} = \mathbb{R}_{\geqq}$.

If $\mathcal{Z} = \mathbb{R}^p, p > 1$ and $\mathcal{Y} = \mathbb{R}^p$ it is not clear how to find such an ordering cone with $P_\mathcal{Y} \cap (-P_\mathcal{Y}) = \{0\}$ and $P_\mathcal{Y} \cup (-P_\mathcal{Y}) = \mathcal{Y}$. A cone $P_\mathcal{Y} \subset \mathbb{R}^p$ is called *acute* if and only if its closure is contained by an open half-space $\mathcal{H}$ and the origin, i.e. $cl(P_\mathcal{Y}) \subset \mathcal{H} \cup \{0\}$. The cone $P_\mathcal{Y}$ is acute if and only if $cl(P_\mathcal{Y})$ is pointed, see Yu (1985). For an acute cone $P_\mathcal{Y}$ we have $P_\mathcal{Y} \cup (-P_\mathcal{Y}) \subset \mathcal{H} \cup (-\mathcal{H}) \cup \{0\} \neq \mathcal{Y}$. If the cone $P_\mathcal{Y}$ is closed and pointed, then $cl(P_\mathcal{Y}) = P_\mathcal{Y}$ and therefore it is also acute. Hence, $P_\mathcal{Y} \cup (-P_\mathcal{Y}) = \mathcal{Y}$ cannot hold. A pointed open cone must be an open half-space together with the origin or an open subset thereof: $P_\mathcal{Y} \subsetneqq \mathcal{H} \cup \{0\}$. Again $P_\mathcal{Y} \cup (-P_\mathcal{Y}) \neq \mathcal{Y}$ follows. If the cone $P_\mathcal{Y}$ is open, it represents only weak dominance.

Yang and Yu (2005)

Yang and Yu (2005) show existence of a solution of VVI, relate the problem to VOP, and present some so-called gap functions. A dynamic traffic equilibrium is introduced where flow and demand are time dependent. This is extended to a dynamic vector equilibrium for which sufficient conditions (in terms of VVI and WVVI) are given. Note that a proposition Yang and Yu (2005, Proposition 5.2) appears to be incorrect, see also the discussion of Goh and Yang (1999).

Konnov (2005)

Konnov (2005) studies VVIs with set-valued functions F. The paper contains a section on the application of vector equilibrium principles and traffic assignment, where $F(x)$ always takes a single value. After introducing a WVEQ problem, the corresponding WVVI is presented, a solution of which is always a solution of WVEQ, and its scalar reformulation is given, formulated as single WVVI with set-valued function.

Khan and Raciti (2005)

Khan and Raciti (2005) study a time-dependent vector equilibrium problem, where flow, capacity constraints, and demand may depend on time. The concept of (weak) vector equilibrium is formulated similar to (W)VEQ in this thesis. Corresponding vector variational inequalities are also given, solutions of which are shown to be (weak) vector equilibrium solutions.

Chen et al. (2005)

Chapter 6 of Chen et al. (2005) is on vector network equilibrium problems. Findings from previous papers are collected in a section on WVEQ, one on VEQ, and one on dynamic VEQ. It should be noted that some incorrect results as discussed for Goh and Yang (1999) are included in Chen et al. (2005, Chapter 6).

Cheng and Wu (2006)

Even though the problem studied in Cheng and Wu (2006) is not exactly what we are interested in, it is mentioned here as there seem to be several flaws in the presentation.

Cheng and Wu (2006) formulate a problem where multiple products traverse a network and each product incurs different multi-objective costs. The products can also be interpreted as user classes. Initially, the multi-product problem is considered with a single objective, i.e. $p = 1$. This means, they consider a network in which certain goods are produced by suppliers and need to be shipped to certain destination points. There also are warehouses, i.e. nodes through which products can be shipped, without staying there. There is a certain demand for every product. The cost of transporting different products along an arc may differ. We first need some additional notation. Let q be the number of different products, then the amount of product $j = 1, \ldots, q$ to be shipped between OD pair w is d_w^j and the flow of product j on arc a is $\overline{f}_a^j$. Similarly, the cost of product j on arc a is $\overline{c}_a^j$, and the corresponding cost on path r is $c_r^j = \sum_{a \in r} \overline{c}_a^j$.

Cheng and Wu (2006) define f_r^j, the flow of product j on path $r \in \mathcal{R}_w$, by

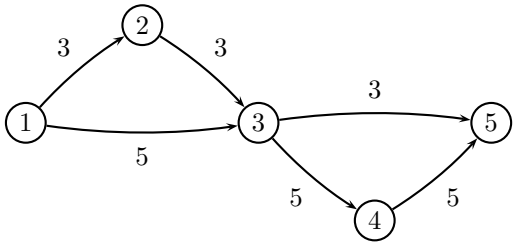
$$f_r^j = \min\{\overline{f}_a^j : a \in r\}. \tag{4.10}$$

Then, they state a flow satisfies the demand if

$$\sum_{r \in \mathcal{R}_w} f_r^j = d_w^j \quad \forall w \in \mathcal{W}, j = 1, \ldots, q. \tag{4.11}$$

Considering the following example, this definition does not seems intuitive:

Example 4.4.3 *Consider the following network with the amount of flow of a single product, so $q = 1$, indicated next to each arc. The demand for this product between OD pair $1 = (1, 5)$ is $d_1^1 = 8$, so intuitively satisfied by the arc flow indicated in the figure.*



For the OD pair $(1, 5)$, there are the following paths: $r_1 = (1, 2), (2, 3), (3, 5)$, $r_2 = (1, 2), (2, 3), (3, 4), (4, 5)$, $r_3 = (1, 3), (3, 5)$, and $r_4 = (1, 3), (3, 4), (4, 5)$. Evaluating (4.10) for the four paths yields

$$f_{r_1}^1 = 3, \quad f_{r_2}^1 = 3, \quad f_{r_3}^1 = 3, \quad f_{r_1}^1 = 5,$$

and therefore (4.11) yields $\sum_{r \in \mathcal{R}_1} f_r^1 = 14$, which does not satisfy $d_1^1 = 8$. Instead of trying to derive path flow from arc flow as in (4.10), one should simply introduce a variable for the flow of every product j on every path r. It is well known that arc flow can be uniquely defined in terms of path flow, but not vice versa.

The set $\mathcal{K}$ now denotes all demand feasible solutions. Cheng and Wu (2006) define $m_w^j(f)$ as minimum path cost of product j between OD pair w, i.e. $m_w^j = \min_{r \in \mathcal{R}_w} c_r^j(f)$. Both $c_r^j(f)$ and $m_w^j(f)$ are grouped into vectors by $c_r(f) =$

$(c_r^1(f), c_r^2(f), \ldots, c_r^q(f))^\top$ and $m_w(f) = (m_w^1(f), m_w^2(f), \ldots, m_w^q(f))^\top$. Subsequently, an equilibrium flow pattern is introduced.

A vector $v \in D$ is called *equilibrium flow pattern* if and only if

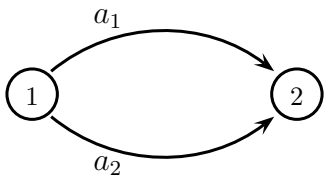
$$\forall w \in \mathcal{W}, \forall r \in \mathcal{R}_w \quad c_r(f) - m_w(f) \begin{cases} = 0 & \text{if } f_r \in \mathbb{R}^q_{\geq}, \\ \geqq 0 & \text{if } f_r = 0. \end{cases} \tag{4.12}$$

According to Cheng and Wu (2006, Prop. 2.1), (4.12) is equivalent to the following:

$$c_r(f) - c_s(f) \in \mathbb{R}^q_{\geq} \Rightarrow f_r = 0 \text{ for each } w \in \mathcal{W} \text{ and any } r, s \in \mathcal{R}_w. \tag{4.13}$$

Note that $f_r = (f_r^1, \ldots, f_r^q)^\top$ here is the vector of path flow of products $1, \ldots, q$ on path r. We present a simple example to show that that (4.13) and (4.12) are not equivalent.

Example 4.4.4 *In this example we assume $q = 2$, $\mathcal{W} = \{1 = (1, 2)\}$ and that there are exactly two paths connecting the OD pair $w = 1$, namely $r_1 = a_1$ and $r_2 = a_2$. Furthermore we assume $d_1^1 = 1$ and $d_1^2 = 1$.*



For the two paths $r_1 = a_1, r_2 = a_2$ and the two products $j = 1, 2$ we have the following costs:

$$\begin{aligned} c_{r_1}^1(f) &= 10(f_{a_1}^1 + f_{a_1}^2) \quad c_{r_2}^1(f) = 7(f_{a_2}^1 + f_{a_2}^2) \\ c_{r_1}^2(f) &= 3(f_{a_1}^1 + f_{a_1}^2) \quad c_{r_2}^2(f) = 7(f_{a_2}^1 + f_{a_2}^2) \end{aligned}$$

Choosing $h_{r_1}^1 = 0, h_{r_1}^2 = 1, h_{r_2}^1 = 1, h_{r_2}^2 = 0$, which is a demand feasible solution, we obtain

$$c_{r_1}(h) = (10, 3)^\top, \quad c_{r_2}(h) = (7, 7)^\top, \quad m_1(h) = (7, 3)^\top.$$

Clearly, the above solution h is not an equilibrium flow pattern according to (4.12) as

$$c_{r_1}(h) - m_1(h) = (3,0)^\top \geqq 0 \text{ and } c_{r_2}(h) - m_1(h) = (0,4)^\top \geqq 0,$$

but both $h_{r_1} \in \mathbb{R}^q_{\geq}$ and $h_{r_2} \in \mathbb{R}^q_{\geq}$. The vector h does, however, satisfy (4.13) as both $c_{r_1} - c_{r_2} \notin \mathbb{R}^q_{\geq}$ and $c_{r_2} - c_{r_1} \notin \mathbb{R}^q_{\geq}$. In fact, an equilibrium flow pattern satisfying (4.12) can only exist if the cost for each product is minimal along the *same* path, so if there exists a unique optimal path. In Example 4.4.4 there exists no feasible solution satisfying the equilibrium flow pattern by Cheng and Wu.

Cheng and Wu (2006, Theorem 2.1.) state the following: A vector flow $f^* \in \mathcal{K}$ is an equilibrium pattern flow as in (4.12) if and only if f^* is a solution to the following vector variational inequality:

$$\text{find } f^* \in \mathcal{K}, \text{ s.t. } c(f^*)(h - f^*)^\top \in \mathbb{R}^{q\times q}_{\leqq} \quad \forall h \in \mathcal{K}. \tag{4.14}$$

For $\rho = |\mathcal{R}|$, the cost matrix is defined as $c(f) := (c_1(f), \ldots, c_\rho(f)) \in \mathbb{R}^{q\times\rho}$, and the flow matrix by $f := (f_1, \ldots, f_\rho) \in \mathbb{R}^{q\times\rho}$, where every $c_r(f)$ and f_r is a column vector of class cost and class flow, respectively. Throughout the proof the term equilibrium pattern flow as defined in (4.13) is used, so we will show that with this definition, their claim is incorrect. Again, we use the feasible solution from Example 4.4.4, which satisfies (4.13), and show that the variational inequality (4.14) is not satisfied:

$$\begin{aligned} c(h)(u-h)^\top &= \begin{pmatrix} 10 & 7 \\ 3 & 7 \end{pmatrix} \begin{pmatrix} u^1_{r_1} & u^2_{r_1} - 1 \\ u^1_{r_2} - 1 & u^2_{r_2} \end{pmatrix} \\ &= \begin{pmatrix} 10u^1_{r_1} + 7u^1_{r_2} - 7 & 10u^2_{r_1} - 10 + 7u^2_{r_2} \\ 3u^1_{r_1} + 7u^1_{r_2} - 7 & 3u^2_{r_1} - 3 + 7u^2_{r_2} \end{pmatrix} \end{aligned}$$

Choosing the feasible solution $\hat{u}^1_{r_1} = 1, \hat{u}^2_{r_1} = 0, \hat{u}^1_{r_2} = 0, \hat{u}^2_{r_2} = 1$, we obtain:

$$c(h)(\hat{u}-h)^\top = \begin{pmatrix} 3 & -3 \\ -4 & 4 \end{pmatrix} \notin \mathbb{R}^{q\times q}_{\leqq}.$$

More importantly in the context of this thesis, Cheng and Wu (2006) consider the multi-product, multi-objective equilibrium problem defined similar to (4.13) by extending scalar-valued cost functions to p-dimensional cost vectors for every product j (Cheng and Wu 2006, Def 3.1): A vector $f^* \in \mathcal{K}$ is said to be an *equilibrium pattern flow in the generalised context of the multi-product supply-demand network equilibrium problem with vector-valued cost function* if and only if

$$C_r(f^*) - C_s(f^*) \in \mathbb{R}^{q\times p}_{\geq} \Rightarrow f^*_r = 0 \quad \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w. \tag{4.15}$$

They claim that an equilibrium according to this is equivalent to a ξ_e-equilibrium pattern. The function ξ_e is just ξ_{ea}, see (4.8), without the a component: Given a fixed $e \in \mathbb{R}^p_{>}$, $\xi_e : \mathbb{R}^p \to \mathbb{R}$ is defined here by

$$\xi_e(y) = \min\left\{\lambda \in \mathbb{R} : y \in \lambda e - \mathbb{R}^p_{\geqq}\right\} \quad \forall y \in \mathbb{R}^p.$$

The usage of ξ_e gives rise to the definition of ξ_e-equilibrium. This was attempted similarly by Chen et al. (1999), and shown to be incorrect by Li et al. (2006). A vector $f^* \in \mathcal{K}$ is said to be an ξ_e-equilibrium pattern flow in the vector valued network equilibrium problem for multiple products if there exists an $e \in int(\mathbb{R}^p_{\geqq})$ such that (Cheng and Wu 2006, Def 3.3):

$$\xi_e \circ C_r(f^*) - \xi_e \circ C_s(f^*) \in \mathbb{R}^q_{\geq} \Rightarrow f^*_r = 0 \quad \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w. \tag{4.16}$$

Despite their attempts to prove the contrary (Cheng and Wu 2006, Theorem 3.1), the two concepts (4.15) and ξ_e-equilibrium (4.16) are not equivalent. This can be shown by the same counterexample used by Li et al. (2006).

Li et al. (2006)

Li et al. (2006) show incorrectness of a weighted sum scalarisation by Goh and Yang (1999) and a scalarisation based on the function ξ_{ea} by Chen et al. (1999). Li et al. (2006) give correct re-formulations of those scalarisations, which give rise to the concept of weakened parametric (weighted sum) equilibrium and weak ξ_{ea}-equilibrium.

Li et al. (2007)

Li et al. (2007) extend the concept of ξ_{ea}-equilibrium to vector equilibrium problems with path capacity constraints, elastic demand, and different user classes. First, a system of VVIs is introduced, a solution of which always satisfies (W)VEQ. It is shown that a feasible solution is in weak ξ_{ea}-equilibrium if and only if it satisfies WVEQ. They also show that a solution satisfying the original concept of (non-weak) ξ_{ea}-equilibrium (Chen et al. 1999) does also satisfy VEQ, but not vice versa. It is furthermore shown that from the above weak ξ_{ea}-equilibrium principle a VVI formulations can be derived, which is equivalent to WVEQ. For the (non-weak) ξ_{ea}-equilibrium a VVI is derived that implies WVEQ, but not vice versa.

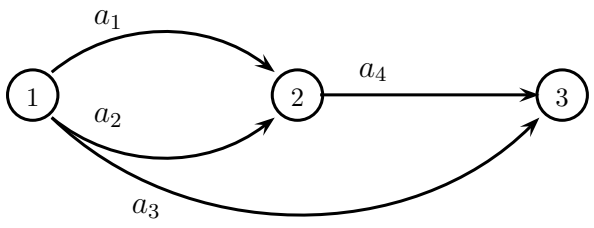
Li et al. (2008)

Li et al. (2008) aim at relating a capacitated version of VEQ to a special multi-objective minimum cost flow problem defined in the following. It is assumed that there are lower and upper bounds $\overline{l}, \overline{u}$ on arc flow. From those they derive limits for path flow on path r by

$$l_r = \max\{\overline{l}_a \phi_{ar} : a \in \mathcal{A}\} \text{ and } u_r = \max\{\overline{u}_a \phi_{ar} : a \in \mathcal{A}\}. \tag{4.17}$$

Clearly, this does not guarantee that the resulting arc flow is within the bounds $\overline{l}, \overline{u}$. The following simple example demonstrates this.

Example 4.4.5 *Let the demand for the single OD pair* $(1,3)$ *be* 2 *and the upper bound for flow on each arc* 1*, i.e.* $\overline{u} = (1,1,1,1)$*. There are three paths,* $r_1 = a_1, a_4$*,* $r_2 = a_2, a_4$*, and* $r_3 = a_3$*. According to* (4.17)*, the upper bound for flow on each path is also* 1*, i.e.* $u = (1,1,1)$*.*



Clearly, the path flow solution $f_{r_1} = 1$, $f_{r_2} = 1$, $f_{r_3} = 0$ *satisfies* $f \leqq u$, *but the resulting arc flow has* $\overline{f}_{a_4} = 2$, *which violates the upper bound on arc flow.*

Li et al. (2008) formulate the following optimisation problem

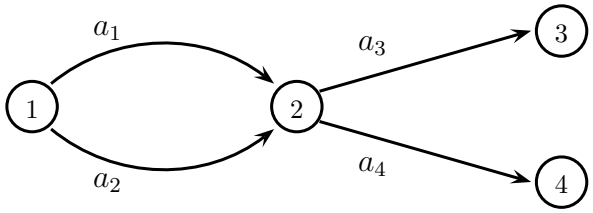
$$\begin{array}{ll} \min & \sum_{r\in\mathcal{R}} C_{\cdot r}(f_r) \\ \text{s.t.} & l \leqq f \leqq u \\ & \sum_{r\in\mathcal{R}_w} f_r = d_w \quad \text{for all } w \in \mathcal{W}, \end{array} \tag{4.18}$$

where path cost is assumed to be additive, i.e. $C_{\cdot r}(f_r) = \sum_{a\in\mathcal{A}} \overline{C}_{\cdot a}(\overline{f}_a)$. Also, arc costs are assumed to be given by linear functions depending on arc flow. A feasible f^* is in capacitated vector equilibrium if and only if:

$$\forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad C_{\cdot s}(f^*) \geq C_{\cdot t}(f^*) \Rightarrow f^*_s = l_s \text{ or } f^*_t = u_t. \tag{4.19}$$

Li et al. (2008, Theorem 2.1) claim that an efficient solution of (4.18) always satisfies (4.19). Within the proof, the assumption that path cost functions are linear is made (without explicitly stating it), which generally does not follow from the fact that arc cost functions are linear. In Example 4.4.6 linear arc costs $\overline{C}_{\cdot a}(\overline{f})$ do not lead to linear path costs $C_{\cdot r}(f_r)$. The next example presents a network with cost functions where a solution of (4.18) does not satisfy (4.19).

Example 4.4.6 *We consider two OD pairs,* $(1, 3)$ *and* $(1, 4)$ *each with demand* 1. *The paths* $r_1 = a_1, a_3$ *and* $r_2 = a_2, a_3$ *connect the first OD pair and paths* $r_3 = a_1, a_4$ *and* $r_4 = a_2, a_4$ *connect the second one. We assume the lower bounds are zero and upper bounds are* $u_{r_i} > 1$.



We consider cost functions $\overline{C}_{\cdot a_i} = (\overline{f}_{a_i}, 1)^\top; i = 1, 3, 4$ *and* $\overline{C}_{\cdot a_2} = (2\overline{f}_{a_2}, 1)^\top$.

Now the objective in (4.18) *is* $\sum_{r\in\mathcal{R}} C_{\cdot r}(f_r) = (2\overline{f}_{a_1} + 4\overline{f}_{a_2} + 2\overline{f}_{a_3} + 2\overline{f}_{a_4}, 8)^\top$. *Clearly, an efficient solution with minimal first component is* $f_{r_1} = 1, f_{r_2} = 0, f_{r_3} = 1, f_{r_4} = 0$, *as this solution avoids the most expensive arc* a_2. *The corresponding path costs are*

$$\begin{array}{ll} C_{\cdot r_1}(f_{r_1}) = (3,2)^\top & \\ C_{\cdot r_2}(f_{r_2}) = (1,2)^\top & \textit{and} \\ & C_{\cdot r_3}(f_{r_3}) = (3,2)^\top \\ & C_{\cdot r_4}(f_{r_4}) = (1,2)^\top. \end{array} \tag{4.20}$$

For both OD pairs, the equilibrium conditions are violated as there is positive flow on non-efficient paths r_1, r_3. *Even in the special case of constant path costs* $C_{\cdot r}(f_r) = C_{\cdot r}$, *a solution of the optimisation problem does not have to satisfy the equilibrium conditions: constant path costs mean that the objective of* (4.18) *is constant and therefore any feasible solution is optimal, whereas a solution that satisfies the equilibrium conditions should only use paths with non-dominated costs.*

Li et al. (2008, Theorem 2.1) claim that an efficient solution of (4.19) satisfies (4.18) when $|\mathcal{R}_w| \leqq 2$ holds for all $w \in \mathcal{W}$. The network given in Example 4.4.6 satisfies those assumptions. The feasible path flow solution $h_{r_1} = \frac{2}{3}, h_{r_2} = \frac{1}{3}, h_{r_3} = \frac{2}{3}, h_{r_4} = \frac{1}{3}$ has path cost

$$\begin{array}{ll} C_{\cdot r_1}(h_{r_1}) = (\frac{7}{3},2)^\top & \\ C_{\cdot r_2}(h_{r_2}) = (\frac{7}{3},2)^\top & \text{and} \\ & C_{\cdot r_3}(h_{r_3}) = (\frac{7}{3},2)^\top \\ & C_{\cdot r_4}(h_{r_4}) = (\frac{7}{3},2)^\top, \end{array} \tag{4.21}$$

and therefore satisfies the equilibrium conditions (4.19). The objective function value of (4.18) for this solution is $(\frac{28}{3}, 8)^\top$, whereas the objective function value of feasible solution f in Example 4.4.6 is $(8,8)^\top \leq (\frac{28}{3}, 8)^\top$. Therefore, h is not an efficient solution of the optimisation problem. It also follows that Li et al. (2008, Theorem 2.3 and 2.4) are incorrect as they are similar to the above theorems addressing weak efficiency and weak vector equilibrium. Also, Li et al. (2008, Proposition 2.1) state that the scalar version of problems (4.18) and (4.19) are equivalent is incorrect, which can be seen by considering Example 4.4.6 without the second objective.

Li et al. (2008) generalise (4.19): a feasible f^* is in *generalised capacitated vector equilibrium* if and only if for all OD pairs $w \in \mathcal{W}$ and for any positive

integers n_1, n_2 with $n_1 + n_2 \leqq |\mathcal{R}_w|$:

$$\begin{aligned}&\textstyle\sum_{i=1}^{n_1} \lambda_i C_{\cdot s_i}(f^*_{s_i}) \geq \sum_{j=1}^{n_2} \mu_j C_{\cdot t_j}(f^*_{t_j})\\ &\quad \Rightarrow \exists i_0 \in \{1,\dots,n_1\}\ f^*_{s_{i_0}} = l_{s_{i_0}} \text{ or } \exists j_0 \in \{1,\dots,n_2\}\ f^*_{t_{j_0}} = u_{t_{j_0}},\end{aligned}$$

for all $r_j, s_i \in \mathcal{R}_w$ and $\lambda_i, \mu_j > 0, i = 1,\dots,n_1, j = 1,\dots,n_2$ and $\sum_{i=1}^{n_1} \lambda_i = 1, \sum_{j=1}^{n_2} \mu_j = 1$. In Li et al. (2008, Theorem 3.1), it is claimed that with this definition the optimisation problem and the generalised vector equilibrium problem are equivalent. However, the counter example presented in Example 4.4.6 still applies with $n_1 = n_2 = 1$.

Raciti (2008)

Raciti (2008) studies traffic assignment with different user classes as well as multi-objective cost functions for each of the user classes on each path. The models initially include capacity constraints, which are later omitted. The aim is establishing a vector equilibrium principle that is equivalent to a VVI formulation, similar to the previous work of Oettli (2001). In the following we present some of the mentioned variational equilibrium concepts and attempt an economic interpretation similar to Wardrop's.

Raciti (2008) defines a *vector variational equilibrium* according to (4.8). Note that in his paper, Raciti (2008) actually presents problem (4.8) in a more general framework assuming the space of flows (here $\mathbb{R}^\rho$), and the space of costs (here $\mathbb{R}^p$) are topological vector spaces and also that order relations are given by ordering cones. In (4.8) and everything that follows, we replace those general vector spaces by the set of q class flows, $\mathbb{R}^q$, and the cost space by $\mathbb{R}^p$ so that p objectives are considered.

Two other equilibrium concepts are also introduced by Raciti (2008), which are basically VEQ and WVEQ with additional user classes and the WVEQ concept is also formulated with capacity constraints on path flow.

Raciti (2008) shows that (4.8) is equivalent to the following equilibrium conditions:

$$\begin{gathered}\forall i \in \{1,\dots,p\}, \forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w\\ C^j_s(f)_i > C^j_r(f)_i \Rightarrow f^j_s = 0, \quad \forall j = 1,\dots,q.\end{gathered} \tag{4.22}$$

Entry i of the p-dimensional path cost vector $C_r^j(f)$ for user class j is denoted by $C_r^j(f)_i$. Therefore, the aim of identifying equilibrium conditions that are equivalent to a variational inequality formulation is achieved. However, a closer look at (4.22) reveals that those equilibrium conditions can only be satisfied if there is exactly one efficient path for every OD pair, i.e. there exists a unique minimal solution. In this case, however, it is not necessary to consider a vector equilibrium.

Fixing a product j, (4.22) is a set of scalar VIs for every component i of the objective vector. For each j, (4.22) represents p scalar VIs, each of which implies that path flow can only be positive if the i^{th} cost component is minimal on the corresponding path. So a solution exists only if all components attain their minimal cost for the same path(s).

Let us show that (4.22) does not have a solution if any OD pair w with positive demand for a user class j, $d_w^j > 0$, has more than one non-dominated path cost vector $C_r^j(f)$ for all $f \in \mathcal{K}$. Now, for user class j and for every pair of efficient paths r, s with $C_r^j(f) \neq C_s^j(f)$, there exist two indices $i_1, i_2 \in \{1, \ldots, p\}$ such that

$$C_r^j(f)_{i_1} < C_s^j(f)_{i_1} \text{ and } C_r^j(f)_{i_2} > C_s^j(f)_{i_2}. \tag{4.23}$$

From (4.23) and (4.22) we can conclude that

$$C_s^j(f)_{i_1} > C_r^j(f)_{i_1} \Rightarrow h_s^j = 0$$
$$C_r^j(f)_{i_2} > C_s^j(f)_{i_2} \Rightarrow h_r^j = 0.$$

Repeating this process for every pair of efficient paths r, s with $C_r^j(f) \neq C_s^j(f)$, the flow on all those paths must become zero to satisfy (4.22). But then, demand $d_w^j > 0$ is not satisfied. Therefore, there exists no feasible solution satisfying (4.22) whenever there is more than one efficient path per OD pair w and class j.

This completes our discussion of literature on VEQ and related problems. In Sections 4.4.3 and 4.4.4 we will collect some known facts on the relationship between VEQ and VOP as well as VEQ and VVI and present some new insights that help understand what sets VEQ apart from VOP and VVI. Although solution algorithms of the single-objective TA problem take advantage of equivalent optimisation and VI formulation, this equivalence does not hold

in the multi-objective case, which will be demonstrated in the two subsequent sections.

4.4.3 Relationships between VOP and VEQ

Before the relation of VOP and VEQ is discussed, we need to introduce the term *properly efficient* solutions for a VOP problem with objective vector $z(\overline{f}) = (z_1(\overline{f}), \ldots, z_p(\overline{f}))$.

Definition 4.4.10 (Geoffrion 1968). A feasible solution $\overline{f}^*$ is called *properly efficient*, if it is efficient and if there is a real number $M > 0$ such that for all i and $\overline{f} \in \mathcal{K}_\mathcal{A}$ for which $z_i(\overline{f}) < z_i(\overline{f}^*)$, there exists an index j such that

$$\frac{z_i(\overline{f}^*) - z_i(\overline{f})}{z_j(\overline{f}) - z_j(\overline{f^*})} \leqq M. \tag{4.24}$$

Properly efficient solutions are efficient solutions with bounded trade-offs between the objectives. Properly efficient solutions can be obtained as optimal solutions of minimisation problems with weighted sum objective, given that all weights are positive and the functions and underlying feasible set are convex (Geoffrion 1968, Theorem 2).

Theorem 4.4.3 *Every properly efficient solution of the multi-objective optimisation problem VOP with convex objectives* $z_i = \sum_{a\in\mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a^i(v)dv, i = 1, \ldots, p$ *based on positive and continuous functions* $\overline{c}_a^i$*, is a VEQ solution.*

Proof For every properly efficient solution $\overline{f}^*$ there exist positive weights $\omega_1, \ldots, \omega_p$ such that $\overline{f}^*$ is optimal for the single-objective optimisation problem

$$\begin{aligned} \min \quad & \omega_1 \sum_{a\in\mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a^1(v)dv + \ldots + \omega_p \sum_{a\in\mathcal{A}} \int_0^{\overline{f}_a} \overline{c}_a^p(v)dv \\ \text{s.t.} \quad & \overline{f} \in \mathcal{K}_\mathcal{A}. \end{aligned} \tag{4.25}$$

Problem (4.25) can be re-written as

$$\begin{aligned} \min \quad & \sum_{a\in\mathcal{A}} \int_0^{\overline{f}_a} (\omega_1 \overline{c}_a^1(v) + \ldots + \omega_p \overline{c}_a^p(v))\, dv. \\ \text{s.t.} \quad & \overline{f} \in \mathcal{K}_{\mathcal{A}}. \end{aligned}$$

This is the equivalent optimisation formulation corresponding to a standard SEQ problem with arc cost function $\omega_1\overline{c}_a^1(\overline{f})+\ldots+\omega_p\overline{c}_a^p(\overline{f})$. This cost function is positive and continuous as all its components $\overline{c}_a^p$ are positive and continuous, and the weights are positive. At equilibrium all used paths for any OD pair w have the same minimal generalised cost value, η_w, see also (4.4):

$$\begin{aligned} \omega_1 c_r^1(f^*) + \ldots + \omega_p c_r^p(f^*) &= \eta_w \quad \text{if } f_r^* > 0, \\ \omega_1 c_r^1(f^*) + \ldots + \omega_p c_r^p(f^*) &\geqq \eta_w \quad \text{if } f_r^* = 0. \end{aligned} \tag{4.26}$$

If we now assume that there exists a path $s \in \mathcal{R}_w$ with positive flow $f_s^* > 0$ that is dominated by another path $t \in \mathcal{R}_w$, then

$$c_t^1(\overline{f}^*) \leqq c_s^1(\overline{f}^*),\ c_t^2(\overline{f}^*) \leqq c_s^2(\overline{f}^*),\ \ldots,\ c_t^p(\overline{f}^*) \leqq c_s^p(\overline{f}^*)$$

holds with at least one strict inequality. From this and $\omega_i > 0$ it follows that

$$\omega_1\overline{c}_t^1(\overline{f}^*) + \ldots + \omega_p\overline{c}_t^p(\overline{f}^*) < \omega_1\overline{c}_s^1(\overline{f}^*) + \ldots + \omega_p\overline{c}_s^p(\overline{f}^*),$$

which contradicts (4.26), and therefore $\overline{f}^*$ satisfies VEQ. □

Remark 4.4.1 Let Ω be an open subset of $\mathbb{R}^m$ and h a function differentiable on Ω. Furthermore, $\mathcal{K}$ is a convex subset of Ω. Then, h is convex on $\mathcal{K}$ if and only if its gradient ∇h is *monotone*, i.e. satisfies

$$\left(\nabla h(x) - \nabla h(y)\right)^\top (x - y) \geqq 0, \tag{4.27}$$

for all $x, y \in \mathcal{K}$ (Hiriart-Urruty and Lemaréchal 2001, Theorem 4.1.4). Therefore, Theorem 4.4.3 is valid assuming that each cost function $\overline{c}_a^i$ is positive, continuous, monotone, and separable as this implies that the objectives z_i are convex functions. A similar theorem appears in Goh and Yang (1999).

Remark 4.4.2 A solution with one or more of the weights equal to zero does not necessarily satisfy VEQ. If, for example, the only two objectives are time $\overline{t}$ and cost $\overline{m}$, then the solution of SEQ with cost function $\omega_1\overline{t} + 0 \cdot \overline{m}$ is not a solution that satisfies BEQ. At equilibrium, all used paths for an OD pair will have the same travel time, but if the costs for those paths differ from each other, then BEQ is not satisfied. In BEQ, there should only be flow on the path with cheapest cost as this path dominates the other paths with same travel time.

Next, we show that the reverse of Theorem 4.4.3 is not true, even for convex functions z_i. We give an example in which there exists a solution of BEQ, that cannot be obtained as solutions of a BOP problem even though the objectives of BOP are convex. We again consider Example 4.4.2. Here solution

$$f^* = (0, 607, 393)^\top \text{ with } C(f^*) = \begin{pmatrix} 19 & 35.9 & 36 \\ 20 & 15 & 0 \end{pmatrix}$$

satisfies BEQ. But there exists a solution h that dominates solution f^* in BOP:

$$h = (117.5, 450, 432.5)^\top \text{ with } C(h) = \begin{pmatrix} 19.3 & 24.8 & 41.2 \\ 20 & 15 & 0 \end{pmatrix}.$$

The BEQ solution f^* is not an efficient solution of BOP (4.6), as the objective vector $z(f^*)$ of BOP is dominated by $z(h)$:

$$z(h) = \begin{pmatrix} 22825.4 \\ 9100 \end{pmatrix} \leq \begin{pmatrix} 24764.4 \\ 9105 \end{pmatrix} = z(f^*).$$

In the next section, we explore why the problems VVI and VEQ are significantly different by characterising solutions of VVI and showing which solutions of VEQ cannot be obtained by a VVI.

4.4.4 Relationships between VVI and VEQ

It is well-known that a solution of VVI also satisfies VEQ, which we repeat here. Equivalence of VVI and VEQ is established under very strong assumptions, namely that for each OD pair all efficient paths have the same path cost vector. We then characterise which properties of VEQ solutions prohibit them from satisfying VVI. This answers the question whether solving VVI can help us understand and solve VEQ. Then, we give assumptions which are weaker than those previously made (in the literature), that guarantee that a solution of VEQ also solves VVI.

It is well-known that every solution of VVI is a solution of VEQ, as the following theorem confirms. However, the reverse is not true in general.

Theorem 4.4.4 *(Yang and Goh 1997) If $f^* \in \mathcal{K}$ is a solution of VVI, then f^* is also a solution of VEQ.*

Proof (similar to Yang and Goh 1997) Assume f^* satisfies VVI but not VEQ. Then there exists some $w \in \mathcal{W}$ with $r \in \mathcal{R}_w, f_r^* > 0$ so that the path cost vector $C_{\cdot r}(f^*)$ is dominated by the path cost vector of some path $s \in \mathcal{R}_w$, i.e.

$$C_{\cdot s}(f^*) \leq C_{\cdot r}(f^*) \text{ or } C_{\cdot s}(f^*) - C_{\cdot r}(f^*) \leq 0. \tag{4.28}$$

Choose solution h as

$$h_t = \begin{cases} f_t^* & \text{if } t \neq r, s \\ 0 & \text{if } t = r \\ f_s^* + f_r^* & \text{if } t = s \end{cases}$$

It follows that $h \in \mathcal{K}$ as demand for OD pair w is still satisfied. As f^* satisfies VVI:

$$\begin{aligned} C(f^*)(h - f^*) &= \sum_{w \in \mathcal{W}} \sum_{t \in \mathcal{R}_w} C_{\cdot t}(f^*)(h_t - f_t^*) \\ &= C_{\cdot r}(f^*)(h_r - f_r^*) + C_{\cdot s}(f^*)(h_s - f_s^*) \\ &= \underbrace{f_r^*}_{>0} \underbrace{(C_{\cdot s}(f^*) - C_{\cdot r}(f^*))}_{\leq 0 \text{ by } (4.28)} \leq 0. \end{aligned}$$

The latter implies that f^* does not satisfy VVI, a contradiction.

□

Every solution of VVI is a solution of VEQ. Chen and Yen (1993) show the following equivalence provided that the set $\mathcal{Z}_N^w(f)$ is singleton, i.e. $|\mathcal{Z}_N^w(f)| = 1$, for all $w \in \mathcal{W}$.

Theorem 4.4.5 *(attributed to Chen and Yen 1993) Let $\mathcal{Z}_N^w(f)$ be singleton for all $w \in \mathcal{W}$. If $f \in \mathcal{K}$ satisfies VEQ, then f satisfies the following modified VVI:*

$$C(f)(h - f) \ngeq 0, \forall h \in \mathcal{K}. \tag{4.29}$$

Therefore, equivalence of the modified VVI in (4.29) and VEQ is established. Unfortunately, the singleton assumption renders this equivalence worthless. If for each OD pair there is always a single path that is optimal for each of the two or more objectives, we do not have to consider a multi-objective problem at all. At the end of this section we give a variation of Theorem 4.4.5 that has weaker assumptions for a solution of VEQ also being a solution of VVI.

Lee et al. (1998) show how the sets of solutions of VVI, WVVI, and VI_ξ are related. They give a proof of the following theorem, where the set of solutions of VVI (WVVI, VI_ξ) is denoted by $sol(\text{VVI})$ ($sol(\text{WVVI})$, $sol(\text{VI}_\xi)$):

Theorem 4.4.6 *The following properties hold:*

$$\bigcup_{\xi \in \mathbb{R}^p_>} sol(\,VI_\xi) \subset sol(\,VVI) \subset sol(\,WVVI) = \bigcup_{\xi \in \mathbb{R}^p_\geq} sol(\,VI_\xi)$$

Theorem 4.4.6 shows that all solutions of VVI can be obtained by applying a weighted sum scalarisation of the objectives $\sum_{i=1}^p \xi_i C_{i\cdot}(f)$. It appears that this gives a first indication that VVI is not suitable to solve VEQ. We emphasised previously that the definition of VEQ does not make any behavioural assumptions such as combining the different objectives into a weighted sum generalised cost function. This indicates that path cost vectors $C_{\cdot r}$ that are not optimal for a generalised cost function with some weighting factors cannot be included (i.e. have positive f_r) into a solution of VVI. We confirm this

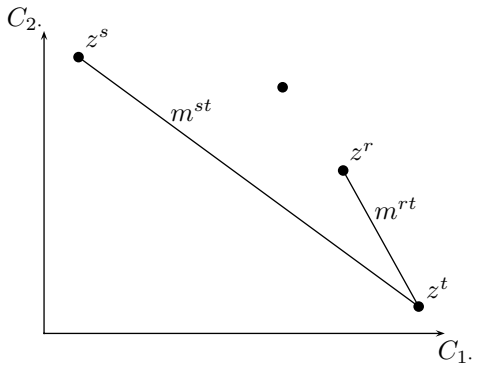


Figure 4.4. Points z^s and z^t are supported, z^r is one of the non-supported points "between" them.

conjecture in the following.

Theorem 4.4.6 shows that $sol(\text{VVI})$ is a subset of the solutions of WVVI, which in turn can be obtained via $sol(\text{VI})_\xi$ when ξ varies over $\xi \in \mathbb{R}^p_{\geqq} \setminus \{0\}$. This indicates that VVI and WVVI do not permit solutions in which for some $w \in \mathcal{W}$ one of the efficient paths is *non-supported*. We first give a proof of this result for $p = 2$ and then extend it to $p \geqq 2$.

Theorem 4.4.7 *Assume $p = 2$, f is a solution of VEQ and there exists $w \in \mathcal{W}$ such that the set of non-dominated points for w, $\mathcal{Z}_N^w(f)$, contains at least one* non-supported *point z^r with corresponding path $r \in R^w$ such that $z^r = C_{\cdot r}(f)$ and $f_r > 0$ ($C_{\cdot r}(f)$ denotes column r of the path cost matrix). Then f does not solve VVI.*

Proof As indicated in Figure 4.4, we choose path indices s and t to define the neighbouring supported points of $z^r = (C_{1r}(f), C_{2r}(f))$. Index s is chosen so that $z^s = (C_{1s}(f), C_{2s}(f))$ is a *supported point* in $\mathcal{Z}_N^w(f)$ with maximal value $C_{1s}(f) < C_{1r}(f)$. Similarly, index t is chosen so that $z^t = (C_{1t}(f), C_{2t}(f))$ is a *supported point* in $\mathcal{Z}_N^w(f)$ with minimal value $C_{1t}(f) > C_{1r}(f)$.

As z^r is a non-supported non-dominated point and z^s, z^t are neighbouring

supported non-dominated points, we have

$$C_{1s}(f) < C_{1r}(f) < C_{1t}(f) \text{ and } C_{2s}(f) > C_{2r}(f) > C_{2t}(f) \tag{4.30}$$

and for slopes m^{st} and m^{rt} we have $m^{rt} < m^{st}$:

$$m^{st} = \frac{C_{2s}(f) - C_{2t}(f)}{C_{1s}(f) - C_{1t}(f)} > \frac{C_{2r}(f) - C_{2t}(f)}{C_{1r}(f) - C_{1t}(f)} = m^{rt}. \tag{4.31}$$

We proceed by constructing $h \in \mathcal{K}$ such that $C(f)(h-f) \in -\mathbb{R}^2_{\geq}$. Let path-flow vector h be given as

$$\begin{aligned} h_u &= f_u \text{ for } u \neq r, s, t \\ h_s &= f_s + a\mu \\ h_r &= f_r - \mu \\ h_t &= f_t + (1-a)\mu, \end{aligned} \tag{4.32}$$

with $0 < a < 1$ and $0 < \mu \leqq f_r$. As $h_u - f_u = 0$ for $u \neq r, s, t$ the LHS of VVI reduces to

$$\begin{aligned} C(f)(h-f) &= \begin{pmatrix} C_{1s}(f)(h_s - f_s) + C_{1r}(f)(h_r - f_r) + C_{1t}(f)(h_t - f_t) \\ C_{2s}(f)(h_s - f_s) + C_{2r}(f)(h_r - f_r) + C_{2t}(f)(h_t - f_t) \end{pmatrix} \\ &= \begin{pmatrix} a\mu C_{1s}(f) - \mu C_{1r}(f) + (1-a)\mu C_{1t}(f) \\ a\mu C_{2s}(f) - \mu C_{2r}(f) + (1-a)\mu C_{2t}(f) \end{pmatrix}. \end{aligned}$$

To show that f does not satisfy VVI, i.e. $C(f)(h-f) \in -\mathbb{R}^2_{\geq}$, it suffices to show that a and μ exist so that

$$a\mu C_{1s}(f) - \mu C_{1r}(f) + (1-a)\mu C_{1t}(f) = 0 \tag{4.33}$$

$$\text{and} \quad a\mu C_{2s}(f) - \mu C_{2r}(f) + (1-a)\mu C_{2t}(f) < 0. \tag{4.34}$$

From (4.33) $a = \frac{C_{1r}(f) - C_{1t}(f)}{C_{1s}(f) - C_{1t}(f)}$ follows and (4.30) implies that $0 < a < 1$. With this choice of a, it remains to verify that (4.34) holds. For the LHS of (4.34),

we have

$$\begin{aligned}
& aC_{2s}(f) - C_{2r}(f) + (1-a)C_{2t}(f) \\
= \; & a(C_{2s}(f) - C_{2t}(f)) + (C_{2t}(f) - C_{2r}(f)) \\
= \; & \frac{C_{1r}(f) - C_{1t}(f)}{C_{1s}(f) - C_{1t}(f)}(C_{2s}(f) - C_{2t}(f)) + (C_{2t}(f) - C_{2r}(f)) \\
= \; & \underbrace{\underbrace{\frac{C_{2s}(f) - C_{2t}(f)}{C_{1s}(f) - C_{1t}(f)}}_{<0} \underbrace{(C_{1r}(f) - C_{1t}(f))}_{<0} + (C_{2t}(f) - C_{2r}(f))}_{>0} \\
= \; & \left|\frac{C_{2s}(f) - C_{2t}(f)}{C_{1s}(f) - C_{1t}(f)}\right| |C_{1r}(f) - C_{1t}(f)| + (C_{2t}(f) - C_{2r}(f)) \\
< \; & \frac{C_{2r}(f) - C_{2t}(f)}{C_{1r}(f) - C_{1t}(f)}(C_{1r}(f) - C_{1t}(f)) + (C_{2t}(f) - C_{2r}(f)) = 0,
\end{aligned}$$

as (4.31) implies $\left|\frac{C_{2s}(f)-C_{2t}(f)}{C_{1s}(f)-C_{1t}(f)}\right| < \left|\frac{C_{2r}(f)-C_{2t}(f)}{C_{1r}(f)-C_{1t}(f)}\right|$. Therefore (4.34) is true when choosing $a = \frac{C_{1r}(f)-C_{1t}(f)}{C_{1s}(f)-C_{1t}(f)}$, and any $0 < \mu \leqq f_r$, we obtain a feasible $h \in \mathcal{K}$ as defined in (4.32) such that (4.33) and (4.34) hold, therefore f does not satisfy VVI. □

Even when there are only supported objective vectors in a solution of VEQ, this solution might not be a solution of VVI as we show next.

Theorem 4.4.8 *Assume $p = 2$, f is a solution of VEQ and there exists $w \in \mathcal{W}$ such that the set of non-dominated points for w, $\mathcal{Z}_N^w(f)$, contains at least three* supported *points z^r, z^s, z^t that are not optimal for a weighted sum problem with the* same *weighting factor. Furthermore, we assume for the paths $r, s, t \in \mathcal{R}^w$ that $z^u = C_{\cdot u}(f)$ and $f_u > 0, u = r, s, t$ ($C_{\cdot u}(f)$ denotes column u of the path cost matrix). Then f does not solve VVI.*

Proof The proof is similar to that of Theorem 4.4.7 above. The situation is illustrated in Figure 4.5.

Again, we have relationship (4.30), i.e. supported non-dominated solutions are in some order. For the slopes we have $m^{st} < m^{rt}$ and therefore

$$m^{st} = \frac{C_{2t}(f) - C_{2s}(f)}{C_{1t}(f) - C_{1s}(f)} < \frac{C_{2t}(f) - C_{2r}(f)}{C_{1t}(f) - C_{1r}(f)} = m^{rt}. \tag{4.35}$$

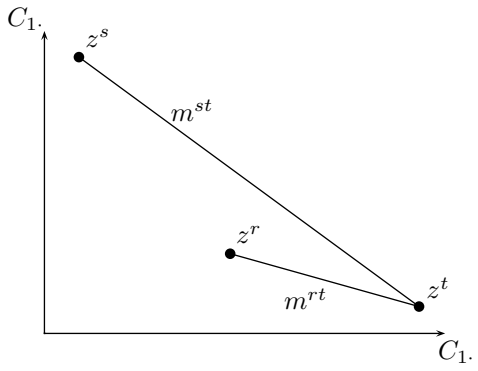


Figure 4.5. Points z^s, z^t, and z^r are supported, but do not lie on the same face of $bd(conv(\mathcal{Z}_N^w(f)))$.

We proceed by constructing $h \in \mathcal{K}$ such that $C(f)(h-f) \in -\mathbb{R}^2_{\geq}$. Let path-flow vector h be given as

$$\begin{aligned} h_u &= f_u \text{ for } u \neq r, s, t \\ h_s &= f_s - a\mu \\ h_r &= f_r + \mu \\ h_t &= f_t - (1-a)\mu, \end{aligned} \tag{4.36}$$

with appropriate choice of $0 < a < 1$ and $\mu > 0$, which we will comment on later. As $h_u - f_u = 0$ for $u \neq r, s, t$ the LHS of VVI reduces to

$$C(f)(h-f) = \begin{pmatrix} -a\mu C_{1s}(f) + \mu C_{1r}(f) - (1-a)\mu C_{1t}(f) \\ -a\mu C_{2s}(f) + \mu C_{2r}(f) - (1-a)\mu C_{2t}(f) \end{pmatrix}.$$

To show that f does not satisfy VVI, i.e. $C(f)(h-f) \in -\mathbb{R}^2_{\geq}$, it suffices to show that a and μ exist so that

$$-a\mu C_{1s}(f) + \mu C_{1r}(f) - (1-a)\mu C_{1t}(f) = 0 \tag{4.37}$$

$$\text{and} \quad -a\mu C_{2s}(f) + \mu C_{2r}(f) - (1-a)\mu C_{2t}(f) < 0. \tag{4.38}$$

From (4.37) we conclude $a = \frac{C_{1t}(f) - C_{1r}(f)}{C_{1t}(f) - C_{1s}(f)}$ and $0 < a < 1$ by (4.30). It remains

to verify that (4.38) holds for this choice of a:

$$\begin{aligned}
& -aC_{2s}(f) + C_{2r}(f) - (1-a)C_{2t}(f) \\
= \;& a(C_{2t}(f) - C_{2s}(f)) + (C_{2r}(f) - C_{2t}(f)) \\
= \;& \frac{C_{1t}(f) - C_{1r}(f)}{C_{1t}(f) - C_{1s}(f)}(C_{2t}(f) - C_{2s}(f)) + (C_{2r}(f) - C_{2t}(f)) \\
= \;& \frac{C_{2t}(f) - C_{2s}(f)}{C_{1t}(f) - C_{1s}(f)}(C_{1t}(f) - C_{1r}(f)) + (C_{2r}(f) - C_{2t}(f)) \\
\overset{(4.35)}{<} \;& \frac{C_{2t}(f) - C_{2r}(f)}{C_{1t}(f) - C_{1r}(f)}(C_{1t}(f) - C_{1r}(f)) + (C_{2r}(f) - C_{2t}(f)) = 0
\end{aligned}$$

Therefore (4.38) is true when choosing $a = \frac{C_{1t}(f)-C_{1r}(f)}{C_{1t}(f)-C_{1s}(f)}$. It remains to choose μ so that h is feasible. As $f_s, f_t > 0$, by choosing $\mu \leqq \min\left\{\frac{f_s}{a}, \frac{f_t}{1-a}\right\}$, we obtain a feasible $h \in \mathcal{K}$ as defined in (4.36) such that (4.37) and (4.38) hold, therefore f does not satisfy VVI. □

Solutions of VEQ that satisfy the assumptions of Theorems 4.4.7 and 4.4.8 do exist. For instance, in Example 4.4.2 we consider a network in which the first cost function is the travel time, which increases with traffic flow. The second cost function, however, is given by toll cost, which we assume to be independent of traffic flow, so it remains constant. Here, non-supported solutions may occur as well as more than two supported solutions, which are not optimal for the same weighting factors.

In Example 4.4.2 the solution $\hat{f}$ has three supported points, whereas solution f^* has a non-supported point. Clearly both solutions satisfy VEQ as there is only flow on efficient paths. This is illustrated in Figure 4.6.

$$\hat{f} = (300, 300, 400)^\top \text{ with } C(\hat{f}) = \begin{pmatrix} 17.59 & 20.95 & 36.85 \\ 20 & 15 & 0 \end{pmatrix}$$

$$f^* = (300, 400, 300)^\top \text{ with } C(f^*) = \begin{pmatrix} 17.59 & 23.00 & 28.75 \\ 20 & 15 & 0 \end{pmatrix}$$

In the following Theorems 4.4.7 and 4.4.8 are extended for $p > 2$.

Theorem 4.4.9 *Assume $p \geqq 2$, f is a solution of VEQ and there exists $w \in \mathcal{W}$ such that the set of non-dominated points for w, $\mathcal{Z}_N^w(f)$, contains at least*

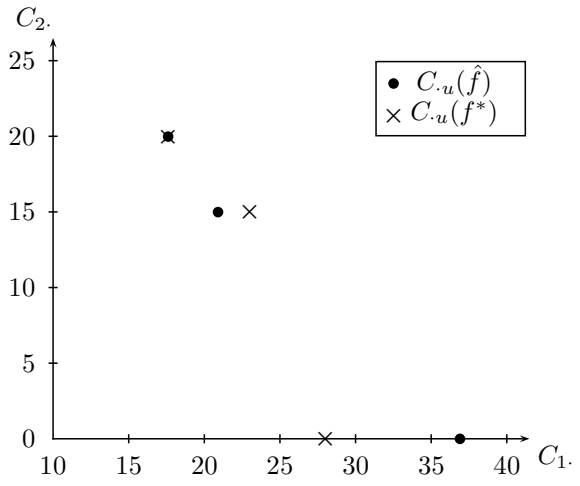


Figure 4.6. Example for solutions of VEQ: objective vectors of all three different paths of solution $\hat{f}$ and f^*.

one non-supported *point* z^r *with corresponding path* $r \in \mathcal{R}^w$ *such that* $z^r = C_{\cdot r}(f)$ *and* $f_r > 0$ *(*$C_{\cdot r}(f)$ *denotes column* r *of the path cost matrix). Then* f *does not solve VVI.*

Proof We know that $conv(\mathcal{Z}_N^w(f))$ is a polyhedron. All supported solutions lie on the boundary of $conv(\mathcal{Z}_N^w(f)+\mathbb{R}^p_{\leqq})$, whereas the non-supported solutions lie in the interior of $conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq})$. We need to establish that the non-supported solution z^r is dominated by (at least) one point on a face of the polyhedron. This point on the face is, however, infeasible for the problem of finding shortest paths with given costs $C(f)$.

The set $\mathcal{Z}_N^w(f)$ is discrete and finite, therefore $conv(\mathcal{Z}_N^w(f))$ is a compact set. A set $\mathcal{Z}$ is called $\mathbb{R}^p_{\leqq}$-*compact* if for all $z \in \mathcal{Z}$ the section $(z - \mathbb{R}^p_{\leqq}) \cap \mathcal{Z}$ is compact (Ehrgott 2005, Definition 2.13). The set $\mathcal{Z}_N$ of all non-dominated points of $\mathcal{Z}$ is called *externally stable* if for each $z \in \mathcal{Z} \setminus \mathcal{Z}_N$ there is $\hat{z} \in \mathcal{Z}_N$ such that $z \in \hat{z} + \mathbb{R}^p_{\leqq}$, i.e. for every non-dominated point z there is always a point $\hat{z}$ that dominates z (Ehrgott 2005, Definition 2.20).

Clearly, $conv(\mathcal{Z}_N^w(f))$ is a $\mathbb{R}^p_{\leqq}$-compact set. As $conv(\mathcal{Z}_N^w(f)) \subset \mathbb{R}^p_{\leqq}$ is nonempty and $\mathbb{R}^p_{\leqq}$-compact, Theorem 2.21 in Ehrgott (2005) implies that $conv(\mathcal{Z}_N^w(f))$ is externally stable.

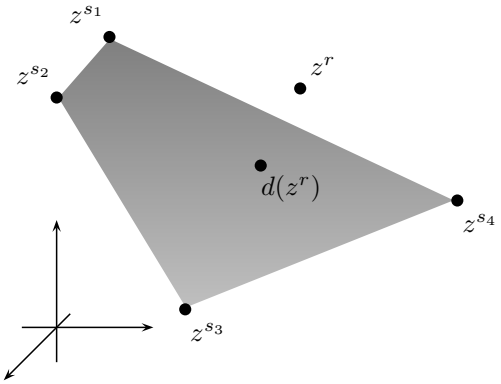


Figure 4.7. Illustration of the situation described in Theorem 4.4.9 for $p = 3$.

Therefore, the point z^r is dominated by some point on one of the faces of $conv(\mathcal{Z}_N^w(f))$, we call this face the *face associated* with z^r and we denote by $d(z^r)$ the point that dominates z^r. The distance between $d(z^r)$ and z^r is $\|d(z^r) - z^r\| > 0$ as z^r is non-supported. Every face is defined by a set of supported extreme points $z^{s_1} = C_{\cdot s_1}(f), \ldots, z^{s_q} = C_{\cdot s_q}(f) \in \mathcal{Z}_N^w(f)$ with $q \geqq 2$. The point $d(z^r)$ can be obtained as a convex combination of $z^{s_1}, \ldots, z^{s_q}$. For an illustration in case $p = 3$, refer to Figure 4.7.

Along the lines of the proof of Theorem 4.4.7, we construct a feasible solution $h \neq f$ so that the VVI condition is violated. Choose

$$\begin{aligned} h_u &= f_u \text{ for } u \neq s_1, \ldots, s_q, r \\ h_{s_1} &= f_{s_1} + a_1\mu \\ \vdots &= \vdots \\ h_{s_q} &= f_{s_q} + a_q\mu \\ h_r &= f_r - \mu, \end{aligned} \tag{4.39}$$

for some $0 \leqq a_1, \ldots, a_q$ with $\sum_{i=1,\ldots,q} a_i = 1$ and $0 < \mu \leqq f_r$. Solution h is feasible as for OD pair w a flow of μ is removed from path r and re-distributed to paths $s_1, \ldots, s_q$. As $h_u - f_u = 0$ for $u \neq s_1, \ldots, s_q, r$ the LHS of VVI reduces

to

$$\begin{aligned} C(f)(h-f) &= \begin{pmatrix} a_1\mu C_{1s_1}(f) + \ldots + a_q\mu C_{1s_q}(f) - \mu C_{1r}(f) \\ a_1\mu C_{2s_1}(f) + \ldots + a_q\mu C_{2s_q}(f) - \mu C_{2r}(f) \\ \vdots \\ a_1\mu C_{ps_1}(f) + \ldots + a_q\mu C_{ps_q}(f) - \mu C_{pr}(f) \end{pmatrix} \\ &= \mu\left(a_1 z^{s_1} + \ldots + a_q z^{s_q} - z^r\right) \end{aligned}$$

As the point $d(z^r)$ can be expressed as a convex combination of $z^{s_1}, \ldots, z^{s_q}$, factors $0 \leqq a_1, \ldots, a_q \leqq 1$ can be chosen so that $d(z^r) = a_1 z^{s_1} + \ldots + a_q z^{s_q}$ and $\sum_{i=1,\ldots,q} a_i = 1$, which simplifies the previous equation to

$$C(f)(h-f) = \mu\left(d(z^r) - z^r\right). \tag{4.40}$$

The vector $d = z^r - d(z^r)$ is in $\mathbb{R}^p_{\geq}$ as z^r is dominated by $d(z^r)$. Replacing z^r by $d(z^r) + d$ in (4.40) yields

$$\begin{aligned} C(f)(h-f) &= \mu\left(d(z^r) - (d(z^r) + d)\right) \\ &= -\underbrace{\mu}_{>0}\underbrace{d}_{\in\mathbb{R}^p_{\geq}} \in -\mathbb{R}^p_{\geq}. \end{aligned}$$

□

Theorem 4.4.10 *Assume $p \geqq 2$, f is a solution of VEQ and there exists $w \in \mathcal{W}$ such that the set of non-dominated points for w, $\mathcal{Z}^w_N(f)$, contains at least $p+1$* extreme supported *points $z^{s1}, \ldots, z^{s_p}, z^r$ with corresponding paths $s_1, \ldots, s_p, r \in \mathcal{R}^w$ such that $z^u = C_{\cdot u}(f)$ and $f_u > 0, u = s_1, \ldots, s_p, r$ (by $C_{\cdot u}(f)$ we mean column u of the path cost matrix). Furthermore, assume the objective vectors $z^{s_1}, \ldots, z^{s_p}$ lie on the same facet of $conv(\mathcal{Z}^w_N)$ and all points are optimal for* exactly one *common weighting factor $\xi \in \Lambda$ (there may exist weighting factors for which some of the vectors are optimal, but only a unique common one). If the solution z^r is not optimal for this weighting factor ξ, then f does not solve VVI.*

Proof The objective vectors $z^{s_1}, \ldots, z^{s_p}$ lie on the same facet $\mathcal{F}$ of $conv(\mathcal{Z}^w_N)$ and all are optimal for exactly one common weighting factor $\xi \in \Lambda$. Therefore,

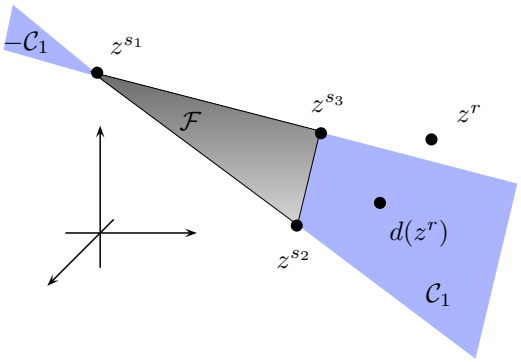


Figure 4.8. Illustration of the situation described in Theorem 4.4.10 for $p = 3$.

we have $\xi^\top z^{s_1} = \ldots = \xi^\top z^{s_p}$ and $\xi^\top z^r > \xi^\top z^{s_i}, i = 1, \ldots, p$.

The points $z^{s_1}, \ldots, z^{s_p}$ generate a hyperplane $\mathcal{H}$ of dimension $p-1$ in $\mathbb{R}^p$ (as $\xi \in \Lambda$ with $\xi^\top z^{s_1} = \ldots = \xi^\top z^{s_p}$ is unique). The point z^r is dominated by a point $d(z^r)$ in $\mathcal{H}$, but not in $\mathcal{F}$ (as z^r is supported). As the point z^r does not lie in the hyperplane, It follows that there exists $d \in \mathbb{R}^p_{\geq}$ such that $z^r = d(z^r) + d$.

A $p-1$ dimensional simplex is obtained by $conv(\{z^{s_1}, \ldots, z^{s_p}\})$. The point $d(z^r)$ does not lie within this simplex (as $\xi^\top z^r > \xi^\top z^{s_i}, i = 1, \ldots, p$), i.e. it cannot be obtained as a convex combination of $z^{s_1}, \ldots, z^{s_p}$. We choose one of the vertices z^{s_i} of the simplex and then obtain a cone $\mathcal{C}_i = \{z \in \mathbb{R}^p : z = z^{s_i} + a_1(z^{s_1} - z^{s_i}) + \ldots + a_{i-1}(z^{s_{i-1}} - z^{s_i}) + a_{i+1}(z^{s_{i+1}} - z^{s_i}) \ldots + a_p(z^{s_p} - z^{s_i}), a_j \geqq 0\}$ with apex z^{s_i}. The corresponding opposite cone $-\mathcal{C}_i$ is obtained by only allowing coefficients $a_j \leqq 0$. Now $d(z^r)$ lies within at least one of the cones $\mathcal{C}_i, -\mathcal{C}_i, i = 1, \ldots, p$. Without loss of generality we now assume that $d(z^r) \in \mathcal{C}_1$. For an illustration of $\mathcal{C}_1$ and $-\mathcal{C}_1$ in case $p = 3$, refer to Figure 4.7. We can write

$$\begin{aligned} d(z^r) &= z^{s_1} + a_2(z^{s_2} - z^{s_1}) + \ldots + a_p(z^{s_p} - z^{s_1}) \\ &= (1 - a_2 - \ldots - a_p)z^{s_1} + a_2 z^{s_2} + \ldots + a_p z^{s_p}. \end{aligned} \tag{4.41}$$

Along the lines of the proof of Theorem 4.4.8, we construct a feasible solution

$h \neq f$ so that the VVI condition is violated. Choose

$$\begin{aligned} h_u &= f_u \text{ for } u \neq s_1, \ldots, s_p, r \\ h_r &= f_r - \mu, \\ h_{s_1} &= f_{s_1} + (1 - a_2 - \ldots - a_p)\mu \\ h_{s_2} &= f_{s_2} + a_2\mu \\ \vdots &= \vdots \\ h_{s_p} &= f_{s_p} + a_p\mu \end{aligned}$$

for some $0 \leqq a_2, \ldots, a_p$ and $0 < \mu \leqq \min\left\{f_r, \frac{f_{s_1}}{|1-a_2-\ldots-a_p|}\right\}$. Note that $a_2 + \ldots + a_p > 1$ as $d(z^r)$ does not lie within the simplex, so it cannot be obtained as a convex combination. Solution h is feasible as for OD pair w a flow of μ is removed from paths r and s_1 and re-distributed to paths $s_2, \ldots, s_p$. As $h_u - f_u = 0$ for $u \neq s_1, \ldots, s_p, r$ the LHS of VVI reduces to

$$C(f)(h - f) = \mu((1 - a_2 - \ldots - a_p)z^{s_1} + a_2 z^{s_2} + \ldots + a_p z^{s_p} - z^r),$$

where the details of this step are analogous to the corresponding step in the proof of Theorem 4.4.9. Using (4.41) and $z^r = d(z^r) + d$, we obtain

$$\begin{aligned} C(f)(h - f) &= \mu(d(z^r) - z^r) \\ &= -\underbrace{\mu}_{>0}\underbrace{d}_{\in\mathbb{R}^p_{\geq}} \in -\mathbb{R}^p_{\geq}, \end{aligned}$$

which confirms that f is not a solution of VVI. □

Remark 4.4.3 We suspect that it is possible to drop the assumption that points $z^{s_1}, \ldots, z^{s_p}$ must lie on the *same* facet in Theorem 4.4.10. We believe that it may be possible to show a version of this theorem based on the assumption that there exist $p+1$ points that are not all optimal for the same weighting factor $\xi \in \Lambda$. However, we are not able to give a proof of this situation.

To summarise, we established that VVI does not yield any solutions with positive flow on non-supported efficient paths in Theorem 4.4.9. Furthermore, solutions with positive flow on at least p efficient paths that are not optimal

for the same weighting factor (so they do not lie on the same face in objective space) cannot be obtained, see Theorem 4.4.10. VVI only provides solutions that can be obtained by using a single weighting factor (for every OD pair) and therefore has no advantage over solving the problem with a generalised cost function with single VOT, as discussed in Section 4.3.2.

We can now state under which conditions we can obtain equivalence of VEQ and VVI.

Theorem 4.4.11 *Assume that $f \in \mathcal{K}$ is a solution of VEQ. Also assume there exists $\xi \in \Lambda \cap \mathbb{R}_>$ so that for each $w \in \mathcal{W}$ among all the efficient paths $r \in \mathcal{X}_E^w(f)$ there is only positive flow on those paths r with weighted cost value $\xi^\top C_{\cdot r}(f)$ equal to $C_{\min} = \min\{\xi^\top C_{\cdot r}(f) : r \in \mathcal{X}_E^w(f)\}$, i.e. those paths with minimal weighted cost. Then f is a solution of VVI.*

Proof We have to show that f is a solution of VVI. First observe that the assumptions imply the following

$$\forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad \xi^\top C_{\cdot r}(f) < \xi^\top C_{\cdot s}(f) \Rightarrow f_s = 0.$$

Therefore, SEQ is satisfied for the single-objective cost function $\xi^\top C$. Equivalence of SEQ and VIp (Theorem 4.2.1) implies that f also satisfies

$$(\xi^\top C(f)) \cdot (h - f) \geqq 0 \quad \forall h \in \mathcal{K}.$$

The latter can be re-written as VI_ξ:

$$\left(\sum_{i=1}^{p} \xi_i C_{i\cdot}(f)\right)(h - f) \geqq 0 \quad \forall h \in \mathcal{K}.$$

As f satisfies VI_ξ with $\xi \in \mathbb{R}_>$ it follows by Theorem 4.4.6 that f is a solution of VVI. □

Remark 4.4.4 Note that ξ needs to be identical for all $w \in \mathcal{W}$. Also, the assumptions of Theorem 4.4.11 do not require that all non-dominated points $\mathcal{Z}_N^w(f)$ lie on the boundary of the convex hull $bd(conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\geqq}))$ – it is sufficient that no efficient path with objective vector lying in the interior of

$conv(\mathcal{Z}_N^w(f) + \mathbb{R}^p_{\leqq})$ has positive flow.

Remark 4.4.5 Clearly, the assumptions of Theorem 4.4.11 are weaker than those of Theorem 4.4.5, and no modification of the VVI is required. With VEQ and VVI defined as in this thesis (and throughout the related literature), it appears that Theorem 4.4.11 is the strongest link between VEQ and VVI that can be established. This is because we know from Theorem 4.4.6 that all f that satisfy VVI can be obtained by solving VI_ξ with varying parameter $\xi \in \mathbb{R}^p_{\leq}$. In particular, the value of ξ must be equal for all w.

In conclusion, we have seen that, although a VVI solution is always a VEQ solution, solutions of VVI have very limited structural properties. They cannot have non-supported efficient paths and also no efficient paths that solve different weighted sum problems. One motivation for studying VEQ is that no assumptions on a linear choice function are made. Solving VEQ by obtaining solutions of VVI only, however, implicitly makes these assumptions which renders the approach through VVI unsuitable for our purposes.

4.5 Solving Bi-objective Traffic Assignment

In this section we extend some solution algorithms that are well-known for the standard (single-objective) TA to the bi-objective case. We do not present a proof of the convergence of the given algorithms but show how to observe that an equilibrium solution has been found for the bi-objective path equilibration algorithm in Section 4.5.3. It is, in general, easy to confirm convergence of a TA equilibrium algorithm when path flow variables are used throughout the algorithm. Convergence can be checked by simply confirming that positive flow exists only on efficient paths. When only aggregate arc flow variables are used, the determination of convergence criteria is not straight-forward in the bi-objective case. For single-objective TA, the objective of the optimisation formulation (4.5) can be used to measure convergence. Unfortunately there is no obvious extension of this approach to the bi- or multi-objective case. In this section, we discuss only the case of BEQ, and comment on a possible extension to VEQ at the end of the section.

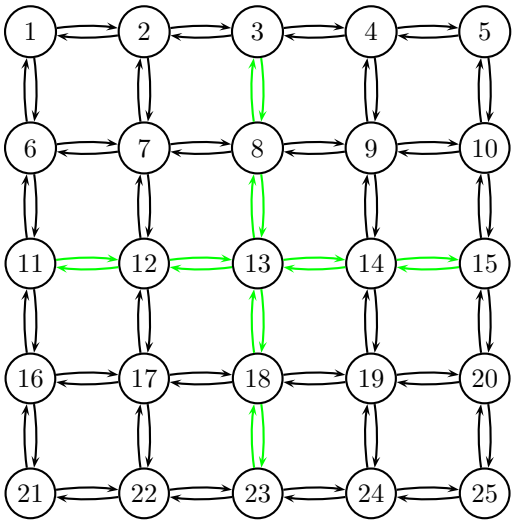


Figure 4.9. The 5×5 grid network.

The proposed heuristic algorithms are implemented in the C programming language. We use a small network to illustrate the results obtained by the different algorithms. Unless noted otherwise, results are based on performing 1000 iterations.

The network used to illustrate the output of the heuristics has a grid structure similar to the ones in Dial (1999a,b). Tests are performed on a 25-node network with grid structure as shown in Figure 4.9. The arcs are un-tolled except for the central ones highlighted green in the figure. Each tolled arc has the same toll of \$5. We consider a single OD pair $w = (1, 25)$ with a travel demand of 4000. All arcs have similar length, a value that varies around 5 km with a minimum of 4.6 km, a maximum of 5.3 km, and an average of 4.98 km. The speed limit on un-tolled roads is 30 km/h, whereas it is 55 km/h on tolled roads. Every arc a has a travel time $\bar{t}_a$ given by the so-called BPR function (Bureau of Public Roads 1964) of the form

$$\bar{t}_a = \bar{t}_a^0 \left(1 + \alpha \left(\frac{\overline{f}_a}{k_a}\right)^\beta\right), \tag{4.42}$$

where $\bar{t}_a^0$ is the *free-flow* travel time when there is no traffic on the road, and

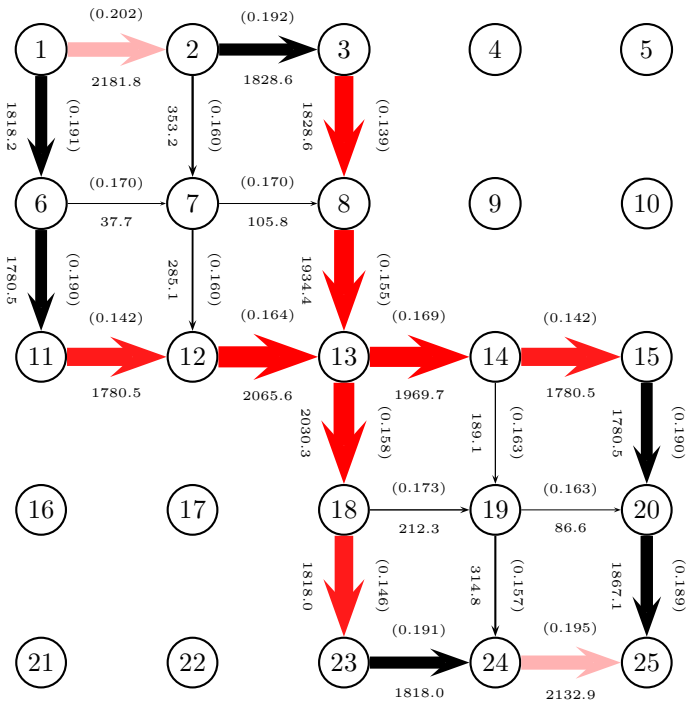


Figure 4.10. Single-objective TA: user equilibrium solution with time objective.

k_a is the *practical capacity* of the road. In our grid instance it is assumed that the practical capacity is 2000 on un-tolled roads and 1300 on tolled roads. The free flow travel time for each arc in the network is derived by dividing the arc's length by its free-flow speed. The function parameters α, β are chosen as $\alpha = 0.15, \beta = 4$.

We explain the output format based on the user equilibrium solution obtained for the TA problem with a single objective, namely travel time. The solution is obtained using the path equilibration algorithm (Algorithm 12 on page 150). Arc flow results are shown in Figure 4.10. Above each arc, in brackets, is the travel time of each arc (in hours), the toll is omitted from this figure. In the following, there will be two values in the brackets as we consider bi-objective TA problems. The first number is travel time (hours) and the second one is toll cost. Below the arc is the amount of traffic flow traversing the arc. The amount of flow is also visualised by the thickness of the arcs. All arcs missing from the figure have zero traffic flow. In transportation planning an important

measure of congestion is the ratio of arc flow and practical capacity $\frac{\overline{f}_a}{k_a}$. When this ratio exceeds 1, the arc is clearly congested, which is indicated in the figure by red arcs – the darker the red, the higher the ratio.

Figure 4.10 shows that the fastest arcs are congested as the toll is not taken into account by the solution algorithm. This leads to highly congested arcs in the centre of the network, whereas most other arcs are not used at all or receive only a small amount of flow.

4.5.1 Bi-objective Traffic Assignment and Non-linear VOT

BEQ can be solved by applying one of the solution approaches proposed for the TA problem with generalised cost function with non-linear weighting function as discussed in Section 4.3.2. We assume the first objective is a travel time objective and the second one is a fixed cost objective. We are able to establish the equivalence of BEQ and a scalar equilibrium problem SEQ with generalised cost function that has a non-linear weighting function as proposed in Larsson et al. (2002). They consider the two objectives travel time and travel cost. Path travel cost is assumed fix and converted into an equivalent time value via a non-linear weighting function v_w for each OD pair w. The arising generalised time objective has the form

$$g_r^t(f) = C_{1r}(f) + v_w(C_{2r}),$$

for $r \in \mathcal{R}_w$. They then formulate the *non-linear scalar equilibrium* according to Wardrop's principle as

$$\text{(nlSEQ)} \quad \forall w \in \mathcal{W}, \forall s \in \mathcal{R}_w \quad f_s > 0 \Rightarrow g_s^t(f) = \min_{r \in \mathcal{R}_w} \left\{ g_r^t(f) \right\},$$

which is equivalent to the formulation SEQ presented earlier. This non-linear scalar equilibrium can equivalently be written as

$$\forall w \in \mathcal{W}, \forall r, s \in \mathcal{R}_w \quad g_r^t(f) < g_s^t(f) \Rightarrow f_s = 0.$$

Under the assumptions that v_w is increasing and non-negative, and $C_{2\cdot} \geqq 0$

and fix, as well as the assumptions that the flow-dependent component $C_{1\cdot}(f)$ is positive, continuous, and separable, they establish equivalence of nlSEQ and the following optimisation problem, similar to (4.5).

$$\begin{aligned} \min \quad & \sum_{a\in\mathcal{A}} \int_0^{\overline{f}_a} C_{1a}(x)dx + \sum_{w\in\mathcal{W}} \sum_{r\in\mathcal{R}_w} v_w(C_{2r}) f_r \\ \text{s.t.} \quad & f \in \mathcal{K} \\ & \overline{f} = \Phi f. \end{aligned} \tag{4.43}$$

Next, we establish equivalence between BEQ and nlSEQ. We show that for every solution $(\overline{f}^*, f^*)$ of BEQ, we can derive a function v_w such that $(\overline{f}^*, f^*)$ also solves nlSEQ. On the other hand, we show that a solution $(\overline{f}^*, f^*)$ of nlSEQ with given v_w is also a solution of BEQ. Note that in order to establish equivalence, we need to assume that the functions v_w are *strictly* increasing. This is necessary to guarantee that there are no weakly efficient paths in the BEQ solution.

Theorem 4.5.1 *If $(\overline{f}^*, f^*)$ is a solution of BEQ with fixed second objective $C_{2\cdot} \geqq 0$, then there exists a strictly increasing, non-negative function v such that the vector $(\overline{f}^*, f^*)$ is a solution of nlSEQ.*

Proof Assume $(\overline{f}^*, f^*)$ solves BEQ. Without loss of generality, choose one OD pair $w \in \mathcal{W}$. $\mathcal{X}_E^w(f^*)$ is the set of efficient paths. There may be equivalent efficient paths with identical path cost vectors. All distinct path cost vectors can be sorted by increasing first component and they are indexed accordingly:

$$\begin{array}{ccccccc} C_{1s_{11}}(\overline{f}^*) = \ldots = C_{1s_{1m_1}}(\overline{f}^*) & < & \ldots & < & C_{1s_{n1}}(\overline{f}^*) = \ldots = C_{1s_{nm_n}}(\overline{f}^*) \\ C_{2s_{11}} = \ldots = C_{2s_{1m_1}} & > & \ldots & > & C_{2s_{n1}} = \ldots = C_{2s_{nm_n}}. \end{array} \tag{4.44}$$

BEQ implies that there can only be positive flow on efficient paths. For $(\overline{f}^*, f^*)$ to be a solution of nlSEQ, we need to construct v_w such that $g^t_{s_{11}}(f^*) = g^t_{s_{12}}(f^*) = \ldots = g^t_{s_{nm_n}}(f^*) = \min_{r\in\mathcal{R}_w}\{g^t_r(f^*)\}$. With $T > C_{1s_{n1}}(f^*)$, we can choose v_w as piecewise linear function through the following points

$$(0,0), (C_{2s_{n1}}, T - C_{1s_{n1}}(f^*)), \ldots, (C_{2s_{21}}, T - C_{1s_{21}}(f^*)), (C_{2s_{11}}, T - C_{1s_{11}}(f^*)),$$

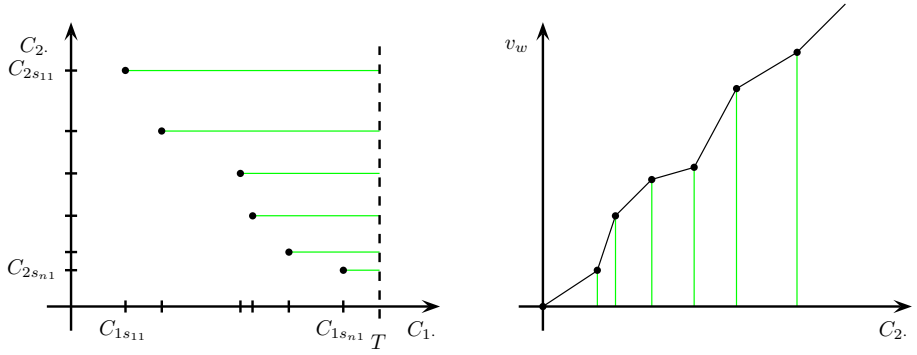


Figure 4.11. Constructing v_w from efficient solutions.

the last segment of the function goes through points $(C_{2s_{11}}, T - C_{1s_{11}}(f^*))$ and $(C_{2s_{11}} + 1, T - C_{1s_{11}}(f^*) + 1)$ to infinity. We define the function only for values $\geqq 0$ as it is assumed that $C_{2\cdot} \geqq 0$. Note that if $C_{2s_{n1}} = 0$, point (0,0) can be omitted. This function v_w is strictly increasing and non-negative. The construction of v_w is illustrated in Figure 4.11. With the above choice of v_w, we obtain $g^t_{s_{11}}(f^*) = g^t_{s_{12}}(f^*) = \ldots = g^t_{s_{nm_n}}(f^*) = T$, so all efficient paths have identical generalised costs. It remains to verify that their generalised cost is also minimal.

Choose a non-efficient path $s \in \mathcal{R}_w \setminus \mathcal{X}^w_E(f^*)$. The path cost vector of s is dominated by that of an efficient path s_{i1}. There are two cases:

- $C_{1s_{i1}}(f^*) \leqq C_{1s}(f^*)$ and $C_{2s_{i1}} < C_{2s}$. It follows that $T = g^t_{s_{i1}}(f^*) = C_{1s_{i1}}(f^*) + v_w(C_{2s_{i1}}) < C_{1s}(f^*) + v_w(C_{2s}) = g^t_s(f^*)$ as v_w strictly increasing.
- $C_{1s_{i1}}(f^*) < C_{1s}(f^*)$ and $C_{2s_{i1}} \leqq C_{2s}$. It clearly follows that $T = g^t_{s_{i1}}(f^*) < g^t_s(f^*)$.

The generalised path cost vector of any non-efficient path $s \in \mathcal{R}_w \setminus \mathcal{X}^w_E(f^*)$ always has a higher value than T, and therefore $(\overline{f}^*, f^*)$ also solves nlSEQ.

□

Theorem 4.5.2 *If the vector $(\overline{f}^*, f^*)$ is a solution of nlSEQ with fixed second objective $C_{2\cdot} \geqq 0$ and strictly increasing, non-negative function v_w then $(\overline{f}^*, f^*)$*

is a solution of BEQ.

Proof Assume $(\overline{f}^*, f^*)$ solves nlSEQ. Without loss of generality, choose one OD pair $w \in \mathcal{W}$. Let $\mathcal{R}_w^{\min}(f^*)$ be the set of paths for which g^t attains its minimum:

$$\mathcal{R}_w^{\min}(f^*) = \operatorname{argmin}_{r \in \mathcal{R}_w} \left\{ g_r^t(f^*) \right\}.$$

For all paths in $\mathcal{R}_w^{\min}(f^*)$, g^t has the same minimal value. Paths in $\mathcal{R}_w^{\min}(f^*)$ are also the only paths that may have positive flow. Whenever $C_{1s_i}(f^*) = C_{1s_j}(f^*), i \neq j$ it follows that $v_w(C_{2s_i}) = v_w(C_{2s_j})$ and therefore $C_{2s_i} = C_{2s_j}$ as v_w is strictly increasing. We assume the paths are indexed as in (4.44) with increasing first component, therefore their second component is decreasing. For BEQ to be satisfied we need to show that the paths $s_{11}, s_{12}, \ldots, s_{nm_n}$ are efficient. Assume there exists a path $s \in \mathcal{R}_w \setminus \{s_{11}, s_{12}, \ldots, s_{nm_n}\}$ that dominates path(s) $s_{i1}, \ldots, s_{im_i}$ (they all have the same objective vector). We distinguish the following two cases

- $C_{1s}(f^*) \leqq C_{1s_{i1}}(f^*)$ and $C_{2s} < C_{2s_{i1}}$. It follows that $g_s^t(f^*) = C_{1s}(f^*) + v_w(C_{2s}) < C_{1s_{i1}}(f^*) + v_w(C_{2s_{i1}}) = g_{s_{i1}}^t(f^*)$ as v_w is strictly increasing.
- $C_{1s}(f^*) < C_{1s_{i1}}(f^*)$ and $C_{2s} \leqq C_{2s_{i1}}$. Again, $g_s^t(f^*) < g_{s_{i1}}^t(f^*)$ follows.

$g_s^t(f^*) < g_{s_{i1}}^t(f^*)$ is a contradiction to all $g_{s_{ij}}^t(f^*)$ having minimal value. So a path s that dominates any of the paths $s_{11}, s_{12}, \ldots, s_{nm_n}$ does not exist. Furthermore, the paths $s_{11}, s_{12}, \ldots, s_{nm_n}$ do not dominate one another by (4.44). Therefore, $(\overline{f}^*, f^*)$ solves BEQ as there is only flow on efficient paths. □

Given such a function v_w, we can solve our bi-objective TA problem with the solution approach proposed in Larsson et al. (2002). The only difficulty is determining a good function v_w. When a BEQ solution is given, v_w can be easily derived. It is, however, unclear how to derive v_w without any a priori knowledge of the structure of the non-dominated path cost vectors one may obtain at certain levels of flow.

In the following, we propose several heuristic approaches to solve BEQ. We assume that the bi-objective cost function consists of a travel time component $\overline{t}$ and a monetary cost component $\overline{m}$ which may be fixed or flow dependent. It is also assumed that path flow is additive.

Algorithm 13 MSA method for solving BEQ

1: **input**: Graph $(\mathcal{V}, \mathcal{A})$, arc cost functions $\overline{C} = (\overline{t}, \overline{m})$, set of OD pairs $\mathcal{W}$, and demand d.
2: Calculate fixed arc costs $\overline{C}(0)$.
3: Identify efficient paths.
4: Assign flow to efficient paths yielding path flow vector $\overline{f}^0$.
5: $i = 0$
6: **while** Convergence criterion not satisfied **do**
7: Calculate fixed arc costs $\overline{C}(f^i)$.
8: Identify efficient paths.
9: Assign flow to efficient paths yielding path flow vector $\overline{h}$.
10: Compute new arc flow $\overline{f}^{i+1} = \left(1 - \frac{1}{i+1}\right) \overline{f}^i + \frac{1}{i+1} \overline{h}$.
11: $i = i + 1$
12: **end while**
13: **output**: Arc flow vector $\overline{f}$.

4.5.2 Method of Successive Averages (MSA)

MSA is a basic algorithm to solve TA, and its formulation does not require any equivalence relationship with an optimisation problem or a VI.

In order to extend the MSA method to a solution algorithm for BEQ as in Algorithm 13, we need to replace the single-objective AON assignment (where for each OD pair all flow is assigned to the minimal path) by some bi-objective counterpart. Assuming fixed arc costs (initially based on zero traffic flow), the set of efficient paths is determined using any known bi-objective shortest path algorithm as discussed in Chapter 2.

Now the question is which portion of the total demand for each OD pair to assign to each of the efficient paths. A great degree of liberty lies in this step as there are many different ways to split the total flow for an OD pair between the efficient paths. To obtain a good solution, we believe that a realistic assignment method applied in each iteration of MSA will yield an overall good solution of the TA problem. With a correctly calibrated assignment setup, it should be possible to closely model the true situation. We propose different assignment strategies in the following, but it is of course possible to develop many more reasonable strategies. This solution approach for bi-objective TA is heuristic.

Equal Share Assignment

A first assignment strategy that comes to mind is *equal share* (EQS) assignment. Here, the total demand for each OD pair is split up evenly between all efficient paths. The path flow vector $\overline{h}$ is constructed as follows in every iteration: let $r_1^w, \ldots, r_{k_w}^w$ be the efficient paths for OD pair w at the current cost level $\overline{C}(f^i)$. We also assume that the objective vectors associated with the paths are distinct. Then, for every OD pair w, h is obtained as

$$h_s = \begin{cases} 0 & \text{if } s \neq r_1^w, \ldots, r_{k_w}^w \\ \frac{d_w}{k_w} & \text{if } s \in \{r_1^w, \ldots, r_{k_w}^w\}. \end{cases}$$

Finally the arc flow vector is obtained as $\overline{h} = \Phi h$.

When there are equivalent paths among $r_1^w, \ldots, r_{k_w}^w$, they can be separated into groups of paths with same objective vector

$$t_{r_{11}^w}(f^i) = \ldots = t_{r_{1l_1}^w}(f^i) < \ldots < t_{r_{k1}^w}(f^i) = \ldots = t_{r_{kl_k}^w}(f^i)$$
$$m_{r_{11}^w}(f^i) = \ldots = m_{r_{1l_1}^w}(f^i) > \ldots > m_{r_{k1}^w}(f^i) = \ldots = m_{r_{kl_k}^w}(f^i)$$

Now each group with equal objective vector is assigned a fraction of the total demand d_w, namely $\frac{d_w}{k_w}$. The path flow vector h is obtained by splitting this between the number of equivalent paths:

$$h_s = \begin{cases} 0 & \text{if } s \neq r_{11}^w, \ldots, r_{kl_k}^w \\ \frac{1}{l_\iota} \cdot \frac{d_w}{k_w} & \text{if } s = r_{\iota\kappa}^w. \end{cases}$$

EQS has some obvious disadvantages. A main one is that the actual values of the objective vectors do not influence the assignment at all. For example, the cheapest cost path is always efficient, even though it may have excessively long travel time. In reality, this path may not be chosen but it is treated just like all other paths in the case of EQS assignment. Therefore, EQS assignment may not be realistic, particularly when efficient solutions with very large values in one objective are present (the problematic ones are often the $lex(1,2)$ and the $lex(2,1)$-best solutions).

The heuristic approach MSA with EQS assignment is applied to the grid net-

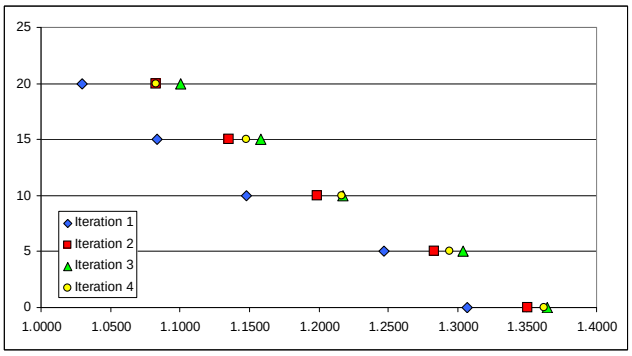


Figure 4.12. Non-dominated path cost vectors in first four iterations of MSA with EQS assignment.

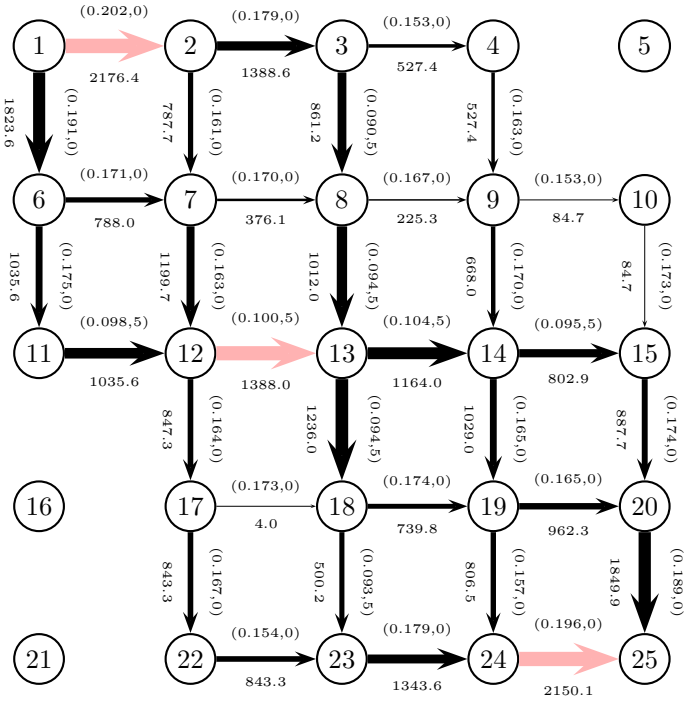


Figure 4.13. MSA with EQS assignment for 5×5 grid network.

work instance. Figure 4.12 shows the non-dominated path cost vectors obtained in the first four iterations of MSA. A fifth of the total travel demand is assigned to the paths corresponding to one of the non-dominated points in each iteration. The resulting arc flows, after 1000 iterations, are shown in Figure 4.13. It appears that it is not necessary to perform 1000 iterations. In fact, after 50 iterations, the resulting flow pattern is very similar to that observed after 1000 iterations. Although it does not seem necessary to perform 1000 iterations, we do so anyway as even 1000 iterations run quickly, in less than one second, for this small instance. As mentioned earlier, the travel time (hours) and toll ($) can be found in brackets above each arc, whereas the arc flow is below the arc.

Cost per Unit Time Saving

Another assignment method is called *cost per unit time saving* (CTS). Here, all efficient paths are compared to the cheapest one. In decision space, the slope of the lines obtained by connecting the objective vector of the cheapest path to all others indicates the cost per unit time saving – a steep slope means that the price paid for saving a unit of time is higher than when the slope is less steep. Given a distribution of CTS values, flow can be assigned to efficient paths. We explain this by means of the setup from Example 4.3.1, see Figure 4.14. The non-dominated path costs vectors z^i in the figure are associated with paths r_i.

We denote by m^{56} the slope between points z^6 and z^5, by m^{46} the slope between points z^6 and z^4, etc. Given a CTS distribution the number of users on each efficient path can be determined. Users who are willing to pay more than m^{16} to save one time unit, all choose path r_1. Users willing to pay between m^{16} and m^{26}, choose path r_2, and so forth for paths r_3, r_4, r_5. Finally, the remaining users, who are not willing to pay at least m^{56}, choose path r_6. Similarly to EQS assignment, this approach can be extended to the case when there are equivalent efficient paths that have the same path cost vector.

The slopes are not necessarily increasing as in Figure 4.14. Figure 4.15 shows an example where m^{46} is larger than m^{36} and m^{26}. Also, the efficient paths with distinct objective vectors $\tilde{z}^2$ and $\tilde{z}^3$ have the same slope, therefore they are each assigned half of the portion of network users with CTS between m^{25}

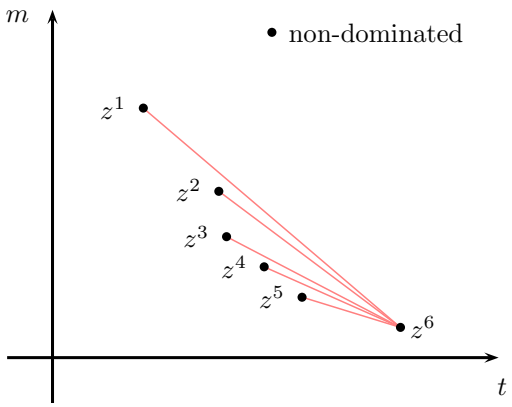


Figure 4.14. Example 4.3.1 for CTS assignment.

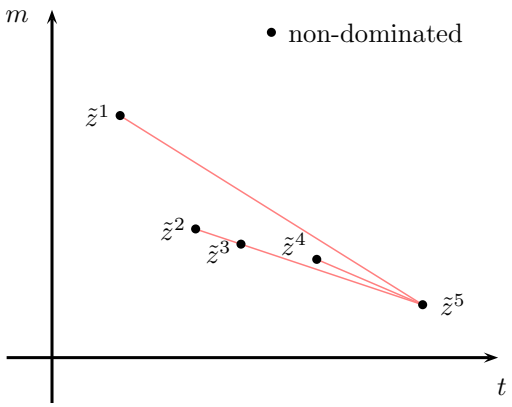


Figure 4.15. Another example for CTS assignment.

and m^{45}.

Procedure 6 outlines this assignment procedure for a given probability density function g when all efficient paths have distinct path cost vectors and also all resulting slopes are distinct. This procedure is easily adapted to deal with equivalent efficient paths and non-distinct slopes. Note that S is defined as $S(x) = \int_0^x g(v)dv$.

An advantage of CTS assignment over EQS assignment is that it does matter what the actual non-dominated path values are. If, for example, the cost of path r_1 is large, only few users choose this path. Compared to conventional weighted-sum approaches, CTS assignment enables the assignment of flow to non-supported paths (the one with cost vector z^2 in Figure 4.14) as well as non-extreme supported paths (the one with cost vector z^3 in Figure 4.14).

On the other hand, CTS does not take into account the total path cost values. It is not necessarily true that a user who is willing to pay \$0.8 to save 2 minutes of travel time, would pay \$8 to save 20 minutes and vice versa.

To demonstrate the solution algorithm we choose the share of flow to assign to each path based on a uniform distribution of CTS values according to the probability density function

$$g(x) = \begin{cases} \frac{1}{b-a} & a \leqq x \leqq b \\ 0 & \text{otherwise.} \end{cases}$$

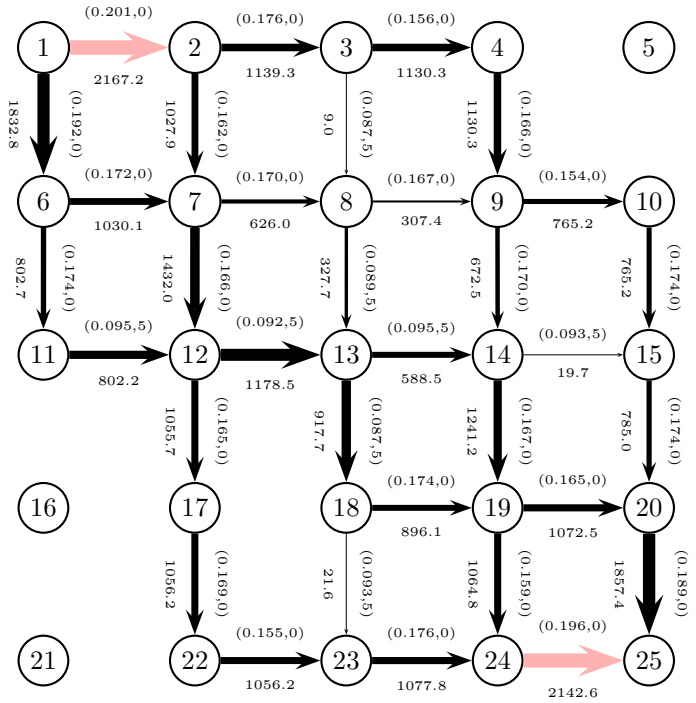


Figure 4.16. MSA with CTS assignment based on uniform distribution with $a = 60, b = 70$ for 5×5 grid network.

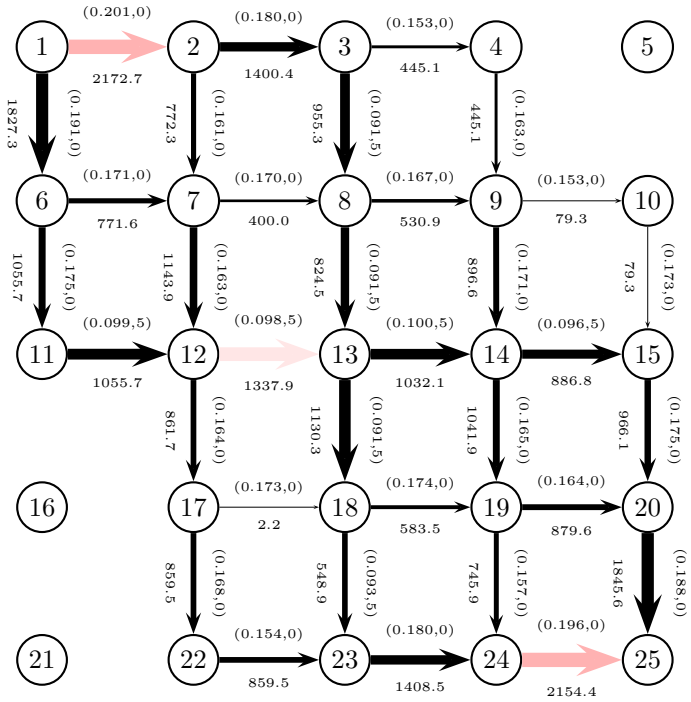


Figure 4.17. MSA with CTS assignment based on uniform distribution with $a = 60, b = 90$ 5×5 grid network.

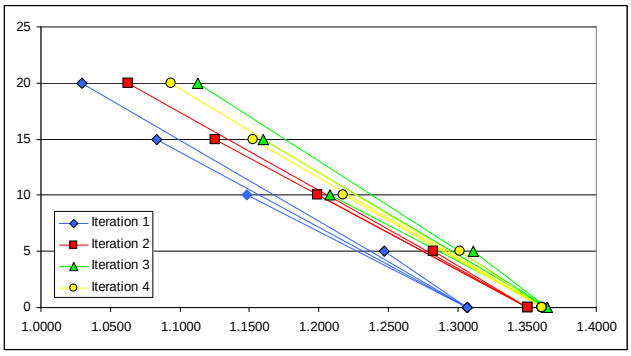


Figure 4.18. Non-dominated path cost vectors in first four iterations of MSA with CTS assignment as in Figure 4.17.

Procedure 6 CTS Assignment

input: Efficient paths $r_1^w, \ldots, r_{k_w}^w$ for each OD pair $w \in \mathcal{W}$, feasible solution $f \in \mathcal{K}$, demand d_w, and probability density function g of CTS-value τ.
The path with maximum travel time for OD pair w path is $r_{\max}^w$.
for all $w \in \mathcal{W}$ **do**
  for all $r_l^w \neq r_{\max}^w$ **do**
    $\tau_{r_l}^w = \left| \frac{m_{r_l^w}(f) - m_{r_{\max}^w}(f)}{t_{r_l^w}(f) - t_{r_{\max}^w}(f)} \right|$ /* Calculate CTS value $\tau_{r_l}^w$ */
    Rank values of τ^w: denote by $\tau_{(1)}^w$ the lowest and by $\tau_{(k_w-1)}^w$ the highest value.
  end for
  for all $l = 1, \ldots, k_w - 2$ **do**
    $h_{(l)} = \left[S\left(\tau_{(l+1)}^w\right) - S\left(\tau_{(l)}^w\right) \right] \cdot d_w$ /* Assign share of flow to each efficient path */
  end for
  $h_{r_{\max}} = \left[S\left(\tau_{(1)}^w\right) \right] \cdot d_w$ /* Assign share of flow to cheapest path */
  $h_{(k_w-1)} = \left[1 - S\left(\tau_{(n-1)}^w\right) \right] \cdot d_w$ /* Assign share of flow to most expensive path */
  $h_r = 0, r \in \mathcal{R}_w \setminus \{r_1^w, \ldots, r_{k_w}^w\}$ /* No flow on non-efficient paths */
end for
output: Components of path flow vector corresponding to OD pair w.

Again, 1000 iterations of the CTS assignment algorithm were performed. In Figure 4.16 a uniform distribution with $a = 60, b = 70$ is assumed, whereas in Figure 4.17 we assume $a = 60, b = 90$. In the first case, the CTS values that receive a positive share of flow are fairly low with $b = 70$, this is reflected in Figure 4.16 by less usage of tolled arcs when compared to Figure 4.17 with $b = 90$. For the solution from Figure 4.17, we show efficient path cost vectors for the first four iterations in Figure 4.18 and corresponding slopes between the non-dominated point corresponding to the fastest path and the other ones. In this example, the obtained CTS values initially range between 63.1 and 84.2.

Reference Point Assignment

Reference point (RPT) assignment works by defining a reference point in decision space for each OD pair w that acts as an attractor, a point to which the network users are attracted. The attractiveness of each non-dominated path cost vector is determined by its distance to the reference point while also

taking into account the distance of all other points to the reference point. The smaller this attractiveness measure, the higher the share of users on the corresponding path and vice versa. The reference point also serves as an indicator of a reasonable solution, which allows to identify paths whose travel time or cost is too far from what is reasonable and that should therefore not be assigned any flow or at least not much flow.

Different attractiveness measures are possible, each more or less sensitive to the difference in path cost vectors and their distance to the reference point. We present three possible ways of calculating the share of demand d_w that is assigned to each efficient path $r_1^w, \ldots, r_{k_w}^w$. Again, we initially assume that there are no equivalent paths and that all path cost vectors are distinct. Denote the reference point for OD pair w by z^w and the Euclidean distance between point z^j and the reference point z^w by $d_j^w = d(z^j, z^w)$. The share of flow on efficient path r_l^w is s_l^w. The first method to compute s_l^w is a sum-based approach

$$s_l^w = \frac{\sum_{j=1}^{k_w} d_j^w - d_l^w}{(k_w - 1)\sum_{j=1}^{k_w} d_j^w}. \tag{4.45}$$

Alternatively, we may compute s_l^w by following a sum of squares approach

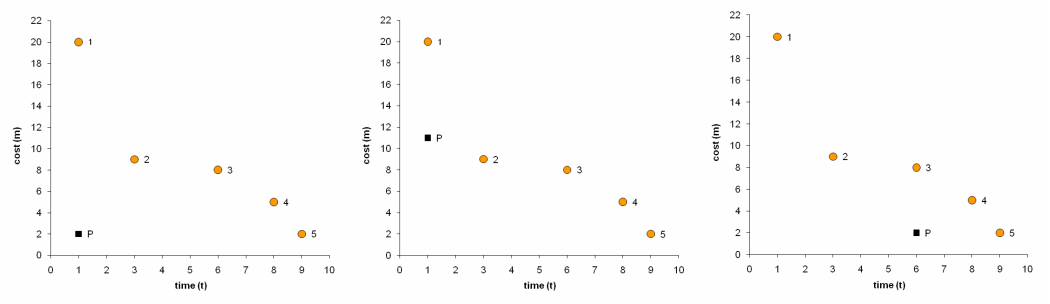
$$s_l^w = \frac{\sum_{j=1}^{k_w} (d_j^w)^2 - (d_l^w)^2}{(k_w - 1)\sum_{j=1}^{k_w} (d_j^w)^2}. \tag{4.46}$$

Finally, we suggest computing s_l^w by a product approach

$$s_l^w = \frac{\prod_{j \neq l} d_j^w}{\sum_{j_1=1}^{k_w} \left(\prod_{j_2 \neq j_1} d_{j_2}^w \right)}. \tag{4.47}$$

We compare the different methods of calculating the share of flow for an example with just five non-dominated path cost vectors. The reference point is located in different positions in objective space, as indicated in the three parts of Figure 4.19. Each non-dominated path cost vector in the figure is numbered and point number j represents path r_j, whereas the reference point is labelled by P. The resulting shares are calculated in Table 4.2.

It becomes apparent how the location of P changes the shares for each path. It appears that when calculating shares with the sum method (4.45), the shares



(a) (b) (c)

Figure 4.19. Different positions of reference point P for RPT assignment.

Table 4.2. Share calculated by different methods in RPT assignment.

Reference point as in Figure 4.19	path	shares calculated according to		
		sum (4.45)	sum of squares (4.46)	product (4.47)
(a)	r_1	0.16	0.11	0.1
	r_2	0.21	0.23	0.24
	r_3	0.21	0.22	0.22
	r_4	0.21	0.22	0.23
	r_5	0.21	0.22	0.22
(b)	r_1	0.19	0.19	0.13
	r_2	0.23	0.24	0.43
	r_3	0.21	0.23	0.21
	r_4	0.19	0.19	0.13
	r_5	0.17	0.15	0.10
(c)	r_1	0.13	0.06	0.06
	r_2	0.20	0.22	0.14
	r_3	0.21	0.23	0.17
	r_4	0.23	0.24	0.29
	r_5	0.23	0.25	0.35

of all paths are of similar magnitude. This is less true for the sum of squares method (4.46) and the product method (4.47) appears to assign much higher shares to paths with cost vector close to P. The appropriate method to calculate shares within the RPT approach needs to be determined depending on the TA problem to be solved and the characteristics of network users. The RPT approach can be extended to the case of equivalent paths by just splitting the demand to be assigned between all paths with the same path cost vector. An extension of the RPT approach to include several reference points to allow for different locations that attract network users is also imaginable. Furthermore, weighting factors for the two objectives can be included to ensure the objectives are balanced. For example when time is measured in seconds and monetary cost is measured in dollars then the distance measure should not be biased by the fact that points are hundreds of seconds apart, whereas the costs differ only by a few dollars.

The choice of a different reference point influences the amount of flow on the paths. For the grid instance we illustrate how the choice of the reference point's location influences the resulting arc flow. We choose two different reference points and obtain a solution using MSA with RPT assignment where (4.47) is used to calculate the share of flow that is assigned to every path.

Choosing the point $(1.2, 3)$ with long travel time and small toll costs should cause preference of longer and cheaper paths. Consequently, there should be relatively little flow on tolled arcs. We observe this behaviour in Figure 4.20. Placing the reference point at $(1.0, 13)$, on the other hand, favours faster and more expensive paths. The resulting arc flow solution is illustrated in Figure 4.22.

In Figures 4.21 and 4.23 the behaviour of path cost vectors during the first four iterations of MSA with RPT assignment is illustrated. The reference point is included in both figures.

Area of Domination

Every non-dominated path cost vector dominates part of the objective space. This gives rise to another potential assignment approach denoted by area of domination (ADO). A survey of road users can be conducted asking every user

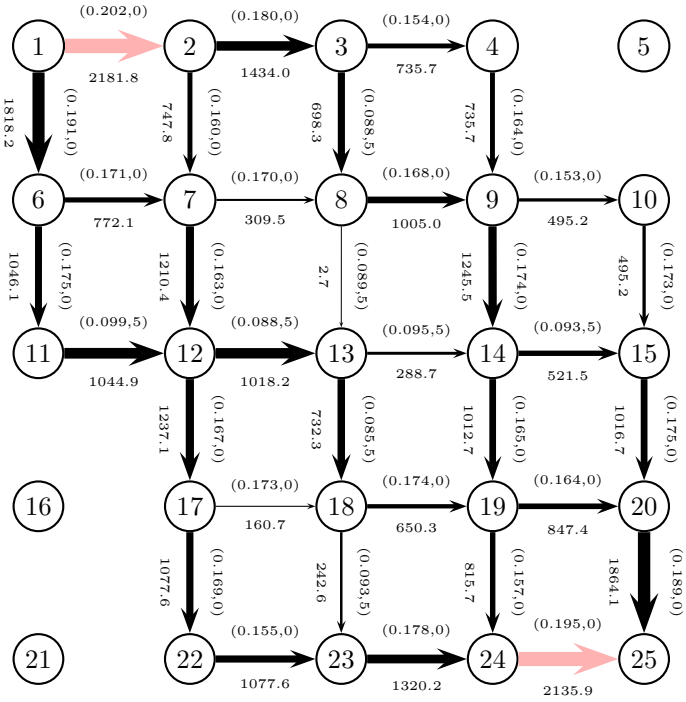


Figure 4.20. MSA with RPT assignment where the reference point is $(1.2, 3)$ and shares are calculated according to (4.47).

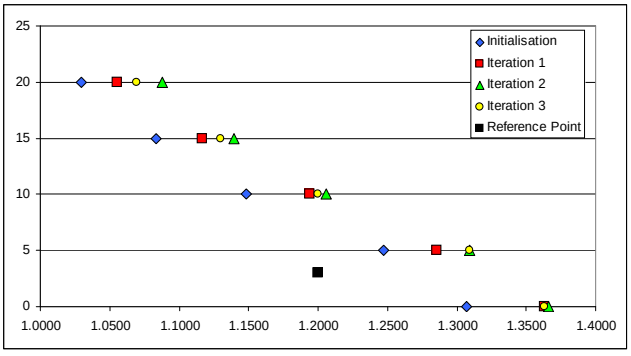


Figure 4.21. Non-dominated path cost vectors in first four iterations of MSA with RPT assignment and reference point $(1.2, 3)$.

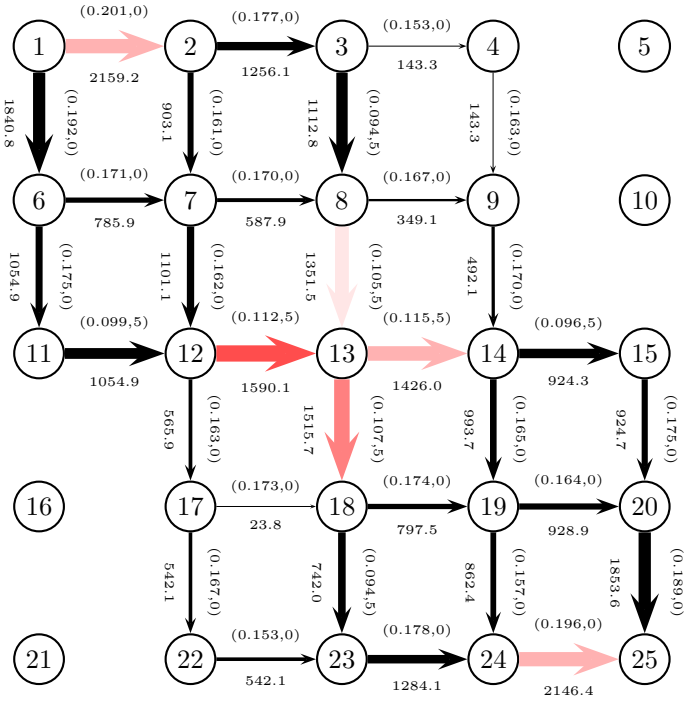


Figure 4.22. MSA with RPT assignment where the reference point is $(1, 13)$ and shares are calculated according to (4.47).

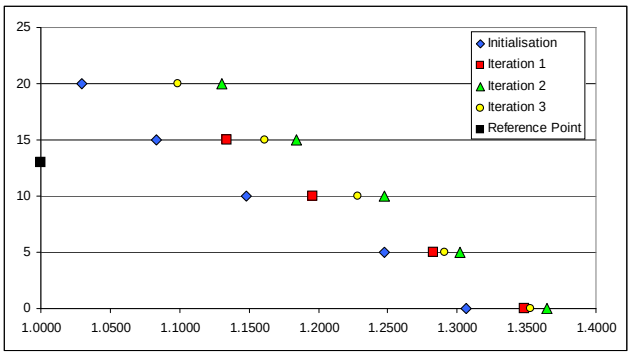


Figure 4.23. Non-dominated path cost vectors in first four iterations of MSA with RPT assignment and reference point $(1, 13)$.

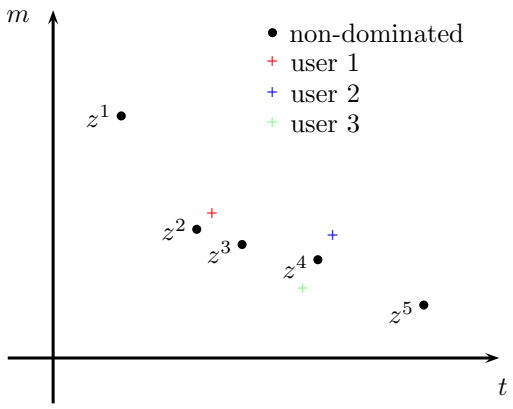


Figure 4.24. Different user ideal points and non-dominated path cost vectors.

the question "How much travel time do you spend at the moment, how much would you (realistically) want to save and how much money are you prepared to pay for it?". Then every road user can be represented by their own ideal point in objective space. Such a road user would select any of the paths with cost vector dominating their ideal point.

Figure 4.24 shows five different non-dominated path cost vectors z^j for one OD pair, each associated with an efficient path r_j together with the ideal points of three different road users. Clearly, user 1 will choose path r_2 as this path is both faster and cheaper than the user's ideal. User 2 has the choice between two paths as both paths r_3 and r_4 are faster and cheaper than his ideal. User 3 has an ideal that cannot be achieved by any of the options available – this user could either choose the fastest path with cost less than that of their ideal or choose the closest path to the ideal.

From this observation we can devise an ADO assignment method. Given a two-dimensional density function in decision space derived from the user survey, portions of the demand can be assigned to the different efficient paths as indicated in Figure 4.25. The yellow rectangles are each dominated by exactly one non-dominated point, therefore the share corresponding to the demand arising in this area is assigned to the corresponding efficient path(s). The red rectangles are dominated by two different non-dominated points and the arising demand should be split between the two. The blue rectangles are dominated by three points, and therefore upcoming demand from this area should be split between the corresponding three efficient paths. This process

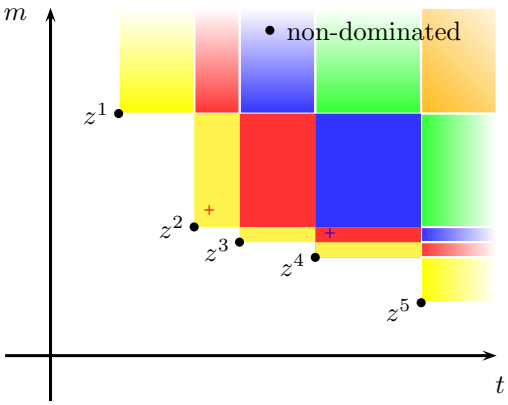


Figure 4.25. Dominated area.

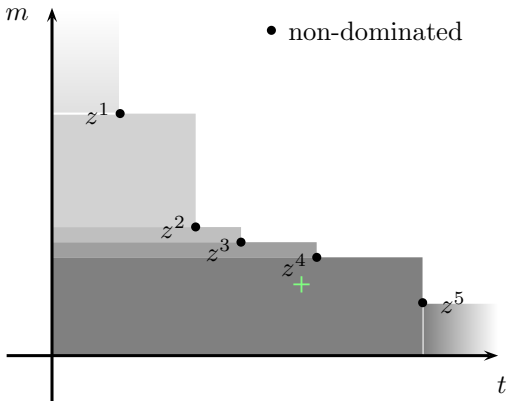


Figure 4.26. Area that is not dominated by points $z^1, \ldots, z^5$.

continues until, in the figure, the orange area is reached that is dominated by all non-dominated points.

It can be expected that most demand occurs in the yellow areas of Figure 4.25, as network users would not want to pay large amounts of money to save just a little time. Therefore, the orange area should have hardly any users' ideal points. It remains to decide how to assign users with ideal points that are not dominated by any non-dominated path cost vector such as the one represented by a green plus in Figures 4.24 and 4.26. One can assign a portion of those users to the fastest path with cost less than that of their ideal, as mentioned above. This is illustrated in Figure 4.26: users with ideal point in the light-gray area at the top of the graph would choose path r_1, users with ideal point in the darker gray area just below the previous one would choose path r_2, etc. Finally, users with ideal point in the dark gray area at the bottom of the graph would choose path r_5. Alternatively, users can be assigned to their closest non-dominated path cost vector.

Let $g(x, y)$ be the probability density function that represents network users' ideal points such that $\int_0^\infty \int_0^\infty g(u, v)dudv = 1$. Then the share of users in a certain rectangle $\mathcal{B} = [x_1; x_2] \times [y_1; y_2]$ is $S(\mathcal{B}) = \int_{x_1}^{x_2} \int_{y_2}^{y_2} g(u, v)dudv$, where the upper limits may be ∞. We want to calculate the portion of demand d_w that is assigned to each efficient path $r_1^w, \ldots, r_{k_w}^w$ with non-dominated path cost vectors $z^{1,w}, \ldots, z^{k_w,w}$. Again, we initially assume that there are no equivalent

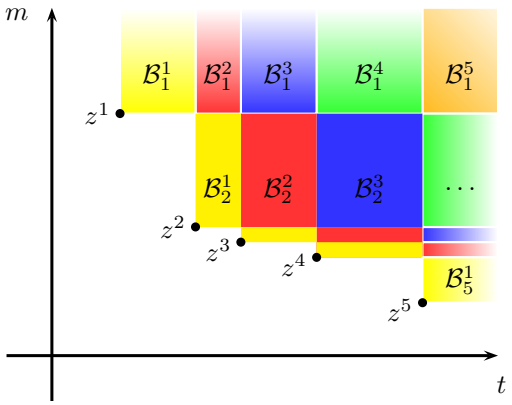


Figure 4.27. Rectangles $\mathcal{B}_j^i$ in dominated area.

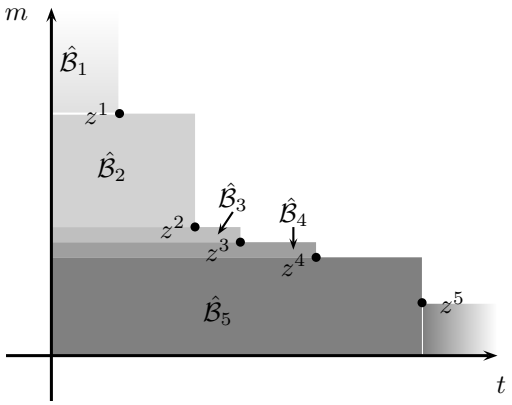


Figure 4.28. Rectangles $\hat{\mathcal{B}}_j$ in area that is not dominated by points $z^1, \ldots, z^5$.

paths and that all path cost vectors are distinct. We also assume that the paths are numbered so that

$$z_1^{1,w} < z_1^{2,w} < \ldots < z_1^{k_w,w} \text{ and } z_2^{1,w} > z_2^{2,w} > \ldots > z_2^{k_w,w}.$$

From the non-dominated points $z^{1,w}, \ldots, z^{k_w,w}$, k_w rectangles $\mathcal{B}_1^1, \ldots, \mathcal{B}_{k_w}^1$ are derived that are dominated by only one non-dominated point:

$$\begin{aligned}
\mathcal{B}_1^1 &= [z_1^{1,w}; z_1^{2,w}] \times [z_2^{1,w}; \infty) \\
\mathcal{B}_2^1 &= [z_1^{2,w}; z_1^{3,w}] \times [z_2^{2,w}; z_2^{1,w}] \\
\mathcal{B}_3^1 &= [z_1^{3,w}; z_1^{4,w}] \times [z_2^{3,w}; z_2^{2,w}] \\
&\vdots \\
\mathcal{B}_{k_w}^1 &= [z_1^{k_w,w}; \infty) \times [z_2^{k_w,w}; z_2^{k_w-1,w}]
\end{aligned} \tag{4.48}$$

Similarly, $k_w - 1$ rectangles $\mathcal{B}_1^2, \ldots, \mathcal{B}_{k_w}^2$ that are dominated by two neighbouring non-dominated points can be formulated. This process continues until the final rectangle $\mathcal{B}_1^{k_w} = [z_1^{k_w,w}; \infty) \times [z_2^{1,w}; \infty)$ is defined that is dominated by all points $z^{1,w}, \ldots, z^{k_w,w}$. The rectangles are also marked in Figure 4.27.

The share of demand on each path arising from dominated user ideal points is

$$s_{r_j^w} = S(\mathcal{B}_j^1) + \frac{1}{2}\sum_{i=\max\{1,j-1\}}^{\min\{k_w-1,j\}} S(\mathcal{B}_i^2) + \frac{1}{3}\sum_{i=\max\{1,j-2\}}^{\min\{k_w-2,j\}} S(\mathcal{B}_i^3) + \ldots + \frac{1}{k_w} S(\mathcal{B}_1^{k_w}). \tag{4.49}$$

We need to add the share for paths with non-dominated user ideal points. For simplicity, we assume that all network users choose the fastest path with lower cost than their user ideal among paths $r_1^w, \ldots, r_{k_w}^w$, if their ideal point is not dominated by a point $z^{1,w}, \ldots, z^{k_w,w}$. Then, the rectangle $\hat{\mathcal{B}}_j$ is the rectangular area that represents users choosing path r_j^w:

$$\begin{aligned}
\hat{\mathcal{B}}_1 &= [0; z_1^{1,w}] \times [z_2^{1,w}; \infty) \\
\hat{\mathcal{B}}_2 &= [0; z_1^{2,w}] \times [z_2^{2,w}; z_2^{1,w}] \\
\hat{\mathcal{B}}_3 &= [0; z_1^{3,w}] \times [z_2^{3,w}; z_2^{2,w}] \\
&\vdots \\
\hat{\mathcal{B}}_{k_w-1} &= [0; z_1^{k_w-1,w}] \times [z_2^{k_w-1,w}; z_2^{k_w-2,w}],
\end{aligned}$$

and finally $\hat{\mathcal{B}}_{k_w} = [0; z_1^{k_w,w}] \times [z_2^{k_w,w}; z_2^{k_w-1,w}] \cup [0; \infty) \times [0; z_2^{k_w,w}]$, see also Figure 4.28. This leads to shares

$$\hat{s}_{r_j^w} = S(\overline{\mathcal{B}}_j). \tag{4.50}$$

By combining (4.49) and (4.50) the path flow for efficient paths $r_1^w, \ldots, r_{k_w}^w$ becomes

$$h_{r_j^w} = \left(s_{r_j^w} + \hat{s}_{r_j^w}\right) d_w.$$

For all other paths $r \in \mathcal{R}_w \setminus \{r_1^w, \ldots, r_{k_w}^w\}$ $h_r = 0$.

In order to run MSA with ADO assignment for the grid network, we need a probability density function or a distribution of users. Instead, we divide the area of the objective space that contains the non-dominated path cost vectors into a grid. For every cell of the grid, we assume there is a certain number of network users whose ideal points lie within this cell. Figure 4.29 shows how many users choose each ideal point, indicated by the height of the bar in the cell. This data can be obtained as a the result of a survey of network users who are asked to state their ideal point for a trip in the grid network instance. The resulting arc flow solution is shown in Figure 4.30.

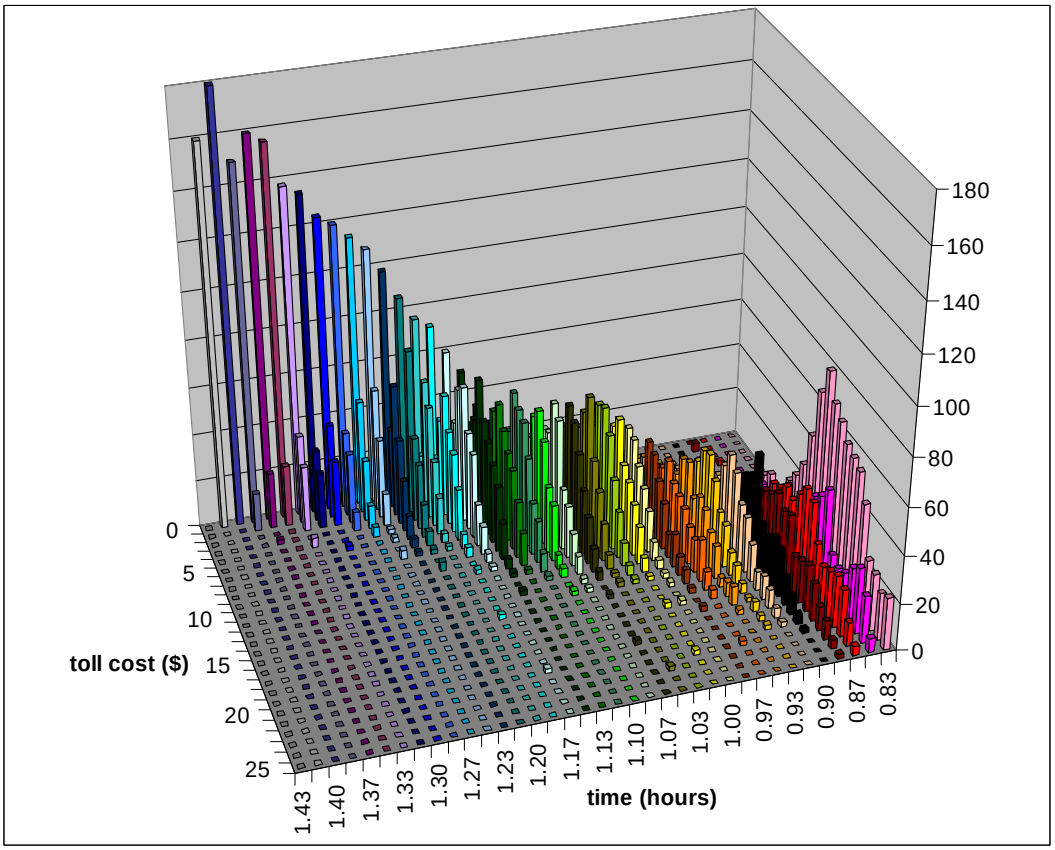


Figure 4.29. Possible distribution of users for ADO assignment.

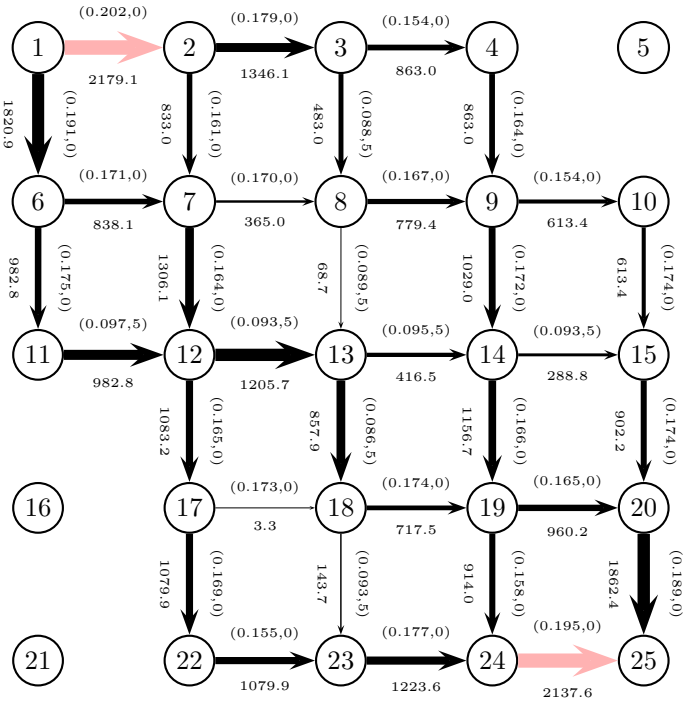


Figure 4.30. MSA with ADO assignment based on distribution from Figure 4.29. Solving 5×5 grid network.

Assignment Based on Non-linear Valuation Function

When a non-linear valuation function v_w is given as in Section 4.5.1, the single-objective MSA approach can be used as introduced above. One can again perform AON assignment, by identifying the shortest path for each w with respect to $g^t(f)$ and assigning all demand d_w to it.

Due to the non-linear component of g^t that evaluates path cost, it is necessary to use a bi-objective shortest path algorithm to obtain all efficient paths with non-dominated path cost vectors (t_r, m_r). Then, the generalised time is computed as $g_r^t = t_r + v_w(m_r)$ and the minimal path is identified. As an alternative to assigning all flow to the cheapest path, one might consider assigning flow to the first few cheapest paths to potentially obtain faster convergence of MSA.

Comparison of Different Assignment Methods

Clearly, EQS is a very simple assignment method that is not expected to be very useful in practice. Assigning equal portions of travel demand to each efficient path in the network no matter what the actual path costs are, cannot be a very realistic approach.

CTS overcomes this disadvantage by introducing the ratio between additional travel cost (compared to the cheapest solution) and saved travel time into the assignment process. Only the ratio but not the absolute costs and times are included, which may overestimate the number of users on very expensive paths. A difficulty here is finding an appropriate distribution of the CTS values.

ADO is similar to CTS in requiring the distribution of user ideal points that indicate how much time a user would like to spend when travelling to their destination and how much they are prepared to pay for it. Compared to CTS, this may present an advantage as absolute values are included in the assignment process and network users can give upper bounds on how much money they are prepared to spend.

From a practical point of view, a distribution of CTS values and a distribution of user ideal points needed for ADO assignment similar to the one shown in Figure 4.29 can be obtained through a network user survey. Similar surveys are common practice in transportation planning.

RPT allows to select a single reference point as an attractor. However, this can be modified to use more than one reference point. Paths close to this point are more attractive than those far away and should therefore obtain more flow. We presented three different measures of distance to this point that allow stronger and weaker influence of the location of the point and all non-dominated path cost vectors on the portion of total travel demand that is assigned to each path. An advantage of RPT over CTS and ADO is that a reference point is easily selected, but it is not clear how to select a point to model certain user behaviour. If network user surveys show certain trends in preference, such as clusters of similar user preference, a reference point could be selected for each of those clusters.

Only practical tests can show how realistic the above assignment methods are, and how practical they are to use.

Other Assignment Methods

There are numerous other assignment methods one can think of. Each of them potentially leads to a different BEQ solution. This makes the BEQ approach highly adaptable to any kind of real-world situation - a correctly calibrated model can lead to any kind of equilibrium solution that satisfies BEQ, and can therefore model the behaviour of road users in different situations accurately.

Whichever assignment method is used, its calibration to ensure that reality is modelled as accurately as possible, remains difficult.

4.5.3 Bi-objective Path Equilibration

The idea of path equilibration (Section 4.2.4) can also be extended to BEQ. As this is a path flow based method, the obtained path flow variables can be used to check that an obtained solution is actually an equilibrium solution as explained above. We assume that we have two objectives, namely flow dependent time t and a fixed cost function m.

In the single-objective case, the essence of path equilibration is to find the current shortest path for each OD pair and the longest with positive flow. Then flow is re-distributed between them until they have identical costs or the longer path has zero flow. Translating this to a bi-objective setting means that efficient paths are identified and one of the non-efficient paths with positive flow. Among all paths with positive flow, the one with largest distance from the efficient paths can be selected, but other selection criteria are possible as well. Having selected such a non-efficient path, the task is to re-distribute flow from the non-efficient path to the efficient ones (or at least to one of them) until the longest path is efficient or has flow zero. Unlike the single-objective case, there are multiple ways of re-distributing flow, which we discuss in the following.

We base this discussion on several examples. Consider the situation in Figure 4.31. There are four non-dominated points $z^1, \ldots, z^4$ corresponding to efficient paths $r_1, \ldots, r_4$ for some OD pair w. Point $\tilde{z}$ is dominated with corresponding non-efficient path $\tilde{r}$. Assume that $f_{\tilde{r}} > 0$. This is a situation one may face in bi-objective path equilibration: there is positive flow on a non-efficient path

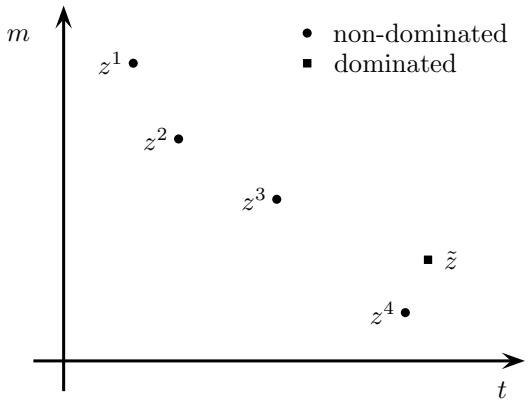


Figure 4.31. Non-dominated path cost vectors for OD pair w and dominated path cost vector $\tilde{z}$ with positive flow on corresponding path.

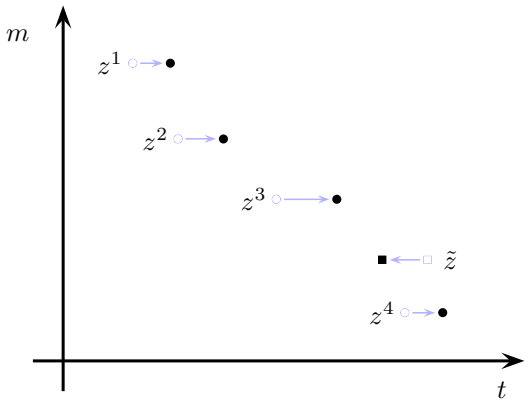


Figure 4.32. New path cost vectors resulting from feasible flow re-distribution.

that should be re-distributed to efficient paths. As we assume the path costs $m_r, r \in \mathcal{R}_w$ are fixed, only path travel time t changes as path flow changes. In particular, path flow $t_{r_i}(f)$ increases when increasing f_{r_i}, whereas path flow $t_{\tilde{r}}(f)$ decreases as $f_{\tilde{r}}$ decreases. A possible resulting change is depicted in Figure 4.32. Due to the fixed monetary costs, path objective vectors move horizontally.

Given the flow change indicated in Figure 4.32 is feasible, it achieves the goal of making the formerly non-efficient path $\tilde{r}$ efficient. Unlike the single-objective case, it is not clear how much of the flow on path $\tilde{r}$ should be re-distributed. The re-distribution indicated in Figure 4.33 is also feasible and it is unclear which one is preferable. A reasonable re-distribution should ensure that the point $\tilde{z}$ becomes non-dominated, therefore the point needs to move into the red rectangle indicated in Figure 4.34. Of course, the rectangle may change as flow is assigned to paths r_3 and r_4, which is disregarded in the figure.

A different situation arises if point $\tilde{z}$ has the same monetary cost as one of the non-dominated points z^i, see Figure 4.35. A sensible re-distribution of flow seems to be one that makes the travel time on paths r_3 and $\tilde{z}$ equal, for the results of a possible flow re-distribution see Figure 4.36.

It is apparent that there is a lot of flexibility in the flow re-distribution step.

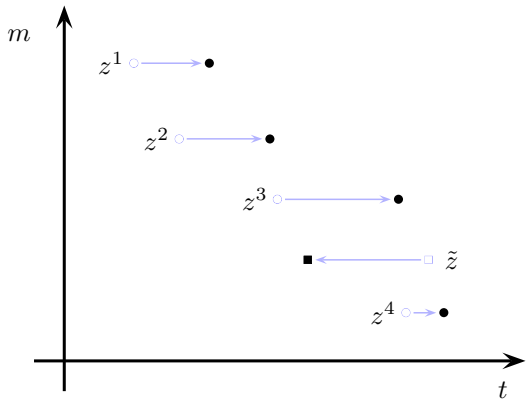


Figure 4.33. New path cost vectors resulting from feasible flow re-distribution.

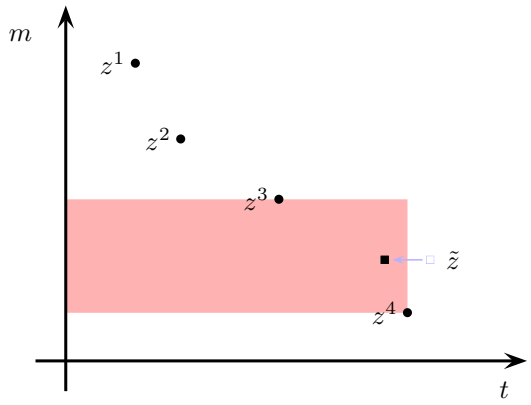


Figure 4.34. Area into which point $\tilde{z}$ needs to move to become non-dominated.

Assume there is a feasible path flow vector f that does not satisfy VEQ meaning that there exists $w \in \mathcal{W}$ and a non-efficient path $\tilde{r} \in \mathcal{R}_w$ with positive flow $f_{\tilde{r}} > 0$. One has to decide on a feasible flow re-distribution onto the efficient paths $r_1, \ldots, r_k$ of flow $0 < \sigma \leqq f_{\tilde{r}}$ of the form

$$h_r = \begin{cases} f_r & \text{if } r \notin \{\tilde{r}, r_1, \ldots, r_k\} \\ f_r - \sigma & \text{if } r = \tilde{r} \\ f_r + \omega_i \sigma & \text{if } r = r_i, i = 1 \ldots, k, \end{cases}$$

with $\omega_i \geqq 0, \sum_{i=1}^{k} \omega_i = 1$. The re-distribution of flow should make point $\tilde{z}$ non-dominated, but otherwise it is very flexible. Possible re-distribution strategies, which means determining values for all ω_i, include

- Re-distribute flow equally to all efficient paths, i.e. choose $\omega_i = \frac{1}{k}$.
- Add flow only to paths with objective vector close to point $\tilde{z}$. In Figure 4.36 those would be paths r_2, r_3, r_4, whereas in Figure 4.34 flow would be re-distributed to paths r_3, r_4.
- In a situation where the monetary cost of $\tilde{z}$ is equal to that of some other non-dominated point z^i only move flow to path r_i. Using this rule, it cannot be guaranteed that flow can be re-distributed to make point $\tilde{z}$

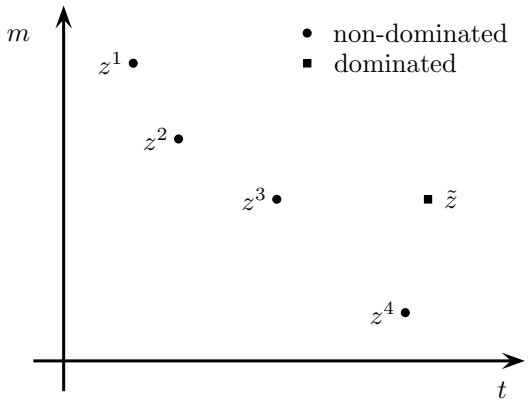


Figure 4.35. Non-dominated path cost vectors for OD pair w and dominated path cost vector with monetary cost equal to that of another point, z^3.

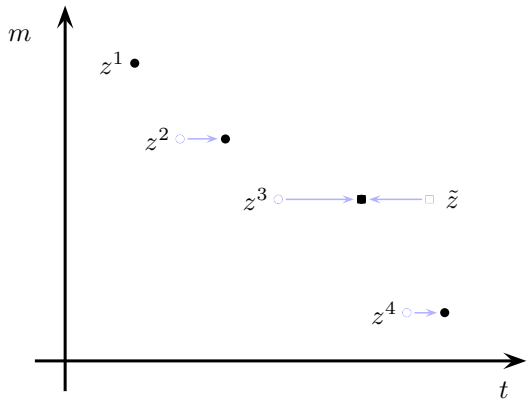


Figure 4.36. New path cost vectors resulting from feasible flow re-distribution.

non-dominated as the re-distribution may equate the costs of $\tilde{z}$ and z^i and at the same time render the two points dominated.

There are also different strategies to determine how much flow to re-distribute, including

- Re-distribute all flow, $\sigma = f_{\tilde{r}}$.
- In a situation where the monetary cost of $\tilde{z}$ is equal to that of some other non-dominated point z^i move as much flow as necessary to make points z^i and $\tilde{z}$ coincide or flow on $\tilde{r}$ is zero.
- If the monetary cost of $\tilde{z}$ is not equal to that of any other non-dominated point z^i, the goal is to make $\tilde{r}$ efficient. One could move $\tilde{z}$ to be just non-dominated, so that $\tilde{z} + \epsilon$ is dominated when a small value of $\epsilon \in \mathbb{R}^2$ is added to $\tilde{z}$. Conversely, one could move the point as far as possible while ensuring that it does not dominate its neighbouring non-dominated points.
- If the monetary cost of $\tilde{z}$ is not equal to that of any other non-dominated point z^i, one could aim at moving point $\tilde{z}$ depending on the position of its neighbouring non-dominated points z^i and z^{i+1}. Then, $\tilde{z}$ could move

as far as possible while it does not dominated its neighbours, but $\tilde{z} + \epsilon$ does dominated at least one of the neighbours. Alternatively, the aim could be to move $\tilde{z}$ to obtain a travel time of $t_{\tilde{r}} = \frac{1}{2}(t_{z^i} - t_{z^{i+1}})$, assuming $t_{z^i} > t_{z^{i+1}}$. Any other fraction is also possible.

When a non-linear valuation function v_w is given as in Section 4.5.1, the single-objective approach for path equilibration can be used as it is. Firstly, the shortest path r with respect to generalised time g^t and the longest one s with positive flow are identified. Then flow is moved from path s to path r until their generalised time g^t becomes equal or until there is no more flow on path s, whichever comes first.

As discussed earlier, it is necessary to find all efficient paths first and then to compute their generalised time value $g_r^t = t_r + v_w(m_r)$. Instead of re-assigning flow only between the shortest and the longest path, one can re-assign flow from the longest to several shortest paths in order to obtain faster convergence of the method.

We make a basic implementation of bi-objective path equilibration. In each iteration, the efficient paths are identified. Then the non-efficient path $\tilde{r}$ (with positive flow), whose path travel time has largest distance d from that of any non-dominated point, is selected. Flow is re-distributed from path $\tilde{r}$ to the current efficient paths with the aim of reducing the travel time on path $\tilde{r}$ by d (or reducing flow on $\tilde{r}$ to zero). The flow that is removed from path $\tilde{r}$ is re-distributed in equal shares between the efficient paths. This path equilibration runs until it can be guaranteed that a BUE solution is obtained. We stop the algorithm when the maximal distance d between the cost vector of a non-efficient path with positive flow and a non-dominated point is less than 0.00005.

The BUE solution obtained depends on the initial assignment of flow. We implement two different initial assignment procedures. The solution shown in Figure 4.37 is based on an initial EQS assignment, i.e. flow is split equally between the efficient paths. The solution in Figure 4.38 is based on initially assigning all the flow to the fastest path, this yields a solution with higher usage of tolled arcs.

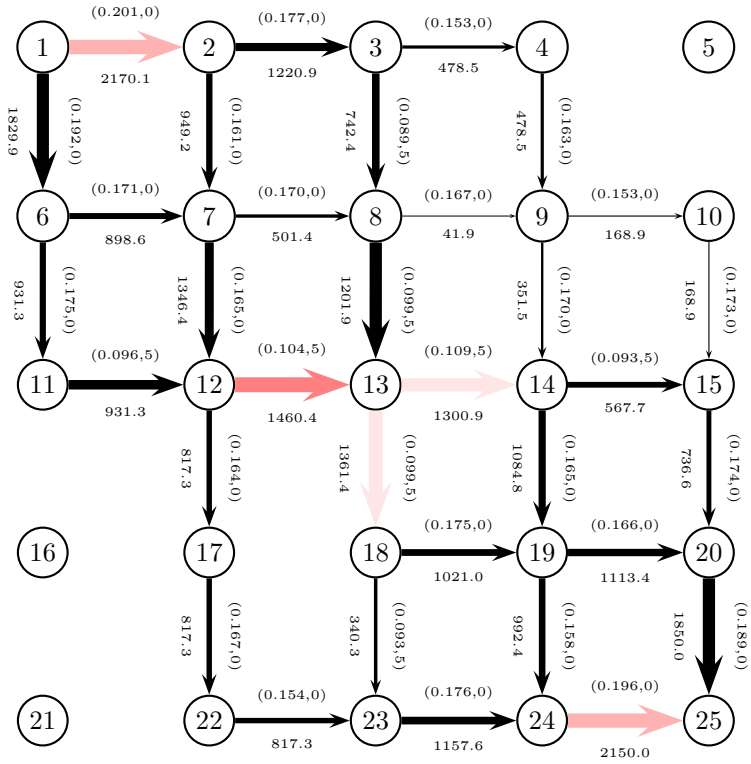


Figure 4.37. Path Equilibration, initially assigning equal portions of flow to all efficient paths.

4.6 Conclusions on Bi-objective Traffic Assignment

The study of the traffic assignment (TA) problem that explicitly deals with at least two objective functions has taken two general lines of approach. On the one hand, there is the practical approach where the problem is reduced to the single-objective TA by combining the two (or more) objectives into a generalised cost function. We are able to highlight some shortcomings of approaches that assume such a generalised cost function. On the other hand, vector equilibrium (VEQ) problems have been studied from a theoretical point of view. The traffic assignment problem with multiple criteria has been identified as an application of VEQ, but no attempt has been made to actually solve the problem using VEQ.

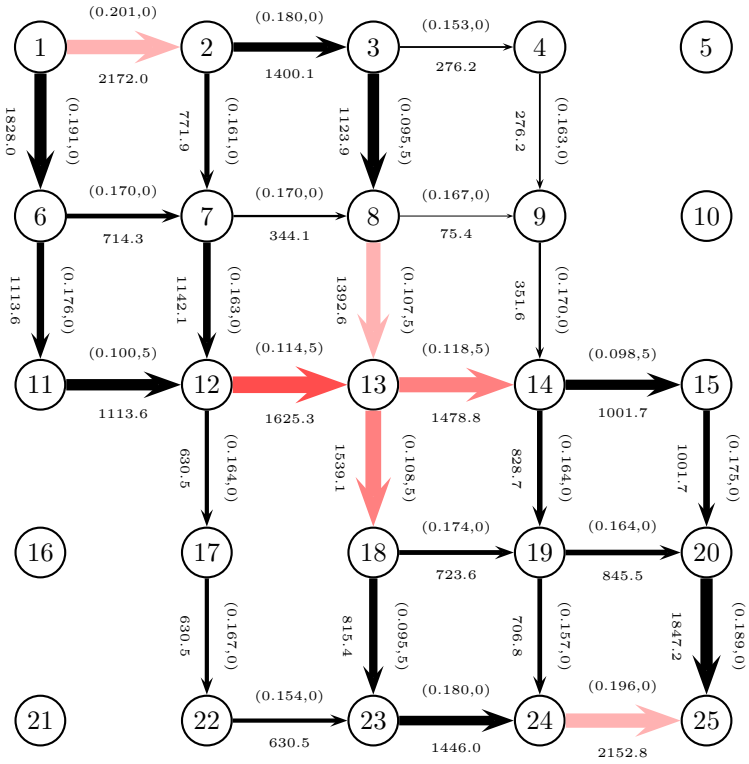


Figure 4.38. Path Equilibration, initially assigning all flow to the fastest efficient path.

In this chapter, we first present theoretical aspects of VEQ. It is well-known that the single-objective equilibrium problem is equivalent to a variational inequality problem and an optimisation problem (with the assumptions outlined in Section 4.2.2). Throughout the literature attempts are made to show a similar equivalence between VEQ and the corresponding vector variational inequality (VVI) and vector optimisation problems (VOP) but it has been found that there exists no equivalence relation between the respective problems. Our efforts to build on previous research on VEQ, VVI, VOP, and related problems were complicated by many erroneous statements in published articles as discussed in Section 4.4.2.

In the literature, attempts have been made to exploit the fact that a solution of VVI is a solution of VEQ. In this thesis we are able to characterise the structural properties of VVI solutions. Compared to a VEQ solution, the solution of a VVI has a very restricted structure. This fact indicates that

the whole diversity of VEQ solutions cannot be obtained via VVI. In fact, it appears that although a VVI can have many solutions, each one of them could also be characterised by a single-objective VI with generalised cost function (or weighted sum cost function).

To summarise, the single-objective TA problem with generalised cost function will yield a single solution. As an alternative solution approach for TA, VVI has many solutions, but they are still based on generalised cost functions and could also be obtained by solving TA with different weighting factors. VEQ on the other hand has many solutions that are not restricted to being obtainable via a generalised cost formulation as solutions may contain non-supported paths and multiple paths that are not optimal for the same weighting factor. Therefore, multi-objective TA has solutions that cannot be found by repeatedly solving the conventional single-objective TA, thus considering multi-objective TA enriches the scope of attainable solutions greatly.

We also present several heuristic solution algorithms for the bi-objective TA problem, which can easily be extended to the multi-objective problem. Our algorithms are extensions of algorithms for the single-objective TA problem, for which convergence can be shown. We hope to be able to show convergence of our algorithms in the future. At the moment, we see the proposed heuristics as a first step towards more general and more realistic solution approaches for bi-objective TA. We propose different algorithms, but without classifying them for their applicability and usefulness in solving a real-life problem. This is an important aspect of further studies. The quality of the approaches will be assessed by applying the proposed models for user behaviour in road networks and comparing the resulting traffic flows with behaviour observed in real life.

From a practical point of view, it may not be necessary to restrict considerations to a single final solution. The fact that VEQ permits many solutions can be beneficial in the analysis of a road network. The bi- or multi-objective nature of the problem reflects the diversity of route choice decisions made in real life, and it is widely acknowledged that there is not one best route to choose. As a result, there exist different plausible traffic volumes. By considering VEQ, we permit many final solutions to the bi- or multi-objective TA problem, each representing different possible traffic volumes on roads. Instead of generating only a single solution of multi-objective TA, many solutions can

be generated, which represent a range of different possible outcomes. With respect to tolling, for example, the model may yield a range of feasible tolling revenues – giving decision makers a much broader picture of what may happen than just presenting a single scenario. Especially when large scale tolling schemes are introduced in a region for the first time, as would be the case in New Zealand, this may prove valuable as network user behaviour cannot be observed to calibrate a model.

References

R.K. Ahuja and H.W. Hamacher. A network flow algorithm to minimize beam-on time for unconstrained multileaf collimator problems in cancer radiation therapy. *NETWORKS*, pages 36–41, 2005.

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

Y.P Aneja and K.P.K. Nair. Bicriteria transportation problem. *Management Science*, 25:73–78, 1979.

L. Aultman-Hall, F.L. Hall, and B.B. Baetz. Analysis of bicycle commuter routes using geographic information systems – implications for bicycle planning. *Transportation Research Record*, 1578:102–110, 1997.

J.A. Azevedo and E.Q.V. Martins. An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigagão Operational*, 11(1):52–69, 1991.

M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows*. Wiley, 2005.

M. Beckmann, C.B. McGuire, and C.B. Winsten. *Studies in the economics of transportation*. Yale University Press, 1956.

D. Bernstein and S.A. Gabriel. Solving the nonadditive traffic equilibrium problem. In P.M. Pardalos, D.W. Hearn, and W.W. Hager, editors, *Network Optimization*, pages 72–102. Springer, 1997.

D.P. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2):216–224, 1989.

Bureau of Public Roads. *Traffic assignment manual*. U.S. Department of Commerce, Urban Planning Division, Washington D.C., 1964.

W.M. Carlyle and R.K. Wood. Near-shortest and k-shortest simple paths. *Networks*, 46(2):98–109, 2005.

R.L. Carraway, T.L. Morin, and Moskowitz H. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44:95–104, 1990.

G. Chen, X. Huang, and X. Yang. *Vector Optimization*. Springer, 2005.

G.Y. Chen and N.D. Yen. On the variational inequality model for network equilibrium. Technical Report 3.196 (724), Department of Mathematics, University of Pisa, 1993.

G.Y. Chen, C.J. Goh, and X.Q. Yang. Vector network equilibrium problems and nonlinear scalarization methods. *Mathematical Methods of Operations Research*, 49:239–253, 1999.

T.C.E. Cheng and Y.N. Wu. A multiproduct, multicriterion supply-demand network equilibrium model. *Operations Research*, 54:544–554, 2006.

B.V. Cherkassy, A.V. Goldberg, and T. Radzik. Shortest path algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73:129–174, 1996. Implementations of algorithms obtainable at `http://www.avglab.com/andrew/soft.html`.

J.C.N. Clímaco and E.Q.V. Martins. A bicriterion shortest path problem. *European Journal of Operational Research*, 11:399–404, 1982.

J.L. Cohon. *Multiobjective Programming and Planning*. Academic Press, New York, 1978.

H.W. Corley and I.D. Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46(1):79–86, 1985.

W.H. Cunningham. A network simplex method. *Mathematical Programming*, 11:105–116, 1976.

S. Dafermos. A multicriteria route-mode choice traffic equilibrium model. *Bulletin of the Greek Mathematical Society*, 24:13–32, 1983.

S. Dafermos and F.T. Sparrow. The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards–B. Mathematical Sciences*, 73B:91–118, 1969.

G.B. Dantzig and M.N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer Series in Operations Research. Springer, 1997.

E.V. Denardo and B.L. Fox. Shortest-route methods: 1. reaching, pruning, and buckets. *Operations Research*, 27:161–186, 1979.

I. Diakonikolas and M. Yannakakis. Small approximate pareto sets for bi-objective shortest paths and other problems. In *10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007. ProceedingsApproximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 74–88. Springer Berlin / Heidelberg, 2007.

R.B. Dial. A model and algorithm for multicriteria route-mode choice. *Transportation Research*, 13B:311–316, 1979.

R.B. Dial. Bicriterion traffic assignment: Basic theory and elementary algorithms. *Transportation Science*, 30:93–111, 1996.

R.B. Dial. Bicriterion traffic assignment: efficient algorithms plus examples. *Transportation Research B*, 31:357–379, 1997.

R.B. Dial. Network-optimized road pricing: Part I: A parable and a model. *Operations Research*, 47:54–64, 1999a.

R.B. Dial. Network-optimized road pricing: Part II: Algorithms and examples. *Operations Research*, 47:327–336, 1999b.

J. Dill and T. Carr. Bicycle commuting and facilities in major US cities – if you build them, commuters will use them. *Transportation Research Record*, 1828:116–123, 2003.

M. Ehrgott. *Multicriteria Optimization*. Springer, second edition, 2005.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22:425–460, 2000.

M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization – theory, methodology, and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, International Series in Operations Research & Management Science, pages 369–444. Kluwer, 2002.

J. Eliasson. *Transport and location analysis*. PhD thesis, Kungliga Tekniska Hogskolan (Sweden), 2000. Chapter "The use of average values of time in road pricing. A note on a common misconception".

A. Eusébio and J.R. Figueira. Why is it difficult to find non-dominated solutions in bi-criteria network flow problems when using simplex based methods? Technical report, Instituto Superior Técnico, Portugal, 2006.

A. Eusébio and J.R. Figueira. Finding non-dominated solutions in bi-objective integer network flow problems. *Computers & Operations Research*, 36:2554–2564, 2009a. doi: 10.1016/j.cor.2008.11.001.

A. Eusébio and J.R. Figueira. On the computation of all supported efficient solutions in multi-objective integer network flow problems. *European Journal of Operational Research*, 199:68–76, 2009b.

Florida Department of Transportation. Quality / level of service handbook. available online, 2002. URL `http://www.dot.state.fl.us/planning/systems/sm/los/los_sw2.shtm#handbook`.

V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139(2):533–542, 2002.

S.A. Gabriel and D. Bernstein. The traffic equilibrium problem with nonadditive path costs. *Transportation Science*, 31:337–348, 1997.

G. Gallo and S. Pallotino. Shortest path algorithms. *Annals of Operations Research*, 13:3–79, 1988.

A.M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

F. Gianessi. Theorems of alternative, quadratic programs and complementarity problems. In R.W. Cottle, F. Giannessi, and J.L. Lions, editors, *Variational inequalities and complementarity problems : theory and applications*. Wiley, 1980.

C.J. Goh and X.Q. Yang. Vector equilibrium problem and vector optimization. *European Journal of Operational Research*, 116:615–628, 1999.

F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3):589–613, 2001.

H.W. Hamacher. A note on k best network flows. *Annals of Operations Research*, 57:65–72, 1995.

H.W. Hamacher, C.R. Pedersen, and S. Ruzika. Multiple objective minimum cost flow problems: A review. *European Journal of Operational Research*, 176:1404–1422, 2007.

P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making, Theory and Application. Proceedings of the 3rd International Conference, Hagen/Königswinter 1979*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer Verlag, Berlin, 1980.

D.L. Harkey, D.W. Reinfurt, and Knuiman M. Development of the bicycle compatibility index. *Transportation Research Record*, 1636:13–20, 1998.

R. Hartley. Vector optimal routing by dynamic programming. In P. Serafini, editor, *Mathematics of Multiobjective Optimization*, CISM International Centre for Mechanical Sciences – Courses and Lectures, pages 215–224. Springer Verlag, Wien, 1985.

R.V. Helgason and J.L. Kennington. Primal simplex algorithms for minimum cost network flows. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, volume 7, pages 85–133. Elsevier, 1995.

M. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25:281–291, 1985.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2001.

H.-J. Huang and Z.-C. Li. A multiclass, multicriteria logit-based traffic equilibrium assignment model under atis. *European Journal of Operational Research*, 176:1464–1477, 2007.

F. Huarng, P. Pulat, and Ravindran A. An algorithm for bicriteria integer network flow problem. In *Proceedings of the 10th International Conference on Multiple Criteria Decision Making, Taipei, Taiwan*, volume 3, pages 305–318, 1992.

F. Huarng, S. Pulat, and L.-H. Shih. A computational comparison of some bicriterion shortest path algorithms. *Journal of the Chinese Institute of Industrial Engineers*, 13(2):121–125, 1996.

H. Isermann. Proper efficiency and the linear vector maximum problem. *Operations Research*, 22:189–191, 1974.

A. Khan and F. Raciti. On time dependent vector equilibrium problems. In F. Giannessi and A. Maugeri, editors, *Variational Analysis and Applications*, pages 579–587. Springer, 2005.

D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale assignment, transportation, and minimum cost flow problems. *Management Science*, 20:814–821, 1974.

I. Konnov. A scalarization approach for vector variational inequalities with applications. *Journal of Global Optimization*, 32:517–527, 2005.

P. Korhonen, S. Salo, and R.E. Steuer. A heuristic for estimating nadir criterion values in multiple objective linear programming. *Operations Research*, 45:751–757, 1997.

Land Transport Safety Authority. Cycle network and route planning guide. available online, 2004. URL `http://www.landtransport.govt.nz/road-user-safety/walking-and-cycling/cycle-network/index.html`.

B.W. Landis, V.R. Vattikuti, R.M. Ottenberg, T.A. Petrisch, M. Guttenplan, and L.B. Crider. Intersection level of service for the bicycle through movement. *Transportation Research Record*, 1828:101–106, 2003.

T. Larsson, P.O. Lindberg, M. Patriksson, and C. Rydergren. On traffic equilibrium models with a nonlinear time/money relation. In M. Patriksson and M. Labbé, editors, *Transportation Planning*, pages 19–31. Kluwer, 2002.

G.M. Lee, D.S. Kim, B.S. Lee, and N.D. Yen. Vector variational inequality as a tool for studying vector optimization problems. *Nonlinear Analysis*, 34: 745–765, 1998.

H. Lee and P.S. Pulat. Bicriteria network flow problems. *European Journal of Operational Research*, 51:1190126, 1991.

H. Lee and P.S. Pulat. Bicriteria network flow problems: Integer case. *European Journal of Operational Research*, 66(1):148–157, 1993.

F. Leurent. Route choice and urban tolling: the Prado-Carénage tunnel in Marseille. *Recherche Transports Sécurité*, 71:21–23, 2001.

F. Leurent. Cost versus time equilibrium over a network. *European Journal of Operational Research*, 71:205–221, 1993.

F. Leurent. The practice of dual criteria assignment model with continuously distributed values-of-time. In *23rd European Transport Forum, 11-15 September 1995, Proceedings of Seminar E*, 1995.

F. Leurent. The theory and practice of a dual criteria assignment model with a continuously distributed value-of-time. In *Transportation and traffic theory: proceedings of the 13th International Symposium on Transportation and Traffic Theory, Lyon, France, 24-26 July 1996*, 1996.

F.M. Leurent. Multicriteria assignment modeling: Making explicit the determinants of mode or path. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, pages 153–174. Kluwer Academic Publishers, 1998.

T. Leventhal, G. Nemhauser, and J. Trotter. A column generation algorithm for optimal traffic assignment. *Transportation Science*, 7:168–176, 1973.

S.J. Li, X.Q. Yang, and G.Y. Chen. A note on vector network equilibrium principles. *Mathematical Methods of Operations Research*, 64:327–334, 2006.

S.J. Li, K.L. Teo, and X.Q. Yang. Vector equilibrium problems with elastic demands and capacity constraints. *Journal of Global Optimization*, 37:647–660, 2007.

S.J. Li, K.L. Teo, and X.Q. Yang. A remark on a standard and linear vector network equilibrium problem with capacity constraints. *European Journal of Operational Research*, 184:13–23, 2008.

A. Löbel. MCF version 1.3 - a network simplex implementation. Available for academic use free of charge via WWW at `http://www.zib.de/Optimization/Software/Mcf/`, 2004.

P. Marcotte. Advantages and drawbacks of variational inequalities formulations. In F. Gianessi and A. Maugeri, editors, *Variational Inequalities and Network Equilibrium Problems*, pages 179–194. Plenum Press, 1995.

P. Marcotte and D.L. Zhu. Equilibria with infinitely many differrentiated classes of customersl. In M.J. Ferris and J.-S. Pang, editors, *Complementarity and Variational Problems–State of the Art*, Proceedings of the International Conference on Complementarity Problems. siam, 1997.

E.Q.V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16:236–245, 1984.

E.Q.V Martins and J.C.N. Clímaco. On the determination of the nondominated paths in multipibjective network problem. *Methods of Operations Research*, 40:255–258, 1981.

E.Q.V Martins and J.L. Santos. The labelling algorithm for the multiobjective shortest path problem. Technical report, Universidade de Coimbra, Portugal, Departamento de Matématica, 2000. `http://www.mat.uc.pt/~eqvm/cientificos/investigacao/mo_papers.html`.

E.Q.V Martins, J.M. Paixão, M.S. Rosa, and J.L. Santos. Ranking multiobjective shortest paths. Technical Report 07–11, Universidade de Coimbra, 2007.

M.G. McNally. The four-step model. In D.A. Hensher and K.J. Button, editors, *Handbook of Transport Modelling*, volume 1, pages 35–52. Pergamon, 2000.

P. Modesti and A. Sciomachen. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111:495–508, 1998.

J. Mote, I. Murthy, and D. L. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53:81–92, 1991.

M. Müller-Hannemann and K. Weihe. On the cardinality of the Pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147: 269–286, 2006.

I. Murthy and D. L. Olson. An interactive procedure using domination cones for bicriterion shortest path problems. *European Journal of Operational Research*, 72:417–431, 1994.

A. Nagurney. *Network Economics A Variational Inequality Approach*, volume 1 of *Advances in Computational Economics*. Kluwer Academic Publishers, 1993.

A. Nagurney. A multiclass, multicriteria traffic network equilibrium model. *Mathematical and Computer Modelling*, 32:393–411, 2000.

A. Nagurney and J. Dong. A multiclass, multicriteria traffic network equilibrium model with elastic demand. *Transportation Research B*, B 36:445–469, 2002.

A. Nagurney, J. Dong, and P.L. Mokhtarian. Teleshopping versus shopping: A multicritieria network equilibrium framework. *Mathematical and Computers Modelling*, 34:783–798, 2001.

A. Nagurney, J. Dong, and P.L. Mokhtarian. Multicriteria network equilibrium modeling with variable weights for decision-making in the information age with applications to telecommuting and teleshopping. *Journal of Economic Dynamics & Control*, 26:1629–1650, 2002a.

A. Nagurney, J. Dong, and P.L. Mokhtarian. Traffic network equilibrium and the environment. a multicriteria decision-making perspective. In E.J. Kontoghiorghes, B. Rustem, and S. Siokos, editors, *Computational Methods in Decision-Making, Economics and Finance*, pages 501–523. Kluwer Academic Publishers, 2002b.

W. Oettli. Necessary and sufficient condtitions of wardrop type for vectorial traffic equilibria. In F. Gianessi and P.M. Maugeri, A. amd Pardalos, editors, *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models*, pages 223–229. Kluwer, 2001.

J.de D. Ortúzar and L.G. Willumsen. *Modelling Transport*. Wiley, 3 edition, 2002.

J.M. Paixão and J.L. Santos. Labelling methods for the general case of the multi-objective shortest path problem - a computational study. Technical Report 07–42, Universidade de Coimbra, 2007.

J.M. Paixão and J.L. Santos. A new ranking path algorithm for the multi-objective shortest path problem. Technical Report 08–27, Universidade de Coimbra, 2008.

S. Pallottino and M.G. Scutellà. Shortest path algorithms in transportation models: classical and innovative aspects. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, pages 245–281. Kluwer Academic Publishers, 1998.

D. Palmer, E. Ai-Uzaizi, J. Campbell, S. Killips, V. Lawson, S. Reid, J. Lee, S. Mason, T. McLoughlin, P. Noble, P. Philippou, T. Russell, and B. Sabey. Guidelines for cycle audit and cycle review. Published by The Institution of Highways & Transportation, 1998.

U. Pape. Implementation and efficiency of Moore-algorithms for the shortest route problem. *Mathematical Programming*, 7:212–222, 1974.

M. Patriksson. *The Traffic Assignment Problem Models and Methods*. VSP, 1994. URL `http://www.math.chalmers.se/~mipat/traffic.html`.

D. Philip. An algorithm for the biobjective integer transportation problem. In *Proceedings of the 43rd Annual Conference ORSNZ'08*, 2008.

W.B. Powell and Y. Sheffi. The convergence of equilibrium algorithms with predetermined step sizes. *Transportation Science*, 16:45–55, 1982.

A. Przybylski, X. Gandibleux, and M. Ehrgott. The biobjective integer minimum cost flow problem–incorrectness of Sedeño-Noda and González-Martín's algorithm. *Computers & Operations Research*, 33(5):1459–1463, 2006.

A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multiobjective integer programming and its application to the assignment problem with three objectives. Technical report, LINA - Laboratoire d'Informatique de Nantes Atlantique, Université de Nantes, 2007.

A. Przybylski, X. Gandibleux, and M. Ehrgott. Two-phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509–533, 2008.

A. Przybylski, X. Gandibleux, and M. Ehrgott. Two recursive algorithms for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 2009. To appear.

P.S. Pulat, F. Huarng, and H Lee. Efficient solutions for the bicriteria network flow problem. *European Journal of Operational Research*, 19(7):649–655, 1992.

R.E. Quandt. *Studies in Travel Demand*, chapter A probabilistic abstract mode model, pages 127–149. Mathematica, Inc., Princeton, NJ, 1967.

F. Raciti. Equilibrium conditions and vector variational inequalities: a complex relation. *Journal of Global Optimization*, 40:353–360, 2008.

A. Raith. A comparison of solution strategies for biobjective shortest path problems. In *Proceedings of the 41st Annual Conference ORSNZ'06*, pages 101–111, Christchurch, 2006.

A. Raith. Traffic assignment with travel time and toll cost objectives. In *Proceedings of the 43rd Annual Conference ORSNZ'08*, pages 42–51, Wellington, 2008a.

A. Raith. Bounded labelling for biobjective shortest path problems. Technical report, The University of Auckland, 2008b.

A. Raith and M. Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36:1299–1331, 2009a. doi: 10.1016/j.cor.2008.02.002.

A. Raith and M. Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36:1945–1954, 2009b. doi: 10.1016/j.cor.2008.06.008.

A. Raith and M. Ehrgott. Solving the biobjective integer minimum cost flow problem. In *Proceedings of the 42nd Annual Conference ORSNZ'07*, pages 99–108, Auckland, 2007.

A. Raith, C. Van Houtte, J.Y.T. Wang, and M. Ehrgott. Applying bi-objective shortest path methods to model cycle route-choice. In *ATRF 2009 Proceedings*, 2009.

G. Ruhe. Complexity results for multicriteria and parametric network flows using a pathological graph of Zadeh. *Zeitschrift für Operations Research*, 32: 9–27, 1988.

V.N. Sastry, T.N. Janakiraman, and S.I. Mohideen. New algorithms for multi objective shortest path problem. *Opsearch*, 40(4):278–298, 2003.

H. Schittenhelm. On the integration of an effective assignment algorithm with path and path-flow management in a comined trip distribution and traffic assignment algorithm. In *Proceedings of Seminar H held at the PTRC Transport and Planning Summer Annual Meeting*, pages 203–214, University of Sussex, 1990.

M. Schneider. *Urban Development Models*, chapter Access and land development, pages 164–177. Highway Research Board Special Report, 1968.

D. Schultes. Tiger Road Networks for 9th DIMACS Implementation Challenge – Shortest Path. `http://www.dis.uniroma1.it/~challenge9/data/tiger/`, 2005.

A. Sedeño-Noda and C. González-Martín. The biobjective minimum cost flow problem. *European Journal of Operational Research*, 124:591–600, 2000.

A. Sedeño-Noda and C. González-Martín. An algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 28 (2):139–156, 2001.

A. Sedeño-Noda and C. González-Martín. Am alternative method to solve the biobjective minimum cost flow problem. *Asia-Pacific Journal of Operational Research*, 20:241–260, 2003.

A. Sedeño-Noda, C. González-Martín, and J. Gutiérrez. The biobjective undirected two-commodity minimum cost flow problem. *European Journal of Operational Research*, 164:89–103, 2005.

P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent advances and historical development of vector optimization*, volume 294 of *Lecture Notes in Economics and Mathematical Systems*, pages 222–232. Springer Verlag, Berlin, 1986.

Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, 1985.

A.J.V. Skriver. A classification of bicriterion shortest path (BSP) algorithms. *Asia-Pacific Journal of Operational Research*, 17:199–212, 2000.

A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27: 507–524, 2000.

M.J. Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research B*, 13:295–304, 1979.

B.S. Stewart and C.C. White. Three solution procedures for multiobjective path problems. *Control Theory and Advanced Technology*, 5:443–470, 1989.

M.A. Stinson and C.R. Bhat. Commuter bicylist route choice – analysis using a stated preference survey. *Transportation Research Record*, 1828:107–115, 2003.

Z. Tarapata. Selected multicritieria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17:269–287, 2007.

G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. In *17th International Symposium, ISAAC 2006, Kolkata, India, December 18-20, 2006. Proceedings. Algorithms and Computation*, pages 389–398, 2006.

C.T. Tung and K.L. Chew. A bicriterion Pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5:166–172, 1988.

C.T. Tung and K.L. Chew. A multicriteria Pareto-optimal path algorithm. *European Journal of Operational Research*, 62:203–209, 1992.

E. L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1995.

US Census. US Census 2000 TIGER/Line Files. `http://www.census.gov/geo/www/tiger/tigerua/ua_tgr2k.html`, 2000.

J.Y.T. Wang, A. Raith, and M. Ehrgott. Tolling analysis with biobjective traffic assignment. *Accepted for publication in MCDM08 proceedings*, 2008.

A. Warburton. Approximation of Pareto optima in multiple-objective shortest path problems. *Operations Research*, 35(1):70–79, 1978.

J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II*, 1:325–378, 1952.

M.A. Weiss. Binary heap implementation in c. online, last visited 02/2009. URL `http://www.cs.sunysb.edu/~algorith/implement/weisses/distrib/c-implementation/`. Files binheap.c and binheap.h.

L.G. Willumsen. Travel networks. In D.A. Hensher and K.J. Button, editors, *Handbook of Transport Modelling*, volume 1, pages 165–180. Pergamon, 2000.

L.A. Wolsey. *Integer Programming*. Wiley, 1998.

X.Q. Yang and C.J. Goh. Vector variational inequalities, vector equilibrium flow and vector optimization. In F. Gianessi, editor, *Vector Variational Inequalities and Vector Equilibria*. Kluwer, 2000.

X.Q. Yang and C.J. Goh. On vector variational inequalities: Applications to vector equilibria. *Journal of Optimization Theory and Applications*, 95: 431–443, 1997.

X.Q. Yang and H. Yu. Vector variational inequalities and dynamic traffic equilibria. In F. Giannessi and A. Maugeri, editors, *Variational analysis and applications*, pages 1141–1157. Springer, 2005.

J.Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, 27 (4):526–530, 1970.

P.L. Yu. *Multiple Criteria Decision Making*. Plenum Press, 1985.

F.B. Zhan and C. E. Noon. Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998.