University of Southampton
Faculty of Social Sciences
Mathematical Sciences

A trust-region method for nonlinear bilevel optimisation problems with an application in transportation

by

Laura Murray

A thesis submitted for the degree of
Doctor of Philosophy

Jan 2022

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL SCIENCES
MATHEMATICAL SCIENCES

Thesis for the degree of Doctor of Philosophy

A TRUST-REGION METHOD FOR NONLINEAR BILEVEL OPTIMISATION PROBLEMS WITH AN APPLICATION IN TRANSPORTATION

by Laura Murray

A bilevel optimisation problem is an optimisation problem which has a second optimisation problem embedded in its constraints. It aims to model problems and decision processes that are hierarchical, which are problem structures that occur frequently in real-life. Thus, due to the wide range of applications of bilevel problems, there is a strong motivation to solve them. The aim of this thesis is to develop an approach to solving bilevel programs by utilising the less commonly used optimal value reformulation. The work can be split into two main contributions. First, a novel trust-region approach to solving nonlinear bilevel problems is proposed, which solves an exact penalisation of the optimal value reformulation. Second, an application of bilevel programming to the London congestion pricing problem is explored, investigating the application of the proposed trust-region method to solve a bilevel formulation of the road pricing problem.

One of the most common approaches to solving a bilevel program is to first transform the problem into a single level program. The most popular way of doing so is by replacing the lower level problem by its Karush-Kuhn Tucker conditions. That being said, the reformulation requires strong assumptions on the bilevel program for it to be equivalent to the original problem. An alternative method to transform the problem into a single level problem is to use the optimal value function of the lower level problem. This problem is known to be fully equivalent. However, due to the difficulties in solving it, approaches in the literature that utilise this reformulation are relatively scarce. Under a regularity condition known as the partial calmness condition, an exact penalty problem can be built from the optimal value reformulation. The first contribution of this thesis is the investigation of solving this exact penalty problem to find local solutions of the associated bilevel problem. A novel trust-region algorithm is proposed to solve it, and convergence analysis is explored. The implementation and performance of the algorithm is investigated via extensive numerical

experiments on a large test set of nonlinear bilevel problems. This provides strong numerical results that validate the approach for solving bilevel problems. Based on the results and analysis, the performance and limitations of the algorithms are discussed.

The second contribution of this thesis is exploring the application of the aforementioned trust-region method on the bilevel optimisation formulation of the road pricing problem. Road pricing is a method used by governments to alleviate congestion in an overcrowded network. The problem has a hierarchical structure, and therefore naturally forms as a bilevel program. We investigate a case study of the problem to the London congestion zone charge: a fixed cordon road pricing scheme implemented in the center of London. Although successful on initial implementation in 2003, congestion in the city has returned to pre-charge levels. Due to recent advances in technology, the Mayor of London is looking to update the congestion charge to a more sophisticated tolling scheme that can charge for distance, time, emissions or other factors. A formulation of the London congestion problem as a bilevel program is presented, which considers the aims and objectives set out by the current Mayor of London. We then show how the trust-region algorithm can be applied to solve a simplified form of the road pricing model commonly seen in the literature. This is a novel approach to the problem, solving a single level continuous exact penalty problem to find local solutions of the road pricing bilevel model. The performance of the algorithm is tested and verified numerically on three network examples of varying sizes, and the efficiency and robustness of the algorithm is assessed.

Table of Contents

List of Figures

List of Tables

Declaration of Authorship

I, Laura Murray, declare that this thesis entitled *A trust-region method for nonlinear bilevel optimisation problems with an application in transportation* and the work presented in it are my own and have been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others, this is always clearly attributed;

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

5. I have acknowledged all main sources of help;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. None of this work has been published before submission.

Signed:

Date:

Acknowledgements

First and foremost, I would like to thank my supervisors Jörg Fliege and Alain Zemkoho, for introducing me to the interesting topic of bilevel optimisation and providing me with their expertise. Their invaluable help and guidance throughout my PhD no doubt has got me to where I am today.

A big thank you to my fellow PhD students, and especially to Karl, Simos, Ruth, Marton, Lily, and all the other "nice maths folk". My PhD experience would not have been the same without the many coffee breaks and crossword puzzles.

I am extremely grateful to my friends and family who have supported me throughout my research. The work in this thesis could not have been achieved without their support. A special thanks to my sisters Jo and Molly, for being there when I needed them, and to my parents and brother, for their love and support. To Odette and Hannah, who bring sunshine to my life. To Emma, Hope and Maddie for helping me through my time at the University of Southampton. And finally, to Joe, who I am eternally grateful for. Thank you for always being there for me, supporting me, and bringing happiness to each and every one of my days.

Lastly, I would like to express my sincere gratitude to The Engineering and Physical Sciences Research Council (EPSRC), who provided the funding for this research.

Nomenclature

Abbreviations

BLPP	Bilevel Programming Problem
BLTR	Trust-region algorithm to solve bilevel problems that uses the model $P_k(x, y)$ for the trust-region subproblem
BLTR-Q	Trust-region algorithm to solve bilevel problems that uses the model $\tilde{P}_k(x, y)$ for the trust-region subproblem
BLTR-L	Trust-region algorithm to solve bilevel problems that uses the model $\bar{P}_k(x, y)$ for the trust-region subproblem
BOLIB	Bilevel Optimisation LIBrary of test problems
BPR	Bureau of Public Roads
CCP	Convex-Concave Procedure
DC	Difference of Convex functions
EDO	Equilibrium Decomposition Optimization
KKT	Karush-Kuhn Tucker
LICQ	Linearly Independent Constraint Qualification
LLVF	Lower Level Value Function
MFCQ	Mangasarian-Fromowitz Constraint Qualification
MPCC	Mathematical Optimisation problem with Complementarity Constraints
SNKKT	Semi-smooth Newton type method for bilevel problems using the KKT reformulation
SNLLVF	Semi-smooth Newton type method for bilevel problems using the LLVF reformulation
SOSC	Second Order Sufficient Condition

TfL	Transport for London
VOT	Value of Time

Vectors

$x \in \mathbb{R}^n$	column vector in $\mathbb{R}^n$
x_i	the i-th component of x
x^T	transpose of vector x
$x^T y$	standard inner product of x and y with $x, y \in \mathbb{R}^n$
(x, y)	column vector $(x^T, y^T)^T$
$\|x\|$	any norm of x
$\|x\|_\infty$	infinity norm of x

Spaces

$\mathbb{R}$	the real numbers
$\mathbb{R}^n$	the n-dimensional real vector space

Mathematical functions and sets

$\nabla v(x)$	the gradient of the function $v : \mathbb{R}^n \to \mathbb{R}$ at x
$\nabla_x v(x, y)$	the gradient of the function $v : \mathbb{R}^n \to \mathbb{R}$ with respect to x
$v'(x; r)$	the directional derivative of the function $v : \mathbb{R}^n \to \mathbb{R}$ at x
$\partial v(x)$	the Clarke generalised gradient of the function $v : \mathbb{R}^n \to \mathbb{R}$ at x
$v^o(x; r)$	the Clarke directional derivative of the function $v : \mathbb{R}^n \to \mathbb{R}$ at x
$f \circ g$	the composite function of the functions f and g
conv Ω	convex hull of the set Ω
$\Omega_1 \subseteq \Omega_2$	Ω_1 is a subset of Ω_2
$\|\Omega\|$	cardinality of the set Ω

Functions, sets and variables of the general formulation of a bilevel program

$x \in \mathbb{R}^n$	the upper level variables
$y \in \mathbb{R}^m$	the lower level variables
$F(x, y)$	the upper level objective function

$f(x, y)$	the lower level objective function
K	constraint set of the bilevel program
$S(x)$	the feasible set mapping for the followers problem
$\Psi(x)$	the followers rational reaction set
$\varphi(x)$	the lower level optimal value function
γ	the exact penalty parameter
$\mathcal{L}(x, y, \lambda, \mu)$	$f(x, y) + \lambda^T g(x, y) + \mu^T h(x, y)$ the Lagrangian function of the lower level problem
$\Lambda(x, y)$	$\{(\lambda, \mu) \in \mathbb{R}^p \times \mathbb{R}^q : \lambda \geq 0,\ \lambda^T g(x, y) = 0,\ \nabla_y \mathcal{L}(x, y, \lambda, \mu) = 0\}$ the lower level Lagrange multipliers set

Sets for the road pricing bilevel model

A	set of links in network
$\bar{A}$	subset of tolled links in network
I	set of cordon zones
M	set of user classes
N	set of nodes in network
R	set of routes in the network
R_w	set of routes between O-D pair $w \in W$
W	set of O-D pairs

Variables for the road pricing bilevel model

q_{rw}	route flow on route $r \in R_w$
q_{rw}^m	route flow on route $r \in R_w$ for user $m \in M$
v_{aw}	link flow on link $a \in A$ between O-D pair $w \in W$
v_a	total link flow on link $a \in A$
v_a^m	total link flow on link $a \in A$ for user $m \in M$
λ_a	queueing delay on link $a \in A$
τ_a	toll charge for link $a \in \bar{A}$
τ_a^m	toll charge for link $a \in \bar{A}$ for user $m \in M$
τ_{rw}	toll charge for route $r \in R_w$

τ_{rw}^m	toll charge for route $r \in R_w$ for user $m \in M$
$\bar{\tau}_{rw}^m$	toll charge for route $r \in R_w$ for user $m \in M$ that includes both a distance and delay charge
τ^i	toll charge to use cordon $i \in I$

Functions for the road pricing bilevel model

$c_a(v_a, \tau_a)$	cost function of arc a
$c_a^m(v_a^m, \tau_a^m)$	cost function of arc a for user $m \in M$
$c_{rw}(q_{rw}, \tau_{rw})$	cost function of route $r \in R_w$
$c_{rw}^m(q_{rw}^m, \tau_{rw}^m)$	cost function of route $r \in R_w$ for user $m \in M$
$d_w = g_w(\pi)$	the elastic demand function of the cheapest route costs π
$t_a(v_a)$	travel time function on link $a \in A$
$t_a^m(v_a)$	travel time function on link $a \in A$ for user $m \in M$
$t_{rw}^m(q_{rw})$	travel time function on route $r \in R_w$ for user $m \in M$
$\Phi^m(l)$	nonlinear distance toll charge function

Constants for the road pricing bilevel model

C_a	capacity of link $a \in A$
D_w	fixed demand vector for O-D pair $w \in W$
$\bar{d}_w^m$	leaders desired level of demand for user $m \in M$ between O-D pair $w \in W$
$\bar{D}$	leaders desired level of demand for whole network
l_a	length of route $a \in A$
l_{rw}	length of route $r \in R_w$
l_{rw}^i	length of route $r \in R_w$ in cordon $i \in I$
t_a^0	free-flow traffic time on link $a \in A$
β^m	value of time for user $m \in M$
$\Delta^t = (\delta_{ar})$	link-route incidence matrix
θ^m	operating cost for user $m \in M$

Chapter 1

Introduction

A bilevel programming problem is an optimisation problem which has a second optimisation problem embedded in its constraints. It aims to model real-life problems where there are two decision makers, whom act sequentially. Thus it is of a hierarchical nature, where a higher level authority makes a decision first, known as the leader, but whose optimal decision depends on the choice of the second decision maker, known as the follower. The motivation behind the research of bilevel programs can largely be attributed to the great number of real-world applications. The hierarchical structure naturally occurs in many real-world problems, including areas within transportation, defense, management, and optimal design. In fact, the first introduction of a bilevel programming problem was as an application in economics, in a game theory problem known as the Stackelberg games.

Although there is a strong motivation to solve bilevel programming problems, the hierarchical structure makes them complex and difficult to handle mathematically. They are found to NP-hard, and are in general nonconvex and nonsmooth. Therefore finding a global solution can be challenging. The focus of this research is to investigate novel solution approaches for the bilevel problem, and explore an application of bilevel programs within transportation.

1.1 Objectives

To solve bilevel programs, a large number of solution approaches first reformulate the problem into a single level problem. The most common way to do this is via the Kurush-Kuhn-Tucker (KKT) reformulation, which replaces the lower level problem with its KKT conditions. This reformulation however has its limitations. It has recently been shown by Dempe and Dutta [26] that the problem is only equivalent to the bilevel program when the lower level problem is convex and satisfies the Slater's constraint qualification. These assumptions are relatively strong. On the other hand, the optimal value reformulation of the bilevel problem is shown to be fully equivalent, first introduced by Outrata [94]. That being said, the literature on solution algorithms that solve bilevel problems using the optimal value reformulation is relatively scarce, due to the difficulties in solving it. This is because of the optimal value function, which is in general a nonsmooth and nonconvex function that usually cannot be explicitly given. Nonetheless, due to the relatively weak assumptions required for equivalence to the bilevel program, the approach holds potential for solving more complex nonlinear bilevel programs. In addition, under a regularity condition known as the partial calmness condition, an exact penalty problem can be built from the optimal value reformulation. The main objective of the work presented in this thesis is to develop a new approach to solving nonlinear bilevel programs using the optimal value function reformulation. In particular, we want to investigate how the exact penalty problem built from the optimal value reformulation can be solved to find local solutions of the associated bilevel problem.

The second objective of the work presented in this thesis is to investigate how bilevel programming can be applied to the road pricing problem, with the aim of solving the problem using the solution approach developed for bilevel problems via the optimal value function. The London congestion zone is a fixed cordon tolling scheme implemented by Transport for London (TfL) in the center of London, where users incur a fixed toll to use roads within the cordon for the day. It is a form of road pricing: a common method implemented by governments on overcrowded roads with the aim to alleviate negative externalities such as congestion and pollution. Upon setting a toll on the road network, users of the road network take the route that incurs the least cost to themselves. Road pricing problems can be formed as a bilevel program as, due to their hierarchical structure, they naturally form as one. Although successful on initial implementation, congestion in London has risen to pre-charge prices. The advancement of technology means there is potential to implement a more sophisticated toll scheme that can charge for distance, time, emissions, road danger and other factors. We investigate formulating a bilevel program that models an updated tolling scheme for the London congestion zone, which considers the aims and objectives set

out by the current Mayor of London.

1.2 Contributions

The main contributions of this thesis are:

1. We propose a novel trust-region method to solve nonlinear bilevel problems. The algorithm solves an exact penalty problem built from the optimal value reformulation under the partial calmness condition. The proposed method is tested experimentally via extensive numerical experiments on a large test set of nonlinear bilevel problems, with promising results that validate the approach for solving bilevel problems. Convergence analysis is presented under strong assumptions.

2. By investigating the exact penalty problem built from the optimal value reformulation, we show how under strong assumptions the problem becomes a difference of convex functions (DC) problem. A known DC algorithm called the convex-concave procedure is applied to solve it, and the approach is implemented on a test set of bilevel problems with fully convex lower level objective functions.

3. We present a formulation of a road pricing bilevel program that models the London congestion zone charge and the current problems it faces. This involves a model that suggests an updated cordon tolling scheme which charges for the distance travelled and the level of congestion on a route, and that considers the aims and objectives the current mayor of London is pursuing.

4. We propose a novel method to solve a simplified road pricing bilevel model commonly seen in the literature. We show that local solutions to the problem can be found by solving a single level continuous exact penalty problem and apply the proposed trust-region algorithm. Results on the performance of the algorithm tested on three network examples are presented, supporting the convergence of the algorithm to a stationary point of the road pricing bilevel program.

1.3 Structure of thesis

The structure of the thesis is as follows.

Chapter 2

This chapter introduces bilevel programming, providing an overview of the problem and the current solution approaches taken to solve it. To begin, we define the general formulation of the bilevel problem and notation used throughout this thesis, along with a discussion on the structure and complexities of the problem. Following this, we give a short summary on some of the wide ranges of applications to bilevel programming. The remaining of the chapter gives a review on the different solution approaches that have been taken to solve the problem. Commonly, to solve the bilevel program it is first transformed into a single level problem. The well-used KKT reformulation and some of its limitations are discussed, before we give a detailed introduction on the optimal value reformulation. This includes a discussion on a regularity condition known as the partial calmness condition. Finally, we give a comprehensive literature review on the solution approaches and algorithms designed to solve continuous bilevel programs. It contains an in-depth review on algorithms developed that use the optimal value reformulation, and a briefer coverage on other classical solution methods.

Chapter 3

Based on the optimal value reformulation of the bilevel program introduced in Chapter 2, this chapter presents a trust-region method to solve continuous nonlinear bilevel programs. We begin the chapter with a comprehensive account on the properties of the optimal value function. We then give a short introduction on the classical trust-region method for smooth problems, before presenting the proposed trust-region method to solve bilevel problems. This focuses on solving an exact penalty problem built from the optimal value reformulation of the bilevel program. The algorithm is accompanied with convergence analysis under strong assumptions on the bilevel problem. In addition, we show how a DC algorithm known as the convex-concave procedure (CCP) can be applied to solve the exact penalty problem. We analyse the performance of the proposed trust-region algorithm and the CCP heuristic, presenting extensive numerical results on 124 nonlinear bilevel problems taken from the bilevel literature. The performance of the algorithms are assessed via comparison to known solutions of the problems and to a number of other bilevel algorithms from the literature. Further analysis on the sensitivity of the algorithms to the choice of parameters and starting points are provided.

Chapter 4

This chapter investigates an application of bilevel programming within transportation, looking at a case study of the London congestion zone charge. We begin the chapter with an introduction to the road pricing problem. This is followed by a literature review on how static bilevel models for the road pricing problem have been formulated and the solution methods developed to solve them. We then give an account of the London congestion zone charge, which currently implements a fixed tolling scheme in the center of London. This includes a discussion on the aims and objectives the current Mayor of London has set out. Considering these, we present a bilevel model that encapsulates the current problem that the London congestion zone faces. The model introduces an updated tolling scheme which charges users for the distance travelled within the cordon and the level of congestion on the road. Finally, by simplifying a number of the complexities in the road pricing bilevel model, it is shown that under the assumptions made the optimal value function becomes differentiable. Using this property, we show how the trust-region algorithm proposed in Chapter 3 can be used to solve the problem. Numerical results on three network examples are presented to assess its performance.

Chapter 5

The final chapter presents a summary of the conclusions drawn from the work presented in the preceding chapters and provides suggestions for future work.

Chapter 2

An overview of bilevel programming

Bilevel programming is two-level mathematical program. This means it is an optimisation problem which has a second optimisation problem embedded into its constraints. There are a wide range of real-world applications of bilevel optimisation, driving the strong motivation to find solution approaches to solve them. However, due to the complex structure of the problem, they are challenging to solve.

This chapter provides an overview of bilevel optimisation. We first introduce the problem, defining the general formulation and notation of the problem used throughout this thesis. We also note some of the complexities that can arise from the structure of the problem, before giving a short description on some of the applications of bilevel programming. Next, we discuss the solution approaches taken to tackle the problem, describing the methods taken to transform the problem into a single level problem and the necessary optimality conditions. We end the chapter with a comprehensive literature review on the solution algorithms designed to solve bilevel optimisation.

2.1 Introduction to bilevel programming

The beginnings of bilevel optimisation came from an application to a problem in game theory, known as the Stackelberg games. They were introduced by Stackelberg in 1952 [113], and were formed to describe an oligopolistic market economy concerning a leader and follower firms. The firms are competing and need to make a decision on the quantity of products to optimise profits. The leader makes the first decision, looking to optimise their own objective,

whilst taking into account the follower's response, leading to a hierarchical optimisation problem. This problem can be seen as an example of a bilevel problem, however, the first introduction of bilevel programming into the mathematical community was by Bracken and McGill [13] in 1973, known then as "Mathematical programs with optimisation problems in the constraints". It was later referred to as bilevel optimisation by Candler and Norton [16].

The general formulation of the bilevel program, and the notation maintained throughout this thesis, is as follows. Let the upper level decision variables be $x \in \mathbb{R}^n$ and the lower level decision variables be $y \in \mathbb{R}^m$. Then the general form of the bilevel programming problem (BLPP) is

$$\begin{aligned} \min_{x,y} \quad & F(x,y) \\ \text{subject to} \quad & G(x,y) \leq 0, \\ & \qquad\qquad \min_{y} \quad f(x,y) \\ & y \text{ solves} \quad \text{subject to} \quad g(x,y) \leq 0, \\ & \qquad\qquad\qquad\quad\ \ h(x,y) = 0. \end{aligned} \tag{2.1}$$

The outer level problem is known as the upper level problem, representing the leaders problem, whilst the nested problem is known as the lower level problem, representing the followers problem. The upper level objective function is defined by $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and the lower level objective function by $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$. Similarly we have the upper level constraint function $G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$, and the lower level constraint functions $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$, $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$. We could also separate the upper level equality constraints if required, defining the upper level equality constraint function by $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^l$.

We define the following sets of the bilevel problem.

- The constraint region is defined as

$$K = \{(x,y) : G(x,y) \leq 0,\ g(x,y) \leq 0,\ h(x,y) = 0\}. \tag{2.2}$$

- The followers rational reaction set mapping $\Psi : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is defined as

$$\Psi(x) = \arg\min_{y} \{f(x,y) : g(x,y) \leq 0,\ h(x,y) = 0\}. \tag{2.3}$$

- The feasible set mapping for the followers problem $S : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ is defined as

$$S(x) = \{y \in \mathbb{R}^m : g(x,y) \leq 0,\ h(x,y) = 0\}. \tag{2.4}$$

From the definition of the sets, we can see a feasible solution to the bilevel program is one that satisfies the upper level constraints and is optimal for the lower level problem. We also see that the lower level problem is a parametric optimisation problem, where the upper level decision variables are parameters in the lower level problem. Throughout this thesis, it is assumed that the feasible constraint region is nonempty and compact, and that for all x there exists a y that the follower can respond with, that is $\Psi(x)$ is nonempty.

To solve the bilevel problem, we want to find the best solution (x^*, y^*) that gives the best possible function value for the upper level objective function F, and where $(x^*, y^*) \in K,\ y \in \Psi(x^*)$. When the lower level optimal solution is uniquely determined for all parameter values, that is the rational reaction set $\Psi(x)$ is single valued, then the problem we want to solve is the following

$$\min_x \{F(x, y) :\ G(x, y) \leq 0,\ y \in \Psi(x)\}. \tag{2.5}$$

However, if the set $\Psi(x)$ is not a singleton for some x, it leads to some ambiguity in the computation of the upper level objective function value. Essentially, the upper level objective function value becomes an element in the set $\{F(x, y) :\ y \in \Psi(x)\}$. When this is the case, an assumption on cooperation between the leader and follower needs to be made. There are two main solutions that are considered in the literature: the optimistic and the pessimistic solution concept.

- The *optimistic* bilevel program assumes that the leader can have some form of influence on the follower, so that for each upper level decision x that the leader makes, the follower will choose the solution from $\Psi(x)$ that is best for the leader. Thus, the choice function for the follower is

$$\Psi^O(x) = \arg\min_y \{F(x, y) : y \in \Psi(x)\}.$$

- The *pessimistic* bilevel program assumes the leader cannot influence the follower, and so we assume that the leader will choose the x that bounds the damage from an unfavourable selection of y. It can be seen as a risk averse approach, as the leader protects themselves against the worst possible scenario. We therefore assume the follower chooses the optimal solution that is least preferable for the leader, and thus the choice function for the follower is

$$\Psi^P(x) = \arg\max_y \{F(x, y) : y \in \Psi(x)\}.$$

Under the optimistic case an optimal solution can be guaranteed under fairly reasonable assumptions, described in Section 2.4. On the other hand, the pessimistic bilevel problem is often far more complicated and harder to deal with. It is difficult to reduce the pessimistic problem into a single level program and stronger assumptions are required to guarantee the existence of an optimal solution. This thesis looks at solving the optimistic bilevel program. For more on the properties and optimality conditions of the pessimistic bilevel program see [33], [122].

The hierarchical structure of bilevel programs makes them complex and difficult to work with. They are in general nonconvex and nonsmooth, and as such local minima can exist. They have also been shown to be NP-hard [38]. In fact, linear bilevel problems, which can be seen as the simplest form of a bilevel problem, have been shown to be strongly NP-hard [39]. We refer to a linear bilevel problem as one where all functions in the problem are linear. As an example of how even the most simple bilevel problems can become complicated, Calamai and Vicente [15] construct linear bilevel examples which have an exponential number of local minima. To simplify the problem, a number of algorithms investigate the case when the rational reaction set $\Psi(x)$ is single valued. For example, this is the case when the lower level objective function $f(x, y)$ is assumed to be strictly convex in y for fixed x [7].

2.2 Applications

Bilevel programs can be used to model many practical problems where a hierarchical decision or action occurs. As discussed, the Stackelberg games were one of the first applications of a bilevel problem, applied to an economic problem involving an oligopolistic market. There have been many other economic applications of bilevel programming, for example the principal-agent problem [38]. In this problem the follower, known as an agent, takes actions or makes a decision on behalf of the leader, known as the principal. The problem arises when the follower acts on their own best interest, an interest that may not align with the leader. This problem can be modelled as a bilevel problem due to the hierarchical nature.

The area of transportation has seen many applications of bilevel programming, due to the hierarchical processes within transportation planning. In general, transportation planning involves a governing body, acting on behalf of society, who wish to make a decision to improve efficiency of a road network, whilst users of the road network make decisions such as where they want to go, which route they will travel on, and what mode of transport to take. This can depend on factors such as travel time and monetary costs. Examples of this are in road-pricing problems, which look at easing congestion by imposing a toll on a road or cordon; in network design problems, which look at improving transport networks by

increasing capacity such as adding roads or bike lanes into the network; and in the signal setting problem, which looks at increasing the efficiency of a road network by trying to utilise the existing infrastructure in a better way. This thesis gives in-depth detail and discussion on the road-pricing problem. Further reading on the other applications of bilevel problems in transportation can be found in a paper by Migdalas [87].

Many defense problems form as a hierarchical structure and hence can be formulated as a bilevel program. For example, when defending we may seek to minimise the maximum damage from an attacker, yet when offending we may seek to maximise the damage to an opponent, whilst taking into account their reactions. Engineering has seen many bilevel applications, such as in optimal design of a structure or inverse control theory. Other sectors with common applications include the energy sector, for example within electric power pricing or oil production; environmental economics, for example taxation of an organization that is polluting the environment; and chemistry, for example when looking to optimise chemical equilibria. Other problems which can be formed as bilevel problems include problems within facility location, revenue management and machine learning. Details of research into these applications can be found in the following review papers [109], [19] and the book [32].

2.3 Transformation to a single level problem

To solve the bilevel problem, a large majority of solution methods first transform the problem into a single level problem, with the aim to then apply known optimisation methods. The most commonly used transformations are the following.

1. *The Karush-Kuhn-Tucker (KKT) reformulation.* This replaces the lower level problem by its KKT optimality conditions, which are well known first order necessary conditions for optimal solutions.

2. *The optimal value reformulation.* By using the optimal value function of the lower level objective function, the lower level problem can be formed into a set of constraints.

3. *The primal KKT reformulation.* This replaces the lower level problem with a generalised equation that can be considered a compact form of the KKT conditions of the lower level problem.

4. *Implicit function theorems.* If there is a unique global solution for each $x \in X$, we can form an implicit function $y(x)$ which can be inserted into the problem.

The reformulations typically require some constraint qualification of the bilevel problem

or lower level problem to be satisfied in order for the solution sets of both problems to be equivalent. A description of these reformulations can be found in [38], [37]. We shall take a closer look at two of the most common reformulation approaches: the Karush-Kuhn-Tucker reformulation and the optimal value reformulation.

2.3.1 The Karush-Kuhn-Tucker reformulation

The KKT reformulation is the most commonly used transformation in the literature, and a vast amount of bilevel research focuses on solving this problem. The reformulation requires relatively strong regularity conditions for the lower level problem to be satisfied at all feasible points, and requires all lower level functions to be continuously differentiable and convex [19]. By replacing the lower level problem with its KKT optimality conditions, we get the following reformulation.

$$\min_{x,y,\lambda,\mu} \; F(x,y) \tag{2.6a}$$

$$\text{s.t. } G(x,y) \leq 0, \tag{2.6b}$$

$$g(x,y) \leq 0, \; h(x,y) = 0, \tag{2.6c}$$

$$\nabla_y \mathcal{L}(x,y,\lambda,\mu) = 0, \tag{2.6d}$$

$$\lambda^T g(x,y) = 0, \tag{2.6e}$$

$$\lambda \geq 0, \tag{2.6f}$$

where we define the Lagrangian function for the lower level problem as

$$\mathcal{L}(x,y,\lambda,\mu) = f(x,y) + \lambda^T g(x,y) + \mu^T h(x,y), \tag{2.7}$$

and the set of Lagrange multipliers corresponding to the point (x,y) by

$$\Lambda(x,y) = \{(\lambda,\mu) \in \mathbb{R}^p \times \mathbb{R}^q : \lambda \geq 0, \; \lambda^T g(x,y) = 0, \; \nabla_x \mathcal{L}(x,y,\lambda,\mu) = 0\}. \tag{2.8}$$

The KKT reformulation of the bilevel problem can be seen as a mathematical optimisation problem with complementarity constraints (MPCC). Although the bilevel problem has been reformulated into a single level problem, this problem can still be difficult to solve for the following reasons. Firstly, the addition of the complementarity constraints (2.6e) makes the problem a mixed integer program. Furthermore, the reformulation introduces additional variables λ and μ into the problem, and the Lagrangian constraints can lead to non-convexity of the problem even when all functions are convex [34]. Finally, as shown by Scheel and Scholtes [102], it is found that the Mangasarian-Fromowitz constraint qualification (MFCQ),

given by Definition 2.4.1, is violated at all feasible points. This is a standard constraint qualification in the literature.

It is known that when the lower level problem is nonconvex, the KKT reformulation has a larger feasible set than the bilevel program, and thus the bilevel program and the MPCC are not equivalent [88]. On the other hand, it was believed for a long time that if the lower level problem is convex then the two problems are equivalent. Because of this, a large proportion of the literature use the KKT reformulation to solve bilevel problems, and in particular those with linear, quadratic and convex lower level problems. It was however shown by Dempe and Dutta in 2012 [26] that when the lower level problem is convex, a local optimal solution of the MPCC reformulation may not always be a local optimal solution of the bilevel program. They do nonetheless show that the equivalence is true if the lower level problem is convex and satisfies a stronger assumption known as Slater's constraint condition.

2.3.2 The optimal value reformulation

The optimal value reformulation replaces the lower level optimisation problem by using its optimal value function $\varphi(x)$, first introduced in a paper by Outrata in 1990 [94]. The optimal value function for the lower level problem, or the lower level value function (LLVF), is defined as

$$\varphi(x) := \min_y \{f(x,y) |\ g(x,y) \leq 0,\ h(x,y) = 0\}. \tag{2.9}$$

The optimal value reformulation for the BLPP (2.1) is given by

$$\begin{aligned}
\min_{x,y} \quad & F(x,y) && \text{(2.10a)}\\
\text{s.t.} \quad & G(x,y) \leq 0, && \text{(2.10b)}\\
& f(x,y) - \varphi(x) \leq 0, && \text{(2.10c)}\\
& g(x,y) \leq 0, && \text{(2.10d)}\\
& h(x,y) = 0. && \text{(2.10e)}
\end{aligned}$$

This problem is equivalent to the bilevel problem, and in contrast to the KKT optimality conditions, we do not require convexity of the lower level problem for a local optimistic solution of the BLPP (2.1) to be a local optimal solution of the LLVF reformulation (2.10). This single level program is however more difficult to solve due to the constraint containing the optimal value function (2.10d). This is because the optimal value function is typically a nonconvex nondifferentiable function, which usually can not be explicitly given. Well known constraint qualifications such as the MFCQ are violated [35]. Solution approaches to solve this problem usually require an approximation of the optimal value function.

To overcome the difficulties in the constraint set, Ye and Zhu [131] introduce a property known as the partial calmness condition. They show that if the problem satisfies this condition, an exact penalty problem can be formed that moves the constraint containing the optimal value function into the objective function.

The partial calmness condition

We first introduce a regularity condition on the bilevel problem, known as the partial calmness condition. Let us consider the following partially perturbed problem of the LLVF reformulation (2.10), where $u > 0$ is a parameter

$$\min_{x,y} \quad F(x,y) \tag{2.11a}$$
$$\text{s.t.} \quad G(x,y) \leq 0, \tag{2.11b}$$
$$f(x,y) - \varphi(x) + u = 0, \tag{2.11c}$$
$$g(x,y) \leq 0, \tag{2.11d}$$
$$h(x,y) = 0. \tag{2.11e}$$

We can see that the LLVF reformulation and the partially perturbed bilevel problem coincide when $u = 0$. As defined by Ye and Zhu [131], the partial calmness condition is given by the following.

Definition 2.3.1 (Partial Calmness)**.** *Let (x^*, y^*) be a (local) optimal solution of (2.10). Then it is partially calm at (x^*, y^*) if there exists a $\gamma > 0$ and a neighbourhood $U(x^*, y^*, 0) \subset \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$ such that for all $(x', y', u') \in U(x^*, y^*, 0)$ where (x', y') is feasible for (2.11) for $u = u'$ then the following inequality is satisfied*

$$F(x', y') - F(x^*, y^*) + \gamma|u| \geq 0.$$

The partial calmness condition is a strong qualification condition that can be seen as quite a restrictive property. The condition can also be difficult to verify, however there are some classes of bilevel problems where the partial calmness condition is known to hold. Two sufficient conditions for partial calmness to hold are the following [24]:

1. The lower level problem is a parametric linear programming problem.

2. The upper level objective function is locally Lipschitz continuous with respect to y uniformly in x and the lower level problem has a uniformly weak sharp minimum.

A discussion on uniformly weak sharp minima, and conditions for this property to hold can be found in [27]. Henrion and Surowiec [55] find a weaker condition than the uniformly weak sharp minimum condition that ensures partial calmness, known as the value function constraint qualification. Dempe, Dutta and Mordukhovich [27] generate a fairly broad class of nonlinear bilevel problems that are partially calm, which have nonlinear lower-level problems where the partial calmness condition holds independently of the upper level. Further discussion and conditions that ensure partial calmness can be found in [48], [55].

Exact penalisation using the optimal value reformulation

Penalty methods are a widely used solution approach to solving constrained optimisation problems. They replace the constrained optimisation problem with an unconstrained one by penalising the objective function when a constraint is violated. For a minimisation problem, the penalised constraint measure is positive when the constraint is violated and zero if the constraint is satisfied, and is multiplied by a positive coefficient, known as the penalty parameter. Exact penalisation of an optimisation problem refers to a penalty problem where there exists a finite penalty parameter such that the optimal solution of both problems coincide.

Ye and Zhu [131] show that the concept of partial calmness at one of its local minimisers is actually equivalent to an exact penalty problem, given by the following theorem.

Theorem 2.3.2. *[131] Let (x^o, y^o) be a local optimal solution of the bilevel problem (2.1). Then problem (2.10) is partially calm at (x^o, y^o) if and only if there exists a constant $\gamma > 0$ such that (x^o, y^o) is a local optimal solution of the exact penalty problem*

$$\min_{x,y} \quad F(x,y) + \gamma(f(x,y) - \varphi(x)) \tag{2.12a}$$

$$s.t. \quad G(x,y) \leq 0, \tag{2.12b}$$

$$g(x,y) \leq 0, \tag{2.12c}$$

$$h(x,y) = 0. \tag{2.12d}$$

We can see that there is a close relationship between the value function reformulation (2.10) and the exact penalty problem (2.12), however, the exact penalty problem does not contain the problematic optimal value constraint $f(x,y) - \varphi(x) \leq 0$. In fact, the constraint set is simplified to the constraint region for the bilevel problem K, and standard constraint qualifications for the exact penalty problem can hold. This makes the problem attractive to work with.

In this thesis we focus our attention on solving this exact penalty problem, investigating how the problem could be solved to find local optimal solutions of the bilevel problem. By finding a local solution to the problem (2.12), we hope to choose a γ such that the local solution is also one of the bilevel problem (2.12). This approach of solving the exact penalty problem to find local solutions of the bilevel problem has been used in other algorithms in the literature, although the number is still relatively scarce. This could partly be due to the restrictive conditions of partial calmness, and the relative ease of using the KKT conditions as opposed to dealing with the optimal value function. A discussion of these algorithms are presented in the literature review in Section 2.5.

An important consideration when solving the exact penalty problem (2.12) is the value of the penalty parameter. This can be difficult. For classical exact penalty problems there are a number of methods that are used in an attempt to find suitable penalty parameters, as described in [20]. Some methods look at ways to approximate the critical value of the penalty parameter. For example, for a nonsmooth exact penalty function it is found that the parameter needs to be at least as large as the vector dual norm of the associated Lagrange multiplier of an optimal solution. When finding an approximation of the penalty parameter is difficult, an alternative approach is to minimise the penalty function with an arbitrary penalty parameter and assess the solution. If the solution is feasible then it should be a stationary point of the original problem, if not then we increase (or decrease) the parameter. For the exact penalty problem (2.12), there is currently no known rule on how best to choose the value [48].

2.4 Optimality Conditions

Many different approaches have been taken to derive optimality conditions for bilevel problems. Often, the bilevel problem is first reformulated into a single level problem. In this section we give some necessary optimality conditions for the bilevel problem, derived from the optimal value reformulation, as this is the problem we look to solve throughout this thesis. For further reading on other optimality conditions of the bilevel problem we refer the reader to the book by Dempe [24].

The existence of optimal solutions for the optimistic bilevel problem can be shown under fairly weak assumptions. It requires that the lower level problem satisfies the Mangasarian-Fromovitz constraint qualification (MFCQ), a standard regularity condition in the literature defined as the following.

Definition 2.4.1 (MFCQ)**.** *The Mangasarian-Fromowitz Constraint Qualification is satis-*

fied at (x^0, y^0) *if there exists a direction* $d \in \mathbb{R}^m$ *satisfying*

$$\begin{aligned} &\nabla_y g_i(x^0, y^0)d < 0, \quad \forall\, i \in I(x^0, y^0) := \{j : g_j(x^0, y^0) = 0\}, \\ &\nabla_y h_j(x^0, y^0)d = 0, \quad \forall\, j = 1, \dots, q, \end{aligned} \tag{2.13}$$

and the gradients $\{\nabla_y h_j(x^0, y^0) : j = 1, \dots, q\}$ *are linearly independent.*

The existence of optimal solutions for the optimistic bilevel problem is then given by the following theorem.

Theorem 2.4.2. *[24] Let us assume that all functions are sufficiently smooth, the feasible set* K *is nonempty and compact, and MFCQ holds for the lower level problem. Then a global optimistic solution of the bilevel problem (2.1) exists if there is a feasible solution.*

Having established the existence of optimal solutions, we return back to optimality conditions for bilevel problems using the optimal value reformulation. The difficulty with this problem is that standard constraint qualifications are violated due to the nonsmooth optimal value function [24]. It can nevertheless be shown that under the partial calmness condition, optimality conditions using the KKT type can be obtained. Let $\partial\varphi(\cdot)$ be the Clarke generalised gradient of the optimal value function, defined in Section 3.2. Then the optimality conditions are given by the following, first derived by Ye and Zhu [131].

Theorem 2.4.3. *Let* (x^o, y^o) *be a local optimal solution of the problem (2.10). Assume all functions* F, G, f, g, h *are sufficiently smooth and MFCQ is satisfied for the lower level problem for all points* (x, y) *with* $x = x^0$ *and* $y \in \Psi(x^0)$*. Assume the feasible set* $S(x^0)$ *is nonempty and compact. Let the problem (2.10) be partially calm at* (x^0, y^0)*. Then there exists* $\gamma > 0$*,* $\alpha \in \mathbb{R}^p_+$*,* $\beta \in \mathbb{R}^q_+$*,* $\xi \in \mathbb{R}^k_+$ *such that*

$$\begin{aligned} 0 \in &\nabla F(x^0, y^0) + \gamma\big(\nabla f(x^0, y^0) - \partial\varphi(x^0) \times \{0\}\big) + \alpha^T \nabla g(x^0, y^0) + \beta^T \nabla h(x^0, y^0) \\ &+ \xi \nabla G(x^0, y^0) \times \{0\}, \end{aligned} \tag{2.14}$$

$$0 = \alpha^T g(x^0, y^0), \tag{2.15}$$

$$0 = \xi^T G(x^0, y^0). \tag{2.16}$$

Using properties of the Clarke generalised gradient these conditions can be developed even further, and we get the following conditions for local optimality.

Corollary 2.4.3.1. *Let* (x^o, y^o) *be a local optimal solution of the problem (2.10). Assume all functions* F, G, f, g, h *are sufficiently smooth and MFCQ is satisfied for the lower level problem for all points* (x, y) *with* $x = x^0$ *and* $y \in \Psi(x^0)$*. Assume the feasible set* $S(x^0)$ *is nonempty and compact. Let the problem (2.10) be partially calm at* (x^0, y^0)*. Then*

there exists $\gamma > 0$, $\alpha \in \mathbb{R}^p_+$, $\beta \in \mathbb{R}^q_+$, $\xi \in \mathbb{R}^k_+$, $\zeta \in \mathbb{R}^{n+1}_+$, $\sum_{k=1}^{n+1} \zeta_k = 1$ *and* $y^k \in \Psi(x^o)$, $(\lambda^k, \mu^k) \in \Lambda(x^0, y^k)$, $k = 1, \dots, n+1$ *such that*

$$\begin{aligned} 0 =& \nabla_x F(x^0, y^0) + \gamma\big(\nabla_x f(x^0, y^0) - \sum_{k=1}^{n+1} \zeta_k \nabla_x \mathcal{L}(x^0, y^k, \lambda^k, \mu^k)\big) + \alpha^T \nabla_x g(x^0, y^0) \\ &+ \beta^T \nabla_x h(x^0, y^0) + \xi \nabla_x G(x^0, y^0), \end{aligned} \tag{2.17}$$

$$0 = \nabla_y F(x^0, y^0) + \gamma \nabla_y f(x^0, y^0) + \alpha^T \nabla_y g(x^0, y^0), \tag{2.18}$$

$$0 = \alpha^T g(x^0, y^0), \tag{2.19}$$

$$0 = \xi^T G(x^0, y^0). \tag{2.20}$$

Lastly we note that necessary optimality conditions of the Fritz John type can be derived using the optimal value reformulation that does not require the partial calmness condition. For an extensive review on the optimality conditions for the LLVF reformulation, we refer the reader to [35]. For further reading on necessary and sufficient optimality conditions of other reformulations of the bilevel problem, see [24].

2.5 Literature review on solution approaches

The following gives a literature review on the different solution approaches taken and algorithms designed to solve continuous bilevel programs. Although comprehensive, we give a particular in-depth review of algorithms solving the LLVF reformulation of the problem, with an initial briefer coverage on other classical approaches and a final note on heuristic algorithms. For more information on solution methods of bilevel programs, as well as key concepts, applications and optimality conditions, we refer the reader to the following review papers [109], [19], [25] and the books [24], [7], [32].

We begin with a discussion on the different solution approaches taken to solve bilevel problems that do not consider first reformulating the problem into a single level problem. As bilevel problems are difficult to solve, initial studies looked at algorithms to solve the linear bilevel program, before more complex functions were considered. It can be shown that the optimal solution for the linear bilevel program occurs at the vertex of the constraint region [25]. Therefore one of the first approaches to solve the linear bilevel program involved vertex enumeration. One such method, known as the k-th best algorithm, was proposed by Bialas and Karwan [12]. This enumerates all vertices of the constraint region with respect to the upper level optimisation function, whilst checking for feasibility of the lower level, until the optimal solution is found. Although enumeration techniques solve the problem globally, they can require a large number of computations. Other global optimisation algorithms

that utilise this property of optimal solutions for linear bilevel problems are complementary pivoting algorithms, for example this approach was taken by Júdice and Faustino [60].

Descent methods are commonly applied to search for a local optimal solution of the bilevel program. Descent algorithms work by searching for a new feasible point that decreases the upper level optimisation function. For bilevel problems however, searching for feasible points can be difficult as it requires the lower level problem to be optimal. Dempe [22] therefore proposed a descent method for the linear bilevel program, which optimises the lower level problem at each step. Savard and Gauvin [101] introduced a steepest descent approach for the general bilevel program where the optimal solution of the lower level set is unique and no upper level constraints exist. They found an upper-level descent direction using the directional derivative of the rational reaction function $y(x)$, and show the direction of steepest descent coincides with the optimal solution of a linear-quadratic bilevel program. Known global optimisation algorithms can then be used to solve this bilevel problem. Vicente et al. [118] modified this approach to apply it to a bilevel function with a strictly convex lower level problem.

Another common approach to solve bilevel programs is to use penalty methods. For example, Anandalingam and White [4] introduced a penalty method for the linear bilevel program. The penalty function uses the duality gap of the lower level problem, which is penalised in the leaders objective function. When the duality gap is zero the solution is optimal in the lower level, and thus by minimising the penalty problem we find a feasible solution that optimises the leaders objective function. They show in a later paper that this algorithm can find global optimal solutions [121]. Aiyoshi and Shimizu [2] introduced a method that replaces the lower level problem with an unconstrained problem using a penalty function. However as the bilevel structure remained, the followers penalty problem had to be solved at each iteration, slowing convergence. A double penalty function was therefore introduced in [58] which builds on this problem, but penalises the upper problem as well.

The application of trust-region methods has been explored with bilevel programs, typically without reformulating the problem into a single level problem. Trust-region methods approximate the objective function within a given region with some local model function, which is minimised at each step. At each iteration the trust-region is expanded or contracted depending on how good the approximation of the local model function is. Liu et al. [72] proposed an initial trust-region algorithm where the lower level problem is strongly convex and linearly constrained, and there are no upper level constraints. This was extended by Colson et al. [18], who developed a more general trust-region method to solve a nonlinear bilevel program where the upper level constraints involved only upper level decision variables.

In the latter paper they formed linear models of all functions except the lower level objective function, which was approximated by a quadratic model. The trust-region subproblem then formed a simplified bilevel model which can be solved using known algorithms.

As discussed in Section 2.3, a large number of approaches in the literature look to reformulate the problem into a single level problem first. The vast majority of these use the KKT reformulation, which requires the lower level problem to be convex and sufficiently regular. Due to the complementarity constraints the problem becomes a mixed integer program, and many solution algorithms use enumeration algorithms to address this. A common approach is the branch-and-bound method, which can be applied to linear and quadratic bilevel programs. For example, a branch-and-bound algorithm was developed by Bard and Falk [9] for linear bilevel problems, by Edmunds and Bard [42] for bilevel problems where the lower level objective is quadratic and its constraints are linear, and by Bard [8] for bilevel problems where the lower level problem and constraints are quadratic, and the upper level problem is convex. Although they have been successfully applied, branch-and-bound methods can be slow in convergence if the number of integer variables is large. That being said there are ways this can be improved, for example in the paper by Bard [8] they exploit properties of the inducible region to find a tight upper bound, reducing the number of subproblems that need to be solved.

If it is assumed that if the lower level optimal solution is uniquely determined, then the lower level variables form an implicit function of the form $y(x)$, which is Lipschitz continuous. Using this function, the bilevel model can be reformulated into a single level problem. A common approach to solving this problem is by applying bundle methods. Bundle algorithms have been developed to solve nondifferentiable Lipschitz continuous functions, and involve approximating the subdifferential of the function with a bundle consisting of subgradients from previous iterations. Bundle algorithms have been applied to the bilevel problem with unique lower level solutions by using subgradient information of the lower level implicit function. This approach is taken in the papers by Dempe [38] and Falk and Liu [45]. Dempe [23] extended this to bilevel problems with non-unique lower level solutions using a regularisation approach.

In contrast to the KKT reformulation, the literature on solution algorithms that use the LLVF reformulation to reformulate the bilevel problem into a single level problem is relatively scarce. The reason for this is because the problem is difficult to solve. As the difficulties are due to the typically nonsmooth optimal value function, the majority of approaches look at finding an approximation of the optimal value function, which is usually updated at each iteration. However as discussed, there are advantages of using the LLVF reformulation over

the KKT reformulation. As solutions to the single level program coincide with the BLPP under weaker assumptions then that of the KKT reformulation, it could be used to solve more complicated nonlinear bilevel problems.

Shimizu and Lu [105] introduced one of the first algorithms to solve a bilevel problem using the LLVF reformulation. They presented a global optimisation algorithm for the bilevel problem where the upper level functions F and G are convex, and the lower level problem is jointly convex in x and y. Under these assumptions the optimal value function is convex. They introduced auxiliary variables and then, by penalising the nonlinear constraints containing the auxiliary variables, they transform the problem into a concave program, which has a concave objective function and convex constraints. They then applied an outer approximation method by cutting planes to solve the concave program. We note that this is not an exact penalisation of the problem.

A common solution approach used to tackle the LLVF reformulation is to relax it by using a function that upper bounds the optimal value function. Dempe and Franke [31] introduced an algorithm to solve a bilevel program globally, where the lower level problem is jointly convex in x and y. Under this assumption, the optimal value function is convex, and a function that provides an upper bound on the optimal value function can be found. Using this upper bound approximation, they formed a relaxed LLVF reformulation to solve. If a solution is optimal for the relaxed problem and feasible for the bilevel problem, then it is optimal for the bilevel problem. If not the approximation is updated and the relaxed problem is solved again.

Dempe and Franke [29] presented another relaxation scheme to solve a bilevel problem where the upper and lower objective functions are linear, and where the only constraints are in the lower level, which are linear and do not depend on the upper level variables. In this problem, the optimal solutions of the lower level are supergradients of the optimal value function. Dempe and Franke therefore formed an upper bound on the optimal value function by approximating the optimal value function using a subset of supergradients, which is extended successively at each iteration. Mitsos et al. [90] proposed a relaxation method which doesn't assume the lower level problem is convex. They approximated the value function with a piecewise but discontinuous approximation that provides an upper bound on the optimal value function. The proposed algorithm finds an approximate of the global optimal solution.

A second common approach to solve the LLVF reformulation is by using smoothing methods. Lin et al. [70] presented an algorithm to solve a bilevel problem with no upper level

constraints, and where the lower level constraint set is convex and does not depend on x. The objective functions can however be nonconvex. They approximated the optimal value function using a smoothing approximation function, and successfully applied a smoothing projected gradient algorithm. Two other papers have looked at smoothing methods to solve the same bilevel problem. Xu et al. [124] applied a smoothing sequential quadratic programming method, whist Xu and Ye [123] applied a smoothing augmented Lagrangian method. The latter paper however solved a combined reformulation that uses both the KKT conditions and the value function. This is because the assumptions required for the necessary optimality conditions to hold for the combined reformulation are much more likely to hold than for each of the reformulations on their own. Further research has looked into solving a combined model to overcome some of the stronger conditions, for example in a paper by Ye and Zhu [132].

Recently, Fischer et al. [48] presented a semismooth newton-type method to solve the exact penalised problem built from the LLVF reformulation under the partial calmness condition. They did this by first reformulating the problem into a semismooth system of equations, and providing conditions for when solutions for this system of equations are a local solution of the bilevel problem. The algorithm does not require the direct calculation of the lower level function or its derivatives. They give extensive numerical results, testing on a range of different discrete penalty parameter values. In another recent paper, Ye et al. [130] assume that the lower level problem is fully convex, and reformulate the bilevel problem into a difference of convex problem by using the convex optimal value function. They then apply two difference of convex algorithms to solve it.

Kleniati and Adjiman [63] proposed a branch-and-sandwich algorithm to globally optimise nonconvex bilevel problems using the LLVF reformulation. The algorithm computes lower and upper bounds for both the optimal value reformulation and the optimal value function in a single branch-and-bound tree, which converges to find a global optimal solution. They provided convergence analysis and numerical results on 34 nonlinear bilevel programs in [64], [96]. Finally, by taking a different approach, Lampariello and Sagratella [66], [67] looked at the relationship between the value function approach of the bilevel program with generalised Nash equilibrium problems. They defined classes of the bilevel problem where the solutions can be found by finding equilibria of a generalised Nash equilibrium problem, and proposed numerical methods that solve them.

We conclude the review on the research of solution approaches that utilise the LLVF reformulation by mentioning briefly some work on other bilevel problems not discussed in this review. Some investigation using the LLVF reformulation has been done to solve the

pessimistic bilevel problem. Wiesemann et al. [122] presented an algorithm to solve this, showing that the pessimistic value approach becomes an infinite-dimensional optimisation problem. Their algorithm solves a sequence of relaxed finite-dimensional approximations of the value reformulation at each iteration. There has also been some recent research on solving mixed integer bilevel programming problems using the value reformulation in the following papers [79], [49].

To close, we give a final discussion on some of the heuristic approaches to bilevel programming. Because bilevel programs are difficult to solve, exact approaches can be limited to solving moderately sized problems. Heuristics can allow the opportunity to solve larger problems at a faster speed, using the power of computers. They have therefore recently gained a lot of traction as approaches to solving bilevel problems. Genetic algorithms, simulated annealing algorithms, particle swarm optimization and differential evolution algorithms have all been applied to bilevel problems to search for a solution. Some of these algorithms are applied as a nested algorithm, which is an algorithm that solves the lower level problem for each and every upper level variable. For example, a nested genetic algorithm was developed by Mathieu et al. [81] for a linear bilevel problem and by Yin [133] for a nonconvex bilevel problem, whilst Anandalingam et al. [3] presented a nested simulated annealing algorithm. These algorithms can also be applied to the single level KKT reformulated problem. For example, Wang et al. [119] used a simplex-based genetic algorithm on the KKT reformulated problem to solve linear-quadratic bilevel problems, and Jiang et al. [59] used particle swarm optimization.

Another common heuristic approach to solve bilevel problems is to use metamodelling based methods. These methods use an approximation of the problem which can be computed quickly, and are usually used on the functions that are expensive to calculate. To apply the method, either a small sample of the problem is selected to train the approximated function, which can then be used to optimise the whole problem, or a local approximation is formulated which is updated at each iteration. Metamodelling methods have been used to solve the LLVF reformulation by approximating the optimal value function. Algorithms doing this have been developed by Sinha et al. in [108], [106], where the latter paper looked at approximating both the optimal value function and the reaction set. Other metamodelling algorithms for the bilevel problem have looked at just approximating the rational reaction set, for example this was done in [107].

Chapter 3

Value function reformulation based trust-region method for bilevel optimisation

3.1 Introduction

As discussed in Chapter 2, by using the partial calmness condition an exact penalty problem can be built from the optimal value reformulation. In this chapter, we propose a novel trust-region algorithm that solves this exact penalty problem, which approximates the optimal value function locally at each iteration point. Additionally, under strong assumptions the exact penalty function becomes a difference of convex functions problem, known as a DC problem. We apply a known algorithm developed for DC problems to solve the bilevel problem, called the convex-concave procedure (CCP), which follows a similar approach used in the trust-region algorithm proposed. Extensive numerical results are given on the performance of both algorithms.

This chapter begins by discussing some of the properties of the optimal value function which are utilised to develop the algorithms, followed by an introduction on trust-region methods for smooth problems. The proposed trust-region algorithm developed for bilevel problems is then presented, along with a discussion on the convergence. Following this, the CCP method for bilevel problems is introduced. Finally, extensive numerical results on 124 nonlinear bilevel problems taken from the literature are presented. Performance of the algorithms are assessed, comparing the results to known solutions in the literature and to a small number of other bilevel algorithms tested. Further investigation on the sensitivity of the choice of

penalty parameter is detailed, before concluding on the performance.

3.2 Properties of the optimal value function

We collate together a comprehensive account on the properties of the optimal value function known from the literature. We begin by defining the problem and the constraint qualifications used. The following properties are stated in [24]. Using the same notation as presented in Chapter 2, the lower level problem is defined as

$$\begin{aligned} \min_y \quad & f(x,y) \\ \text{subject to} \quad & g(x,y) \leq 0, \\ & h(x,y) = 0, \end{aligned} \tag{3.1}$$

where $g(x,y) = (g_1(x,y),\ldots,g_p(x,y))^T$, $h(x,y) = (h_1(x,y),\ldots,h_q(x,y))^T$, and we have $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ and $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^q$. We assume all functions are sufficiently smooth.

The lower level problem can be seen as a parametric optimisation problem. As discussed, the optimal value function $\varphi : \mathbb{R}^n \to \mathbb{R}$ returns the optimal value of the lower level problem for a given upper level solution and is defined as

$$\varphi(x) = \min_y \{f(x,y) : g(x,y) \leq 0,\ h(x,y) = 0\}. \tag{3.2}$$

We assume that $|\varphi| < \infty$. Additionally, we have the solution set mapping $\Psi : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ and the feasible set mapping $S : \mathbb{R}^n \to 2^{\mathbb{R}^m}$, which were introduced in Section 2.1.

The Lagrangian function of the lower level problem is defined by the following

$$\mathcal{L}(x,y,\lambda,\mu) := f(x,y) + \lambda^T g(x,y) + \mu^T h(x,y), \tag{3.3}$$

where (λ,μ) are Lagrange multipliers corresponding to the point (x,y), which are in the set

$$\Lambda(x,y) := \{(\lambda,\mu) \in \mathbb{R}^p \times \mathbb{R}^q : \lambda \geq 0,\ \lambda^T g(x,y) = 0,\ \nabla_y \mathcal{L}(x,y,\lambda,\mu) = 0\}. \tag{3.4}$$

Two standard constraint qualifications from the literature are considered in this thesis. The first is the Mangasarian-Fromowitz constraint qualification, which was introduced in Definition 2.4.1. The second is the stronger Linear Independent Constraint Qualification (LICQ), defined as the following.

Definition 3.2.1 (LICQ)**.** *The Linearly Independent Constraint Qualification holds if the gradients*

$$\{\nabla_y g_i(x^0, y^0), i \in I(x^0, y^0)\} \cup \{\nabla_y h_j(x^0, y^0), j = 1, \ldots, q\}$$

are linearly independent.

Clearly if LICQ holds, this implies that MFCQ also holds. A final strong assumption we also consider in this thesis is the following.

Definition 3.2.2 (SOSC)**.** *The Second Order Sufficient Condition is said to be satisfied at* (x^0, y^0) *if for each* $(\lambda, \mu) \in \Lambda(x^0, y^0)$ *and for each* $d \neq 0$ *satisfying*

$$\begin{aligned} &\nabla_y g_i(x^0, y^0)d = 0, \quad \forall\, i \in J(\lambda) := \{j : \lambda_j > 0\}, \\ &\nabla_y g_i(x^0, y^0)d < 0, \quad \forall\, i \in J(\lambda) := \{j : \lambda_j = 0\}, \\ &\nabla_y h_j(x^0, y^0)d = 0, \quad \forall\, j = 1, \ldots, q, \end{aligned} \tag{3.5}$$

we have

$$d^T \nabla_{yy} \mathcal{L}(x^0, y^0, \lambda, \mu) d > 0.$$

3.2.1 Convexity of the optimal value function

Convexity is fundamental in optimisation. When minimising a convex program, we can guarantee that a stationary point is a global minimiser. We explore the convexity and concavity of the optimal value function. For completeness, we state the following standard definition in convex analysis.

Definition 3.2.3. *The function* f *is a convex function if its domain* $dom(f)$ *is a convex set and if for any two points* $x, y \in dom(f)$ *and* $\alpha \in [0, 1]$ *we have*

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y).$$

For the lower level parametric problem we define the following. The lower level problem is said to be a convex parametric problem if for each fixed $x \in \mathbb{R}^n$, $f(x, \cdot)$ and $g_i(x, \cdot)$, $i = 1, \ldots p$ are convex, and $h_j(x, \cdot)$, $j = 1, \ldots q$ is affine linear on $\mathbb{R}^m$. If the function $f(x, y)$ is convex in both x and y, we call it *fully convex*. A lower level problem is a fully convex problem if $f(x, y)$ and $g_i(x, y)$, $i = 1, \ldots, p$ are fully convex and $h_j(x, y)$, $j = 1, \ldots q$ is affine linear in (x, y).

Although often nonconvex, there are examples of bilevel problems when the optimal value

function of the lower level problem is found to be convex. The following proposition is a well known property of the optimal value function.

Proposition 3.2.4. *[7] The optimal value function $\varphi(x)$ is convex when the lower level problem is fully convex. That is, when $f(x,y)$ and $g_i(x,y)$, $i=1,\dots,p$ are fully convex, and $h_j(x,y)$, $j=1,\dots q$ is affine linear in (x,y).*

For additional sufficient conditions and properties relating to convexity and concavity of the general parametric optimisation problem we refer to the papers [46], [73].

3.2.2 Continuity and differentiability of the optimal value function

Under reasonable assumptions we find $\varphi(x)$ to be Lipschitz continuous, a property that can allow us to explore nonsmooth optimisation algorithms. We define the notion of Lipschitz continuity as the following.

Theorem 3.2.5. *A function $f:\mathbb{R}^n\to\mathbb{R}^m$ is locally Lipschitz continuous at a point $x^0\in\mathbb{R}^n$ if there exists on open neighbourhood $V_\epsilon(x^0)$ of x^0 and a constant $l<\infty$ such that*

$$||f(x)-f(x')||\le l||x-x'||, \qquad \forall\ x,\ x'\in V_\epsilon(x^0). \tag{3.6}$$

Then under the following assumptions the optimal value function is Lipschitz continuous.

Theorem 3.2.6. *[24] Let the assumption MFCQ hold for the lower level problem at all points (x,y) with $x=x^0$ and $y\in\Psi(x^0)$ and assume the feasible set $S(x^0)$ is nonempty and compact. Then the optimal value function φ is locally Lipschitz continuous at x^0.*

In addition, using the following theorems, the optimal value function can be found to be directionally differentiable.

Theorem 3.2.7. *[52] Let the assumption LICQ hold for the lower level problem at all points (x,y) with $x=x^0$ and $y\in\Psi(x^0)$, and assume the feasible set $S(x^0)$ is nonempty and compact. Then the optimal value function φ is directionally differentiable at $x=x^0$ and*

$$\varphi'(x^0;r)=\min\{\nabla_x\mathcal{L}(x^0,y,\lambda^0,\mu^0)r : y\in\Psi(x^0)\}, \qquad \forall\ r\in\mathbb{R}^n, \tag{3.7}$$

where for each $y\in\Psi(x^0)$ the set $\Lambda(x^0,y)$ reduces to the singleton (λ^0,μ^0) continuously depending on y.

Theorem 3.2.8. *[24] Let $f(x,\cdot)$ and $g(x,\cdot)$, $i=1,\dots p$ be convex and $h(x,\cdot)$, $j=1,\dots q$*

be affine linear on $\mathbb{R}^m$ *for each fixed* $x \in \mathbb{R}^n$. *Let MFCQ hold at points* (x^0, y^0) *and assume the feasible set* $S(x^0)$ *is nonempty and compact. Then the optimal value function* φ *is directionally differentiable at* $x = x^0$ *and*

$$\varphi'(x^0; r) = \max_{(\lambda,\mu)\in\Lambda(x^0,y)} \min_{x\in\Psi(x^0)} \nabla_x \mathcal{L}(x^0, y, \lambda^0, \mu^0) r, \qquad \forall\, r. \tag{3.8}$$

As has been discussed, the optimal value function is often a nonsmooth function. To explore its properties, we look at the Clarke generalised gradient and the Clarke directional derivative of Lipschitz continuous functions. These definitions were introduced by Clarke [17] to extend the notion of gradients and directional derivatives to nonsmooth functions, and are defined as the following.

Definition 3.2.9. *[17] Let* $z : \mathbb{R}^p \to \mathbb{R}$ *be a locally Lipschitz continuous function. Then the Clarke generalised gradient of the function* z *at* x^0 *is defined as*

$$\partial z(x^0) = \text{conv}\{\lim_{k\to\infty} \nabla z(x)^k : \lim_{k\to\infty} x^k = x^0,\ \nabla z(x^k) \text{ exists } \forall\, k\}.$$

Definition 3.2.10. *[17] Let* $v : \mathbb{R}^m \to \mathbb{R}$ *be a locally Lipschitz continuous function with generalized gradient* $\partial v(x^o)$ *at* $x^o \in \mathbb{R}^n$. *Then the Clarke directional derivative* $v^o(x^0; r)$ *at* $x^0 \in \mathbb{R}^m$ *is defined as*

$$v^o(x^o; r) = \lim_{x\to x^o} \sup_{t\to 0^+} t^{-1}[v(x + tr) - v(x)].$$

Lastly, we define a function to be Clarke regular with the following definition.

Definition 3.2.11. *The function* v *is Clarke regular provided the ordinary directional derivative* $v'(x^o; r)$ *exists and* $v'(x^o; r) = v^o(x^o; r)$ *for all* $r \in \mathbb{R}^n$.

As the optimal value function is Lipschitz continuous under the assumptions of Theorem 3.2.6, we can define the Clarke generalised gradient of the optimal value function.

Theorem 3.2.12. *[52] Let the assumption MFCQ hold for the lower level problem at all points* (x, y) *with* $x = x^0$ *and* $y \in \Psi(x^0)$ *and assume the feasible set* $S(x^0)$ *is nonempty and compact. Then the optimal value function is locally Lipschitz continuous and the generalised gradient for the optimal value function* φ *satisfies*

$$\partial\varphi(x^0) \subset \textit{conv} \bigcup_{y\in\Psi(x^0)} \bigcup_{(\lambda,\mu)\in\Lambda(x^0,y)} \nabla_x \mathcal{L}(x^0, y, \lambda, \mu).$$

Furthermore, under stronger assumptions the following corollaries can be derived.

Corollary 3.2.12.1. *[52] Let the assumption LICQ hold for all points (x, y) with $x = x^0$ and $y \in \Psi(x^0)$ and assume the feasible set $S(x^0)$ is nonempty and compact. Then φ is Clarke regular and we get the following*

$$\partial\varphi(x^0) = \operatorname{conv} \bigcup_{y \in \Psi(x^0)} \nabla_x \mathcal{L}(x^0, y, \lambda, \mu),$$

where $\{(\lambda, \mu)\} = \Lambda(x^0, y)$.

Corollary 3.2.12.2. *[104] Let $f(x, \cdot)$ and $g(x, \cdot)$ be convex and $h(x, \cdot)$ be affine linear on $\mathbb{R}^m$ for each fixed $x \in \mathbb{R}^n$. Let MFCQ hold at points (x^0, y^0) and assume the feasible set $S(x^0)$ is nonempty and compact. Then if $\Psi(x^0)$ reduces to a singleton then*

$$\partial\varphi(x^0) = \bigcup_{(\lambda,\mu) \in \Lambda(x^0, y^0)} \nabla_x \mathcal{L}(x^0, y^0, \lambda, \mu).$$

Corollary 3.2.12.3. *[104] Let f and g_i, $i = 1, \dots p$ be convex in both (x, y) and h_j, $j = 1, \dots q$ be affine linear in both variables. Let assumption MFCQ hold at points (x^0, y^0) and assume the feasible set $S(x^0)$ is nonempty and compact. Then*

$$\partial\varphi(x^0) = \bigcup_{(\lambda,\mu) \in \Lambda(x^0, y^0)} \nabla_x \mathcal{L}(x^0, y^0, \lambda, \mu).$$

Finally, under even stronger assumptions the optimal value function can be found to be locally continuously differentiable. This occurs when we have a uniquely determined lower level solution, and unique Lagrange multipliers, To do so, we introduce the notion of a *strongly stable* local optimal solution.

Definition 3.2.13. *[24] We define a local optimal solution $y^0 \in \Psi(x^0)$ as strongly stable if there exist open neighbourhoods $U_\delta(x^0)$ of x^0, $V_\epsilon(y^0)$ of y^0, where $\delta > 0$, $\epsilon > 0$, and a uniquely determined continuous vector-valued function $y : U_\delta(x^0) \to V_\epsilon(y^0)$ such that $y(x)$ is the unique local optimal solution of the lower level problem 3.1 in $V_\epsilon(y^0)$ for all $x \in U_\delta(x^0)$.*

It can be shown that if both MFCQ and SOSC hold at (x^0, y^0) with $y^0 \in \Psi(x^0)$, then the local solution y^0 is strongly stable [24]. Then as LICQ ensures uniqueness of the Lagrange multipliers, the following proposition can be found.

Proposition 3.2.14. *[84] For each $x \in \mathbb{R}^n$, let $f(x, \cdot)$ be convex and let $g(x, \cdot)$ be component-*

wise convex. Fix a point $(x^0, y^0) \in K$ *where LICQ and SOSC hold for the lower level problem. Let* $(\lambda^0, \mu^0) \in \mathbb{R}^p \times \mathbb{R}^q$ *be the uniquely determined associated lower level Lagrange multipliers. Then* φ *is continuously differentiable at* x^0 *and it holds*

$$\nabla\varphi(x^0) = \nabla_x \mathcal{L}(x^0, y^0, \lambda^0, \mu^0).$$

3.3 A trust-region method

3.3.1 Introduction to trust-region methods

There are two fundamental strategies for iterative methods in optimisation: trust-region methods and line search methods. While both these methods generate steps by solving an approximated, typically quadratic, model of the objective function, a line search method uses the model to generate a search direction and then looks to find a suitable step length along the direction. On the other hand, trust-region methods define a region around the current iterate that we trust the model, and solve the model within this region. Trust-region methods can be seen as a dual to line search methods, as in a trust-region method the step size is chosen first, which can be seen as the size of the trust-region, and the direction is found after, whilst the reverse method is taken in a line search method. We take a trust-region approach to solve the bilevel problem, due to the difficulties of finding a suitable step size for the problem we look to solve.

We shall give a brief overview of the classical trust-region method for smooth problems, before presenting our trust-region method for bilevel problems. The following can be found in the book by Conn, Gould, and Toint [20]. Let us consider the minimisation of the following unconstrained problem.

$$\min_x \quad f(x)$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function.

The trust-region method looks to build a model $m_k(x_k + s)$ about the point x_k that approximates $f(x)$. We define the trust-region about the point x_k to be the set of all points

$$\mathcal{B}_k = \{x \in \mathbb{R}^n | \; ||x - x_k|| \leq \Delta_k\},$$

where $\Delta_k \in \mathbb{R}$ is the trust-region radius and $||\cdot||$ is any norm. Usual candidates for the norm are ℓ_1, ℓ_2 and ℓ_∞. A common choice is the infinity norm as the feasible trust-region becomes

a rectangular box, and the constraint can be formed into the following simple bounds

$$x_k + s \geq 0, \qquad s \geq -\Delta_k e, \qquad s \leq \Delta_k e,$$

where $e = (1, 1, \ldots, 1)^T$. This can be easily solved with standard optimisation procedures.

The trust-region subproblem we solve at each iteration is then the following

$$\begin{aligned} \min_s \quad & m_k(x_k + s) \\ \text{s.t.} \quad & ||s|| \leq \Delta_k. \end{aligned} \tag{3.9}$$

The model $m_k(x_k + s)$ of $f(x)$ should be at least a good first-order smooth approximation of the objective. Commonly, the model is taken to be a linear or quadratic approximation of the function f, based on its Taylor series expansion about the point x_k. The first order linear model has the form

$$m_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s, \tag{3.10}$$

and the second order quadratic model has the form

$$m_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T B_k s, \tag{3.11}$$

where B_k is some symmetric matrix approximating the Hessian H_k. Typically, the quadratic model is used, as it provides a better approximation to the function and faster convergence.

Throughout the iterations of the algorithm, the trust-region is increased or decreased, depending on how well the model performs during the previous iteration. If we find the model to be accurately predicting the behaviour of the objective function, the model is a good fit and the trust-region is increased to allow larger steps. If however we find the model to be a poor fit, we decrease the region and we solve the subproblem again, with the hope that the model provides a better prediction in the smaller region.

Using a model for the trust-region subproblem that is a good first-order smooth approximation of the objective function, such as the models discussed, the algorithm can be shown to converge to a stationary point of the problem as the number of iterations tend to infinity [20], [53]. For practical reasons, however, a suitable stopping criteria is usually given. Interestingly, the algorithm can be shown to still converge when the trust-region subproblem is only solved approximately, so long as we find a point that is within the trust-region that gives a sufficient reduction of the model. For the quadratic model, this must be at least as much as the

reduction achieved by the *Cauchy point*. This is the minimiser along the steepest descent direction $-\nabla f(x_k)$, and it is inexpensive to calculate.

The trust-region algorithm can be readily extended to deal with convex constrained optimisation, as shown in [20]. Let us define the following constrained optimisation problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{C}, \end{aligned} \tag{3.12}$$

where $\mathcal{C}$ is a convex set. Then at each iteration we solve the trust-region subproblem (3.9) with the additional constraints $x \in \mathcal{C}$. We also require the initial starting point to be a feasible point of the constraints, that is $x_0 \in \mathcal{C}$. For further reading on trust-region methods for continuous optimisation and for convergence proofs, we refer to the book by Conn, Gould and Toint [20] and the review paper by Yuan [135].

3.3.2 A trust-region method for the bilevel program using the optimal value reformulation

We formulate a novel trust-region method to solve the bilevel problem (2.1) using the optimal value reformulation (2.10). This problem is challenging to solve because of the optimal value constraint $f(x,y) - \varphi(x) \leq 0$, which in general is nonsmooth and nonconvex due to the optimal value function. As discussed in Section 2.3.2, an exact penalty problem can be built from the optimal value reformulation under the partial calmness condition, which enables the ability to move the problematic optimal value constraint from the feasible set into the upper level objective function. Because of this, we focus our attention on solving this exact penalty problem. To our knowledge, a trust-region approach to solving a general bilevel problem using this exact penalty problem has not yet been explored.

By Theorem 2.3.2, the exact penalty problem we look to solve is

$$\min_{x,y} \quad P(x,y) = F(x,y) + \gamma(f(x,y) - \varphi(x)) \tag{3.13a}$$

$$\text{s.t.} \quad G(x,y) \leq 0, \tag{3.13b}$$

$$g(x,y) \leq 0, \tag{3.13c}$$

$$h(x,y) = 0, \tag{3.13d}$$

where $\gamma > 0$ is the penalty parameter. Let the constraint set be

$$K = \{(x,y) : G(x,y) \leq 0,\ g(x,y) \leq 0,\ h(x,y) = 0\},$$

which coincides with the constraint set for the bilevel problem (2.1). By finding a local solution to this exact penalty problem, we hope to choose γ well such that the local solution is also one of the bilevel problem (2.1). It should be stressed that there is no guarantee that the local solution of the penalty problem is also one of the bilevel problem, and careful consideration of the choice of penalty parameter is required. That being said, the exact penalty problem we now look to solve has a smooth constraint set. The problem does however remain difficult to solve due to the typically nonsmooth optimal value function $\varphi(x)$ in the objective function, which cannot be explicitly given. The idea behind the algorithm is to therefore find an approximation of the optimal value function locally at each iteration, and apply a trust-region approach to solve the model within a region we can trust it.

As discussed in Section 3.3, the trust-region method is classically applied to smooth optimisation problems. In fact, the Cauchy point for nonsmooth functions in general fails except for restricted classes of nonsmooth functions [5]. That being said, some research has been done to investigate classes of nonsmooth problems for which the trust-region method can be applied. Convergence of the method is shown for the case when the objective function is both locally Lipshitz continuous and Clarke regular, and the trust-region model is a first order model that uses the full Clarke generalised gradient, as is done in [40]. Qi and Sun [98] show convergence when the regularity assumption is replaced by the slightly weaker subhomogeneity assumption. Recently, Hoseini and Nobakhtian [57] show convergence for their trust-region algorithm for unconstrained nonsmooth functions under mild assumptions, using the Goldstein ϵ-subdifferential.

To overcome the difficulties in calculating the full Clarke generalised gradient at each iteration point, other studies have investigated combining the trust-region method with the bundle method, which requires only a single subgradient to be calculated at each iteration. Algorithms of this type usually require the problem to be weakly semismooth to ensure convergence, for example in [103]. For the trust-region algorithm proposed in this thesis to solve bilevel problems, we use ideas from the work by Apkarian et al. [5], who build a bundle trust-region algorithm for nonsmooth locally Lipschitz functions.

The trust-region method proposed for solving the exact penalty problem (3.13) is as follows. We look to build a model of the optimal value function about the iteration point x_k. As a second order approximation of the function φ could become quite complicated, we focus on forming a first order approximation of the function. We therefore build the following local first order model of the optimal value function about the iteration point x_k, which uses the

Clarke directional derivative of the optimal value function

$$\bar{\varphi}(x) = \varphi(x_k) + \varphi^{\circ}(x_k, x - x_k). \tag{3.14}$$

As it is difficult to find the whole generalised gradient set, we approximate the model at each iteration by finding an arbitrary subgradient $g_k \in \partial\varphi(x_k)$. The approximation is given by

$$\bar{\varphi}_k(x) = \varphi(x_k) + g_k^T(x - x_k). \tag{3.15}$$

for some $g_k \in \partial\varphi(x_k)$.

If we assume that the MFCQ holds for the lower level problem at all points (x, y) where $x = x_k$ and $y \in \Psi(x_k)$, then by Theorem 3.2.12 we can find an arbitrary subgradient $g_k \in \partial\varphi(x_k)$ at each iteration by the following. For a given point x_k, find some $y_k^* \in \Psi(x_k)$ with associated Lagrange multipliers $(\lambda_k^*, \mu_k^*) \in \Lambda(x_k, y_k^*)$, and set

$$g_k = \nabla_x \mathcal{L}(x_k, y_k^*, \lambda_k^*, \mu_k^*). \tag{3.16}$$

Using this, the local approximated model for the optimal value function is the following

$$\bar{\varphi}_k(x) = f(x_k, y_k^*) + \nabla_x \mathcal{L}(x_k, y_k^*, \lambda_k^*, \mu_k^*)^T(x - x_k). \tag{3.17}$$

To formulate the trust-region subproblem, we replace the objective function in the exact penalty problem (3.13) by a locally approximated model function, and constrain the problem to within the trust-region. For the trust-region constraint, we use the infinity norm due to the simplicity of the feasible region and the subsequent relative ease of being able to solve the subproblem, however any norm could be used. For the model of the exact penalty objective function (3.13a), we suggest a number of variants that could be used.

The formulation of the variants for the trust-region subproblem are as follows:

- As the difficulties in solving the problem are due to the optimal value function, the first approximation looks at replacing only the optimal value function in the objective function by the local approximated function $\bar{\varphi}_k(x)$ about the point x_k. The suggested local model of the exact penalty objective function is therefore the following

$$P_k(x, y) = F(x, y) + \gamma(f(x, y) - \bar{\varphi}_k(x)). \tag{3.18}$$

  As this model is only locally approximated around the point x_k, the trust-region

constraint will take the form

$$||x - x_k||_\infty \leq \Delta_k, \tag{3.19}$$

where $\Delta_k > 0$ is the trust-region radius at k.

- Naturally we could also build models of the continuously differentiable functions $F(x, y)$ and $f(x, y)$ about the point (x_k, y_k). We consider both the first order and second order models discussed in Section 3.3.1.

  Let us denote the first order linear models of F and f by $\bar{F}_k(x, y)$ and $\bar{f}_k(x, y)$ respectively, where the linear model is given by equation (3.10). Then the suggested local model of the exact penalty objective function is the following

$$\bar{P}_k(x, y) = \bar{F}_k(x, y) + \gamma(\bar{f}_k(x, y) - \bar{\varphi}_k(x)). \tag{3.20}$$

  We can see the linear model $\bar{P}_k$ takes a more classical trust-region model form, building a first order model around the whole penalty function $P(x, y)$.

  Let us now denote the second order quadratic models of F and f by $\tilde{F}_k(x, y)$ and $\tilde{f}_k(x, y)$ respectively, where the quadratic model is given by equation (3.11), and the Hessian of each function is taken for matrix B_k. Then the final suggested local model of the exact penalty objective function is the following

$$\tilde{P}_k(x, y) = \tilde{F}_k(x, y) + \gamma(\tilde{f}_k(x, y) - \bar{\varphi}_k(x)). \tag{3.21}$$

  As both of these models are locally approximated around the points (x_k, y_k), the trust-region constraint using this model will take the form

$$||(x, y) - (x_k, y_k)||_\infty \leq \Delta_k, \tag{3.22}$$

  where $\Delta_k > 0$.

By using $\bar{P}_k(x, y)$ or $\tilde{P}_k(x, y)$ as opposed to $P_k(x, y)$, the objective function for the trust-region subproblem is a linear or quadratic function. This can be solved efficiently, and the subproblem should be easier and quicker to solve at each iteration. On the other hand, the function $P_k(x, y)$ is a more accurate model of the penalty function, and so it may converge in less iterations. We also note, when F and f are nonlinear, the quadratic model of the functions form a better local approximation than the linear model, and so $\tilde{P}_k(x, y)$ will be a more accurate model of the penalty function than $\bar{P}_k(x, y)$. We investigate the use of all three models in the numerical experiments in Chapter 3.5 to compare their performances.

Using $P_k(x, y)$ for the model in the trust-region subproblem, Algorithm 1 presents the proposed trust-region algorithm to solve the exact penalty problem (3.13).

Algorithm 1 Trust-region algorithm for bilevel programs using the LLVF reformulation

0. **Initialisation.** Set the initial feasible starting points x_0, y_0, trust-region radius Δ_0, parameters $0 < \nu_1 < \nu_2 < 1$ and $0 < \sigma_1 < 1 < \sigma_2$, the penalty parameter $\gamma > 0$, and set $k := 0$. Using Step 2, find an optimal lower level solution for $y_0^* \in \Psi(x^0)$ and the associated unique Lagrange multipliers $(\lambda_0^*, \mu_0^*) \in \Lambda(x_0, y_0^*)$, and calculate the subgradient g_0.
1. **Step calculation.** Solve the following trust-region subproblem to find $(\bar{x}_k, \bar{y}_k)$

$$\begin{aligned} \min_{x,y} \quad & P_k(x, y) = F(x, y) + \gamma(f(x, y) - \bar{\varphi}_k(x)) \\ \text{s.t.} \quad & ||x - x_k||_\infty \leq \Delta_k, \\ & G(x, y) \leq 0, \\ & g(x, y) \leq 0, \\ & h(x, y) = 0. \end{aligned} \tag{3.23}$$

   where $\bar{\varphi}_k(x) = f(x_k, y_k^*) + g_k^T(x - x_k)$.
2. **Find subgradient.** Solve the following lower level program for given $\bar{x}_k$ to find $\bar{y}_k^*$ and the associated Lagrange multipliers $(\bar{\lambda}_k^*, \bar{\mu}_k^*) \in \Lambda(\bar{x}_k, \bar{y}_k^*)$

$$\begin{aligned} \min_{y} \quad & f(\bar{x}_k, y) \\ \text{s.t.} \quad & g(\bar{x}_k, y) \leq 0, \\ & h(\bar{x}_k, y) = 0. \end{aligned}$$

   Set $\bar{g}_k = \nabla_x \mathcal{L}(\bar{x}_k, \bar{y}_k^*, \bar{\lambda}_k^*, \bar{\mu}_k^*)$, where $\bar{g}_k \in \partial\varphi(\bar{x}_k)$.
3. **Acceptance of trial point.** Compute the ratio of achieved versus predicted reduction

$$\rho_k = \frac{P(x_k, y_k) - P(\bar{x}_k, \bar{y}_k)}{P_k(x_k, y_k) - P_k(\bar{x}_k, \bar{y}_k)}$$

   where $P(x, y) = F(x, y) + \lambda(f(x, y) - f(x, y^*))$, $y^* \in \Psi(x)$ and $P_k(x, y)$ is as stated in Step 1.
   If $\rho_k \geq \nu_1$ then the model is deemed accurate and set $(x_{k+1}, y_{k+1}) = (\bar{x}_k, \bar{y}_k)$, $y_{k+1}^* = \bar{y}_k^*$ and $g_{k+1} = \bar{g}_k$. Otherwise set $(x_{k+1}, y_{k+1}) = (x_k, y_k)$, $y_{k+1}^* = y_k^*$ and $g_{k+1} = g_k$.
4. **Trust-region radius update.** Set

$$\Delta_{k+1} = \begin{cases} \sigma_2 \Delta_k & \text{if } \rho_k \geq \nu_2, \\ \Delta_k & \text{if } \rho_k \in [\nu_1, \nu_2), \\ \sigma_1 \Delta_k & \text{if } \rho_k \leq \nu_1. \end{cases} \tag{3.24}$$

5. **Stopping criteria check.** If stopping criteria satisfied, stop and return the points (x_k, y_k^*). Else set $k := k + 1$ and return to Step 1.

Stopping Criteria

For numerical reasons, we suggest a number of stopping criteria. We stop the algorithm if any of the following criteria is met.

SC1 If the stationarity conditions given by (2.14) are all within some tolerance level $\tau > 0$, using the calculated arbitrary subgradient $g_k \in \partial\varphi(x_k)$.

SC2 If after a successful iteration we have either

SC2a
$$||x_{k+1} - x_k|| < \epsilon_1, \tag{3.25}$$

SC2b
$$||F(x_{k+1}, y_{k+1}) - F(x_k, y_k)|| < \epsilon_2, \tag{3.26}$$

for some given tolerance levels $\epsilon_1 > 0$, $\epsilon_2 > 0$.

SC3 If a maximum number of iterations $k_{\max}$ is reached.

SC4 When a maximum number $u_{\max}$ of unsuccessful iterations has been made.

SC5 If the trust-region becomes smaller then some threshold $\Delta_{\min}$.

For **SC1** checking the stationarity conditions (2.14) is difficult due to the generalised gradient of the value function $\partial\varphi(\cdot)$ and the Lagrange multipliers which, in reality, may not be unique. The second criteria is therefore placed to stop the algorithm once it has converged to a point. The remaining stopping criteria are placed to stop the algorithm if no new points are being found, or the algorithm is taking too long to converge.

The penalty parameter

As we know from the exact penalty problem (3.13), the choice of the exact penalty parameter could be crucial for the algorithm to converge to the optimal solution of the bilevel problem. As discussed in Section 2.3.2, for exact penalty problems there are a few approaches that have been taken in the literature to find a suitable $\gamma \in (0, \infty)$. Unfortunately for our problem, approximating the parameter by finding the critical value is difficult due to the construction of the optimality conditions, and currently there is no known rule on how best to choose the value. We therefore choose to test the algorithm with various discrete values of the parameter in the range $\gamma \in (0, \infty)$, and pick the solution that returns the best upper level objective function value, as was suggested in [48].

3.3.3 Convergence analysis

In this section we present the convergence analysis currently undertaken for the proposed trust-region method for bilevel problems. We note that the analysis at present requires strong assumptions on the bilevel program and which models are used.

First, if the optimal value function is continuously differentiable at each iteration point, then the algorithm reduces to a classical trust-region approach. All three suggested models for the trust-region subproblem form at least a good first order model of the objective function, and so convergence follows. We summarise this in the following theorem.

Theorem 3.3.1. *Consider problem (3.13) for a fixed value of $\gamma > 0$ and let f and g be convex in y for each fixed x, while h is affine linear in y for each fixed x. Furthermore, assume that the constraint set K is nonempty, closed, and convex. Assume that $(\hat{x}, \hat{y})$ is an accumulation point of a sequence of successful iterations $\{x_k, y_k\}$ generated by Algorithm 1 using $P_k(x, y)$, $\bar{P}_k(x, y)$ or $\tilde{P}_k(x, y)$, where LICQ and SOSC hold for the lower level problem at each point $\{x_k, y_k\}$. Then $\{\hat{x}_k, \hat{y}_k\}$ is a stationary point of the exact penalty problem (3.13).*

Proof. Under the assumptions taken, it follows from Proposition 3.2.14 that the value function $\varphi(x)$ is continuously differentiable at each point x_k, and so the set $\partial\varphi(x_k)$ at each iteration point reduces to a single point. Hence, the algorithm is reduced to the classical trust-region algorithm for smooth functions with a convex constraint set. For some fixed value of γ, the models of the exact penalty objective function $P_k(x, y)$, $\bar{P}_k(x, y)$ or $\tilde{P}_k(x, y)$, used in the trust-region subproblem (3.20), form at least a good first order model of the objective function. Therefore, the remaining steps of the proof can follow the path used in [20]. □

Remark 3.3.2. In this particular case, when the optimal value function is locally continuously differentiable, further work could explore using the second order derivative of the optimal value function to form a second order model of the whole penalty function.

We next show that the optimal value function does not necessarily have to be smooth for Algorithm 1 to converge to a stationary point. We do this by using a result shown by Apkarian et al. [5] in their recently proposed bundle trust-region algorithm for nonsmooth functions. To do so, we first introduce the notion of an upper-C^1 function, defined as the following.

Definition 3.3.3. *[111] A locally Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$ is lower-C^1 at $x^0 \in \mathbb{R}^n$ if there exists a compact space $\mathbb{K}$, a neighbourhood U of x^0, and a mapping $F : \mathbb{R}^n \to \mathbb{R}$ such that*

$$f(x) = \max_{y \in \mathbb{K}} F(x, y)$$

for all $u \in U$, and F and $\frac{\partial F}{\partial x}$ are jointly continuous. The function f is said to be upper-C^1 at x^0 if $-f$ is lower-C^1 at x^0.

Daniilidis and Georgiev [21] show that locally Lipschitz functions are lower-C^1 functions if and only if they are approximately convex. A function $f : X \to \mathbb{R} \cup \{+\infty\}$ is defined to be approximately convex at $x_0 \in X$ if for every $\epsilon > 0$ there exists $\delta > 0$ depending on x_0 and ϵ such that for all $x, y \in B(x_0, \delta)$ and $t \in (0, 1)$

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) + \epsilon t(1-t)||x - y||.$$

Clearly if a function is convex, it is approximately convex. Additionally, Apkarian et al. show convergence for their algorithm when the model used in the trust-region subproblem is a *strict* first order model, see [5]. This is a stronger condition than a first order model, for example, for the first order model (3.10) of a differentiable function $f(x)$ to be a strict first order model, the function must be strictly differentiable. They also show if a function is concave, the standard first order model using the Clarke generalised gradient is also strict. For a discussion on strict first order models see [93], [5].

With this in mind, we obtain the following theorem of convergence. This theorem currently only shows convergence of the algorithm when using the first order model $\bar{P}_k(x, y)$.

Theorem 3.3.4. *Consider problem (3.13) for a fixed value of $\gamma > 0$, let the functions F and f be strictly differentiable, and let the lower level problem be fully convex. Furthermore, assume the constraint set K is nonempty, closed, and convex. Assume that $(\hat{x}, \hat{y})$ is an accumulation point of a sequence of successful iterations $\{x_k, y_k\}$ generated by Algorithm 1 using $\bar{P}_k(x, y)$, where MFCQ holds for the lower level problem at each point $\{x_k, y_k\}$. Then $\{\hat{x}_k, \hat{y}_k\}$ is a stationary point of the exact penalty problem (3.13).*

Proof. Under the assumptions taken, it follows from Proposition 3.2.4 that the optimal value function $\varphi(x)$ is convex. Therefore, the penalty function of the exact penalty problem (3.13), given by the following equation

$$P(x, y) = F(x, y) + \gamma(f(x, y) - \varphi(x)),$$

is formed of the smooth, strictly differentiable functions F and f, and a concave nonsmooth function $-\varphi(x)$. As the smooth functions F and f are strictly differentiable, then their first order model is strict. Additionally, as the nonsmooth function $-\varphi(x)$ is a concave function, it is upper-C^1, and the first order model is also strict. We can therefore apply the result provided by Apkarian et al. in [5, Theorem 2]. They propose a bundle trust-region algorithm to solve nonsmooth functions with convex constraints, that builds a first order model of the objective function. They show that if the nonsmooth function we wish to optimise is an upper-C^1 function, then the classical trust-region algorithm can be used to solve the problem, where an arbitrary subgradient is used in replace of the Clarke generalised gradient in the first order model function, and the algorithm converges to a stationary point. Hence, by taking the first order model $\bar{P}_k(x, y)$ of the penalty function $P(x, y)$, where for the smooth functions we take the linear model (3.10), and for the optimal value function we take the first order model (3.15), then the model $\bar{P}_k(x, y)$ is strict, and by [5, Theorem 2] the accumulation point $\{\hat{x}, \hat{y}\}$ is a stationary point of the exact penalty problem (3.13). □

Remark 3.3.5. Under these conditions, convergence is only shown for the model that builds a first order model of the whole penalty function, which takes a more classic trust-region model form. We note the other suggested models for the algorithm form a better local approximation of the penalty function, so, although convergence is not yet shown under these conditions, we could expect them to perform better in practise.

The convergence of the trust-region algorithm is shown under relatively strong assumptions on the bilevel problem. By relaxing some of these assumptions, we note the choice of the subgradient for the first order model of the optimal value function may impact the performance of this algorithm. Thus, the evaluation of the $y \in \Psi(x_k)$ and the choice of the associated Lagrange multipliers at x_k may be crucial to the convergence. Because of this, the algorithm may need to be adapted to prove convergence under weaker assumptions. For example, a bundle approach could be explored to enrich the generalised gradient, like in [103]. Alternatively, we find that the trust-region algorithm by Hoseini and Nobakhtian [57] could be applied to a general bilevel problem with no upper or lower level constraints. Using ideas from their algorithm, which makes use of the Goldstein ϵ-subdifferential, could help adapt the algorithm to show convergence for more general problems.

Nevertheless, although convergence is only shown for particular classes of bilevel problems, we test how the algorithm performs on a large test set of nonlinear problems with a range of convex and nonconvex problems, to see how it works in practise. The numerical results are presented in Section 3.5.

3.4 The Convex-Concave Procedure

The following heuristic algorithm uses a similar technique and approach to the trust-region algorithm introduced in Section 3.3. Let us again consider the optimal value reformulation of the bilevel problem, given by equations (2.10). The following assumptions are made on the bilevel problem.

A1 $F(x, y)$ and $f(x, y)$ are fully convex.

A2 The constraint region set $K = \{(x, y) : G(x, y) \leq 0,\ g(x, y) \leq 0,\ h(x, y) = 0\}$ is a convex, nonempty, and compact set.

A3 The MFCQ holds for the lower level problem.

A4 The bilevel problem is partially calm.

Under the assumptions on the lower level problem, the optimal value function is a convex function. We therefore find the constraint in the LLVF reformulation given by $f(x)-\varphi(x) \leq 0$ is a difference of two convex functions, known as a DC function.

We apply a heuristic algorithm known as the convex-concave procedure (CCP), first introduced by Yuille and Rangarajan [136]. The idea behind the algorithm is to convexify the concave part of the function by linearising it. For example, consider the following continuous DC problem

$$\begin{aligned} \min_{x} \quad & f(x) = u_0(x) - v_0(x), & \\ \text{s.t.} \quad & u_i(x) - v_i(x) \leq 0, & \forall\, i = 1, \ldots, m. \end{aligned}$$

where $x \in \mathbb{R}^n$, and the functions $u_i : \mathbb{R}^n \to \mathbb{R}$, $v_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 0, \ldots, m$ are convex and continuously differentiable. The CCP algorithm linearises the concave parts of the DC functions about the iteration point x_k, and the procedure solves the following sequence of convex programs at each iteration.

$$\begin{aligned} \min_{x} \quad & f(x) = u_0(x) - \bar{v}_0(x, x_k), & \\ \text{s.t.} \quad & u_i(x) - \bar{v}_i(x, x_k) \leq 0, & \forall\, i = 1, \ldots, m. \end{aligned}$$

where we have

$$\bar{v}_i(x, x_k) = v_i(x_k) + \nabla g_i(x_k)^T (x - x_k) \qquad \forall i = 0 \ldots, m. \tag{3.27}$$

This linearisation provides an upper bound to the concave function and transforms the problem into a convex optimisation program, a problem which can be solved relatively easily. One suggested stopping criteria for this method is when the improvement on the objective value is less than some threshold δ, that is

$$(f_0(x_k) - g_0(x_k)) - (f_0(x_{k+1}) - g_0(x_{k+1})) \leq \delta.$$

The heuristic can be shown to converge to a critical point of the problem. The proof of convergence can be found in [71] and [112]. It essentially follows the logic that linearising the concave part of the DC function provides an upper bound on the function, ensuring all iterations are feasible. They then show that the algorithm is a descent method. In addition to this, the algorithm can also be shown to converge to a critical point when the concave functions $-v_i$ are nondifferentiable, described in [71]. In this case, the gradient is replaced by an arbitrary subgradient. Because the nonsmooth functions are concave, the linearisations using their subgradients still provide an upper bound on the functions, and the proof follows the same argument.

We use the same idea to convexify the optimal value function in the optimal value reformulation, using the same linear approximation of the optimal value function that we use in the trust-region algorithm, given by equation (3.17). Again, we look to solve the exact penalty problem (3.13), with the hope of choosing the penalty parameter well enough for the local solution to be a local solution of the bilevel problem. However taking a slightly different approach, we formulate the exact penalty problem with a slack variable $s \in \mathbb{R}$, as was suggested in the penalty CCP method in [71]. The slack variable essentially measures the violation of the constraint containing the optimal value function, and it is the slack variable that we then penalise.

Algorithm 1 presents the CCP algorithm to solve the LLVF reformulation of the bilevel problem.

Algorithm 2 CCP algorithm for bilevel programs using the LLVF reformulation

0. **Initialization.** Set the initial feasible starting point x_0, the penalty parameter $\gamma > 0$, and set $k := 0$.
1. **Find subgradient.** Solve the following lower level program for given x_k to find y_k^* and the associated Lagrange multipliers $(\lambda_k^*, \mu_k^*) \in \Lambda(x_k, y_k^*)$

$$\begin{aligned} \min_{y} \quad & f(x_k, y) \\ \text{s.t.} \quad & g(x_k, y) \leq 0, \\ & h(x_k, y) = 0. \end{aligned}$$

   Set $g_k = \nabla_x \mathcal{L}(x_k, y_k^*, \lambda_k^*, \mu_k^*)$, where $g_k \in \partial\varphi(x_k)$.
2. **Convexify.** Set $\bar{\varphi}_k(x) = f(x_k, y_k^*) + g_k^T(x - x_k)$.
3. **Solve.** Solve the following convex penalty subproblem to find x_{k+1}

$$\begin{aligned} \min_{x,y,s} \quad & F(x, y) + \gamma s, \\ \text{s.t.} \quad & f(x, y) - \bar{\varphi}_k(x) \leq s, \\ & G(x, y) \leq 0, \\ & g(x, y) \leq 0, \\ & h(x, y) = 0, \\ & s \geq 0. \end{aligned} \tag{3.28}$$

4. **Update iteration** If stopping criteria satisfied, stop and return the points (x_k, y_k^*). Else set $k := k + 1$ and go to Step 1.

We note that both the trust-region approach and the CCP heuristic presented for the bilevel problem are similar in their method, except the CCP algorithm does not force the iteration point to stay within a trust-region and it requires all functions including F to be fully convex. The CCP subproblem is however a continuous convex problem that can be solved efficiently. In Section 3.5, we provide numerical results of the CCP heuristic on bilevel problems with fully convex lower level problems to compare with the trust-region method. To implement this algorithm, we use the stopping criteria **SC2-SC3** proposed in Algorithm 1.

We conclude this section by highlighting the following. Under the assumptions made, the exact penalty problem we look to solve is a single level DC problem. These problems have been widely researched, and a number of different algorithms in addition to the CCP method have been developed to solve them which could be applied, see for instance [56].

3.5 Numerical Results

This section presents the extensive numerical experiments conducted to test the performance of the trust-region algorithm and the CCP heuristic. We first detail the problems tested and the implementation details of the algorithms. Following this, we present the results of the experiments, before providing detailed analysis on the performance of the algorithms. The analysis includes a comparison to a number of other algorithms in the literature, and an investigation into the choice of penalty parameter, starting points, and stopping criteria parameters.

We implemented the algorithms on the 124 nonlinear bilevel problems provided in the BOLIB library by Zhou et al. [139]. The BOLIB library is a collated list of examples from the bilevel literature, which gives a full description of the problems including the size of each problem and the best known objective values that have been found. The size of each problem is relatively small. All variables have size $n \leq 10$ and $m \leq 10$, and the total number of constraints for each problem is less than 30.

We split the 124 problems of the BOLIB library into three groups. These are the following.

G1 Problems where the lower level objective function $f(x, y)$ is fully convex in both variables x and y

G2 Problems where the lower level objective function $f(x, y)$ is convex in y

G3 Problems where the lower level objective function $f(x, y)$ is nonconvex

In total we find that group G1 contains 40 examples, G2 contains 45 examples, and G3 contains 39 examples. From our current convergence analysis, we anticipate that the trust-region algorithm should perform well on problems where $f(x, y)$ are convex in both x and y, and those where $f(x, y)$ is convex in y and where y is unique for a given x. As the CCP method is developed to work on DC problems, we test this algorithm on problems from G1 only.

3.5.1 Implementation and parameter details

The algorithms were implemented in MATLAB R2019a. We used the built-in MATLAB nonlinear optimisation solver *fmincon* to solve the trust-region subproblem (3.23), the CCP subproblem (3.28), and to solve the lower level problem to calculate a subgradient of the optimal value function. The solver *fmincon* uses an interior point method.

We tested the trust-region algorithm on all three of the local approximations of the penalty function suggested. The first function $P_k(x, y)$, given by equation (3.18), only builds a linear approximation of the optimal value function $\varphi(x)$ in the penalty function. The other two functions build approximations of the whole penalty function, forming a linear model of $\varphi(x)$ and a first or second order model for the functions F and f. The function $\tilde{P}_k(x, y)$, given by (3.21), uses quadratic models for the functions F and f, whilst the function $\hat{P}_k(x, y)$, given by equation (3.20), uses linear models. We implement the approach on all three models to compare their performance, with the aim of helping aid practitioners in choosing which model they feel is most appropriate for their problem. Thus, in total we test four algorithms, which we refer to as the following:

- **BLTR.** The trust-region algorithm for bilevel problems using the objective function model $P_k(x, y)$ for the trust-region subproblem.
- **BLTR-Q.** The trust-region algorithm for bilevel problems using the objective function model $\tilde{P}_k(x, y)$ for the trust-region subproblem, which forms quadratic models of F and f.
- **BLTR-L.** The trust-region algorithm for bilevel problems using the objective function model $\bar{P}_k(x, y)$ for the trust-region subproblem, which forms linear models of F and f.
- **CCP.** The convex-concave procedure for bilevel problems.

As the ability for an algorithm to find a local solution can depend on the choice of starting point, we implement the algorithms using two different starting points. As the algorithms require the starting point to be feasible, the starting points were problem specific and in general chosen using the following approach. Firstly, because the BOLIB problems are from the bilevel literature, if a starting point was provided in the original source of the problem, we used this point. The starting points for these problems including their sources can be found in [47]. For many of these examples however, only one starting point was provided in the source, and in some cases no starting point was provided. If this was the case, simple starting points such as $(0, \ldots, 0)^T$ or $(1, \ldots, 1)^T$ were chosen, provided that they were feasible, or otherwise we chose a point that could easily be gotten to be feasible. For example, if the variables had box constraints then a combination of the upper and lower bounds on the variables could be chosen.

In addition to the starting point, to implement the algorithms a value of the exact penalty parameter $\gamma > 0$ is required. This choice can impact on whether the algorithm finds a good solution of the exact penalty problem that is also a local optimal solution of the bilevel

problem, but the value may also be problem specific. As stated in Fischer et al. [48], there is no known method on how to find the best value. Therefore, the best course of action is to try the algorithm for a finite discrete number of penalty parameter values within the range $(0, \infty)$, and choose the best solution from that. We chose the same set of parameter values as in Fischer et al. [48], which are nine values in the set $\gamma \in \Gamma$, where

$$\Gamma = \{2^{-1}, 2^0, 2^1, \ldots, 2^7\}.$$

We therefore solve each problem at two different starting points for 9 different penalty parameters each, solving each problem a total of 18 times.

For the stopping criteria, we used the criteria **SC1**-**SC5** for the trust-region algorithm and **SC2**-**SC3** for the CCP method. The values chosen for the stopping criteria and trust-region parameters are outlined in Table 3.1.

Table 3.1: Parameters for numerical experiments

Initial trust-region radius for **G1**	Δ_0	5
Initial trust-region radius for **G2** and **G3**	Δ_0	1
Other trust-region parameters	ν_1	0.01
	ν_2	0.9
	σ_1	0.25
	σ_2	2.5
Stopping criteria	ϵ_1	1e-4
	ϵ_2	1e-6
	τ	1e-4
	$k_{\max}$	500
	$u_{\max}$	25
	$\Delta_{\min}$	1e-4

We tried various values for the trust-region parameters to test what impact they had on the solutions. For the initial trust-region radius we tried the values in the set $\Delta_0 = \{1, 5, 10\}$. The tests indicated that the starting radius did not have a large impact on the solutions found, although we did see that a smaller value tended to lead to a slight increase in average computational time and average iterations. We also tended to see a smaller starting radius performing better on the problems where the lower level objective function was nonlinear or only convex in y, and a larger radius performing better when it was fully convex. We therefore chose a starting trust-region radius of 5 for G1 and 1 for G2 and G3, the latter being a value suggested to use by Conn et al. [20]. We also found that the best starting parameter may be problem specific, for example a smaller starting radius may restrict the problem to find a local solution before finding a global solution. For this reason two problems

from groups G2 and G3 needed a larger starting radius of $\Delta_0 = 5$, which were problems `GumusFloudas2001Ex3` and `MacalHurter1997`.

For the choice of the other trust-region parameters, we chose the parameters suggested by Conn et al. [20]. We also tried several other options suggested in the literature, these were $(\nu_1, \nu_2, \sigma_1, \sigma_2) = (\frac{1}{3}, \frac{2}{3}, 0.5, 2)$ suggested in [80], $(\nu_1, \nu_2, \sigma_1, \sigma_2) = (0.01, 0.90, 0.6, 1.4)$ suggested in [18] and $(\nu_1, \nu_2, \sigma_1, \sigma_2) = (0.25, 0.75, 0.5, 2)$, which is a standard choice chosen in the literature as stated in [54]. These parameters tended not to have a large impact on the solution. For the stopping criteria parameters, a discussion on the values and their impact can be found in Section 3.5.3.

3.5.2 Test Results

In order to evaluate the performance of the algorithm we compare our solutions, where possible, with the solutions known in the literature. These are provided in the BOLIB library [139]. Out of the 124 problems, 6 of the problems either have no optimal solution or the solution is unknown. For these problems we compare the solutions between the different algorithms we test. We compare our solutions in the same way that is done in [48]. We refer to the status of the provided solution of a problem as the following. When we know the solution is optimal we denote the status by "O", when a solution is known, but we do not know if it is optimal, we denote the by status "K", when the solution is unknown we denote the status by "U", and when no optimal solution exists we denote the status as "N".

Out of the 118 problems where a solution is provided, 82 of these are known to be optimal, and 36 problems are only known. We note here that the solution status for `DempeLohse2011Ex31a` in Fischer et al. [48] was stated as optimal, with an optimal solution of $F(x, y) = -6.00$. This was found to be incorrect and updated in the BOLIB library [139], with a new suggestion that the actual optimal solution is $F(x, y) = -5.50$. We therefore use this solution, but change the status to from optimal to known. Furthermore, the solution given in the BOLIB library for the example `MitsosBarton2006Ex318`, originally given in Mitsos and Barton [89], is $F^* = -0.25$, $f^* = 0.00$. It is however shown in [96] that the optimal solution is at $(x^*, y^*) = (1.0, 0.0)$, with $F^* = -1.0$ and $f^* = 0.0$.

For some examples in the literature, local solutions are provided in addition to a global solution. As we do not expect our algorithms to solve globally, we also compare our solution to the local solutions provided. Following this we note a further mistake in the provided solutions in the literature. Example `DempeFranke2011Ex42` has three strict local minimum solutions, as shown by Dempe and Franke [28]. However, one of the local solutions given is incorrect. They give the points $x = (0, -1)$, $y = (2, 2)$, to obtain a value of $F(x, y) = 4$,

however the lower level solution point should be $y = (1, 2)$, which correctly gives the objective function value $F(x, y) = 4$.

To compare our solutions to the solutions provided in the literature, we use a similar method to that used in [48]. For the algorithms we test, the solution obtained by all algorithms is (x_k, y_k^*), where $y_k^* \in \Psi(x_k)$. We therefore know that our solution is optimal for the lower level and is therefore feasible for the bilevel problem. Thus, we only need to compare the value of the upper level objective function found by the algorithm with the best known value provided in the literature.

We define the following function as a measure of comparing the results with the best known solution

$$\delta_F := \frac{F(x, y) - F^*}{\max\{1, |F^*|\}}, \tag{3.29}$$

where (x, y) is the solution found by the algorithm, and F^* is the best known solution from the literature. If local solutions are provided, we compare with all values and pick the value that produces the smallest δ. We then set

$$\delta = \begin{cases} |\delta_F|, & \text{if status is optimal,} \\ \delta_F, & \text{otherwise.} \end{cases} \tag{3.30}$$

This allows the measure δ to become negative if a better solution is found then the known optimal. The smallest value of δ considering the 9 different penalty parameters for both starting points is then found, denoted by δ^*.

We present the results in three tables, separated into each group of test problems. In the tables we present the solution for δ^*, which is the best solution found from all penalty parameters and starting points. F^* denotes the best known solution found in the literature, whilst F denotes the solution found by the tested algorithm. In cases where our algorithm finds the local minimum, we mark the solution value F found by the algorithm with an asterisk to show its a local solution. We also report the CPU time, given in seconds, and the number of iterations, which only account for the number of outer iterations of the trust-region method and the CCP heuristic, and does not account for the iterations needed to solve the subproblems.

For further details on the solutions found for the trust-region algorithm, Table A.1 in the appendix gives a detailed table on the upper level objective function values found from all three of the trust-region subproblems tested, on all 124 problems for each penalty parameter and both starting points.

Bilevel problems that have a fully convex lower level objective function

Table 3.2 gives the results for the 40 examples from the BOLIB library in **G1**.

Table 3.2: Best solutions found from algorithms BLTR, BLTR-Q, BLTR-L and CCP for bilevel problems in G1

Example	Solution		BLTR				BLTR-Q				BLTR-L				CCP			
	Status	F^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*
AiyoshiShimizu1984Ex2	O	5.00	5.01	3	0.18	0.00	5.01	4	0.29	0.00	8.38	24	2.15	0.16	5.01	3	0.11	0.00
Bard1988Ex2	O	-6600.0	-6600.0	6	0.11	0.00	-6600.0	20	0.42	0.00	-6600.0	15	1.07	0.00	-6600.0	8	0.26	0.00
Bard1988Ex3	O	-12.68	-12.68	21	0.37	0.00	-12.68	32	0.83	0.00	-12.68	67	4.68	0.00	-12.68	28	0.65	0.00
BardBook1998	O	0.00	0.00	3	0.05	0.00	0.00	2	0.05	0.00	0.00	34	2.05	0.00	0.00	2	0.03	0.00
ClarkWesterberg1990a	O	5.00	5.00	23	0.27	0.00	5.00	45	0.61	0.00	5.00	10	0.28	0.00	5.00	293	4.05	0.00
Colson2002BIPA1	O	250.0	250.0	2	0.03	0.00	249.9	4	0.15	0.00	250.0	2	0.04	0.00	250.0	2	0.04	0.00
Colson2002BIPA3	K	2.00	2.00	11	0.11	0.00	2.00	5	0.07	0.00	2.00	3	0.06	0.00	2.00	34	0.55	0.00
Colson2002BIPA5	K	2.75	2.75	5	0.08	0.00	2.75	10	0.20	0.00	2.75	9	0.24	0.00	2.75	500	6.84	0.00
Dempe1992a	U	0.00	-0.01	10	0.25		-0.01	2	0.06		0.00	5	0.12		-0.01	7	0.12	
Dempe1992b	O	31.25	31.25	1	0.02	0.00	31.25	1	0.02	0.00	31.25	37	0.65	0.00	31.25	4	0.04	0.00
DempeDutta2012Ex31	O	-1.00	-0.98	500	9.66	0.02	-0.99	500	12.86	0.01	-1.46	500	24.72	0.46	-0.97	500	11.11	0.03
DeSilva1978	O	-1.00	-1.00	2	0.06	0.00	-1.00	2	0.10	0.00	-1.00	10	1.34	0.00	-1.00	11	0.24	0.00
FalkLiu1995	O	-2.25	-2.25	29	0.59	0.00	-2.25	29	0.78	0.00	-2.25	16	2.01	0.00	-2.25	2	0.06	0.00
FloudasEtal2013	O	0.00	0.00	18	0.28	0.00	0.00	14	0.30	0.00	0.00	7	0.48	0.00	0.00	4	0.07	0.00
FloudasZlobec1998	O	1.00	1.00	2	0.05	0.00	1.00	2	0.06	0.00	1.00	2	0.07	0.00	1.00	500	12.40	0.00
GumusFloudas2001Ex1	O	2250.0	2307.2*	2	0.06	0.00	2307.2*	10	0.31	0.00	2250.0	6	0.13	0.00	2307.2*	3	0.07	0.00
GumusFloudas2001Ex4	O	9.00	9.00	2	0.03	0.00	9.00	2	0.03	0.00	9.00	10	0.18	0.00	9.00	2	0.03	0.00
HatzEtal2013	O	0.00	0.00	11	0.42	0.00	0.00	11	0.57	0.00	0.30	11	0.52	0.30	-0.01	500	9.60	0.01
IshizukaAiyoshi1992a	O	0.00	0.00	1	0.02	0.00	0.00	1	0.02	0.00	0.00	1	0.02	0.00	0.00	1	0.02	0.00
LamparSagrat2017Ex23	O	-1.00	-1.00	1	0.02	0.00	-1.00	61	1.44	0.00	-1.00	6	0.15	0.00	-1.00	2	0.05	0.00
LamparSagrat2017Ex31	O	1.00	1.00	1	0.01	0.00	1.00	1	0.02	0.00	1.00	3	0.04	0.00	1.00	2	0.06	0.00
LamparSagrat2017Ex32	O	0.50	0.50	2	0.02	0.00	0.50	2	0.03	0.00	0.52	5	0.13	0.02	0.50	2	0.05	0.00
LamparSagrat2017Ex33	O	0.50	0.50	1	0.03	0.00	0.50	1	0.04	0.00	0.50	139	2.50	0.00	0.50	3	0.07	0.00
LamparSagrat2017Ex35	O	0.80	0.80	1	0.02	0.00	0.80	1	0.02	0.00	0.80	15	0.37	0.00	0.80	2	0.03	0.00
LuDebSinha2016c	K	1.12	1.12	6	0.09	0.00	1.12	8	0.24	0.00	1.12	14	1.65	0.00	1.12	32	0.54	0.00
LuDebSinha2016d	U	0.00	-56.27	6	0.13		-56.27	6	0.19		-56.27	6	0.14		-56.27	2	0.07	
LuDebSinha2016e	U	0.00	1.10	4	0.06		1.10	4	0.07		1.10	7	0.11		1.10	4	0.09	
LuDebSinha2016f	U	0.00	-87.50	14	0.27		-87.50	14	0.46		-25.57	19	0.51		-87.50	7	0.21	
MitsosBarton2006Ex323	O	0.18	0.18	2	0.05	0.00	0.18	2	0.05	0.00	0.18	2	0.05	0.00	0.18	3	0.11	0.00
SahinCiric1998Ex2	O	5.00	5.00	9	0.10	0.00	5.00	15	0.20	0.00	5.00	27	0.51	0.00	5.00	25	0.34	0.00
ShimizuAiyoshi1981Ex1	O	100.0	99.83	3	0.04	0.00	99.83	3	0.05	0.00	99.83	31	1.13	0.00	99.83	3	0.06	0.00
ShimizuAiyoshi1981Ex2	O	225.0	225.0	19	0.29	0.00	225.0	22	0.49	0.00	225.0	18	0.78	0.00	225.0	44	0.85	0.00
ShimizuEtal1997b	O	2250.0	2250.0	95	0.95	0.00	2250.0	96	1.03	0.00	2250.0	14	0.26	0.00	2250.0	21	0.34	0.00
SinhaMaloDeb2014TP3	K	-18.68	-18.68	1	0.02	0.00	-18.68	1	0.02	0.00	-18.68	6	0.39	0.00	-18.68	3	0.07	0.00
SinhaMaloDeb2014TP8	O	0.00	0.00	4	0.08	0.00	0.00	4	0.12	0.00	0.03	126	6.16	0.03	0.00	16	0.40	0.00
TuyEtal2007	O	22.50	22.50	1	0.02	0.00	22.50	1	0.02	0.00	22.50	14	0.34	0.00	22.50	2	0.03	0.00
WanWangLv2011	O	10.63	10.63	10	0.22	0.00	10.63	4	0.12	0.00	10.63	10	0.41	0.00	10.63	2	0.06	0.00
Yezza1996Ex41	O	0.50	0.50	2	0.03	0.00	0.50	2	0.04	0.00	0.50	7	0.17	0.00	0.50	3	0.05	0.00
Zlobec2001a	O	-1.00	-1.00	13	0.37	0.00	-1.00	11	1.34	0.00	-1.00	22	4.09	0.00	-0.26	500	10.21	0.74
Zlobec2001b	N	0.00	0.00	1	0.04		0.00	1	0.05		0.00	1	0.12		0.00	2	0.07	

The results are summarised in Table 3.3.

Table 3.3: Summary of results for G1

Algorithm	Number of problems			Iterations		CPU (s)	
	$\delta^* \geq 0.01$	$\delta^* \geq 0.02$	$\delta^* \geq 0.05$	Average	Median	Average	Median
BLTR	1	0	0	21.20	3.5	0.39	0.07
BLTR-Q	1	0	0	23.90	4	0.60	0.12
BLTR-L	5	5	3	31.53	10	1.52	0.38
CCP	2	2	1	77.10	3.5	1.50	0.08

As we can see from the summarised table, the algorithms perform extremely well. In total there are 40 problems, of which there are 5 problems where the solution is either unknown or there is none.

For the trust-region method, the algorithms BLTR and BLTR-Q performed better than BLTR-L. For BLTR and BLTR-Q, only one problem had a larger δ^* value than 0.01, which was problem `DempeDutta2012Ex31`. The algorithms reached the maximum number of iterations, and the solutions obtained were very close to the optimal, with the solutions $F = -0.98,\ f = 3.93$ for BLTR, and $F = -0.99,\ f = 3.95$ for BLTR-Q, where the optimal is $F^* = -1$ and $f^* = 4$. The algorithm BLTR-L also reached the maximum number of iterations for this problem, although it did not come close to finding the optimal solution. It suggests for this problem the maximum number of iterations should have been set higher. A discussion on the stopping criteria parameters chosen can be found in Section 3.5.3. Although BLTR-L did not perform as well, with five problems having a δ^* greater than 0.01, this can still be seen as a high proportion of problems solved. Finally, we found that the CCP method also performed very well on the test set, performing better than BLTR-L. Like the other algorithms, it also did not quite find the optimal solution for problem `DempeDutta2012Ex31`, but could also not find a solution for `Zlobec2001a`.

We found that for the examples where the solutions are unknown, every algorithm found the same answer. It should be noted that `Zlobec2001b` is an example used to illustrate that the feasible set of a bilevel optimization problem is not necessarily closed. As stated in [142], the problem does not have an optimal solution.

The median number of iterations and CPU time for the trust-region algorithm is relatively small, with BLTR-L taking the most number of iterations and BLTR the least. This may be because BLTR forms the best approximation of the penalty function, followed by BLTR-Q, and so less iterations are required to get to the solution of the penalty function. The average CPU time and average number of iterations for the CCP algorithm are on average higher than the trust-region method, although the median values were similar to BLTR and BLTR-Q. This could be attributed to the higher number of problems reaching the maximum number of iterations, a discussion of which can be found in Section 3.5.3.

Bilevel problems that have a lower level objective function that is convex in y

Table 3.4 gives the results for the 45 examples from the BOLIB library in **G2**.

Table 3.4: Best solutions found from algorithms BLTR, BLTR-Q and BLTR-L for bilevel problems in G2

Example	Solution		BLTR				BLTR-Q				BLTR-L			
	Status	F^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*
AllendeStill2013	O	1.00	1.00	14	1.16	0.00	1.01	6	0.51	0.01	1.16	7	0.80	0.16
Bard1988Ex1	O	17.00	17.00	2	0.03	0.00	17.00	2	0.04	0.00	17.00	2	0.05	0.00
Bard1991Ex1	O	2.00	2.00	2	0.04	0.00	2.00	3	0.07	0.00	3.00	2	0.17	0.50
CalamaiVicente1994a	O	0.00	0.00	31	2.19	0.00	0.00	6	0.33	0.00	0.00	3	0.09	0.00
CalamaiVicente1994b	O	0.31	0.31	86	5.89	0.00	0.31	57	5.77	0.00	0.35	30	7.23	0.03
CalamaiVicente1994c	O	0.31	0.31	62	4.57	0.00	0.46	12	0.89	0.14	0.34	15	2.38	0.03
Colson2002BIPA2	K	17.00	17.00	2	0.03	0.00	17.00	2	0.04	0.00	17.00	2	0.05	0.00
DempeDutta2012Ex24	O	0.00	0.00	1	0.12	0.00	0.00	1	0.15	0.00	0.00	1	0.27	0.00
DempeEtal2012	O	-1.00	-1.00	3	0.05	0.00	-1.00	1	0.03	0.00	-1.00	3	0.07	0.00
DempeFranke2011Ex41	O	5.00	5.00	9	0.26	0.00	5.00	10	0.97	0.00	5.00	7	0.23	0.00
DempeFranke2011Ex42	O	2.13	2.13*	4	1.32	0.00	4.00*	14	1.06	0.00	3.03	9	0.56	0.01
DempeFranke2014Ex38	O	-1.00	-1.00	1	0.03	0.00	-1.00	1	0.03	0.00	-1.00	3	0.09	0.00
DempeLohse2011Ex31a	K	-5.50	-5.75	5	0.13	-0.05	-5.75	7	0.20	-0.05	-5.45	9	1.42	0.01
DempeLohse2011Ex31b	O	-12.00	-12.00	1	0.09	0.00	-12.00	2	0.09	0.00	-12.00	14	2.24	0.00
HendersonQuandt1958	K	-3266.7	-3266.7	53	0.90	0.00	-3266.7	20	0.35	0.00	-3266.6	61	0.87	0.00
HenrionSurowiec2011	O	0.00	0.00	1	0.01	0.00	0.00	1	0.01	0.00	0.00	1	0.01	0.00
LucchettiEtal1987	O	0.00	0.00	2	0.04	0.00	0.00	2	0.04	0.00	0.00	2	0.03	0.00
MacalHurter1997	O	81.33	81.33	12	0.20	0.00	81.98	12.00	0.28	0.01	123.3	210	3.97	0.52
MitsosBarton2006Ex38	O	0.00	0.00	13	0.28	0.00	0.01	14	0.52	0.01	1.00	12	0.33	1.00
MitsosBarton2006Ex313	O	-1.00	-1.77	201	3.14	0.77	-0.50	3	0.07	0.50	-2.00	2	0.03	1.00
MitsosBarton2006Ex317	O	0.19	0.19	6	0.21	0.00	0.19	5	0.14	0.00	0.24	4	0.14	0.05
MitsosBarton2006Ex324	O	-1.75	-1.75	2	0.03	0.00	-1.75	6	0.11	0.00	-1.67	5	0.14	0.05
MitsosBarton2006Ex325	K	-1.00	-1.00	3	0.78	0.00	-1.00	3	0.17	0.00	-1.00	12	0.47	0.00
MitsosBarton2006Ex326	O	-2.35	-2.35	2	0.06	0.00	-2.35	2	0.08	0.00	-2.35	2	0.14	0.00
MorganPatrone2006a	O	-1.00	-1.00	18	0.49	0.00	-1.00	11	0.33	0.00	-1.00	13	0.49	0.00
MuuQuy2003Ex1	K	-2.08	-2.09	44	0.72	0.00	-2.05	6	0.14	0.01	-2.07	28	1.91	0.00
MuuQuy2003Ex2	K	0.64	0.64	6	0.09	0.00	0.64	6	0.11	0.00	0.64	20	0.73	0.00
NieWangYe2017Ex34	O	2.00	2.00	2	0.22	0.00	2.00	2	0.25	0.00	2.00	2	0.28	0.00
NieWangYe2017Ex54	O	-0.44	-0.44	74	9.94	0.00	-0.44	35	3.78	0.00	-0.07	6	2.05	0.37
Outrata1990Ex1c	K	-12.00	-12.00	13	0.81	0.00	-12.00	7	0.57	0.00	-4.94	6	0.63	0.59
Outrata1990Ex1d	K	-3.60	-3.60	11	0.30	0.00	-3.60	20	0.97	0.00	-3.52	20	2.01	0.02
Outrata1990Ex1e	K	-3.15	-3.79	137	9.47	-0.20	-3.90	25	1.83	-0.24	-2.85	13	1.54	0.09
Outrata1990Ex2a	K	0.50	0.50	1	0.03	0.00	0.50	1	0.04	0.00	0.50	2	0.05	0.00
Outrata1990Ex2b	K	0.50	0.50	5	0.13	0.00	0.50	12	0.36	0.00	0.50	7	0.22	0.00
Outrata1990Ex2c	K	1.86	1.85	8	0.31	0.00	1.85	16	0.68	-0.01	1.86	7	0.25	0.00
Outrata1990Ex2d	K	0.92	0.85	16	0.61	-0.07	0.85	21	1.05	-0.07	0.85	23	1.06	-0.07
Outrata1990Ex2e	K	0.90	0.89	44	2.06	-0.01	0.89	46	1.89	-0.01	0.90	18	1.68	0.00
Outrata1993Ex31	K	1.56	1.58	84	1.50	0.01	1.59	69	1.47	0.02	1.56	17	0.63	0.00
Outrata1993Ex32	K	3.21	3.20	21	0.45	0.00	3.20	32	0.80	0.00	3.22	161	7.16	0.00
Outrata1994Ex31	K	3.21	3.20	18	0.38	0.00	3.20	20	0.51	0.00	3.20	95	4.07	0.00
OutrataCervinka2009	O	0.00	0.00	8	0.57	0.00	0.00	7	0.64	0.00	0.00	4	0.20	0.00
PaulaviciusEtal2017b	O	-2.00	-2.00	2	0.03	0.00	-2.00	3	0.06	0.00	-2.00	2	0.04	0.00
ShimizuEtal1997a	U	0.00	16.89	7	0.07		16.89	7	0.07		16.89	14	0.28	
SinhaMaloDeb2014TP6	K	-1.21	-1.21	2	0.04	0.00	-1.21	3	0.09	0.00	-1.21	54	1.58	0.00
Yezza1996Ex31	O	1.50	1.50	8	0.17	0.00	1.50	10	0.23	0.00	1.50	10	0.27	0.00

The results are summarised in Table 3.5.

Table 3.5: Summary of results for G2

Algorithm	Number of problems			Iterations		CPU (s)	
	$\delta^* \geq 0.01$	$\delta^* \geq 0.02$	$\delta^* \geq 0.05$	Average	Median	Average	Median
BLTR	2	1	1	23.27	8	1.08	0.22
BLTR-Q	5	3	2	12.24	7	0.62	0.25
BLTR-L	13	13	9	20.89	7	1.09	0.33

The algorithms BLTR and BLTR-Q performed very well. From the summarised table, we see BLTR performed better than BLTR-Q, although it was on average a bit slower with more iterations. Again we find the algorithm BLTR-L did not perform as well as the other two in comparison. In total there are 45 problems, of which there is one problem where the solution is unknown. Therefore, out of the 44 problems, BLTR solved 42 problems to a degree of accuracy where $\delta^* < 0.01$, whilst BLTR-Q solved 39 and BLTR-L solved 31.

For the two problems BLTR did not solve to a degree of accuracy where $\delta^* < 0.01$, example `Outrata1993Ex31` was nearly solved. BLTR found an objective function value of 1.58 and BLTR-Q found a value of 1.59, whereas the best solution found in the literature is 1.56. This solution was actually found by BLTR-L. All algorithms however could not find the optimal solution for `MitsosBarton2006Ex313`. For the remaining three examples BLTR-Q could not solve to a degree of accuracy of $\delta^* \leq 0.01$, problems `AllendeStill2013` and `MuuQuy2003Ex1` were very nearly solved to the best known solution. Actually we see that BLTR found a solution slightly better than the known solution for the problem `MuuQuy2003Ex1`. BLTR-Q could not find the optimal solution for `CalamaiVicente1994c`. BLTR-L did not find the solution for a number of problems. For a large proportion, the solution was nearly found, however, there were some problems it did struggle to find a solution that the other models solved.

Algorithms BLTR and BLTR-Q found a feasible solution that is better than the provided known solution for problem `DempeLohse2011Ex31a`. This is at the points $x = (0, 0.5)$, $y = (2, 0)$, which gives an upper level objective function value of -5.75. This is opposed to the solution that was suggested in the BOLIB library [139] at the points $x = (0, 0)$, $y = (1, 1)$, giving a value of $F(x, y) = -5.50$. In addition, both these algorithms found better solutions than the known best for problems `Outrata1990Ex1e`, `Outrata1990Ex2c`, and `Outrata1990Ex2e`, and all three trust-region algorithms found a better solution than the best known for problem `Outrata1990Ex2d`. For the example where the solution is unknown, all algorithms found the same solution.

Lastly, it is interesting to note that for problem `DempeFranke2011Ex42`, out of the three

strict local minima of the problem, all three algorithms found, or nearly found, different ones. It was found that the algorithms for this problem, and a small number of others, struggled when the solution set of the lower level problem $\Psi(x)$ was nonsingular and sometimes would not return the best $y \in \Psi(x)$ that was optimal for the upper level objective function. This led to incorrect optimal solutions for some values of the penalty parameter.

Bilevel problems that have a nonconvex lower level objective function

Table 3.6 gives the results for the 39 examples from the BOLIB library in **G3**.

Table 3.6: Best solutions found from algorithms BLTR, BLTR-Q and BLTR-L for bilevel problems in G3

Example	Solution		BLTR				BLTR-Q				BLTR-L			
	Status	F^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*	F	Iter	CPU	δ^*
AnEtal2009	O	2251.6	2251.6	7	0.16	0.00	2251.6	7	0.21	0.00	2251.6	7	0.27	0.00
CalveteGale1999P1	O	-29.20	-29.20	2	0.05	0.00	-29.20	2	0.05	0.00	-29.20	2	0.07	0.00
Colson2002BIPA4	K	88.79	88.79	3	0.06	0.00	88.79	7	0.14	0.00	88.79	12	0.42	0.00
GumusFloudas2001Ex3	O	-29.20	-29.2	11	0.21	0.00	-29.20	6	0.15	0.00	-29.20	3	0.11	0.00
GumusFloudas2001Ex5	O	0.19	0.19	3	0.08	0.00	0.19	3	0.11	0.00	0.19	3	0.13	0.00
KleniatiAdjiman2014Ex3	O	-1.00	-0.98	8	0.22	0.02	-1.00	7	0.23	0.00	-1.00	6	0.14	0.00
KleniatiAdjiman2014Ex4	K	-10.00	-10.00	4	0.23	0.00	-10.00	4	0.45	0.00	-10.00	2	0.08	0.00
LuDebSinha2016a	K	1.14	0.98	6	0.19	-0.14	1.14	6	0.50	0.00	1.14	12	0.66	0.00
LuDebSinha2016b	K	0.00	0.05	3	0.07	0.05	0.05	6	0.30	0.05	0.11	6	0.49	0.11
Mirrlees1999	O	1.00	0.08	5	0.10	0.92	0.05	6	0.13	0.95	0.80	9	0.19	0.20
MitsosBarton2006Ex39	O	-1.00	-1.00	2	0.04	0.00	-1.00	2	0.04	0.00	-1.00	2	0.04	0.00
MitsosBarton2006Ex310	O	0.50	0.50	2	0.06	0.00	0.50	4	0.13	0.00	0.50	1	0.03	0.00
MitsosBarton2006Ex311	O	-0.80	-0.80	2	0.03	0.00	-0.80	2	0.08	0.00	-0.80	1	0.04	0.00
MitsosBarton2006Ex312	O	0.00	0.00	4	0.08	0.00	0.00	10	0.25	0.00	0.00	3	0.10	0.00
MitsosBarton2006Ex314	O	0.25	0.25	3	0.07	0.00	0.25	4	0.10	0.00	0.25	4	0.19	0.00
MitsosBarton2006Ex315	O	0.00	0.00	2	0.03	0.00	0.00	2	0.05	0.00	0.00	2	0.05	0.00
MitsosBarton2006Ex316	O	-2.00	-2.00	2	0.04	0.00	-2.00	2	0.04	0.00	-2.00	2	0.06	0.00
MitsosBarton2006Ex318	O	-1.00	-1.00	1	0.03	0.00	-1.00	3	0.09	0.00	-1.00	3	0.10	0.00
MitsosBarton2006Ex319	O	-0.26	-0.26	7	0.18	0.00	-0.26	20	0.74	0.00	0.00	3	0.14	0.26
MitsosBarton2006Ex320	O	0.31	0.09	3	0.06	0.22	0.50	11	1.10	0.19	0.31	13	0.39	0.00
MitsosBarton2006Ex321	O	0.21	0.21	2	0.05	0.00	0.21	8	0.15	0.00	0.21	8	0.30	0.00
MitsosBarton2006Ex322	O	0.21	0.21	4	0.07	0.00	0.21	8	0.15	0.00	0.21	9	0.46	0.00
MitsosBarton2006Ex327	K	2.00	1.00	2	0.32	-0.50	1.00	2	0.67	-0.50	1.08	7	3.07	-0.46
MitsosBarton2006Ex328	K	-10.00	-10.00	95	22.05	0.00	-10.00	22	2.50	0.00	-10.00	5	0.60	0.00
MorganPatrone2006b	O	-1.25	-1.25	9	0.22	0.00	0.50	3	0.08	1.40	0.50	3	0.14	1.40
MorganPatrone2006c	O	-1.00	-1.00	1	0.04	0.00	-1.00	1	0.04	0.00	-1.00	1	0.06	0.00
NieWangYe2017Ex52	O	-1.71	-1.71	2	0.06	0.00	-1.71	1	0.04	0.00	-1.71	7	0.40	0.00
NieWangYe2017Ex57	K	-2.00	-2.00	1	0.05	0.00	-2.00	1	0.05	0.00	-2.00	5	0.26	0.00
NieWangYe2017Ex58	K	-3.49	-3.49	12	0.38	0.00	-3.49	33	5.54	0.00	-3.49	16	0.87	0.00
NieWangYe2017Ex61	K	-1.02	-2.00	9	0.35	-0.96	-1.99	27	0.84	-0.95	-1.02	2	0.16	0.00
Outrata1990Ex1a	K	-8.92	-8.91	27	2.11	0.00	-8.91	12	1.28	0.00	-8.91	34	3.11	0.00
Outrata1990Ex1b	K	-7.56	-7.57	8	0.43	0.00	-7.57	7	0.46	0.00	-7.56	18	2.15	0.00
PaulaviciusEtal2017a	O	0.25	0.00	1	0.02	0.25	0.25	1	0.03	0.00	0.25	1	0.03	0.00
SinhaMaloDeb2014TP7	K	-1.96	-1.96	8	0.28	0.00	-1.96	157	21.39	0.00	-1.96	9	0.62	0.00
SinhaMaloDeb2014TP9	K	0.00	0.00	1	0.24	0.00	0.00	3	0.67	0.00	0.01	15	8.27	0.01
SinhaMaloDeb2014TP10	K	0.00	0.00	1	0.20	0.00	0.00	2	0.63	0.00	0.02	12	6.09	0.02
Vogel2002	O	1.00	0.00	1	0.01	1.00	0.00	1	0.01	1.00	0.00	1	0.01	1.00
YeZhu2010Ex42	O	1.00	1.00	4	0.07	0.00	1.00	4	0.07	0.00	1.00	195	9.10	0.00
YeZhu2010Ex43	O	1.25	1.00	1	0.02	0.20	1.00	1	0.01	0.20	1.25	5	0.12	0.00

The results are summarised in Table 3.7.

Table 3.7: Summary of results for G3

Algorithm	Number of problems			Iterations		CPU (s)	
	$\delta^* \geq 0.01$	$\delta^* \geq 0.02$	$\delta^* \geq 0.05$	Average	Median	Average	Median
BLTR	7	6	5	6.90	3	0.74	0.07
BLTR-Q	6	6	5	26.73	5	1.55	0.21
BLTR-L	7	5	5	11.51	5	1.01	0.19

All of the trust-region algorithms performed similarly for this group of problems. Surprisingly this includes BLTR-L, which, in comparison to the other two algorithms, had not performed so well for problems in groups G1 and G2. In total there are 39 problems, where all problems have a known or optimal solution provided. Although in general the algorithms did not solve as many as the previous groups of bilevel problems, out of 39 problems all methods solved at least 32 problems to a good degree of accuracy, which is about 82% of the problems. This is very good considering the difficulty of the problems, and numerically shows basis that this approach could be extended to bilevel problems formed of nonlinear, nonconvex functions. All algorithms again solved the problems relatively quickly and with few iterations, with BLTR performing slightly quicker with fewer iterations.

We will now look at the problems the algorithms did not find the known or optimal solution provided from the literature. For problem `Mirrlees1999`, all algorithms found the solution point $x = 1.99$, $y = 0.895$. This was stated in the paper by Mirrlees [88] as a solution to the KKT reformulation that is not a solution to the bilevel problem. The problem with this solution is that $y = 0.895$ is a local minimum of the lower level problem for $x = 1.99$, and is not the global minimum. Moreover, for examples `MitsosBarton2006Ex320`, `Vogel2002` and `YeZhu2010Ex43` some of the algorithms also found local optimal solutions of the lower level problem for a given x, instead of finding the global solution. This highlights some of the difficulties when the lower level function is nonconvex. In practise however, using a global solver at each iteration could greatly increase the CPU time.

Examining the remaining problems that the algorithms could not solve, we found algorithms BLTR-Q and BLTR-L found the optimal solution for the problem `PaulaviciusEtal2017a`, but it is unsure why BLTR did not. Interestingly, for the problem `MorganPatrone2006b` all algorithms found the correct upper level variables x, however, the solution set mapping at the optimal solution x is not a singleton. Only algorithm BLTR found the correct y that minimises the upper level function $F(x, y)$.

There are a number of problems where some of the algorithms found better solutions than the

known best, namely `MitsosBarton2006Ex327`, `NieWangYe2017Ex61` and `LuDebSinha2016a`, although some care may need to be taken that the lower level solution for the given x is not a local solution, as has been the case for previous examples.

3.5.3 Analysis of Results

Overall, the trust-region algorithm worked very well on all problems tested. Out of the 118 problems that have solutions in the literature for which we can compare, the trust-region algorithm that used the penalty model that only approximates the optimal value function (BLTR) solved 111 problems to an accuracy where $\delta^* < 0.02$, and 108 where $\delta^* < 0.01$. The trust-region algorithm that used a penalty model that locally approximated all functions in the penalty model with F and f as quadratic models (BLTR-Q) solved 109 problems to an accuracy where $\delta^* < 0.02$ and 106 where $\delta^* < 0.01$. Finally, the trust-region algorithm that used a linear model for F and f in the penalty model (BLTR-L), solved 95 problems to an accuracy where $\delta^* < 0.02$ and 93 where $\delta^* < 0.01$. The CCP method also performed very well on the bilevel problems where the lower level problem is fully convex.

In the following section, we compare the algorithms with a number of other algorithms in the literature. We then compare the performance of the trust-region models with each other, before investigating the sensitivity of the penalty parameter and the starting point. A discussion on the parameters chosen for the stopping criteria is then given.

Comparison with other algorithms

We compare the performance of the algorithms with three other methods in the literature, which we implemented in MATLAB 2019Ra for fair comparison. The first two use a semi-smooth Netwon type method, one using the KKT reformulation and one using the value function reformulation of the bilevel problem. We denote these as SNKKT and SNLLVF respectively. The SNKKT method is presented in a paper by Zhou and Zemkoho [141], and the SNLLVF is presented in a paper by Fischer et al. [48]. We implemented the algorithms in MATLAB using the code provided in the BiOpt Toolbox [140]. We used the default parameters provided in the code, except for the penalty parameters where we used the same parameters that we tested for the trust-region and CCP methods. The third method we look at is a very simple coded algorithm, which uses the MATLAB built-in function *fmincon* to solve the KKT reformulation of the bilevel problem. We refer to this algorithm as the KKT algorithm. For all algorithms, the same starting points as before were used.

We use the same method implemented in Section 3.5.2 to assess the quality of a solution. That is, we compare the solutions found for a test problem against a known solution,

measured by the value δ given by equation (3.29), and pick the smallest delta found from all penalty parameter and starting points, denoted by δ^*.

Table 3.8 presents the percentage of problems solved for each algorithm. In this table we only include the 118 problems from the BOLIB library where we have known solutions, and can therefore calculate a δ^*, and we refer to a solution as solved if $\delta^* < 0.01$.

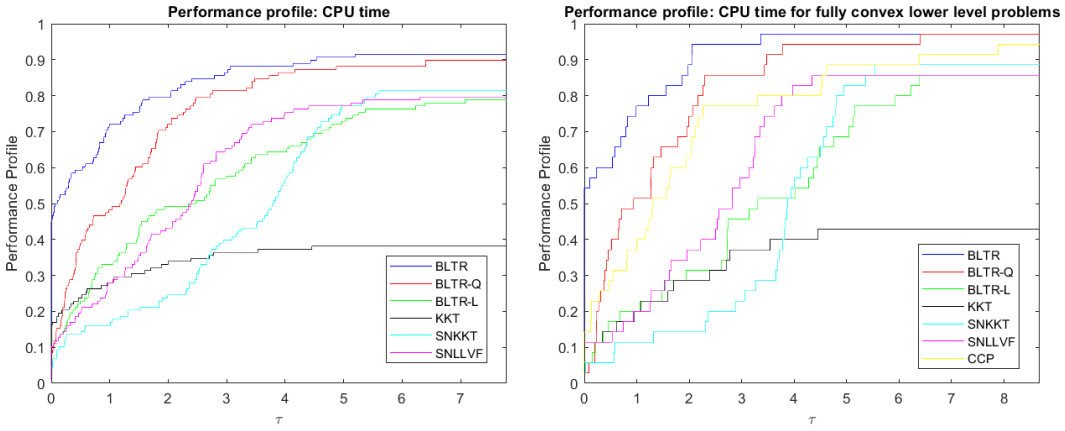
Table 3.8: Summary of percentage of problems solved

Algorithm	Percentage of problems solved to an accuracy of $\delta^* < 0.01$			
	Problems from **G1**	Problems from **G2**	Problems from **G3**	All problems
BLTR	97.14 %	95.45%	82.05%	91.53%
BLTR-Q	97.14 %	88.64%	84.62%	89.83%
BLTR-L	85.71%	70.45%	82.05%	78.81%
SNKKT	88.57%	86.36%	69.23%	81.36%
SNLLVF	85.71%	81.82%	71.79%	79.66%
KKT	42.86%	36.36%	35.90%	38.14%
CCP	94.29%	-	-	-

From the table we can see that BLTR and BLTR-Q solve the highest number of problems in each group, suggesting the trust-region algorithm works extremely well on the problems given. BLTR-L does not solve as many prolems as the semi-smooth Newton type methods for groups G1 and G2, but seems to outperform them for the nonconvex problems in G3. The KKT algorithm does not perform well on any problems, which is perhaps to be expected as the MATLAB solver *fmincon* has not been designed specifically for the purpose of solving MPCCs. The CCP method appears to work well on the fully convex problems.

We use the performance profile developed by Dolan and More [41] to compare the algorithms further. These profiles were developed to compare the performance of a set of solvers $\mathcal{S}$ on a test set of problems $\mathcal{P}$. They compare the performance by a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ with the best performance by any solver on that problem, thus scaling the cost of solving the problem according to the best solver. We investigate the CPU time of the algorithms as the performance measure. Figure 3.1a shows the $\log_2$ scaled performance profile comparing the CPU time taken for each algorithm to solve the 118 problems where a solution is provided. Figure 3.1b shows the $\log_2$ scaled performance profile comparing the CPU time for only problems where the lower level objective function is fully convex, to show how the CCP method compares. Similar to before, we count a problem as being solved if $\delta^* < 0.01$.

Looking at Figure 3.1a, we can see that BLTR dominates all other solvers, as its profile lies above all others for all performance ratios. It is the fastest solver on approximately 50% of all



(a) All problems **(b)** Problems from G1

Figure 3.1: Performance profile of CPU time for all algorithms

problems, and solves the most problems to optimality. BLTR-Q performs slower than BLTR, but still outperforms the other algorithms, and solves almost as many problems as BLTR. BLTR-L does not perform as well as the other trust-region algorithms, but it does perform similarly to both of the semi-smooth Newton algorithms with both CPU time and number of problems solved. This is still a good performance for solving bilevel problems, solving just under 80% of all problems. Looking at Figure 3.1b which shows only problems from Group **G1**, algorithm BLTR still performs the best, solving just over 55% of the problems the fastest. The CCP algorithm performs competitively, performing slower then the trust-region algorithms BLTR and BLTR-Q, but faster than the other algorithms including BLTR-L.

Comparison of trust-region models

For the trust-region algorithm, we have chosen to work with three different models for the penalty objective function, to assess whether any performs better in practise. As we have seen in the previous section, the model that only locally approximates the optimal value function seems to perform the best, both in terms of speed and the number of problems solved, although BLTR-Q solves slightly more problems from **G3**. In order to compare the algorithms further, we plot a $\log_2$ scaled performance profile that uses the number of iterations as the performance measure, given in Figure 3.2. Again, we only include the 118 BOLIB problems where a solution has been provided, and we count a problem as being solved if $\delta^* < 0.01$.

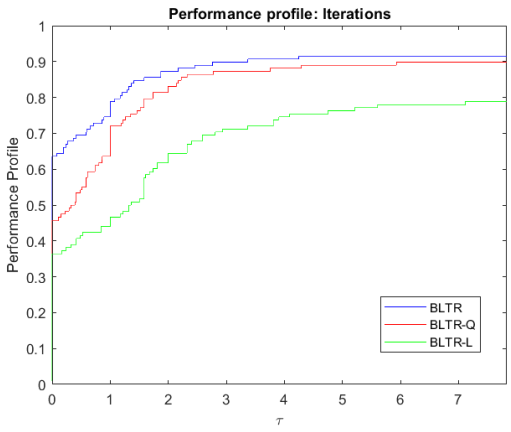
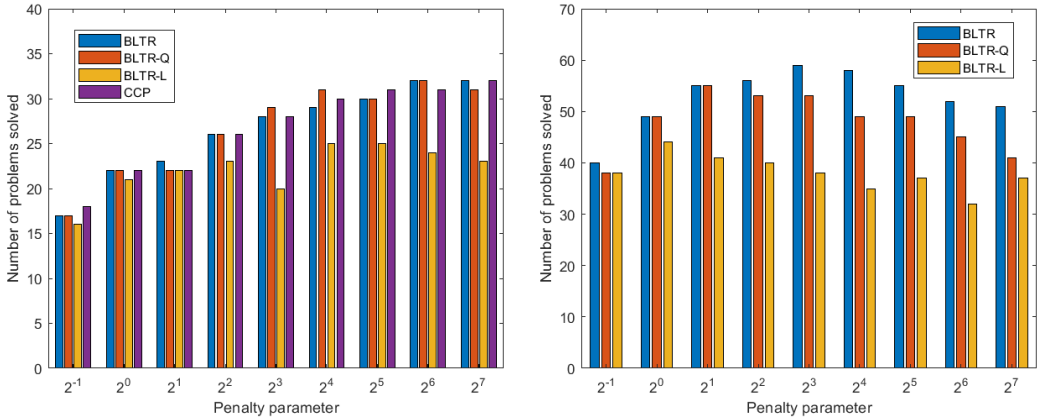


Figure 3.2: Performance profile of iterations for the trust-region algorithm

From the figure, we see that BLTR again dominates BLTR-Q for the number of iterations, and BLTR-Q dominates BLTR-L by a larger margin. BLTR solves 75% of the problems with the same or less iterations. This can explain why the algorithm performs on average fastest. That being said, we may need to take into consideration that the problems in the test set are relatively small. Because the model used in BLTR is a more accurate representation of the penalty function we wish to minimise, we could expect the number of iterations needed to solve the problem is smaller. However, the trust-region subproblem used in BLTR becomes more difficult to solve when the upper and lower level objective functions become more complicated nonconvex and nonlinear functions. On the other hand, the subproblem used in BLTR-Q will always have a quadratic objective function, and BLTR-L will always have a linear objective funcion. Consequently, we may find that as the problem becomes larger and more complicated, the models used in BLTR-Q and BLTR-L may become the better option.

Sensitivity of the penalty parameter

The quality of the solution found by the trust-region algorithm and the CCP method was sensitive to the choice of penalty parameter. Figure 3.3 shows the number of problems solved for each penalty parameter, where we separate the problems into those with fully convex lower level objectives functions, given by group **G1**, and all other problems, given by group **G2** and group **G3**. We say that the algorithm solved a problem for a given parameter if the algorithm solved the problem for at least one starting point. Similar to before, we refer to a problem as solved if $\delta < 0.01$, and we only include the 118 problems where we can compare the solutions.



(a) Problems from G1 **(b)** All other problems

Figure 3.3: Number of problems solved for each penalty parameter

From Figure 3.3a, we can see for problems where the lower level objective function is fully convex, the number of problems solved increases as the penalty parameter is increased for both the trust-region algorithm with BLTR and BLTR-Q, and the CCP method. In fact, a high proportion of problems were solved for $\gamma = 2^6$ and $\gamma = 2^7$, where all of these algorithms solved 31 or 32 out of the 35 problems. From this we may suggest a higher penalty parameter would be a good choice to solve these problems with these algorthims. For BLTR-L, the relationship is not so clear. The algorithm does not perform as well as the others, solving less problems for each penalty parameter, and the highest proportion of problems were solved at $\gamma = 2^4$.

For problems where the lower level objective function was convex in y or nonconvex, we can see from Figure 3.3b that the trust-region algorithm with BLTR and BLTR-Q was more robust for the mid range of penalty parameters, with less examples being solved for low or high penalty parameters. For BLTR-L we find again the relationship is less clear. It appears to solve more problems with a lower penalty parameter, with $\gamma = 2^0$ being the parameter that solves the highest proportion of problems. We find that 94.6% of the problems from groups G1 and G2 that were solved by algorithm BLTR, and 94.4% of the problems from G1 and G2 that were solved by algorithm BLTR-Q, were solved using the set of penalty parameters $\Gamma = \{2^1, 2^2, 2^3, 2^4\}$. In the same set, we find that 79.4% of the problems from G1 and G2 that were solved by algorithm BLTR-L. This is still a relatively large proportion of problems solved within this range. A suggestion on a starting value for the penalty parameter would perhaps be to start initially in this range, however, it is worth noting that the optimal choice for the penalty parameter will depend on the example.

Sensitivity of the starting point

As bilevel problems are inherently nonlinear, the choice of starting point can be crucial to finding a good solution. As discussed in Section 3.5.1, the choice of the two starting points were chosen problem specific in order to find a feasible and sensible starting point. We investigate whether the solutions found by the algorithms were sensitive to the choice of the two starting points used. The figures in Table 3.9 show what proportion out of the total number of problems that the algorithms solved were solved by both starting points. As before, we say that an algorithm has solved a problem when $\delta < 0.01$. We say that an algorithm finds the solution of the problem for a given starting point if for that starting point, there exists one penalty parameter that solves the problem.

Table 3.9: The proportion of solved problems where the solution was found by both starting points

Algorithm	Problems from **G1**	Problems from **G2**	Problems from **G3**	All problems
BLTR	82.35 %	85.71%	78.13%	82.41%
BLTR-Q	85.29 %	89.74%	69.70%	82.08%
BLTR-L	70.00%	70.97%	59.38%	66.67%
CCP	81.82%	-	-	-

We can see a high proportion of the problems that were solved by the algorithms were solved by both starting points, with a notable drop in the proportion of solved problems found by both starting points for BLTR-L. That being said, we still found around 60-70% of solved problems for BLTR-L were found by both starting points. We also found that for problems in G3, where the objective functions become more nonconvex, a smaller proportion of solved problems were found by both starting points for all algorithms. For future work, this analysis would benefit from testing the algorithm with a large number of randomly generated starting points, to enable further assessment on the impact on the choice of starting point.

The stopping criteria

After noting that some runs reached the maximum number of iterations for problems when the lower level objective function was fully convex, in particular for the CCP algorithm, some investigation was taken into the number of runs that stopped for each stopping criteria. Table 3.10 presents these results as a percentage of the total number of test runs. This includes the runs for all 124 problems, with 9 different penalty parameters for two different starting points. Thus, for the trust-region algorithms a total of 2232 runs were made, and for the CCP algorithm a total of 720 runs were made.

Table 3.10: The percentage of test runs that stopped at each stopping criteria

Stopping criteria	BLTR	BLTR-Q	BLTR-L	CCP
SC1	25.09 %	14.34%	9.18%	-
SC2a	59.32%	72.67%	79.21%	1.94%
SC2b	8.87%	5.51%	5.29%	89.86%
SC3	0.85%	1.08%	0.67%	8.19%
SC4	1.52%	1.30%	0.18%	-
SC5	4.35%	5.11%	5.47%	-

For all the trust-region algorithms, around 93% of runs were successful at stopping for **SC1** – **SC2**. Only a small percentage reached the maximum number of iterations, which was around 20 runs for each algorithm. If the algorithm did not converge, most of these runs stopped when the trust-region radius became too small. For the CCP method, around 92% of runs were successful at stopping for **SC2**. However around 8% of runs stopped because the algorithm reached the maximum iterations, suggesting the algorithm may benefit from increasing the maximum number of iterations. For future work, experiments to further analyse the solutions sensitivity to the choice of these parameters would be needed to help practitioners understand what values are best to choose, and to ensure the stopping criteria does not stop the algorithms prematurely.

3.6 Summary

In this section we proposed a novel trust-region algorithm for solving bilevel problems with strong numerical results. The algorithm looks to solve the exact penalty problem that can be built from the optimal value function using the partial calmness condition. The literature on solving this exact penalty problem to find local solutions of the bilevel problem is scarce, and to our knowledge, this is the first trust-region approach to solving a general bilevel problem using this exact penalty problem.

We presented extensive numerical experiments on 124 bilevel problems to assess the performance of the algorithm, which include many bilevel problems with nonconvex lower level problems. The algorithm was tested using three variants of the model for the trust-region subproblem. The first variant approximates only the optimal value function in the penalty objective function, whilst the second and third approximates the whole function, using a quadratic model or a linear model for the smooth parts respectively. Overall, the trust-region algorithm works extremely well on the bilevel problems tested, solving a high proportion of nonconvex problems. In comparison with the other algorithms tested, the trust-region algorithm with the first two aforementioned models performed the best. Considering the

difficulty in solving nonlinear bilevel problems, the results demonstrate the promising performance of the bilevel trust-region algorithm. We found the trust-region model that only approximates the optimal value function performed the best, solving 92% of all of the bilevel problems, and the trust-region model which approximated the smooth parts of the penalty function with quadratic models came close behind it, solving 90% of problems. The third model that used linear models did not perform as well in comparison, solving fewer problems with a higher CPU time. That being said, it still performed well when considering the difficulty of the problems, solving a similar number of problems as other bilevel algorithms from the literature that were tested.

There were some problems that the trust-region algorithm had difficulties in solving. These were problems where the solution set of the lower level problem was nonsingular. There were a number of times the algorithm did not find the optimistic solution of the lower level variable y for a given x. This may be a problem when we approximate the Clarke generalised gradient with only one subgradient. This is an area of research that needs to be investigated. When this happens we may need to enrich the approximation with new subgradients, as we see in bundle-type algorithms.

The trust-region algorithm had further problems when MATLAB found only local solutions of the lower level problem, instead of global solutions. Using a global solver at each iteration in order to find a subgradient could however make the algorithm very slow. This is a concern in the efficiency of the algorithm. The method of solving the lower level function at each iteration could be problematic and slow if the lower level problem is a particularly difficult and complicated problem. In this situation, when the lower level problem is itself a difficult problem to solve, an alternative algorithm may be preferred. A further cause for concern on the speed of finding a solution is the choice of penalty parameter. We found, depending on the problem, the choice of penalty parameter could have a substantial impact on the quality of the solution. The algorithm currently requires testing different penalty parameters and choosing the best solution, and this can make finding a solution slow. Future research could look into how the penalty parameter could be approximated for a given problem such that the solution found is a local solution of the bilevel problem.

We presented convergence of the trust-region algorithm under strong assumptions, with Theorem 3.3.4 only showing convergence using the first order model of the whole penalty function $\bar{P}_k(x, y)$. However, due to the promising numerical results, further research on the convergence theory of this trust-region algorithm should be investigated. This should consider both the convergence of the algorithm under the conditions in Theorem 3.3.4 with the other models for the trust-region problem, and the convergence of the algorithm under

weaker assumptions. The current research on trust-region methods for nonsmooth functions often show that if the function is Clarke-regular and the full generalised gradient is used in the first order model, the algorithm converges to a stationary point, for instance this is shown in [40]. Under the assumptions of Theorem 3.2.12.1, the optimal value function is shown to be Clarke-regular. Therefore in theory under these assumptions, if the full Clarke generalised gradient set of the optimal value function is used in the first order model $\bar{P}_k(x, y)$, the algorithm should converge to a stationary point. Further research could therefore consider whether the convergence of the algorithm still holds when only an approximated generalised gradient set is used.

In this chapter we also introduced the CCP heuristic to solve bilevel problems with fully convex lower level problems. This is a heuristic used to solve DC problems. The algorithm solves the same exact penalty problem as the proposed trust-region algorithm, where it is shown the problem becomes a DC problem under the assumptions made. We tested the algorithm on 40 bilevel problems with a lower level objective function that was fully convex. The algorithm performed very well, solving 94.29% of the problems. It performed slightly better than the trust-region method that used a first order model of the whole penalty function, however, it was a bit slower at converging to an optimal solution than the other two models used in the trust-region method, both in terms of CPU time and number of iterations, and solved slightly fewer problems. The performance endorses the theory that under the assumptions made, a single level DC problem can be solved which, provided that a good choice of penalty parameter is made, can find a local solution of the bilevel problem. With this knowledge, there are many known algorithms in the literature that have been designed to solve DC problems that could be used to solve them [56]. For example, this approach has been taken in the recent paper by Ye et al. [130], where they apply two different DC algorithms to solve the LLVF reformulation of the bilevel problem.

Finally, we note that the size of the bilevel test problems were all relatively small. Future work could look into how the algorithms perform on larger problems. Additional numerical experiments to test the robustness of the algorithm against a large set of starting points should also be done, alongside further analysis on the choice of stopping criteria parameters. However, with all things considered, the method of solving the exact penalty problem built from the optimal value function to find local solutions of the bilevel problem shows promise. Currently, the research on bilevel optimisation that makes use of this exact penalty reformulation is limited. Although the assumption of partial calmness is itself a strong condition, the encouraging results suggest that this approach should be further explored in the future as an approach to solving bilevel problems.

Chapter 4

An application of the trust-region method to the London congestion pricing problem

4.1 Introduction

Bilevel programming can be applied to a vast range of real-life problems. One considerable field of application is within transportation. We look at a problem known as the road pricing, or toll setting, transportation problem. This is a common method implemented by governments to charge for the use of roads, with the aim of helping to alleviate negative externalities such as congestion. The road pricing problem can be modelled as a bilevel program as, due to its hierarchical structure, it naturally forms as one. The upper level problem models the governing body that is looking to set a tolling scheme on a road, or subset of roads, whilst the lower level problem models users of the road network looking to minimise cost of travel to themselves, such as travel time and toll costs.

We investigate the London congestion pricing problem, a real-world example of road pricing. Currently, road users are charged a fixed toll to use the road network within a cordon in the center of London. Introduced first in 2003; Transport for London (TfL), the local governing body responsible for the transport system in London, are now looking to update their pricing system as advancements in technology can allow for a more sophisticated tolling scheme. In this chapter we look at how this problem can be formulated as a bilevel problem, forming a model that takes into consideration the objectives currently being pursued by TfL. Additionally, we show how under simplifications of the road pricing bilevel model we

can apply the trust-region method presented in Chapter 3 to solve it, providing numerical results to test the performance of the approach.

This chapter will first introduce the general road pricing problem followed by a literature review on the models formulated and methods developed to solve it, with a particular focus on how the problem has been formed as a static deterministic bilevel program. We then introduce the London congestion pricing problem, outlining how the pricing is currently implemented in the London congestion zone, how TfL currently model the transportation network, and the objectives they are currently looking to achieve. Following this, the formulation of the bilevel model for the London congestion pricing problem is presented. We then simplify a number of the complexities of the model, showing how properties of this simplified problem allow us to apply the bilevel trust-region algorithm to solve it. Finally, we give detailed numerical results and analysis for three examples of varying sizes to test the quality and robustness of solutions.

4.2 The road pricing problem

Congestion in a road network can cause large costs to an economy including problems such as air pollution, increased accidents and loss of work time. With an increasing population and excessive demand for road space, congestion is becoming a growing problem in urban areas, and there is need for effective measures to help alleviate it. Due to the limited ability to expand networks and increase capacity in cities, a common solution is the use of road pricing. This method introduces a charge to discourage users from using congested routes and encouraging them to take alternative routes or modes of transport. A further benefit is that tolls can raise a large amount of revenue which can be used for maintenance of the road network, and to improve transport systems such as public transport [82].

Many cities around the world have successfully introduced road pricing to tackle congestion and pollution, with a majority of these cases adopting fixed tolling schemes. Cases include cordon schemes, for example in London and Stockholm [43]; bridges and tunnels, for example the Dartford crossing in the UK; and lengths of roads, for example motorways tolls in France, Italy and Spain [100]. Although there are various forms of tolling schemes, the most frequently used tolling scheme in practice is fixed cordon pricing, due to the relative ease in implementing it. It does however have its weaknesses. Once implemented, cordon schemes are inflexible in adapting to changes, for example zones cannot be easily relocated, and congestion can occur around the cordon zone. Furthermore, they fail to differentiate a charge for users taking short and long journeys; once you have paid the cordon toll you can use the roads as much as you like [82].

The rapid improvement in technology means there is now potential to track what route people take and at what time, meaning it is now easier to implement varying toll policies. Tolls could depend on the time of the day, how long users have been using the congested roads for, the distance they have travelled, or the level of congestion. This should in theory give rise to a more effective scheme. Some real-world examples of variable pricing schemes include the lorry road user charges in Switzerland, Germany and Austria [83], which depend on distance travelled and the weight of the vehicle, and the electronic pricing scheme in Singapore [6], which depends on the time of day, the location and the type of vehicle used.

4.2.1 Literature review of road pricing

We provide an overview of the literature on the road pricing problem, looking at how the problem has been formulated and the solution approaches taken to solve the problem.

The first suggestion of road pricing as a mechanism to alleviate congestion was by Pigou [97] in the 1920's who, from a welfare economic view, suggested the use of optimal congestion charges to pay for externalities caused such as pollution and work time lost. Known as the first-best pricing problem, or marginal-cost pricing, it is assumed we can charge a toll on every link in the network. The toll is equal to the difference between the marginal-social cost and the marginal-private cost, to maximise social surplus. The idea is to add a price mechanism in the same way we apply it to other markets, charging higher prices where demand is higher.

Although in theory this method ensures each road is used to maximum efficiency, in reality, it is not very practical to charge different tolls to every link in a network. The second-best pricing problem therefore looks at only tolling a subset of roads. It has far more practical relevance as this problem can be implemented far more easily. Research into the second-best road pricing problem looks at answering three key questions: which pricing scheme best achieves the aims of the governing body, what price to set the toll levels at, and where the tolls should be set.

To identify what pricing scheme is the most effective in alleviating congestion, Smith et al. [110] and May and Milne [82] analyse four pricing systems for the second best road pricing problem: cordon pricing, time-based pricing, delay-based pricing and distance pricing. Delay-based pricing, otherwise known as congestion pricing, refers to a charge that depends on the level of congestion, where we define congestion to be the lost travel time of users on a network from queueing. They test each tolling scheme on a road network of Cambridge, applying various different toll prices for each scheme and evaluating the user equilibrium using a traffic-assignment model. Smith et al. [110] conclude that delay-based charging

has the best effects on congestion in terms of increasing network speeds, with fixed cordon schemes having the worst. May and Milne [82] extend the work to look at different objective functions for the governing body. They find that although congestion pricing increases network speeds the best, it can result in users taking longer minor roads. Their analysis showed that the best pricing system depended on what objective the governing body wished to achieve. Notably however, they found cordon pricing to be the least effective overall, despite it being the most commonly implemented road pricing scheme in practise.

To find the optimal price of tolls, we can formulate a model of the second best road pricing problem in two ways: the static model and the dynamic model. Both models formulate the problem as a bilevel program. The static model considers the long term equilibrium flow of the network, and charges one price over a given time period. On the other hand, the dynamic model charges a varying price over time depending on changing circumstances, such as changes in demand. Recently, the dynamic model has gained traction, with a growing number of research looking towards building this type of model. Both models however have been studied extensively, and the number and breath of research is vast. As we build a static bilevel model, we focus on the development of this model, with detail on the key models that influenced our research.

The static bilevel road pricing problem is formulated as follows. The upper level problem models the governing body setting tolls to achieve a given objective. In general it is assumed that the toll locations are set, and we find model formulations often consider the case of fixed tolls on a fixed subset of roads. The lower level problem is formulated as a traffic assignment problem. This problem has been well studied, and looks at how traffic in a road network is assigned to paths in equilibrium. The model is constructed to satisfy the Wardrop equilibrium conditions, introduced by Wardrop in 1952 [120]. These conditions state that all users of the network will take the path of cheapest cost to themselves, known as the shortest path, from origin to destination, and when in equilibrium no user will gain from changing paths. Shown first by Beckmann et al. in 1956 [10], these conditions can be formed into a nonlinear programming problem. We can assume a varying level of complexity in the traffic assignment problem. Variations can include: deterministic or stochastic user behaviour, linear or nonlinear travel cost functions, fixed or variable demand, homogeneous or multicommodity users, queueing considerations and multi-modal systems. For a thorough account of the models for traffic assignment problems and methods to solve them, we refer the reader to the book by Patriksson [95].

Focusing on models that use the deterministic traffic assignment problem, a large amount of research has investigated developing bilevel models for the road pricing problem with fixed

tolls. Labbé et al. [65] and Brotcorne et al. [14] present a road pricing bilevel model that assumes linear travel costs, along with fixed demand and no congestion. They consider a multicommodity transportation problem with an upper level objective function to maximise profits. The value of time for all users of the population is uniform. Due to the fixed travel times, fixed tolls and fixed demands, the problem has linear constraints with bilinear objective functions at both levels. In the former paper they reformulate the problem into a mixed integer linear program using optimality conditions of the lower level. They then present heuristics to solve specific cases of the problem, that use shortest path and transshipment algorithms. In the latter paper, they reformulate the problem as an exact penalty problem, resulting in a bilinear program, which they solve using a primal-dual heuristic.

Dempe and Zemkoho [36] and Dempe and Franke [30] also both consider a bilevel road pricing problem where the cost function is linear and the model is deterministic. They examine the optimality conditions of the problem, showing that for the given problem, the partial calmness condition holds. Dempe and Zemkoho [36] give further conditions for partial calmness for when the cost function takes a more general form. Dempe and Franke [30] solve the problem by utilising the following property of the problem. They show that the optimal value function, although not differentiable, is concave. Using this property they approximate the optimal value function using supergradients of the function, and by inputting this approximation into the optimal value reformulation of the bilevel problem, it gives an outer approximation of the feasible set. The algorithm sequentially improves the approximation of the feasible set until a solution of the relaxed problem is feasible for the bilevel problem.

As opposed to a linear time function, Yang and Lam [125] develop a model that uses the convex bureau of public roads (BPR) function, one of the most commonly used travel time functions in the literature. They formulate a bilevel model for the deterministic, fixed demand case with fixed link tolls, and introduce queueing and congestion into their model by adding an explicit capacity constraint for each link. It can be shown that the Lagrange multipliers associated with this constraint are equal to the queueing delays. For the upper objective function they consider the following: maximising revenue, minimising total network travel costs and maximising a ratio of the first two. They also consider adding two constraints to the upper level problem. The first constraint adds a maximum queue length allowed, and the second imposes a budget constraint. Two sensitivity analysis based heuristics are suggested to solve the model, of which one of them was applied for when they do not add the extra constraints into the upper level problem. This heuristic was proposed by Suwansirikul et al. [114], known as the equilibrium decomposition optimization algorithm.

Yang and Bell [126] extend this queueing model by considering elastic demand, but use a tolling scheme that charges a fixed toll on all links. They charge a toll that is equal to the queueing delay, and show how the resulting equilibrium with tolls becomes unqueued. Next, they suggest that by limiting the capacity constraint of the links to satisfy an environmental capacity below the actual capacity of the link, traffic demand can be reduced to a desirable level. They impose this environmental capacity on the upper level problem as opposed to the lower level problem, meaning that the lower level problem becomes a standard traffic assignment problem with no queueing. The bilevel model is then solved using a similar sensitivity analysis heuristic as the previous paper. We note that if they imposed the capacity constraint on the lower level problem, this could give rise to a different solution.

As opposed to having fixed tolls, some models consider road pricing problems with variable tolling schemes, where tolls are formed as functions which depend on factors such as the distance travelled or the amount of time spent on a route. For example, Namdeo and Mitchell [91] study a distance based toll that is linearly proportional to the distance travelled, imposing a fixed charge per km. To investigate the problem, they do an empirical study of the road network in Leeds, testing different values for the charge per km. Assuming that the toll is directly proportional to the distance can make solving the model easier, as tolls are link additive. This property allows us to apply the same methods that solve road pricing problems with fixed link tolls fairly easily.

It is however argued that road pricing is often nonlinear, for example this is so in the simple case where there is a minimum or maximum limit you can pay. If this is the case, tolls are no longer link additive. Law and Yin [68] study a nonlinear distance toll function. They simplify the toll function into a piecewise linear function with a maximum of two linear pieces, and form a bilevel model with the deterministic user equilibrium. As they only consider a piecewise linear function with two parameters, they find they are able to apply a coordinate search algorithm to solve the problem.

A number of other studies have also looked at approximating a nonlinear distance toll function with a piecewise linear function. Meng et al. [85] use this approach, but consider a piecewise linear function with more than two intervals. They use the logit-based stochastic user equilibrium as opposed to the deterministic model. As the tolls are not link additive, they transform the model using dummy links to a network where tolls are link additive and solve using a heuristic genetic algorithm. Liu et al. [74] extend this model. They argue that a pricing scheme based only on distance can cause users to take the shortest route, even if its very congested. They therefore add a time toll to the distance toll to deter users picking congested routes. The proposed toll function is a weighted sum of a nonlinear distance

toll with a linearly proportional time toll. They solve this problem by first making tolls link additive using the same method in the previous paper, before transforming it into a semi-infinite programming model by utilising the logit-based stochastic user equilibrium properties. They then solve this using a global optimization method which iteratively solves some relaxed problems.

Finally, Yang et al [129] design a model with tolls that depend on the entry and exit point, meaning tolls are not link-additive. They consider both fixed and distance tolling schemes. The model is deterministic with elastic demand, and the travel time function is the BPR function, a strictly increasing function of link flow. They transform the network into one that is link-additive in tolls, by creating hypothetical links between each entry-exit pair. Then, by using a result first shown by Meng et al. [86] for the network design problem, it's shown that the optimal value function of the lower level is continuously differentiable. Using the optimal value reformulation of the bilevel problem they transform the problem into a single level continuous problem. They solve this problem using the augmented Lagrangian algorithm, using the frank-wolfe algorithm to solve the augmented Lagrangian subproblem.

Although the majority of toll-pricing models assume the toll location is given, some research has been done to consider the optimal location of tolls. Verhoef [117] examined how to select a singular toll-point. He then proceeded to look at how to select multiple toll-points, considering different strategies for how to sequentially add toll points. Zhang and Yang [138] modified the elastic demand bilevel model to search for cordon toll charges, which could be multi-layered or multi-centered cordons. They developed a genetic algorithm to search for the optimal cordon locations and toll levels simultaneously.

A number of papers have investigated the wider issues and objectives of road pricing. For example, Yang and Zhang [137] develop a multiclass user model that includes social and spatial equity issues. They try to tackle the issue that tolls are a larger proportion of income for low-income drivers. They solve this model using a simulated annealing method. Yin and Lawphongpanich [134] formulate a toll setting model that minimises emissions. In particular, they form an upper level objective function that minimises the level of vehicular CO emissions, which depends on the length and time travelled.

More recently, work on the static road pricing models look at further variations of the models mentioned, for example they explore more complex and varying tolling schemes, or look at applying these models to specific problems. The most common approach to solve these more complex models is to use heuristics, such as in the following papers [61], [62], and [69].

4.3 A bilevel model of the London congestion pricing problem

In this section, we present a bilevel model on the London congestion pricing problem. We first introduce the problem, examining the current issues of the cordon scheme and the objectives that Transport for London (TfL) has to improve it. We then present the lower level problem and the upper level problem of the bilevel model separately. A discussion on the formulated model is then given.

4.3.1 Introduction to the London congestion pricing problem

London is one of the most congested cities in the world. To tackle this problem, in 2003 TfL introduced the London congestion zone charge within central London. This is a cordon based scheme that charges users a fixed fee to use the zone. It is one of the largest congestion charge zones in the world. As a car enters or leaves the zone the vehicle number plate is checked against a registered database to check if a user has paid the charge [99].

As stated in their sixth annual report [76], the charge was introduced to tackle four of the then Mayor's ten priorities for transport in 2001, which are the following.

1. To reduce congestion
2. To make radical improvements in bus services
3. To improve journey time reliability for car users
4. To make the distribution of goods and services more reliable, sustainable and efficient

The main objective was to reduce congestion in and around the zone. The London congestion zone has been a very successful project. As stated [75], after the first year of implementation congestion within the zone had reduced by 30%, with no significant changes to outside the zone. It also found an increase in the use of public transport with a 38% increase in users of all TfL buses entering the zone. The majority of revenue raised from the charge is spent to further improve transportation in London.

Initially, drivers paid a daily fee of £5 to drive within the zone between the hours of 07:00 and 18:30 Monday to Friday, for which they can leave and enter the zone multiple times in a day. Motorcycles, buses and taxis were exempt, and certain other users such as electric or zero-emission cars. However, throughout its implementation the charge has seen four increases extending over a longer time, rising in 2020 to £15 a day, every day, between 07:00

and 22:00. Additionally, private hire vehicles such as Uber, but excluding London's black cabs, are no longer exempt from the charge [78]. Part of the reason for this increase is that, since the introduction of the toll, the level of congestion has been steadily increasing, and traffic within the zone has now slowed to precharge speeds. Many factors have contributed to this, including the rapid increase in the use of private hire cars, who were exempt from the charge, and a decrease of road space for cycle lanes [116], [77].

These new high levels of congestion give rise to the need for an updated pricing scheme. The current mayor of London, Sadiq Khan, has set out a number of aims and proposals for the transportation system in London, as stated in the Mayor's Transport Strategy report [77]. The Mayor has a large focus on making London greener and healthier, coined the healthy streets approach. The approach has the following aims. To encourage people to switch from using cars to more sustainable and active travel, such as walking, cycling and public transport; to reduce pollution that occurs from congested roads; and to increase the reliability and efficiency of public transport. The principal objective in the Mayors strategy to tackle congestion in London is to shift road users to different modes of transport, to reduce the current over-demand of the roads.

To help with shifting demand, the Mayor wants to develop a new road user charging system. He suggests that the current camera based system and fixed cordon scheme is now outdated, and suggests a charge per mile. Proposal 21 in the Mayors strategy states that he wants to look for a "next generation of road user charging systems", which uses technology to develop a sophisticated road user charging scheme that can charge for distance, time, emissions, road danger and other factors [77].

Taking everything into account, we would like to present a model that looks to find an effective toll system that attempts to encapsulate the current problem TfL faces, and that can help to achieve the objectives that TfL and the current Mayor of London have set out. Currently, TfL use the SATURN simulation network program to simulate a model of London, known as the London Highway Assignment Model (LoHAM). Their data collecting and modelling follow the guidelines and rules set out in WebTAG, the Transport Analysis Guidance provided by the Department for Transport [50]. Some of the requirements set out by WebTAG are considered when proposing the bilevel model of the London congestion pricing problem.

4.3.2 The lower level problem

To introduce the lower level problem of the London congestion pricing bilevel model, we first present the deterministic traffic assignment problem with capacity constraints, elastic

demand, and fixed tolls. This problem in then extended to consider multi-commodity users, where the cost function for a user to travel on a route is presented. Finally, we propose a formulation for an updated cordon tolling scheme that charges depending on the user, the distance travelled, and the level of queues in the system.

The traffic assignment problem

We model the lower level problem as a deterministic traffic assignment problem, aimed to model the flow of users through a road network by predicting the choice of route made by individual travellers. It is assumed that users choose the route that minimises their own travel cost, for which they have full knowledge of the travel time and delays in the network. The mathematical program is constructed to satisfy the Wardrop user equilibrium conditions [120]. The first principle states that the travel time of the routes that are utilised are equal, and they are less than the cost of an unused route. The second principle states that the average journey time is minimum.

We wish to model a saturated urban road network, and therefore we need to consider the effect that queueing can have on users choice of route. We use the queueing user equilibrium model approach taken by Yan and Lam [125] and Yang and Yagar [127] to model congestion, except we assume that demand is elastic as opposed to fixed. Elastic demand means that demand is a function of the cost, so as costs increase demand for the network decreases. To model elastic demand we use the formulation described in the book by Patriksson [95].

Let us consider the network $G(N, A)$, formed of a set of nodes N and a set of directed arcs (or links) A. We assume users take a route from origin to a destination, and we denote the set of origin-destination (O-D) pairs by $W \subset \mathcal{N}^2$. Each O-D pair $w \in W$ is connected by a set of routes (or paths), denoted by R_w, where each route $r \in R_w$ is a set of sequentially connected arcs. We assume that each O-D pair has at least one route that joins it, that is that the network is *strongly connected*. The set of all routes in the network is $R = \bigcup_{w \in W} R_w$, and we denote the cardinalities of A, W, and R by $\alpha = |A|$, $\omega \in |W|$ and $\rho \in |R|$ respectively.

The network is defined as follows. The matrix $(\Delta = [\delta_{arw}]) \in \mathbb{R}^{\alpha \times \rho}$ denotes the link-route incidence matrix where

$$\delta_{arw} = \begin{cases} 1 & \text{if arc } a \text{ is in route } \ r \in R_w \\ 0 & \text{otherwise} \end{cases} \qquad \forall a \in A,\ r \in R_w,\ w \in W. \tag{4.1}$$

Let v_{aw} be the link flow on link $a \in A$ for O-D pair $w \in W$, and $(q = [q_{rw}]) \in \mathbb{R}^{\rho}_{+}$ be the flow on route $r \in R_w$. The link and route flow are related by the following flow conservation

constraint

$$v_{aw} = \sum_{a \in A} \delta_{arw} q_{rw}, \qquad \forall a \in A,\ \forall w \in W. \tag{4.2}$$

The total link flow $(v = [v_a]) \in \mathbb{R}^{\alpha}$ is given by

$$v_a = \sum_{w \in W} v_{aw}, \qquad \forall a \in A. \tag{4.3}$$

To consider the effect queueing can have on a users choice of route, we add an explicit capacity constraint on the number of vehicles that can be on a given link. Let us denote the capacity of each link $a \in A$ by $C_a \in \mathbb{R}^{\alpha}$. If the flow on a link is greater than the capacity, then the queue length, and therefore the queueing delay λ_a, will grow. As the traffic assignment problem is in equilibrium, that is in steady-state, we will always have that $v_a \leq C_a$. Furthermore there will only be a queue, and hence queueing delay, if $v_a = C_a$. These conditions are summarised in the following equations

$$\begin{cases} \lambda_a = 0, & \text{if } v_a \leq C_a, \\ \lambda_a \geq 0, & \text{if } v_a = C_a, \end{cases} \qquad \forall a \in A. \tag{4.4}$$

The capacity constraints added to the traffic assignment model are given by

$$v_a \leq C_a, \qquad \forall a \in A. \tag{4.5}$$

The Lagrange multipliers associated with the capacity constraints for each $a \in A$ are equal to the equilibrium traffic queueing delay λ_a. To ensure uniqueness of optimal multipliers and therefore delays, it is required that the link capacity constraints that bind are linearly independent, as shown by Bell [11].

Let the subset of tolled arcs in the network be $\bar{A}$. We denote the toll for using link $a \in \bar{A}$ by τ_a, and the toll for using route $r \in R_w$ by τ_{rw}. Let us denote the cost function for a link as $c_a(v_a, \tau_a)$, which takes into account the travel time and toll price for a link, and is assumed to be continuous and positive. In general, the travel time is always assumed to be link additive, but the toll price may not be. For now, we assume that the tolls are link additive, and so the total cost of a route is link additive. Then the route cost function is defined as

$$c_{rw}(q_{rw}, \tau_{rw}) = \sum_{a \in A} \delta_{arw} c_a(v_a, \tau_a), \qquad \forall r \in R_w,\ \forall w \in W. \tag{4.6}$$

For short, we write c_{rw} to be the travel cost on route $r \in R_w$. The definition of the cost function is given in proceeding subsection in Section 4.3.2.

As demand is elastic, demand between O-D pairs is modelled as a function of the least travel cost between the origin and destination, where we denote the minimum cost between each O-D pair $w \in W$ by π_w. We assume that the demand function g_w is nonnegative, continuous and strictly decreasing for each $w \in W$. The demand function of the cheapest route costs $\pi = (\pi_w)_{w \in W}$ is then given by the following function

$$d_w = g_w(\pi), \qquad \forall w \in W. \tag{4.7}$$

Wardrops equilibrium principle for route flows and demands can then be stated as the following equations. For all $w \in W$ we have

$$\begin{cases} c_{rw} + \sum_{a \in A} \delta_{arw} \lambda_a = \pi_w, & \text{if } q_r > 0, \qquad \forall r \in R_w, \\ c_{rw} + \sum_{a \in A} \delta_{arw} \lambda_a \geq \pi_w, & \text{if } q_r = 0, \qquad \forall r \in R_w, \\ d_w = g_w(\pi), & \text{if } d_w > 0, \\ g_w(\pi) \leq 0, & \text{if } d_w = 0. \end{cases} \tag{4.8}$$

The first two equations represent the Wardrop conditions for route flows, which state that if a route is used then it is a route that is of minimal cost to the user, and any route that is not minimum is not used. The last two equations state that the demand between an O-D pair is either equal to the value of the demand function at the shortest route cost, or is equal to zero if the travel cost is too high.

It can be shown that the user equilibrium conditions (4.8), including feasibility and capacity restrictions for the flow, are equivalent to the following optimisation problem [95].

$$\min_{v} \sum_{a \in A} \int_0^{v_a} c_a(x, \tau_a) dx - \sum_{w \in W} \int_0^{d_w} g_w^{-1}(x) dx, \tag{4.9a}$$

$$\text{s.t.} \sum_{r \in R_w} q_{rw} = d_w, \qquad \forall w \in W, \tag{4.9b}$$

$$\sum_{w \in W} \sum_{r \in R_w} \delta_{arw} q_{rw} = v_a, \qquad \forall a \in A, \tag{4.9c}$$

$$q_{rw} \geq 0, \qquad \forall r \in R_w,\ w \in W, \tag{4.9d}$$

$$v_a \leq C_a, \qquad \forall a \in A. \tag{4.9e}$$

This is a standard traffic assignment formulation of the deterministic elastic demand model

with the additional capacity constraint, which as discussed adds a form of queueing into the model. The addition of this constraint also allows the ability to charge a queueing delay toll, otherwise known as a congestion toll. It has been shown that instead of modelling an explicit toll in the model, one can set the toll equal to the queueing delay from the Lagrange multipliers of the capacity constraint. By charging this toll, the queued equilibrium becomes the unqueued equilibrium network flow pattern. In addition, if it is assumed that all users are homogeneous and therefore have the same value of time, the cost to the user is the same in both cases. This is because we substitute the cost of the queueing delay for an equivalent charge in toll, which users are indifferent to. This result was first shown in a paper by Evans [44], and used in the model by Yang and Bell [126].

The model presented is known as the link-route traffic assignment problem, which models the problem with route flows. An alternative way to represent the problem is by the link-node model. This models the network as a set directed links and a set of nodes, which represent intersections of links and origin and destination nodes. The link-route and link-node formulations of the traffic assignment problem are not quite equivalent, as the feasible set for the link-node formulation can include cycle flows whereas the link-route cannot. However, in equilibrium, if travel costs are positive, no user will choose to travel in a cycle and therefore the set of equilibria do indeed coincide. There are some small advantages to each of the formulations. The link-route formulation has the advantage that its constraint structure is relatively simpler with a smaller number of equality constraints [95]. It further has an advantage of dealing with undirected arcs with a small change to the formulation [115]. A disadvantage to the link-route formulation however is that the set of routes for each O-D pair can be very large, meaning the number of route flow variables can be very large, whilst only a small number of variables are positive at a solution. For this reason, we use the link-node formulation when we solve a simplified road pricing bilevel model in Section 4.4.

The cost function with multi-class users

Different users of the road network may have different values of time, time functions and operating costs. For example, we may expect the travel time for heavy goods vehicles to be longer on a length of road than the travel time for smaller vehicles. It is therefore necessary to split the users into different groups that have similar characteristics and functions. By doing so, we can also consider a toll charge that can charge a different toll depending on the user and their purpose of trip.

The UK Department for Transport set out guidelines for highway modelling of London in the WebTAG Highway Assignment Modelling handbook [50]. They state that the users of

the road network should be split into the following classes.

1. Car employers' business, or in-work-time, trips
2. Car other, or out-of-work-time, trips
3. Taxi trips
4. Light Goods Vehicle (LGV) trips
5. Other Goods Vehicle (OGV) trips

We can easily extend the traffic assignment model to incorporate multicommodity users, as is done for example in Brotcorne et al. [14]. Let the set M be the set of all user classes. For user $m \in M$, we denote the flow on link $a \in A$ by v_a^m, the flow on route $r \in R_w$ by q_{rw}^m, and the demand function for each O-D pair $w \in W$ by d_w^m. We get the following relation between link flows

$$v_a = \sum_{m \in M} v_a^m, \qquad \forall a \in A. \tag{4.10}$$

A similar relation holds for the route flows and demands.

For the cost function $c(v, \tau)$, we use the function provided by the Department for Transport in [50]. The costs to a user include the time it takes for the journey, monetary costs, and operating costs. The function is given in time units, where monetary costs are converted into time units using a parameter called the value of time (VOT). The VOT can vary for different users, and is a measure of the amount a user is willing to pay in order to save time.

For a user $m \in M$ to take a route $r \in R_w$, we denote their toll cost by τ_{rw}^m, their cost function by $c_{rw}^m(q_{rw}^m, \tau_{rw}^m)$ and their travel time by $t_{rw}^m(q_{rw}^m)$. Furthermore, for each user $m \in M$ let β^m be their value of time and θ^m be their operating costs. We denote the distance, or length, for route $r \in R_w$ by l_{rw}. Then the generalised cost to each user $m \in M$, on route $r \in R_w$, between O-D pair $w \in W$, is the following

$$c_{rw}^m(q_{rw}^m, \tau_{rw}^m) = t_{rw}^m(q_{rw}^m) + \frac{\tau_{rw}^m}{\beta^m} + \frac{\theta^m}{\beta^m} l_{rw}. \tag{4.11}$$

The travel time and operating costs are always assumed to be link additive. Let us also assume tolls are link additive, so that the cost is link additive. Using similar notation, the

generalised cost to each user $m \in M$ for each link $a \in A$ is

$$c_a^m(v_a^m, \tau_a^m) = t_a^m(v_a^m) + \frac{\tau_a^m}{\beta^m} + \frac{\theta^m}{\beta^m} l_a, \tag{4.12}$$

where $\tau_a^m = 0$ for $a \notin \bar{A}$.

For the travel time function, we use the Bureau of Public Roads (BPR) function. This is a commonly used function, and is also used by the Department of Transport. It is a convex function which is link additive, and gives the following relation between travel time and traffic flow

$$t_a(v_a) = t_a^0 \left(1 + \alpha_1 \left(\frac{v_a}{C_a}\right)^{\alpha_2}\right), \tag{4.13}$$

where t_a^0 is the free-flow traffic time on link $a \in A$ which can be user dependent, and α_1, α_2 are parameters which can be link dependent. These parameters are classically set to $\alpha_1 = 0.15$ and $\alpha_2 = 4$.

Under all assumptions made, we get the following multicommodity traffic assignment model.

$$\min_v \sum_{m\in M} \sum_{a\in A} \left(\int_0^{v_a^m} t_a^m(x)dx + \frac{\theta^m}{\beta^m} l_a v_a^m \right) + \sum_{m\in M} \sum_{a\in \bar{A}} \frac{\tau_a^m}{\beta^m} v_a^m - \sum_{w\in W} \int_0^{d_w^m} g_w^m(x)^{-1} dx \tag{4.14a}$$

$$\text{s.t.} \sum_{r\in R_w} q_{rw}^m = d_w^m, \qquad \forall w \in W,\ m \in M, \tag{4.14b}$$

$$\sum_{w\in W} \sum_{r\in R_w} \delta_{arw} q_{rw}^m = v_a^m, \qquad \forall a \in A,\ m \in M, \tag{4.14c}$$

$$q_{rw}^m \geq 0, \qquad \forall r \in R_w,\ w \in W,\ m \in M, \tag{4.14d}$$

$$v_a \leq C_a, \qquad \forall a \in A, \tag{4.14e}$$

$$v_a = \sum_{m\in M} v_a^m, \qquad \forall a \in A. \tag{4.14f}$$

An updated tolling scheme

We investigate how to model an updated tolling scheme that can charge users based on a travellers usage of the network, as opposed to a fixed daily charge. We do this by considering a tolling scheme that can charge users based on their user class, how far they have travelled, the level of congestion on the route taken and the time of day.

As it is currently implemented in London, we suggest a cordon based scheme, however we

look at multiple cordon networks which can have different toll charge functions in each cordon zone. We assume the cordons are multi-layered, so that one cordon zone contains another. We present a tolling scheme that considers both a distance charge and a delay-based charge, with the latter helping to eliminate the equilibrium queues of the network. Furthermore, as TfL model the demand of the London network by splitting the data into peak and off-peak demand, it is suggested to split the toll into a peak toll function price and an off-peak toll function price. This can be done by solving the model twice for the different demand during the peak and off-peak time period.

In practise, tolls are often found to be nonlinear, for example there may be a daily maximum limit a user can pay, or discounts may be offered on monthly or annual passes. Because of this we assume the distance toll charge is nonlinear, and we assume that we can charge different tolls for different commodity users. We note however by doing this, the cost to a user for taking a route is no longer link additive. Let I be the set of pricing cordons in the network G. We divide the network into an external network $\bar{G} = (\bar{N}, \bar{A})$, which contain external nodes and links that are not subject to any tolls, and an internal network comprised of multiple cordon networks $\bar{G}_i = (\bar{N}_i, \bar{A}_i)$ for $i = 1, \ldots, I$, which contain internal nodes and links.

We define the distance toll charge as the following. Let the distance of the portion of route $r \in R_w$ in cordon $i \in I$ be denoted by $l = (l^i_{rw})$, and the distance for each arc $a \in A$ be denoted by l_a, so that the following relation holds

$$l^i_{rw} = \sum_{a \in \bar{A}_i} l_a \delta_{arw}, \qquad \forall r \in R_w,\ w \in W,\ i \in I. \tag{4.15}$$

Then for any toll charge function $\Phi^m(l)$ for a user $m \in M$, the distance toll charge on some path $r \in R_w$ is given by

$$\tau^m_{rw}(\Phi) = \sum_{i \in I} \Phi^m(l^i_{rw}), \qquad \forall r \in R_w,\ w \in W,\ m \in M. \tag{4.16}$$

A common nonlinear toll function used in the literature is a piecewise linear function. This somewhat simplifies the function and the ability to solve the problem. Furthermore, the simplification may be more applicable in real-life, as it can be argued that it is easier for users to understand and it is more practical to implement.

In addition to the distance toll charge, we impose a further queueing delay charge. This is equal to the multiplier of the capacity constraint. The resulting solution will ensure that no

equilibrium queues will occur. The total charge for using a route will thus be equal to

$$\bar{\tau}_{rw}^{m} = \tau_{rw}^{m}(\Phi) + \sum_{a \in A} \delta_{arw} \lambda_a, \qquad \forall r \in R_w,\ w \in W,\ m \in M. \tag{4.17}$$

Overall, we get the following traffic assignment problem

$$\min_{v} \sum_{m \in M} \sum_{a \in A} \left(\int_0^{v_a^m} t_a^m(x)dx + \frac{\theta^m}{\beta^m} l_a v_a^m \right) + \sum_{w \in W} \sum_{r \in R_w} \sum_{m \in M} \frac{\tau_{rw}^m(\Phi)}{\beta^m} q_{rw}^m - \sum_{w \in W} \int_0^{d_w^m} g_w^m(x)^{-1} dx \tag{4.18}$$

s.t. 4.14b – 4.14f.

4.3.3 The upper level optimisation problem

As discussed in Section 4.3, the Mayor of London has a large focus to shift road users to using alternative modes of transport such as walking and cycling, to reduce the over-demand of the roads and to become healthier. He has aims to decrease road demand, reduce congestion, reduce pollution from congested roads, and increase the efficiency of public transport and cycle lanes. With these in mind, we suggest that the following four upper level objective functions could be interesting to consider.

1. *Maximise total revenue*

   By maximising revenue, these funds can then be used to improve public transportation networks and services. This objective is commonly seen in the literature, where it is usually assumed that tolls are link additive. This would give us the following objective function

   $$F(v, \tau) = \max_{\tau} \sum_{m \in M} \sum_{a \in \bar{A}} \tau_a^m v_a^m. \tag{4.19}$$

   Using the nonlinear toll function (4.17), we obtain the following objective function

   $$F(q, \tau) = \max_{\tau} \sum_{m \in M} \sum_{w \in W} \sum_{r \in R_w} \bar{\tau}_{rw}^m q_{rw}^m. \tag{4.20}$$

2. *Minimise total system travel time*

   Minimising the total travel time can help to find an optimal toll system that creates the most efficient way for combined users to use the network. As the travel time is

link additive, we get the following objective function

$$F(v,\tau) = \min_{\tau} \sum_{m\in M}\sum_{a\in A} v_a^m t_a(v_a). \tag{4.21}$$

The travel time function suggested for use by the UK Department of Transport is the BPR function given in equation (4.13).

3. *Maximise the ratio of total revenue to total travel time*
   This objective function is a way of considering both of the first two objective functions. It is simply a ratio of them both, and is given by the following function for tolls that are link additive
$$F(v,\tau) = \max_{\tau} \frac{\sum_{m\in M}\sum_{a\in \bar{A}} \tau_a^m v_a^m}{\sum_{m\in M}\sum_{a\in A} v_a^m t_a(v_a)}. \tag{4.22}$$
   When we use the nonlinear toll function, we obtain the following objective function
$$F(q,\tau) = \max_{\tau} \frac{\sum_{m\in M}\sum_{w\in W}\sum_{r\in R_w} \bar{\tau}_{rw}^m q_{rw}^m}{\sum_{m\in M}\sum_{a\in A} v_a^m t_a(v_a)}. \tag{4.23}$$

4. *Minimise vehicle emissions*
   Here we address the aim of reducing pollution in London. We consider the emissions function given by Yin and Lawphongpanich [134], given by the following function
$$e_a(v_a) = 0.2038\, t_a(v_a) \exp\left(0.7962 \frac{l_a}{t_a(v_a)}\right). \tag{4.24}$$
   This function only considers carbon monoxide emissions, and is increasing as long as travelling speed is not too large. With this, the objective function is then given by the following
$$F(v,\tau) = \min_{\tau} \sum_{m\in M}\sum_{a\in A} v_a^m e_a(v_a). \tag{4.25}$$

In addition to the objective functions considered, we wish to model the primary aim of the mayor of London, which is to reduce overall demand in the road network. Modelling a shift in modes of transportation usually requires the need for multiple lower level networks for different modes of transport and a modal split logit function. This greatly increases the complications in the model and the data required. Instead, due to the elastic demand function, we consider the effects that an increase in charges leads to a decrease in overall demand. This could be because users leave the transportation network completely, or they, as is the far more likely case in practise, switch to alternative cheaper modes of transport.

One method of modelling the objective to reduce demand of the road network could be to decrease the capacity of each link to some desired level $\bar{C}_a$, where $\bar{C}_a \leq C_a$ for all $a \in A$. This approach was taken in [126], where they impose the constraint in the upper level problem and allow the lower level problem to be unconstrained. This problem then does not consider the effects of queueing. Instead of restraining the flow on each link, we suggest to add a constraint that bounds the demand and forces the overall demand of the road network to be below a certain threshold. The demand function d_w^m for user $m \in M$ and O-D pair $w \in W$ is a strictly decreasing function of the minimum cost route between each O-D pair. Thus the governing body can set a toll that is high enough to force users off the network and lower demand to the level desired. Let the constant $\bar{d}_w^m$ denote the desired demand level the leader wants for a user $m \in M$ between an O-D pair $w \in W$. Then the constraint added to the upper level problem is

$$d_w^m \leq \bar{d}_w^m, \qquad \forall w \in W,\ m \in M. \tag{4.26}$$

Alternatively, if the aim is to decrease the overall demand of the whole network to a certain level, denoted by the constant $\bar{D}$, then the constraint added to the upper level problem is

$$\sum_{w \in W} d_w \leq \bar{D}. \tag{4.27}$$

We could also limit the overall demand in the road network of a user class. Because we have modelled this objective as a constraint, not only can we achieve the Mayors principal objective to deter users off the road network, but by setting the upper level objective function to one of the functions (4.19)-(4.25), we can achieve some of the other objectives such as maximising profit to allow better infrastructure for cyclists and pedestrians. These in conjunction could help the Mayors aim of shifting users to using greener transport.

4.3.4 Discussion of the London congestion pricing model

In this section, we presented a model for the cordon pricing of the London congestion charge zone. Due to technology advances, there is now the ability to implement a system that can charge a varying toll. We therefore have considered a tolling scheme with multiple cordon zones that charges users based on the type of user, the time of day, the distance travelled, and the level of congestion. We thus create a toll that charges a user not only on how much of the road network they use, but also on how much the road network is used by others. One motivation behind this tolling scheme is that distance tolls alone encourages users to take the shortest route possible, which could create congestion on certain routes. By introducing a further delay based charge, it could help encourage users to take a slightly longer route

that is less congested to ease the overcapacitated roads.

We form the lower level problem as a static, deterministic traffic assignment model with elastic demand, which considers the effects of queueing and congestion. This can be seen as an oversimplification. For example, a stochastic model over a deterministic one could be seen as more representative of real life, as it is unlikely that users will have perfect knowledge of costs and delays. We chose to represent the problem with a static model as opposed to a dynamic model. Dynamic models give the ability to charge variable tolls dependent on the time of day and the level of congestion within the network in real time. They have gained increasing traction within recent years. They could allow the capability of having a more efficient toll scheme that charges for exactly how much the road is being used at that point in time. Although technology has rapidly advanced, having a toll that can vary at any point in time requires the constant monitoring of individuals and of the network as a whole, as well as a sophisticated system to price the tolls. This would be an expensive task to monitor and maintain. Furthermore, a dynamic tolling system can be difficult for users to know exactly how much they will be charged before using a route and could cause erratic driving. By having a static tolling system, there is no confusion to users on what they will be charged, as the value of the toll charges would be fixed and known prior to travelling. The static system can be updated after a time period to ensure it is priced most efficiently.

The upper level model looks to encompass the objectives that are currently being pursued by TfL, as set out in the Mayor of London's transport strategy report. The principal aim being to decrease the overall demand of the network, by getting users to switch to greener transport such as walking and cycling. One method of doing this is by modelling the transport networks of all modes of transport, and using a modal split logic function. Due to the large increase in the need for data and modelling of the problem, as well as an increase in complexity of solving the model, we present a different way to dealing with this problem. By using elastic demand, a constraint in the upper level problem is introduced that constrains the demand to be below some desired level of demand. This model could be used instead of the modal-split model when, for example, knowledge of how alternative modes of transport networks will be used is not needed in detail. Additionally, we note that although four different objective functions were proposed to capture the other objectives by TfL, for future work it may be interesting to consider a multi-objective function that combines them all together.

The proposed model is complicated to solve. We form a toll function which has nonlinear distance tolls, which means that the tolls are not link additive. A number of approaches to solving road-pricing problems with nonlinear tolls introduce dummy links to transform

the model into one with link additive tolls. These models usually only consider the case with homogenous users. This is because uniqueness of link flows can not be guaranteed with multiclass users, a property shown by Netter [92], making it difficult to solve. A large number of algorithms that look to solve the road-pricing problem with multicommodity users simplify the problem by using a linear cost function. The addition of capacity constraints and demand constraints further complicate the proposed model. To our knowledge, an algorithm that solves a bilevel model of this formulation has not yet been developed. In the remainder of this chapter, we greatly simplify the model presented to enable us to apply the trust-region algorithm to solve it.

4.4 A trust-region method to solve a simple road pricing transportation problem

We look at how we can apply the trust-region algorithm introduced in Chapter 3 to a simplified version of the London congestion pricing model described in Section 4.3. We simplify a number of the complexities in the model, resulting in a problem that has distinct features which allows us to apply the algorithm. The simplifications compared with the London model are the following. For the followers problem we use an uncapacitated deterministic user equilibrium problem, with homogeneous users and fixed demand. For the tolling scheme we consider a fixed link scheme, a fixed cordon scheme and a distance cordon scheme, where all tolls are link additive. For the upper level problem we analyse the results for all four objective functions described in the previous chapter, but as demand is fixed we cannot impose an upper bound constraint on demand.

We first introduce the simplified model that we look to solve, before examining the properties of this problem that allow us to apply the bilevel trust-region algorithm. We then show detailed calculated steps of the algorithm applied to this problem. Finally, we present numerical results and analysis for three examples of varying sizes to test the quality and robustness of solutions.

4.4.1 The simplified road pricing transportation model

The lower level problem is formed as a deterministic traffic assignment problem with homogeneous users and fixed demand. We use the link-node formulation of the traffic assignment problem, as opposed to the link-route formulation, to solve the model. In this formulation, the network $G = (N, A)$ is represented by a set N of nodes, and a set A of directed links. The nodes correspond to intersections of the road network, whilst the links corresponded to the roads joining the intersections. We denote the cardinality of the set N

by $\nu = |N|$. Assuming demand is fixed, the demand vector $(D_w \in [D_{iw}]) \in \mathbb{R}^\nu$ for O-D pair $w \in W$ is given by

$$D_{jw} = \begin{cases} d_w & \text{if node } j \text{ is in the origin of O-D pair } w \in W, \\ -d_w & \text{if node } j \text{ is in the destination of O-D pair } w \in W, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j \in N,$$

where d_w is the fixed demand for O-D pair $w \in W$. Then, using the same notation as in Section 4.3, the node-link formulation for the deterministic traffic assignment problem with fixed demand is given by the following

$$\min_{v} \sum_{a \in A} \int_0^{v_a} c_a(x, \tau_a) dx, \tag{4.28a}$$

$$\text{s.t.} \sum_{a \in j^+} v_{aw} - \sum_{a \in j^-} v_{aw} = D_{jw}, \qquad \forall w \in W,\ \forall j \in N, \tag{4.28b}$$

$$v_{aw} \geq 0, \qquad \forall w \in W,\ \forall a \in A, \tag{4.28c}$$

$$v_a = \sum_{w \in W} v_{aw}, \qquad \forall a \in A, \tag{4.28d}$$

where v_{aw} is the flow on link $a \in A$ for O-D pair $w \in W$, j^+ denotes the set of arcs leaving node j and j^- denotes the set of arcs entering node j.

We drop the fixed operating costs in the cost function (4.12), and set the cost function $c_a(v_a, \tau_a)$ for using link $a \in A$ to be equal to the travel time of using the link, calculated with the travel time function $t_a(v_a)$, plus the monetary cost from the tolls of using the link, which is converted into time using the value of time β. For the travel time function, we use the BPR function given in equation (4.13), using the commonly used parameters $\alpha_1 = 0.15$ and $\alpha_2 = 4$. The travel time function for using link $a \in A$ therefore becomes

$$t_a(v_a) = t_a^0 \left(1 + 0.15 \left(\frac{v_a}{C_a}\right)^4\right). \tag{4.29}$$

For the cost of the tolls for using a route, we assume that tolls are link additive and consider the following three tolling schemes. For each scheme we state the lower level objective function used.

1. *Distance cordon*

   The toll function is a linear function with respect to distance travelled. We assume there are multiple cordon zones, and a different charging rate per mile is charged in a

different cordon $i \in I$, denoted by τ^i. As before, we denote the length of a link by l_a, and obtain the following lower level objective function

$$f(v, \tau) = \int_0^{v_a} t_a(x)dx + \frac{1}{\beta} \sum_{i \in I} \sum_{a \in \bar{A}_i} \tau^i l_a v_a. \tag{4.30}$$

2. *Fixed cordon*

   We assume there are multiple cordon zones, where we charge a fixed cost when entering a zone. This way the cordon fares are link additive. We therefore charge the same toll charge τ^i for a link that crosses over into the cordon zone $i \in I$. We obtain the following lower level objective function

$$f(v, \tau) = \int_0^{v_a} t_a(x)dx + \frac{1}{\beta} \sum_{i \in I} \sum_{a \in \bar{A}_i} \tau^i v_a. \tag{4.31}$$

3. *Fixed link*

   We assume fixed tolls on given links, denoted by τ_a for a link $a \in \bar{A}$. This is the tolling scheme that is most commonly used in the literature. We obtain the following lower level objective function

$$f(v, \tau) = \int_0^{v_a} t_a(x)dx + \frac{1}{\beta} \sum_{a \in \bar{A}} \tau_a v_a. \tag{4.32}$$

For the upper level objective function, we look at all four objective functions that are stated in Section 4.3.3. That is: maximising total revenue, minimising total system travel time, maximising the ratio of total revenue to total travel time, and minimising vehicle emissions. Lastly, we define the set T to be any upper level constraints on the tolls τ. For this problem, we consider the case where we have upper and lower bounds on the tolls. So for example, for the fixed link tolls τ_a, the constraint set would be given by

$$T = \{\tau_a |\ b_a \leq \tau_a \leq u_a\},$$

where b_a is a lower bound and u_a is an upper bound on the toll τ_a on link $a \in A$.

4.4.2 Properties of the bilevel road pricing problem

As the bilevel problem is nonconvex, local optimal solutions can exist making it very difficult to find a global solution. Additionally we may find that the road pricing bilevel model has multiple global optimal solutions, as noted in [128]. That being said, the simplifications

on the bilevel model outlined in Section 4.4.1 give rise to particular properties of the lower level problem that we can take advantage of. In particular, we show that the optimal value function is continuously differentiable, and we can therefore apply the trust-region algorithm introduced in Chapter 3. The exact penalty problem we then look to solve becomes a continuous single level problem, with a nonconvex objective function and convex constraints. We also examine conditions for when the problem is partially calm.

Differentiability of the optimal value function

Without lost of generality, let us define the cost function for taking arc a as

$$c_a(v, \tau) = t_a(v_a) + \frac{1}{\beta}\tau_a, \tag{4.33}$$

and the lower level objective function as

$$f(v, \tau) = \int_0^{v_a} t_a(x)dx + \frac{1}{\beta}\sum_{a\in\bar{A}} \tau_a v_a. \tag{4.34}$$

The following results on existence and uniqueness of solutions are shown in [95]. Let us assume that the network is strongly connected, the demand is non-negative, and the travel time function is positive and continuous. Then it can be shown using Weierstrass' Theorem that a solution to the problem exists. Additionally, let us assume that the demand is positive for each O-D pair. Then as the travel time function (4.29) is strictly increasing for each $a \in A$, the equilibrium travel times and equilibrium link flows are unique. The proof of the latter result is done by showing that the objective function of the traffic assignment problem is strictly convex with respect to the link flows. Uniqueness of the link flows ensures that the solution function of the lower level problem is continuous.

We note that whilst link flows are unique, the equilibrium route flow variables may not be, since a link flow pattern can correspond to multiple route flow patterns. For this reason, we solve the problem to find the unique equilibrium link flows and not the route flow variables. By using the link-node formulation, the route flow variables are not included in the model. Furthermore we only consider the case with homogeneous users, because uniqueness of link flows can not be guaranteed with multiclass users, as shown by Netter [92]. Finally, we require that the time function is strictly increasing for each $a \in A$. This can be seen as a strong assumption, for example it does not include the simple case where the time function is linear.

We are now in a position to determine the differentiability of the optimal value function. Assuming the above assumptions hold, the problem has unique link flows. In addition,

the problem has the important property that the constraint set is a convex set that does not depend on the upper toll variables τ. Using both of these conditions, we find the optimal value function is differentiable. Differentiability of the optimal value function for this particular problem was shown by Meng et al. [86], however we can see this is the case by looking at the optimal value generalised gradient set given in Theorem 3.2.12. This set clearly reduces to a singleton as for given upper level variables there is a unique lower level solution and the gradients of the lower level constraints with respect to the upper level variables are equal to zero.

Without loss of generality, using the lower level objective function given by equation (4.34) we get the following optimal value function

$$\varphi(x) = \sum_{a\in A} \int_0^{v_a^*(\tau)} c_a(x,\tau_a)dx = \sum_{a\in A} \int_0^{v_a^*(\tau)} t_a(x)dx + \frac{1}{\beta}\sum_{a\in\bar{A}} \tau_a v_a^*(\tau), \tag{4.35}$$

where the user equilibrium flow is

$$v^*(\tau) = \arg\min_{v\in V} \int_0^{v_a} c_a(x,\tau_a)dx, \tag{4.36}$$

and V is the constraint set associated with equations (4.28b)-(4.28d). To find the user equilibrium flow for a given toll, a number of efficient algorithms have been developed to solve the traffic assignment problem. A common method used to solve this problem is the Frank-Wolfe deterministic user equilibrium assignment procedure. Details of this method, and other methods developed to solve the traffic assignment problem, can be found in [95].

Given the optimal value function (4.35) and the definition of its generalised gradient set, the gradient of the optimal value function is found to be equal to

$$\nabla\varphi(\tau) = \nabla_x f(v^*(\tau),\tau) = \frac{1}{\beta} v_a^*. \tag{4.37}$$

The partial calmness condition

As we have seen by Theorem 2.3.2, if the road pricing bilevel problem is partially calm then by solving the following exact penalty problem

$$\min_{v\in V,\tau\in T} P(v,\tau) = F(v,\tau) + \gamma\big(f(v,\tau) - \varphi(\tau)\big), \tag{4.38}$$

we can find some $\gamma > 0$ such that the local solution of the exact penalty problem is also one of the road pricing problem.

Unfortunately it can be difficult to verify that the problem is partially calm. In the literature, a handful of papers look at applying the partial calmness property to solve the road pricing transportation problem. The majority of those assume that the cost is fixed and the objective function takes the form $f(v,\tau) = v^T\tau$, a bilinear function. As the lower level problem is a parametric linear program, the partial calmness condition is satisfied, a result first shown by Ye and Zhu [131]. Our lower level objective function (4.34) however includes both the bilinear toll term $v^T\tau$, and the integral of the travel time function.

In a paper by Dempe and Zemkoho [36], they give conditions for the road pricing problem to be partially calm when the travel time function is increasing for each $a \in A$. Using [36, Theorem 3.3], and applying it with our problem, we find if the following condition holds then the problem is partially calm

$$\sum_{a\in A}\Big(t_a^0(1+0.15\Big(\frac{v_a}{C_a}\Big)^4)+\frac{1}{\beta}\tau_a\Big)h_a \geq \alpha||h||, \quad \forall v\in\Psi(\tau),\ h\in T_V(v)\cap N_{\Psi(\tau)}(v),\ \tau\in T, \tag{4.39}$$

where $\Psi(\tau)$ is the solution set of the lower level problem for a given τ, and T_C, N_C denote the tangent cone and normal cone to a set C respectively, in the sense of convex analysis.

4.4.3 A Trust-Region algorithm for the bilevel road pricing problem

For the presented simplified road pricing problem, we find the optimal value function to be continuously differentiable. Thus, by Theorem 3.3.1, we can apply the trust-region method presented in Algorithm 1 to solve the exact penalty problem (4.38). For the trust-region subproblem, we use the model that linearlises the whole penalty function, given in equation (3.20). We choose this model because it enables us to solve the subproblem efficiently, by allowing the ability to separate the subproblem into two smaller linear problems. For the remaining of this section, we show of the trust-region subproblem can be simplified into these two smaller linear problems, which takes a similar approach as in [129].

For the following calculus and without loss of generality, we use the function to maximise the total revenue defined by equation (4.19) for the upper level objective function, and we use the general cost function defined by equation (4.33).

By differentiating the penalty function $P(v,\tau)$ with respect to v and τ, we get the gradient of the function at some feasible solution v^k, τ^k to be the following

$$\frac{\partial P}{\partial v_a}\Big|_{(v,\tau)=(v^k,\tau^k)} = -\tau_a^k + \gamma(t_a(v_a^k)+\frac{1}{\beta}\tau_a^k), \qquad \forall a\in A, \tag{4.40}$$

$$\left.\frac{\partial P}{\partial \tau_a}\right|_{(v,\tau)=(v^k,\tau^k)} = -v_a^k + \gamma\left(\frac{1}{\beta}v_a^k - \frac{1}{\beta}v_a^*\Big|_{\tau=\tau^k}\right), \qquad \forall a \in A. \tag{4.41}$$

Let us denote the partial derivatives as the following

$$g_{v_a}^k = \left.\frac{\partial P}{\partial v_a}\right|_{(v,\tau)=(v^k,\tau^k)}, \qquad g_{\tau_a}^k = \left.\frac{\partial P}{\partial x_a}\right|_{(v,\tau)=(v^k,\tau^k)}.$$

The trust-region subproblem we solve at each iteration is then given by

$$\min_{v,\tau} \quad \bar{P}_k(v,\tau) = \sum_{a\in A} g_{v_a}^k v_a + \sum_{a\in A} g_{\tau_a}^k \tau_a \tag{4.42a}$$

$$\text{s.t.} \quad b_a \le \tau_a \le u_a, \qquad \forall a \in A, \tag{4.42b}$$

$$\sum_{a\in j^+} v_{aw} - \sum_{a\in j^-} v_{aw} = D_{jw}, \qquad \forall w \in W,\ \forall j \in N, \tag{4.42c}$$

$$v_{aw} \ge 0, \qquad \forall w \in W,\ \forall a \in A, \tag{4.42d}$$

$$v_a = \sum_{w\in W} v_{aw}, \qquad \forall a \in A, \tag{4.42e}$$

$$\|(v,\tau) - (v^k,\tau^k)\|_\infty \le \Delta_k. \tag{4.42f}$$

This subproblem can be broken down into two independent linear sub-problems by separating the upper and lower variables. As the infinity norm is simply a rectangular box, we rewrite the constraint as the equivalent linear constraints. The trust-region subproblems are then given by the following

SP1:

$$\min_{\mathbf{v}} \quad \sum_{a\in A} g_{v_a}^k v_a \tag{4.43a}$$

$$\text{s.t.} \quad \sum_{a\in j^+} v_{aw} - \sum_{a\in j^-} v_{aw} = D_{jw}, \qquad \forall w \in W,\ \forall j \in N, \tag{4.43b}$$

$$v_{aw} \ge 0, \qquad \forall w \in W,\ \forall a \in A, \tag{4.43c}$$

$$v_a = \sum_{w\in W} v_{aw}, \qquad \forall a \in A, \tag{4.43d}$$

$$v_a^k - \Delta_k \le v_a \le v_a^k + \Delta_k, \qquad \forall a \in A. \tag{4.43e}$$

SP2:

$$\min_{\boldsymbol{\tau}} \quad \sum_{a\in A} g_{\tau_a}^k \tau_a \tag{4.44a}$$

$$\text{s.t.} \quad l_a \le \tau_a \le u_a, \qquad \forall a \in A, \tag{4.44b}$$

$$\tau_a^k - \Delta_k \le \tau_a \le \tau_a^k + \Delta_k, \qquad \forall a \in A. \tag{4.44c}$$

We note that because the algorithm is reduced to the classical trust-region algorithm for smooth problems, the subproblems only need to be solved approximately for the algorithm to converge to a stationary point. To solve the subproblems, we can see that subproblem SP2 is a simple linear program with upper and lower bounds which can be solved easily and efficiently. Subproblem SP1 can be more complicated to solve. It is linear program, and as such can be solved by classical linear optimisation algorithms. However, when the network becomes large the problem can become computationally expensive and slow to solve as the number of constraints increases greatly.

Looking closer at SP1 we see it is a fixed cost traffic assignment problem, but with additional upper and lower bound constraints on the link flows due to the trust-region constraint. This problem can be transformed into the form of a capacitated fixed cost traffic assignment problem, which is a traffic assignment problem with an upper bound on the link flows but a lower bound of zero. This can be done by variable transformation, by replacing v_a with $v'_a = v_a - b_a$. These problems are known as linear multi-commodity flow problems, and are known to be prohibitively expensive to solve repeatedly. A number of algorithms have however been designed to try to solve them efficiently, by utilising the structure of the constraint set, including Lagrangian relaxation algorithms, column generation algorithms, and Dantzig-Wolfe decomposition [1].

We have shown how the trust-region subproblem in Algorithm 1 can be broken down into the subproblems SP1 and SP2. Therefore, by solving these two smaller linear problems for the trust-region subproblem in Step 1, the algorithm should converge to a stationary point of the exact penalty problem (4.38). Furthermore, by choosing $\gamma > 0$ well, we hope to find a stationary point of the bilevel road pricing problem.

4.5 Numerical Results

We test the trust-region algorithm on three network examples. The first example is a small network first presented in a paper by Yan and Lam [125], who used a fixed link tolling scheme. This example allows us to compare the quality of the solution from the trust-region algorithm with a known solution, and to test the performance of the algorithm against another. The second example is another small network. In this example we run each problem with 1000 different randomly generated starting points for all four upper level objective functions, again with a fixed link tolling scheme. This allows us to test the robustness of the algorithm for different starting points, and to compare the solutions for different upper level objectives. The final example is a larger network, which is a simplified map of the congestion zone in London. In this example we test all three tolling schemes for two objective functions,

to compare solutions of the different tolling schemes whilst seeing how well the algorithm performs on a larger network.

4.5.1 Implementation and parameter details

We implemented Algorithm 1 on the road pricing transportation problem in MATLAB R2019a. As we use the trust-region model that linearlises the whole penalty function, using the same notation as in Section 3.5, we denote the algorithm by BLTR-L. For the trust-region subproblem in Step 1, we solve the trust-region subproblems SP1 (4.43) and SP2 (4.44). We used the MATLAB built-in linear solver *linprog* to solve them, using the interior-point-legacy algorithm for SP1 and the interior-point algorithm for SP2. To solve the lower level traffic assignment problem for a given toll level in Step 2, we found the solver *fmincon* used for the numerical results in Section 3.5 was too slow, and the solver struggled to solve the problem. We therefore instead used the interior point method provided by the European Aerospace Agency nonlinear programming solver WORHP. This is a powerful solver designed to solve large, high dimensional sparse non-linear optimisation problems.

To stop the algorithm, we use the stopping criteria **SC1** detailed in Section 3.3.2, which stops the algorithm when we reach within some tolerance θ of the stationarity conditions of the exact penalty problem, given by Theorem 2.14. As the optimal value function is differentiable, these conditions can be checked easily. However in the case a stationary point cannot be reached, we use the stopping criteria **SC2a** and **SC3**-**SC5**. We use the same notation for the parameters except for **SC2a**, where we denote the tolerance level by $\epsilon > 0$.

For the choice of penalty parameters, starting points, initial trust-region radius and the stationary point tolerance, these were problem specific and are detailed in the proceeding sections. The remaining parameters and tolerances for the algorithm are outlined in Table 4.1.

Table 4.1: Parameters for numerical experiments

Trust-region parameters	ν_1	0.01
	ν_2	0.9
	σ_1	0.5
	σ_2	1
Stopping criteria	ϵ	1e-10
	$k_{\max}$	20000
	$u_{\max}$	25
	$\Delta_{\min}$	1e-12

We note that in comparison to the parameters used in Chapter 3, we have had to increase

the maximum number of iterations and decrease the stopping criteria tolerances ϵ and $\Delta_{\min}$ as the algorithm was stopping prematurely, and we found the convergence for this problem required on average far more iterations.

4.5.2 Example 1

Description and data of the network

To assess the quality of the solution, we compare our results with an example in the literature. The network consists of 6 nodes and 7 links, and was presented as an example by Yan and Lam [125, Example 2]. In this example, Yan and Lam use a road pricing bilevel model that uses a fixed demand queueing traffic assignment model, however they provide demand for the network that does not reach its capacity and so the solutions should coincide. They solve this using an equilibrium decomposition optimization (EDO) algorithm developed by Suwansirikul et al. [114]. The network we look to solve is shown in Figure 4.1.

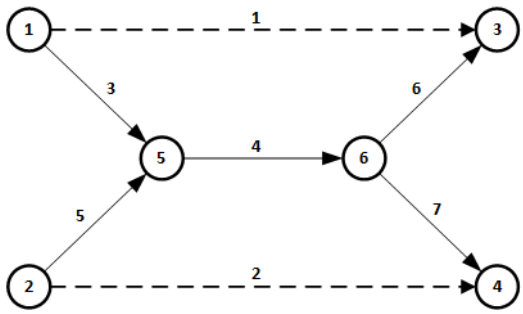


Figure 4.1: Network for Example 1

This example uses a fixed tolling scheme, where links 1 and 2 are tolled, represented by a dashed arrow, and the bounds on all toll variables are $0 \leq \tau \leq 5$. There are two O-D pairs, which are pairs $1 \to 3$ and $2 \to 4$, with demands $D_{13} = D_{24} = 30$. The data for the free-flow travel time and capacity for each link, used for the BPR travel time function, is provided in Table 4.2.

Table 4.2: Data for Example 1

Link a	1	2	3	4	5	6	7
t_a^0	8.0	9.0	2.0	6.0	3.0	3.0	4.0
C_a	20.0	20.0	20.0	40.0	20.0	25.0	25.0

The travel times are given in minutes and the link tolls are calculated in the equivalent minutes. We therefore do not convert the toll into pounds using the users value of time.

Results and Analysis

We used the same starting points as given by Yan and Lam, which are $\tau_1^0 = \tau_2^0 = 2.5$. They consider the first three upper level objective functions described in Section 4.3.3 which are to maximise profit, to minimise total user travel time, and to maximise the ratio of profit to total user travel time. We therefore solve only these three objective functions, where we can compare solutions. We used an initial trust-region radius of $\Delta_0 = 10$ and a stationarity condition tolerance of $\theta = 1e-6$. The algorithm was tested for different values of the penalty parameter to assess the sensitivity of the choice of parameter.

Table 4.3 shows the results for the trust-region algorithm for the road pricing bilevel problem, denoted by BLTR-L, and the equilibrium decomposition optimization algorithm, denoted by EDO, which was applied by Yam and Lam and reported in [125, Table 6]. They give the results for the upper level objective function value F, the upper level toll prices τ_1 and τ_2, and the number of main iterations. We further include the value of the lower level objective function f and the CPU time measured in seconds.

Table 4.3: Solutions using the BLTR-L and EDO algorithm for network Example 1

Objective function	**Alg.**	γ	**F**	**f**	τ_1	τ_2	**Iter.**	**CPU** (s)
Max revenue	BLTR-L	2^1	97.16	762.80	4.472	5.590	354	21.78
		2^2	105.24	741.76	3.611	4.568	779	99.86
		2^3	106.77	733.44	3.333	4.228	2001	250.57
		2^4	107.10	729.71	3.218	4.085	3822	463.24
		2^5	107.18	727.94	3.165	4.018	7529	1007.3
	EDO		107.15		3.164	4.063	13	
Min total travel time	BLTR-L	2^1	628.60	701.72	2.399	3.200	182	22.33
		2^2	628.60	701.72	2.399	3.200	543	67.73
		2^3	628.60	701.72	2.399	3.200	421	70.67
		2^4	628.60	701.72	2.399	3.200	4200	610.38
		2^5	628.60	701.72	2.399	3.200	1871	235.55
	EDO		628.60		2.399	3.200	13	
Max ratio	BLTR-L	0.005	0.155	745.80	3.742	4.758	1217	320.99
		0.01	0.165	731.64	3.257	4.177	1067	290.52
		0.05	0.168	721.71	2.960	3.811	2840	688.74
		0.1	0.168	720.42	2.925	3.768	18590	3978.1
	EDO		0.168		3.125	3.828	13	

All runs of the algorithm were successful in finding a stationary point, where the stationarity conditions were satisfied to a tolerance of 1e-6. From the results we see the solutions found for the trust-region algorithm are consistent with those found by the EDO algorithm. For the objective to maximise revenue, we found a solution with a slightly better upper level

objective value than that found by the EDO algorithm when $\gamma = 32$. For the objective to maximise the ratio of total revenue to total travel time we found the same solution as that found by the EDO algorithm for both $\gamma = 0.05$ and $\gamma = 0.1$. Finally, for the objective to minimise the total travel time, we obtained the same solution as that found by the EDO algorithm for all values of the penalty parameter.

For both the objective functions to maximise revenue and to maximise the ratio, we found that as the penalty parameter was increased, the upper level objective value increased, finding a better solution. However, we also found the CPU time and the number of iterations increased, and although there was an improvement on the upper level objective value, we observed a levelling off on its value, whilst the number of iterations it took to solve the problem increased greatly. For example, whilst maximising the ratio we obtained the same optimal solution as Yan and Lam for both $\gamma = 0.05$ and $\gamma = 0.1$, but it took the algorithm with the larger penalty parameter almost six times longer to find the same solution. In contrast, when we solved the problem with the objective to minimise the total travel time, we obtained the same solution as that found by the EDO algorithm for all values of the penalty parameter. We did however still observe that as the penalty parameter is increased, the computational time and number of iterations also increases. For this particular problem and objective function, it appears a lower penalty parameter would be preferred.

When comparing the performance with the EDO algorithm, the trust-region algorithm had a relatively large number of iterations for each problem run and the convergence was quite slow, particularly for when the penalty parameter was larger. The number of main iterations for the EDO algorithm is 13 iterations for all three objective functions, far fewer than the number of iterations taken for the trust-region algorithm. That being said, Yan and Lam only solve the algorithm to a tolerance of 1e-3, and a large number of iterations are taken in the trust-region algorithm to reduce the stationarity conditions to a tolerance of 1e-6. Furthermore, a slightly better result for the first objective function is found using the trust-region algorithm.

We make a final note on how the choice of penalty parameters was made. We found the first two objective functions responded well between the values chosen. It was found that a penalty parameter chosen too low resulted in not enough penalisation of the penalty gap and too much emphasis on the upper level objective function. For example when maximising profit, a low penalty parameter resulted in the algorithm increasing tolls to the maximum upper bound. On the other hand, if the penalty parameter was set too large then convergence was too slow. The third objective function, to maximise the ratio of the first two functions, required a smaller scale of penalty parameter. This may be because the value of the objective

function is relatively much smaller than the values of the first two upper level objective functions. Using the same penalty parameters as the first two functions resulted in a very slow convergence as the penalty gap was penalised too strongly.

4.5.3 Example 2

Description and data of the network

Example 2 is a simple network with 5 nodes and 6 links. We test the network for 1000 different starting points and three different penalty parameters, which we run for all four objective functions described in Section 4.3.3. The aim of this example is to test the robustness of the algorithm for different starting points. The network we look to solve is shown in Figure 4.2.

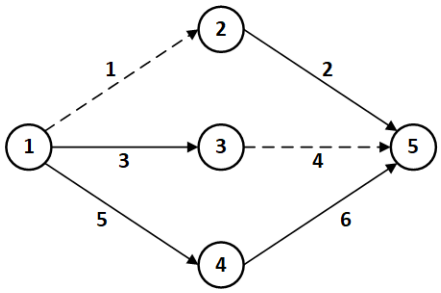


Figure 4.2: Network for Example 2

This example uses a fixed tolling scheme, where links 1 and 4 are tolled, and bounds on all toll variables are given by $0 \leq \tau \leq 20$. We only have one O-D pair, pair $1 \rightarrow 5$, with a demand $D_{15} = 12$. As in Example 1, the travel time is given in minutes, and the tolls for the links are calculated in the equivalent minutes. The data for the free-flow travel time and capacity for each link, used for the BPR travel time function, is provided in Table 4.4.

Table 4.4: Data for Example 2

Link a	1	2	3	4	5	6
t_a^0	5.0	5.0	10.0	10.0	12.5	12.5
C_a	5.0	5.0	7.0	7.0	10.0	10.0

Results and Analysis

We ran each test 1000 times for different starting points, which were randomly generated points between the values of 0 and 15. We used an initial trust-region radius of $\Delta_0 = 10$ and we increased the stationarity condition tolerance from Example 1 to θ=1e-5, due to

the combined length of the runs. All four upper level objective functions that have been discussed were tested, each with 3 different penalty parameters.

Depending on the initial starting point the algorithm converged to a number of different stationary points, where for each example between 1 to 3 different upper level objective values were attained from the stationary points found. The results are summarised in Table 4.5. We present all of the upper level objective values found, denoted by F, with number 1 being the best upper level objective value found. All the stationary points that found each upper level objective function value found the same objective value within a very small range, referred to in the table as the range. We also present the percentage of runs that found each value F. The average CPU time and average number of iterations for all 1000 runs are given in the final two columns.

Table 4.5: Solutions using the BLTR-L for network Example 2

Objective Function	γ	**Stationary Points**				**Average CPU** (s)	**Average Iter**
		No.	F	Range	Runs that found this point (%)		
Max revenue	2^3	1	91.62	4.67e-04	61.4%	10.36	128
		2	73.79	1.6e-04	38.6%		
	2^4	1	92.39	3.86e-04	54.7%	19.46	248
		2	74.30	1.26e-04	45.3%		
	2^5	1	92.57	2.39e-04	55.1%	38.79	493
		2	74.42	9.86e-05	44.9%		
Min total travel time	2^3	1	207.37	2.19e-08	26.1%	19.51	204
		2	208.17	1.38e-08	29.5%		
		3	231.43	9.24e-09	44.4%		
	2^4	1	207.37	5.86e-08	23.7%	52.37	481
		2	208.17	2.26e-08	30.1%		
		3	231.43	7.16e-08	46.2%		
	2^5	1	207.37	7.99e-08	19.8%	51.12	511
		2	208.17	5.40e-08	32.3%		
		3	231.43	1.23e-07	47.9%		
Max ratio	10^{-2}	1	0.279	2.21e-04	100%	5.23	45
	10^{-1}	1	0.445	2.72e-05	55.1%	19.72	205
		2	0.320	4.96e-06	44.9%		
	1	1	0.447	1.64e-05	49.1%	188.98	1935
		2	0.321	5.64e-06	50.9%		
Min emissions	2^2	1	73.18	9.95e-04	51.7%	47.86	444
		2	85.54	3.03e-08	48.3%		
	2^3	1	73.18	8.75e-04	52.4%	75.44	742
		2	85.54	5.20e-08	47.6%		
	2^4	1	73.18	1.14e-03	48.7%	257.68	2500
		2	85.54	2.61e-07	51.3%		

All tests converged to a stationary point, where the stationarity conditions were satisfied to a tolerance of 1e-5, however we found the choice of initial starting point could impact the solution found. In general, the stationary points that were found returned two upper level objective values, with an almost even split between runs that converged to points with each upper level objective value. Only for the objective function to minimise total travel time do we find three upper level objective values, with the best and second best value being relatively close to each other. Just over half of the runs found these two values whilst the other half of the runs found the other objective value. It was found for a number of problems that various toll solutions were found that produced the same upper level objective function value, with varying lower level objective functions.

Like in Example 1, for both the objective functions to maximise revenue and to maximise the ratio, we found that the quality of the solution found improved as the penalty parameter was increased. For the other two objective functions, the solution found was the same regardless of the choice of penalty parameter. Additionally, it was found for all problems that as the penalty parameter was increased, the average CPU time and the average number of iterations for all problems increased. We therefore find again that, for the objective functions to maximise revenue and maximise the ratio, there is a trade-off between the quality of the solution and the time it takes to solve a problem. This does not appear to be the case for the other two objective functions, where the solution of F remained constant regardless on the choice of penalty parameter.

Overall, this example suggests to us that the algorithm is robust in that it successfully stops at a stationary point for all runs. The choice of an initial starting point however should be made with careful consideration. If an estimate or approximation of the optimal tolls are difficult to attain for an initial starting point, a suggestion is to try multiple starting points to provide more assurance in finding the best solution. Furthermore, a choice of penalty parameter has to be made, which looks to depend on the choice of the upper level objective function. It is suggested to try starting with a small number of different penalty parameters, to firstly attain whether the choice has an impact on the value of the upper level objective function. If it does, the parameter may need to be refined further, with a suggestion of increasing the parameter and assessing the solution. When choosing the penalty parameters for this problem, we chose similar values to the parameters chosen in Example 1. When solving the problem to minimise emissions, due to the average length of time it took for $\gamma = 2^4$ to converge for each run, we chose to test a smaller penalty parameter of $\gamma = 2^2$.

4.5.4 Example 3

Description and data of the network

We present this network example to test how the algorithm performs on a larger network with cordon tolling schemes, and to allow a comparison on the solutions found for the different tolling schemes. This example is not to test the quality of the solution or the robustness of the algorithm, but the ability to find a stationary point of a larger network with different tolling schemes. This problem consists of 44 nodes and 154 arcs, and is a simplified map of the major roads within and around the London congestion zone. The network is depicted in Figure 4.3.

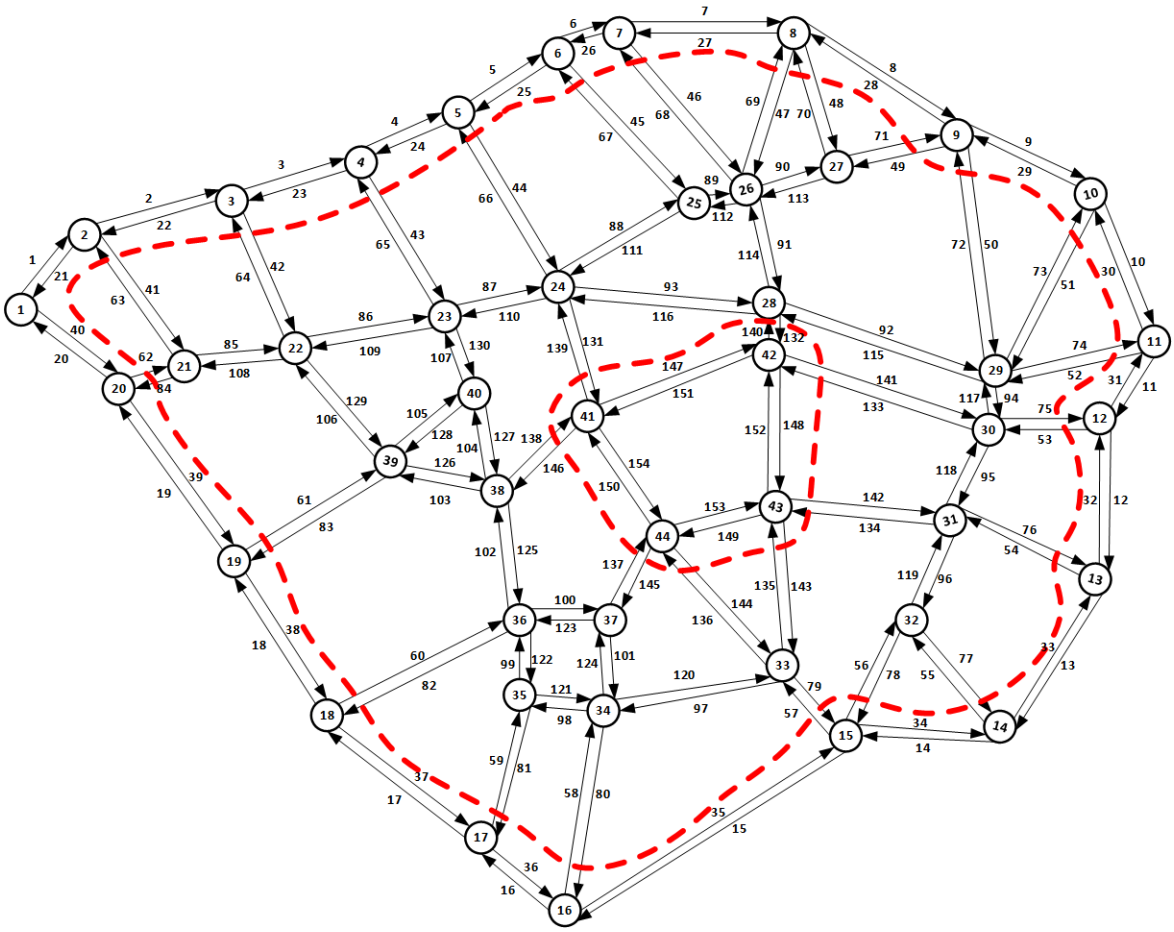


Figure 4.3: Network for Example 3

The outer red ring represents the current congestion zone perimeter in London, and the inner ring is an an additional cordon zone within the center of the network. We have 5 O-D pairs, which are $1 \to 12$, $19 \to 9$, $23 \to 20$, $6 \to 33$, and $28 \to 28$, with demands $D_{1,12} = 800$, $D_{19,9} = 800$, $D_{23,30} = 1200$, $D_{6,33} = 1200$ and $D_{38,28} = 1200$. In reality, we

could reasonably expect there to be a demand between each node in the network.

We calculate the travel time in hours, with capacity given as the number of vehicles per hour. The users value of time is set to be £10 per hour, which is a rounded down figure of the market price value of time per vehicle for the average car over the course of the whole week in the UK in 2010, reported in the transport analysis guidance data book by the Department for Transport [51]. The cost function is calculated in time units, and the toll variables are calculated in pounds. The length of the roads of the network are rounded figures of the length of the London road network, and are given in km. The distance tolls are therefore calculated in pounds per km. This can easily be converted to give a cost per mile. The data for the free-flow travel time, capacity, and lengths for each arc are detailed in Table B.1 in the appendix.

We consider all three toll schemes outlined in Section 4.4.1. The first toll scheme considers distance tolls, given as a charging rate per km. We charge two different rates. The first rate is charged to travel on the links within the outer cordon, given by links 41-130, and the second rate is charged to travel on the links within the inner cordon, given by 131-154. The second toll scheme considers fixed cordon prices. This is implemented by charging a fixed toll to enter inside a cordon ring, but not to leave. We therefore have two fixed toll rates. The first fixed toll rate is charged to enter the outer cordon, given by using links 41-62, and the second fixed toll rate is charged to enter the inner cordon, given by using links 131-138. Lastly, we consider the toll scheme where you can charge fixed tolls on a subset of links. We take these links to be the same as the second tolling scheme, where we assume we can charge different rates on the links entering into each cordon zone. In total this tolling scheme has 30 different toll variables. For all tolling schemes we give the following bounds to all toll variables, $0 \leq \tau \leq 100$.

Results and Analysis

We tested the algorithm on this network example using the first two objective functions: to maximise profit, and to minimise total travel time. We used an initial trust-region radius of $\Delta_0 = 100$, and the initial starting points for all upper level variables were set to $\tau = 5$. We solved each instance to a stationarity condition tolerance level of $\theta = 1e-6$. For the penalty parameter, we used a value of $\gamma = 100$ for all instances except for the distance cordon toll scheme with an upper objective function to minimise total travel time, where it was set to $\gamma = 10$. This is because the algorithm struggled to solve the problem in a reasonable number of iterations, and as we have seen from the previous two examples, a smaller penalty parameter is preferred for this objective function.

The results for all three toll schemes, and both objective functions, are given in Table 4.6. The tolls for the fixed and distance cordon schemes are reported in Table 4.7, where τ_1 refers to the outer ring cordon and τ_2 refers to the inner ring cordon. The fixed cordon tolls are given in pounds, and the distance tolls are given in pounds per km.

Table 4.6: Solutions using the BLTR-L for network Example 3

Objective Function	**Tolling Scheme**	**F**	**f**	**Iter.**	**CPU** (s)
Max revenue	Distance cordon	6931.91	2725.40	54	228.78
	Fixed cordon	6301.24	2657.96	930	3990.71
	Fixed links	14040.83	3478.03	2419	14825.77
Min total travel time	Distance cordon	2744.89	3477.54	6718	18723.24
	Fixed cordon	2679.42	2544.96	1134	5789.54
	Fixed links	2705.56	2357.53	397	2086.12

Table 4.7: Toll schemes for cordon-based pricing

Objective Function	**Tolling Scheme**	τ_1	τ_2
Max revenue	Distance cordon	0.272	0.467
	Fixed cordon	1.840	0.449
Min total travel time	Distance cordon	0.758	0.928
	Fixed cordon	1.553	0.357

All runs were successful in finding a stationary point, where the stationarity conditions were satisfied to a tolerance of 1e-6, although most runs had a relatively slow rate of convergence. From Table 4.6 we see for the objective function to maximise revenue, the fixed link tolling scheme produced the maximum profit. The distance based cordon scheme produced a slightly better profit than the fixed cordon tolling scheme, whilst having a slightly higher lower level objective function. Therefore for this particular simplified example, a distance based toll scheme may be good solution if your aim is to maximise profit, without having to individually price all links. The distance toll scheme on this example was relatively quick at solving the problem, whilst the fixed link tolling scheme took far longer to converge.

For the objective function to minimise total user travel time, surprisingly the fixed cordon tolling scheme finds a better upper level objective function value than the fixed link tolling scheme, even though the solution to the fixed cordon scheme is a feasible solution to the fixed link tolling scheme. This could be an issue with the initial starting point for the toll variables in the fixed link tolling scheme. The distance cordon scheme produced the worst outcome for minimising total user travel time. For this upper level objective function the distance toll scheme took the longest to solve, whilst the fixed link tolling scheme took the

least amount of time.

Comparing toll prices in Table 4.7 we see that for a fixed cordon toll, a higher price is charged for the outer ring for both objectives. On the other hand, when we charge a rate per km, a higher rate is given for the inner ring. This could be due to the fact that there are less km of roads to travel on within the inner ring. When comparing the objective functions, it's found for the fixed tolling scheme smaller tolls are charged when the objective function is to minimise total travel time, whereas for the distance toll scheme larger tolls are charged.

Overall, the algorithm performs well in that it finds a stationary point for all runs tested when dealing with a larger network and with cordon schemes. However, for this example the algorithm has slow convergence for most of the instances. Furthermore, due to some of the runs providing curious results, for example finding a better solution for a fixed cordon scheme over the fixed link scheme, it would be suggested that more runs with different starting points and penalty parameters would need to be taken for a more reliable result.

4.6 Summary

In this chapter we investigated the London congestion pricing problem. We presented a bilevel model for the London road pricing problem that looks to represent the current problem TfL faces. The model looks at an updated tolling scheme that charges based on the type of user, the distance travelled, and the level of congestion, with a nonlinear distance toll function. The principal aim of the Mayor of London is to shift users off the road network to use greener and healthier transport. To reflect this, we suggested four objective functions: to maximise profit, to minimise total system travel time, to maximise the ratio of the first two objectives, and to minimise vehicle emissions. We also suggested a further constraint in the upper level problem which constrains demand by some desired level of demand.

We then simplified the bilevel model of the London congestion zone and presented a new approach to solving the road pricing bilevel model, using the trust-region algorithm introduced in Chapter 3. The bilevel model we solve has a deterministic traffic assignment problem with fixed demand for the lower level problem, and considers fixed links, fixed cordon and distance cordon tolls. The upper level problem considers the same objective functions presented for the London congestion pricing model. We showed how under the strong simplifications and assumptions of the model, including the requirement that the cost function is strictly increasing with respect to link flows, the optimal value function is continuously differentiable. Using this property, the exact penalty problem we solve in the trust-region algorithm becomes a continuously differentiable single level problem with a

nonconvex objective function and a convex constraint set. This problem can then be solved using standard continuous optimisation algorithms.

By using a first order model of the exact penalty problem in the trust-region method, we showed how the trust-region subproblem can be broken down into two smaller linear optimisation problems, by separating the upper and lower variables. The upper level toll variables form a simple optimisation problem with a linear objective function and upper and lower bounds, whilst the flow variables form a linear multicommodity flow problem. The former can be solved quickly and efficiently, whilst the latter can be computationally expensive to solve repeatedly when the network becomes large. Nevertheless there has been a number of efficient algorithms designed to solve these problems.

We provided numerical examples and results of the algorithm on three network examples. Overall, the results show the algorithm performs well on finding a stationary point of the road pricing transportation problem. We did however find that the convergence rate seemed in general relatively slow. This was largely due to the time it took for the solver to find a solution for the lower level problem and trust-region subproblem at each iteration. For the first example, the algorithm found the same solutions, or in one instance slightly better, when compared with the EDO algorithm implemented by Yan and Lam [125]. This is very promising, showing how by solving the exact penalty problem, a local solution to the road pricing bilevel model could be obtained. In the second example, we ran the algorithm 1000 times for different initial starting points. We found the algorithm converged to a number of different stationary points, which was dependent on the initial starting point. In the final example, we presented a simplified road network of the center of London containing two cordons. The algorithm found a stationary point on this larger network example for all the tolling schemes suggested in the model.

One of the most important considerations when implementing this algorithm is the choice of the penalty parameter. How sensitive the algorithm was on this choice depended on the upper level objective function chosen. It was always observed that as the penalty parameter was increased, the number of iterations and the CPU time for convergence greatly increased. However, it was found for some upper level objective functions a change in the penalty parameter did not affect the solution found, in which case a lower penalty parameter would be preferred, whilst for other upper level objective functions an increase in the penalty parameter gave an improvement on the upper level objective function value. In the latter case a higher penalty parameter would be preferred, but not too high that the convergence rate becomes too slow with limited or no improvement on the solution.

For future work on solving this problem a number of suggestions are made. Firstly, investigation into other algorithms that solve the continuous exact penalisation problem could be explored, with a comparison on their performance. Further work could also be done on how one may approximate the value of the penalty parameter. On advancing the implementation of the trust-region method for this problem, future research could investigate implementing a more efficient method to solve the trust-region subproblem that is a linear multicommodity flow problem, and the lower level traffic assignment problem, to see how this may improve CPU time of the algorithm.

Finally, the road pricing bilevel model that is solved using this method is a very simplistic model. Research into a more complicated but realistic model could be done where the conditions for the reformulation still hold. For example, these conditions should still hold for elastic demand. Furthermore, from the analysis of Chapter 3, we may not require the optimal value function to be continuously differentiable for the trust-region algorithm to converge. This could allow some flexibility in relaxing some of the simplifications of the lower level model, such as the assumptions that require uniqueness of a solution in the lower level problem.

Chapter 5

Conclusions and further work

5.1 Conclusions

The work in this thesis can be split into two parts. Firstly, we have explored an approach to finding local optimal solutions of a bilevel problem by solving an exact penalty problem, built from the optimal value reformulation, and proposed a novel trust-region method to solve it. The implementation and performance of the approach has been investigated via numerical experiments, with strong numerical results demonstrated. Secondly, an application of bilevel programming in transportation is explored. We presented a bilevel model for cordon pricing of the London congestion charge zone, suggesting an updated tolling scheme that charges a user based on the distance travelled and the level of congestion of a route. A simplified road pricing bilevel model is then solved, by applying the aforementioned trust-region algorithm. The performance of the proposed algorithm is assessed on three network examples. The main conclusions from each area are separated into the two following sections.

5.1.1 Optimal value function reformulation based algorithms

A common approach to solving bilevel problems is to first transform the problem into a single level problem. A great deal of algorithms do this via the KKT reformulation. However, although it is the most popular reformulation method, it is shown only to be equivalent to the bilevel problem when the lower level problem is convex and satisfies the Slater's constraint qualification, as shown by Dempe and Dutta [26]. On the other hand, the optimal value function reformulation of the bilevel problem is shown to be fully equivalent, first introduced by Outrata [94]. Unfortunately, this reformulation is difficult to solve due to the optimal value function, which is typically a nonsmooth, nonconvex function that usually cannot be

explicitly given. Algorithms developed to solve the optimal value reformulation typically involve approximating the value function, for example this has been done in relaxation schemes, smoothing methods, and metamodelling heuristics.

By introducing a regularity condition known as the partial calmness condition, Ye and Zhu [131] build an exact penalty problem from the optimal value reformulation. They show that if the partial calmness condition holds, there exists some penalty parameter such that a solution to the exact penalty problem is also one of the bilevel problem. Currently the literature on solving this exact penalty problem as a method for finding local solutions of bilevel problems is scarce. In this thesis, we focused our attention on solving this exact penalty problem, to investigate how this problem could be utilised to solve bilevel problems.

By studying the properties of the optimal value function, we proposed a novel trust-region method to solve the exact penalty problem, detailed in Section 3.3.2. To our knowledge, a trust-region approach to solving this problem has not been explored before. The idea behind the algorithm is to form an approximation of the problematic optimal value function, and restrict the model into a region we trust. The approximation of the optimal value function is a first order local model about the iteration point, which uses an arbitrary subgradient to approximate the Clarke generalised gradient. Three variants of the model for the exact penalty function used in the trust-region subproblem were suggested. The first model only approximates the optimal value function in the penalty function, denoted as algorithm BLTR. The second and third models approximate the whole penalty function, where the second model uses a quadratic model for the smooth parts, denoted as algorithm BLTR-Q, and the third model uses a linear model for the smooth parts, denoted as algorithm BLTR-L. Convergence of the trust-region algorithm to a stationary point of the exact penalty problem is shown under strong assumptions on the bilevel problem.

We presented extensive numerical experiments to test the performance of the trust-region algorithm, implementing all three of the suggested models for the trust-region subproblem on 124 nonlinear bilevel problems taken from the BOLIB test set [139]. To analyse the results, we compared them with known solutions from the literature. The trust-region algorithm performed extremely well with all models, solving a large proportion of problems. Considering the difficulty in solving nonlinear bilevel problems, the results of the bilevel trust-region algorithm are very promising. In comparison with a number of other bilevel algorithms tested, the trust-region method using BLTR and BLTR-Q performed the best at solving the highest number of problems. We found BLTR, which uses the model that approximates only the optimal value function, performed the best overall, solving 92% of all bilevel problems. The algorithm BLTR-L did not perform as well, although it still solved a

high proportion of 79% of problems. That being said, for some problems the trust-region algorithm with all models did have some difficulties. These included problems where the solution set of the lower level problem were nonsingular, and problems where there were issues with finding local solutions of the lower level problem as opposed to the global solution. Nevertheless, due to the very strong numerical results, we believe further research on the convergence theory of this trust-region algorithm should be investigated.

In addition to the trust-region algorithm, we showed how, under strong assumptions of the bilevel problem, the exact penalty problem built from the optimal value reformulation becomes a DC problem. The resulting DC problem contains a smooth convex part and a nonsmooth concave part. A handful of algorithms have been developed to solve these problems, including a heuristic known as the convex-concave procedure (CCP) developed by Yuille and Rangarajan [136]. We demonstrated how this algorithm can be applied to the bilevel problem, and implemented the algorithm on 40 bilevel problems with fully convex lower level problems. The CCP heuristic also performed very well, solving 94.29% of the bilevel problems tested. It performed better than BLTR-L but not as well as the other two trust-region algorithms, both in terms of number of problems solved and CPU time.

Overall, both the trust-region algorithm and the CCP algorithm presented showed strong numerical results, validating the method of solving the exact penalty problem as an approach to finding local solutions of the associated bilevel problem. That being said, the solutions were found to be sensitive to the choice of penalty parameter. As there is no known method to find the best penalty parameter, the algorithms were required to be tested on a range of penalty parameter values, and the best solution chosen from that. This can make finding a solution for the bilevel problem slow. Nonetheless, the results show good promise in the approach for nonlinear bilevel problems. Further investigation on the convergence of the trust-region algorithm under weaker assumptions, and suggestions on modifying and extending the algorithms, are discussed in the following section on further work.

5.1.2 An application of bilevel optimisation in transportation

Road pricing is a method implemented by governments to charge for the use of congested roads to help alleviate congestion. It is a common application of bilevel programming as, due to the hierarchical structure of the problem, it naturally forms as one. A large amount of research has gone into formulating the problem under varying simplifications, and developing algorithms to solve them. Solution methods usually depend on the assumptions made on the model, but commonly heuristics are applied to solve them.

We investigated a case study of road pricing on the London congestion zone charge. This

is a fixed cordon scheme implemented in the center of London, governed by Transport for London (TfL). Although successful on initial implementation, the fixed cordon scheme is now outdated, and the Mayor of London is looking to develop a new road charging system that can charge for distance, time, emissions and other factors [77]. We formulated a model that looks to achieve the objectives set out by the current Mayor of London. These objectives are: to reduce overall demand of the road network, to reduce congestion to help increase productivity from wasted time, to reduce pollution, and to increase the efficiency of public transport.

We proposed a multi-cordon tolling scheme that charges for the distance travelled, and the level of congestion on the route taken. The lower level problem is formed as a mulit-commodity, deterministic, capacitated traffic assignment model, with a nonlinear time function and elastic demand. The upper level problem encompasses the objectives that are currently being pursued by TfL, with four upper level objective functions suggested: to maximise revenue, to minimise total system travel time, to maximise the ratio of the first two, and to minimise vehicle emissions. These encapsulate all objectives except the principal aim, which is to reduce overall demand of the network. As demand in the model is elastic, a constraint on the demand in the upper level problem is proposed, which allows the ability to set an upper bound on demand. This can be set to the level of demand TfL wants to achieve in the London road network, to reduce congestion to an acceptable level. To our knowledge, an algorithm to solve a road pricing bilevel model like this formulation has not yet been developed.

By simplifying the road pricing model for the London congestion zone charge, we presented a novel approach to solving the problem, by showing how the trust-region algorithm can be applied. The model solved is a simple road pricing problem commonly seen in the literature. The lower level problem is formed as a deterministic, fixed demand traffic assignment problem. For the upper level problem, we tested the algorithm on all four of the upper level objective functions suggested in the London congestion zone model. Furthermore, we considered three tolling schemes: fixed link tolls, fixed cordon tolls and distance cordon tolls. It is shown that, by assuming that the cost function is strictly increasing with respect to link flows, the optimal value function is continuously differentiable. This enables us to apply the trust-region algorithm proposed for bilevel problems in Section 3.3.2. We note that the exact penalty problem the algorithm looks to solve is a continuously differentiable single level problem, with a nonconvex objective function and a convex constraint set. By applying the trust-region method with the model that forms a linear model of the whole penalty function, we showed how the trust-region subproblem can be broken down into two, smaller,

linear optimisation problems, by separating the upper and lower level variables. The toll variables form a simple linear optimisation problem, whilst the flow variables form a linear multicommodity flow problem. Both problems can be solved efficiently, although the latter can be computationally expensive to solve many times.

The performance of the algorithm was tested on three network examples, testing all four upper level objective functions and three tolling schemes. Overall, the algorithm performed very well in regards to converging to a stationary point of the problem, however the convergence rate was relatively slow. In the first example, the algorithm was compared to the EDO algorithm implemented by Yan and Lam [125] on a small network with fixed link tolls. Our algorithm found the same solutions, and in one instance slightly better. In the second example, we tested the algorithm on a small network with fixed links using 1000 randomly generated starting point, to test the robustness to different starting points. The algorithm successfully converged to a stationary point for all 1000 starting points. The final example was a medium sized network example formulated to be a simplified map of London, where all three tolling schemes were tested. For this problem, the algorithm successfully converged to a stationary point of the problem for all tests run, however the convergence rate was slow.

Finally, as in the previous chapter, it was found that the quality of the solution could be sensitive to the choice of penalty parameter, however, interestingly it was found for this road pricing problem, the sensitivity to the penalty parameter depended on what upper level objective function was used. It was always observed that as the penalty parameter increased, the number of iterations and CPU time for convergence increased. However, for some upper level objective functions, a change in the penalty parameter did not affect the solution found, in which case a lower penalty parameter would be preferred, whilst for other upper level objective functions, an increase in the penalty parameter gave an improvement on the upper level objective function value.

5.2 Suggestions for further work

The work presented in this thesis has highlighted a number of areas for potential further work. The suggestions for further work are separated into the two main areas of focus in the following two sections.

5.2.1 Optimal value function reformulation based algorithms

- Although the numerical results for the trust-region algorithm on nonlinear bilevel programs are very promising, investigation now needs to look into strengthening

the convergence analysis. First, investigation into whether the current convergence analysis can be extended to all the suggested models should be done. Second, further investigation should look at relaxing some of the strong assumptions on the bilevel problem. This may require the algorithm to be modified or extended to be able to handle more complex functions. In particular, research into a better approximation of the Clarke generalised gradient of the optimal value function may be needed. For the current work on convergence theory on trust-region algorithms to nonsmooth functions, the majority of research require that the function is regular or subhomogenous, and that the whole generalised gradient is used, for example this is shown in [40] and [98]. Under the LICQ assumption, the optimal value function is known to be a regular function, however, it is difficult and computationally expensive to calculate the generalised gradient at every iteration. Therefore, further work could investigate whether this trust-region method works when the generalised gradient is only approximated. One approach to approximate it could be to use a bundle approach, where we enrich the generalised gradient with new subgradients if the model is deemed inadequate. Bundle trust-region methods have been explored, see for instance the algorithm by Schramm and Zowe [103], however, they usually require the function to be semismooth, which the optimal value function in general is not. Alternatively, exploration into using the Goldstein ϵ-subdifferential could be investigated, as is done in [57] for unconstrained nonsmooth functions.

- Research regarding the partial calmness condition could be explored further. As we have seen, it is a fairly restrictive property on the bilevel problem, and verifying that the condition holds can be difficult. Further work could investigate cases of problems where the condition holds, as is done in [55] and [34]. That being said, although validating the condition holds is difficult, the work in this thesis shows how the resulting exact penalty problem could be solved to find stationary points of the bilevel problem. The current research that utilise this problem is extremely limited, and so investigation on solution approaches that look to solve the exact penalty problem should be explored. A difficulty with solving this problem is the choice of penalty parameter, which needs careful consideration, as the solution was found to be sensitive to this choice. Future research could therefore investigate how to best approximate the penalty parameter for a given problem.

- Improvements on the implementation and analysis of the algorithms should be looked at. The algorithms were implemented in MATLAB R2019a, and the MATLAB built-in solver *fmincon* was used to solve the lower level problem for a given x at each iteration.

However, when the lower level problem was nonconvex, we found the solver would sometimes find a local, as opposed to global, solution. Research could be done to look at implementing a more powerful solver for nonlinear problems, to improve both the speed and solutions found, for example the WORHP solver implemented on the road pricing problem. Alternatively, research could look at applying the algorithm to a specific bilevel problem, so an efficient solver could be chosen that is tailored to the problem. The trust-region algorithm was tested on 124 bilevel problems from the literature, presented in the BOLIB libary [139]. Although the test set was large, the size of the problems were relatively small. Further analysis of the algorithms presented in this work could explore how the algorithms solve bilevel problems of a larger size. Furthermore, a comparison of the algorithms to a larger range of bilevel optimisation solvers would be beneficial to inspect the performance of the algorithm compared with others. Finally, a more in-depth analysis could be done to examine the sensitivity of a solution on the choice of starting point, or any other parameters of the algorithm.

- We showed how, under strong assumptions, the exact penalty problem we look to solve becomes a DC problem. This problem has been studied extensively. Further work could examine applying other known DC algorithms to solving this problem, as is done in [130], and their performances could be compared. For a discussion on DC algorithms we refer the reader to the following overview paper on the subject [56].

5.2.2 An application of bilevel optimisation in transportation

- In this thesis we presented a complex road pricing bilevel model of the London congestion pricing problem. This model is complicated and difficult to solve, and as such we solved a simplified version of the model. Future work could look at how to solve the complex model presented. The nonlinear toll function means the tolls are not link additive. To solve problems of this form, approaches usually transform the model into one that is link additive, commonly by using dummy links. This approach is taken in [74] and [85]. Lawphongpanich and Yin [68] solve a road pricing model with nonlinear tolls, elastic demand and homogeneous users. They do this by using a piecewise linear toll function and a coordinate search model. However, a model with multicommodity users complicates the model further, as uniqueness of link flows is not guaranteed. Models that consider multicommodity users nearly always only consider linear cost functions, for example in [14] and [65]. Furthermore, the proposed model contains capacity constraints of links in the lower level problem, and demand constraints in the upper level problem, making both the lower and upper level problem more difficult to solve. Research that consider more complicated road pricing problems

usually turn to heuristics to solve them.

- The simplified road pricing problem that we solve is a commonly seen form in the literature. We show how, by solving a continuous single level exact penalisation problem, local solutions to the road pricing problem can be obtained. As opposed to the trust-region algorithm, other standard continuous optimisation algorithms could be applied to solve the exact penalty problem, and their performances could be compared. The quality of the solution may however be dependent on the choice of penalty parameter. Therefore, similar to the discussion of further work in the previous section, exploration into the approximation of the penalty parameter for this problem could be beneficial.

- Although the algorithm converged to stationary point for all test runs, the convergence rate was slow. Part of the reason for this was due to the time it took to solve the lower level problem and trust-region subproblem at each iteration. Improvements on the implementation of the algorithm could be done, to try to improve the CPU time of convergence. We showed how the trust-region subproblem can be formed into two linear optimisation problems, by separating the upper and lower level variables. The linear problem containing the flow variables is a linear multicommodity flow problem. In our analysis, we solve this problem using the MATLAB built-in solver *fmincon*, which uses an interior-point algorithm. Investigation could look at implementing algorithms that have been specifically designed to solve linear multicommodity flow problems, such as Lagrangian relaxation algorithms, column generation algorithms, or Dantzig-Wolfe decomposition approaches [1]. We also solve the lower level traffic assignment problem for a given toll variable at each iteration using the nonlinear optimisation solver by WORHP. Again, investigation could look at implementing an algorithm designed to solve this problem, such as the Frank-Wolfe algorithm, decomposition algorithms, or column generation algorithms. A full description on methods to solve this problem can be found in the book by Patricksson [95].

- We applied the trust-region algorithm to a simplistic model of the road pricing problem. This was done by showing that the optimal value function is differentiable, and solving a continuous exact penalty problem. Further work could explore whether this method still converges if assumptions on the bilevel model are relaxed. For example, the conditions should still hold when the demand is elastic. Additionally, from the research and algorithm proposed in Chapter 3, we do not necessary require differentiability of the optimal value function for the algorithm to converge, and so some of the stronger assumptions on the bilevel model that ensure differentiability could be relaxed.

Part I

Appendix

Appendix A

Detailed numerical results

Table A.1 presents the detailed results of the numerical experiments on the trust-region algorithm discussed in Section 3.5. The table details for each problem in the BOLIB library [139], the status of the best known solution as discussed in Section 3.5.2, and the value of the upper level objective function for the known solution in column F^*. If there any local solutions found, we also report these values in column F^* with the status marked with an asterisk. For example a local optimal solution would be marked with status O*. The solution found from each algorithm is presented for each penalty parameter tested in the set $\Gamma = \{2^{-1}, 2^0, 2^1, \ldots, 2^7\}$, and each starting point. The starting points are labelled as starting point 1 and 2 in the column SP.

As before, we refer to the trust-region algorithm that uses a model for trust-region subproblem that only builds an approximation of the optimal value function as BLTR, the algorithm that uses a model that approximates the whole penalty obective function using quadratic models for the smooth parts as BLTR-Q, and the algorithm that forms a linear model of the whole penalty function as BLTR-L.

Table A.1: Best solutions found from the trust-region algorithm for all penalty parameters and starting points

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
1	AiyoshiShimizu1984Ex2	O	5.00	BLTR	1	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
		O*	0.00		2	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
		O*	10.00	BLTR-Q	1	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
					2	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
				BLTR-L	1	8.38	31.27	30.58	30.27	30.11	30.07	29.98	29.98	29.98
					2	22.55	24.94	24.28	23.20	23.92	24.21	23.92	23.91	23.91
2	AllendeStill2013	O	1.00	BLTR	1	1.44	1.28	1.11	1.25	1.38	1.13	1.06	1.19	2.50
					2	1.44	1.24	1.12	1.39	1.39	1.39	1.31	1.00	1.52

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
				BLTR-Q	1	1.44	1.24	1.10	1.22	1.07	1.06	1.07	1.05	2.02
					2	1.44	1.25	1.39	1.13	1.15	1.07	1.01	1.20	1.32
				BLTR-L	1	2.13	1.36	1.36	1.16	1.43	1.40	1.42	1.53	1.53
					2	2.11	1.39	1.39	1.39	1.26	1.16	1.19	1.20	1.25
3	AnEtal2009	O	2251.6	BLTR	1	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
					2	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
				BLTR-Q	1	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
					2	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
				BLTR-L	1	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
					2	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6	2251.6
4	Bard1988Ex1	O	17.00	BLTR	1	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
		O*	25.00		2	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
				BLTR-Q	1	49.31	57.37	17.00	17.00	17.00	17.00	17.00	17.00	17.00
					2	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
				BLTR-L	1	49.27	53.92	17.00	17.00	17.00	17.00	17.00	17.00	17.00
					2	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
5	Bard1988Ex2	O	-6600.0	BLTR	1	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0
					2	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0
				BLTR-Q	1	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6599.9
					2	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6599.6
				BLTR-L	1	-6570.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0
					2	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6600.0	-6599.9	-6599.9	-6598.7
6	Bard1988Ex3	O	-12.68	BLTR	1	-12.68	-12.68	-12.68	-12.68	-12.68	-12.62	-12.57	-12.65	-12.56
					2	-12.68	-12.68	-12.68	-12.68	-12.68	-12.68	-12.67	-12.67	-12.67
				BLTR-Q	1	-12.68	-12.68	-12.68	-12.68	-12.68	-12.68	-12.67	-12.65	-12.59
					2	-12.68	-12.68	-12.68	-12.68	-12.68	-12.68	-12.68	-12.68	-12.67
				BLTR-L	1	-12.68	-12.67	-12.68	-12.68	-12.68	-12.66	-12.67	-12.61	-12.63
					2	-12.68	-12.67	-12.68	-12.68	-12.68	-12.68	-12.60	-12.59	-12.60
7	Bard1991Ex1	O	2.00	BLTR	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
				BLTR-Q	1	2.00	2.00	2.00	2.42	3.84	3.96	3.96	3.99	3.99
					2	2.50	2.00	4.38	4.84	4.84	4.96	4.96	4.99	4.99
				BLTR-L	1	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
					2	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
8	BardBook1998	O	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-L	1	0.00	0.02	0.02	0.02	0.01	0.01	0.00	0.00	0.00
					2	0.01	0.27	0.12	0.03	0.01	0.01	0.01	0.00	0.00
9	CalamaiVicente1994a	O	0.00	BLTR	1	0.00	0.00	0.05	0.05	0.05	0.05	0.05	0.01	0.31
					2	0.00	0.05	0.00	0.01	0.05	0.05	0.05	0.01	0.31
				BLTR-Q	1	0.00	0.00	0.00	0.02	0.05	0.05	0.05	0.01	0.31
					2	0.00	0.00	0.01	0.04	0.05	0.05	0.05	0.01	0.31
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	CalamaiVicente1994b	O	0.31	BLTR	1	0.38	0.34	0.32	0.32	0.31	0.31	0.33	0.35	1.13
					2	0.38	0.56	0.32	0.32	0.31	0.51	0.53	1.32	1.34
				BLTR-Q	1	0.38	0.34	0.32	0.32	0.32	0.33	0.35	0.35	1.13
					2	0.38	0.34	0.32	0.32	0.32	0.33	0.35	0.35	1.13
				BLTR-L	1	0.55	0.58	0.75	0.57	0.76	0.73	0.79	0.75	0.75
					2	0.50	0.35	0.59	0.56	0.53	0.36	0.66	0.52	0.51
11	CalamaiVicente1994c	O	0.31	BLTR	1	0.38	0.56	1.06	1.06	1.06	1.22	1.23	1.24	1.25
					2	0.38	0.34	0.32	0.32	0.31	0.31	0.53	0.42	0.73
				BLTR-Q	1	0.69	0.62	0.60	0.55	5.98	9.96	10.23	10.37	7.85
					2	0.46	0.49	0.56	0.59	0.57	0.56	0.56	0.56	0.56
				BLTR-L	1	0.56	0.46	0.41	0.40	0.37	109.1	113.8	115.6	271.4
					2	0.52	0.53	0.36	0.34	0.76	0.73	1.09	79.60	79.60
12	CalveteGale1999P1	O	-29.20	BLTR	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-23.00
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20	-29.20	-29.20
				BLTR-Q	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-23.00
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20	-29.20	-29.20

	Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
				BLTR-L	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-23.00
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20	-29.20	-29.20
13	ClarkWesterberg1990a	O	5.00	BLTR	1	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
					2	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
				BLTR-Q	1	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
					2	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.02	5.00
				BLTR-L	1	9.00	5.00	5.00	5.00	5.00	5.00	9.10	9.61	9.04
					2	9.06	9.00	9.00	9.06	9.47	9.00	9.59	9.17	9.56
14	Colson2002BIPA1	O	250.0	BLTR	1	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0
					2	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0
				BLTR-Q	1	249.8	249.8	249.9	249.8	249.8	249.8	249.9	249.9	249.9
					2	249.8	249.8	249.8	249.8	249.9	249.9	249.8	249.8	249.8
				BLTR-L	1	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0
					2	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0
15	Colson2002BIPA2	K	17.00	BLTR	1	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
					2	49.31	43.03	30.32	25.00	25.00	25.00	25.00	25.00	25.00
				BLTR-Q	1	49.31	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
					2	49.31	43.03	30.32	25.00	25.00	25.00	25.00	25.00	25.00
				BLTR-L	1	49.27	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
					2	49.29	43.03	31.02	25.00	25.00	25.00	25.00	25.00	25.00
16	Colson2002BIPA3	K	2.00	BLTR	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
				BLTR-Q	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
				BLTR-L	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
17	Colson2002BIPA4	K	88.79	BLTR	1	89.98	88.93	88.79	88.79	88.79	88.79	88.79	88.79	89.55
					2	89.98	88.93	88.79	88.79	88.79	88.79	88.79	88.79	88.79
				BLTR-Q	1	89.98	88.93	88.79	88.79	88.79	88.79	124.79	124.79	89.53
					2	89.98	88.93	88.79	88.79	88.79	88.79	88.79	88.79	88.79
				BLTR-L	1	90.02	89.02	88.79	88.84	88.79	88.79	88.79	88.79	88.79
					2	89.98	88.93	88.83	89.02	88.79	88.79	88.79	88.79	88.79
18	Colson2002BIPA5	K	2.75	BLTR	1	3.41	3.31	3.26	3.25	3.25	3.25	3.25	3.25	3.23
					2	3.41	2.75	2.75	2.75	2.75	2.75	2.75	2.75	2.75
				BLTR-Q	1	3.40	3.30	3.26	3.25	3.25	3.25	3.25	2.75	3.23
					2	3.40	2.75	2.75	2.75	2.75	2.75	2.75	2.75	2.75
				BLTR-L	1	3.25	3.25	3.26	3.28	3.25	3.25	30.64	30.64	30.64
					2	3.41	3.32	3.28	3.25	3.25	2.75	2.75	2.75	2.75
19	Dempe1992a	U		BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	Dempe1992b	O	31.25	BLTR	1	33.98	33.49	32.41	31.25	31.25	31.25	31.25	31.25	31.25
					2	33.98	33.49	32.41	31.25	31.25	31.25	31.25	31.25	31.25
				BLTR-Q	1	33.98	33.49	32.41	31.25	31.25	31.25	31.25	31.25	31.25
					2	33.99	33.49	32.41	31.25	31.25	31.25	31.25	31.25	31.25
				BLTR-L	1	33.78	33.57	32.83	31.25	28.36	31.25	31.25	31.25	31.25
					2	33.45	35.09	32.58	31.25	28.35	31.25	31.25	31.25	31.25
21	DempeDutta2012Ex24	O	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00
22	DempeDutta2012Ex31	O	-1.00	BLTR	1	-1.99	-1.98	-1.87	-1.80	-1.92	-1.88	-1.86	-1.57	-0.98
					2	-1.99	-1.98	-1.85	-1.81	-1.90	-1.88	-1.86	-1.77	-1.45
				BLTR-Q	1	-1.99	-1.98	-1.85	-1.79	-1.90	-1.88	-1.86	-1.52	-0.99
					2	-1.99	-1.98	-1.97	-1.88	-1.90	-1.88	-1.86	-1.74	-1.40
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	-1.99	-1.98	-1.97	-1.81	-1.80	-1.78	-1.74	-1.70	-1.46
23	DempeEtal2012	O	-1.00	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	0.00	0.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-L	1	-1.00	-1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
24	DempeFranke2011Ex41	O	5.00	BLTR	1	7.00	7.98	6.00	6.00	6.00	7.99	5.00	5.00	5.00
					2	7.00	5.04	6.00	5.00	7.00	7.00	7.00	7.00	7.00
				BLTR-Q	1	7.00	5.00	6.00	6.00	6.00	6.00	6.00	6.00	6.00
					2	7.00	5.03	6.00	6.00	6.00	5.01	7.00	7.00	7.00
				BLTR-L	1	7.05	8.00	5.99	5.02	5.09	6.96	5.00	5.00	5.00
					2	7.00	5.01	6.00	5.15	5.13	5.03	7.99	8.00	8.00
25	DempeFranke2011Ex42	O*	2.13	BLTR	1	3.25	3.24	3.24	2.12	2.13	2.13	2.13	2.13	2.13
		O*	4.00		2	3.24	3.25	3.24	4.99	4.99	4.99	4.99	4.99	5.00
		O*	3.00	BLTR-Q	1	3.24	3.24	3.25	2.15	4.00	4.99	4.99	4.99	4.99
					2	4.60	3.25	4.97	5.00	3.31	3.41	3.47	3.49	3.49
				BLTR-L	1	3.28	3.46	3.24	4.99	4.13	2.16	4.22	2.40	3.13
					2	3.27	3.25	4.54	4.99	3.03	4.99	3.03	5.00	3.03
26	DempeFranke2014Ex38	O	-1.00	BLTR	1	-1.00	-1.00	-1.00	0.99	0.99	0.99	1.00	-1.00	-1.00
					2	-1.00	-1.00	-0.97	-1.00	-1.00	-1.00	0.99	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-0.83	1.00	1.00	1.25	1.25	1.25	1.25
					2	-1.00	-1.00	0.94	1.25	1.25	1.25	1.25	1.25	1.25
				BLTR-L	1	-1.00	-1.00	-1.00	0.00	-0.96	0.75	-1.00	0.99	-1.00
					2	-1.00	-1.00	-0.99	1.00	1.00	0.99	1.00	0.99	-0.98
27	DempeLohse2011Ex31a	K	-6.00	BLTR	1	-5.75	0.25	-5.75	-5.75	-5.75	-5.75	0.23	-5.50	-5.50
					2	0.12	-5.75	-5.75	-5.75	-5.75	-5.75	-5.75	-5.75	0.23
				BLTR-Q	1	-5.75	-5.50	-5.75	0.70	-1.51	-5.43	0.43	0.43	0.42
					2	0.24	-5.75	-5.75	-5.75	-5.75	-5.75	0.23	-5.48	0.23
				BLTR-L	1	0.22	0.45	-5.44	0.48	-5.41	0.49	0.45	0.47	0.50
					2	0.24	0.22	-5.45	0.48	-3.55	0.38	0.46	0.48	0.49
28	DempeLohse2011Ex31b	O	-12.00	BLTR	1	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00
					2	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-11.58	-12.00	-12.00
				BLTR-Q	1	-12.00	-12.00	-11.43	-5.50	-5.51	-5.50	-5.50	-5.50	-5.52
					2	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	0.22
				BLTR-L	1	-11.98	-12.00	-12.00	-11.13	-11.13	-5.50	-5.50	-5.50	-5.50
					2	-12.00	-12.00	-12.00	-11.99	-11.19	-11.49	0.49	0.49	0.49
29	DeSilva1978	O	-1.00	BLTR	1	-0.56	-0.75	-0.89	-0.96	-0.99	-1.00	-1.00	-0.98	-0.80
					2	-0.56	-0.75	-0.89	-0.96	-0.99	-1.00	-1.00	-1.00	-1.01
				BLTR-Q	1	-0.56	-0.75	-0.89	-0.96	-0.99	-1.00	-1.00	-0.98	-0.80
					2	-0.56	-0.75	-0.89	-0.96	-0.99	-1.00	-1.00	-0.98	-0.80
				BLTR-L	1	-0.07	-0.33	-0.81	99234	333.95	-0.08	-0.08	-0.08	-0.08
					2	-0.09	-0.49	-0.77	-0.72	-0.96	-0.97	-1.00	-1.01	-1.00
30	FalkLiu1995	O	-2.20	BLTR	1	-1.56	-2.00	-2.16	-2.22	-2.24	-2.25	-2.25	-2.25	-2.27
					2	-1.56	-2.00	-2.16	-2.22	-2.24	-2.25	-2.25	-2.25	-1.75
				BLTR-Q	1	-1.56	-2.00	-2.16	-2.22	-2.24	-2.25	-2.25	-2.25	-2.27
					2	-1.56	-2.00	-2.16	-2.22	-2.24	-2.25	-2.25	-2.25	-1.75
				BLTR-L	1	-1.20	-1.20	-1.94	-1.97	-2.06	-2.04	-2.00	-2.00	-2.00
					2	-1.30	-1.41	-2.08	-2.11	-2.14	-2.24	-2.25	-2.09	-2.05
31	FloudasEtal2013	O	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	8.25	6.63	5.81	5.41	5.20	5.10	5.05	5.03	5.01
				BLTR-L	1	0.00	0.00	0.00	0.00	38.02	38.00	37.98	37.98	37.90
					2	28.50	26.73	26.16	25.99	25.82	25.78	25.82	25.69	25.73
32	FloudasZlobec1998	O	1.00	BLTR	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
				BLTR-Q	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
				BLTR-L	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
33	GumusFloudas2001Ex1	O	2250.0	BLTR	1	2370.1	2347.9	2332.6	2323.1	2316.5	2312.6	2309.8	2308.0	2307.2

	Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
		O*	2304		2	2370.1	2347.9	2332.6	2323.1	2316.5	2312.6	2309.8	2308.0	2307.2
				BLTR-Q	1	2370.1	2347.9	2332.6	2323.1	2316.5	2312.6	2309.8	2308.0	2307.2
					2	2370.1	2347.9	2332.6	2323.1	2316.5	2312.6	2309.8	2308.0	2307.2
				BLTR-L	1	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0
					2	2368.7	2345.4	2331.3	2324.1	2321.2	2310.9	2309.5	2307.3	2306.3
34	GumusFloudas2001Ex3	O	-29.20	BLTR	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-23.00
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20	-13.00
				BLTR-Q	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-23.00
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20	-13.00
				BLTR-L	1	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-29.20
					2	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-20.00	-16.00
35	GumusFloudas2001Ex4	O	9.00	BLTR	1	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
					2	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
				BLTR-Q	1	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
					2	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
				BLTR-L	1	9.25	9.03	9.00	9.00	9.00	9.07	9.05	9.07	9.07
					2	9.25	9.03	9.00	9.00	9.00	9.07	9.05	9.07	9.07
36	GumusFloudas2001Ex5	O	0.19	BLTR	1	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
					2	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
				BLTR-Q	1	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
					2	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
				BLTR-L	1	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
					2	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.19
37	HatzEtal2013	O	0.00	BLTR	1	1.00	0.50	0.25	0.13	0.06	0.03	0.02	-0.01	0.00
					2	1.00	0.50	0.25	0.13	0.06	0.03	0.02	0.01	0.00
				BLTR-Q	1	1.00	0.50	0.25	0.13	0.06	0.03	0.02	0.01	-0.01
					2	1.00	0.50	0.25	0.13	0.06	0.03	0.02	0.01	0.00
				BLTR-L	1	1.00	0.50	0.30	1.06	1.00	1.00	0.97	0.98	0.92
					2	0.99	0.50	1.16	1.08	1.06	1.02	1.02	0.99	0.99
38	HendersonQuandt1958	K	-3266.7	BLTR	1	-3243.6	-3200.0	-3236.6	-3260.4	-3262.9	-3266.3	-3266.5	-3266.5	-3266.5
					2	-3243.6	-3200.0	-3239.7	-3260.5	-3264.6	-3266.0	-3266.4	-3266.6	-3266.7
				BLTR-Q	1	-3243.6	-3200.0	-3239.7	-3259.3	-3265.0	-3265.5	-3266.5	-3266.6	-3266.3
					2	-3243.6	-3200.0	-3239.7	-3262.2	-3263.6	-3266.6	-3266.6	-3266.6	-3266.7
				BLTR-L	1	-3243.6	-3200.1	-3246.0	-3263.6	-3266.6	-3265.5	-3262.8	-3255.8	-3223.7
					2	-3243.7	-3199.9	-3249.6	-3263.6	-3266.5	-3265.3	-3263.0	-3253.7	-3228.2
39	HenrionSurowiec2011	O	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-L	1	0.00	0.00	0.06	0.25	0.35	0.11	0.88	0.01	0.10
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	IshizukaAiyoshi1992a	O	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
41	KleniatiAdjiman2014Ex3	O	-1.00	BLTR	1	-1.33	-1.33	-1.33	-1.33	-1.33	-0.98	-0.98	-0.90	-0.90
					2	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.33	-1.34
				BLTR-Q	1	-1.33	-1.33	-0.55	-0.99	-1.00	-1.00	-0.99	-0.99	-1.00
					2	-1.33	-1.33	-1.33	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-L	1	-1.33	-1.33	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.33	-1.33	-1.08	0.00	0.00	0.00	0.00	0.00	0.00
42	KleniatiAdjiman2014Ex4	K	-10.00	BLTR	1	-9.00	-9.00	-9.99	-8.11	-10.00	-8.01	-8.00	-2.73	-10.00
					2	-9.00	-9.00	-9.99	-8.11	-10.00	-8.01	-8.00	-2.73	-10.00
				BLTR-Q	1	-10.00	-10.00	-10.00	-10.00	-8.27	-6.48	-7.54	-2.28	-9.00
					2	-9.00	-9.00	-9.99	-8.11	-8.03	-8.00	-7.83	-1.90	-4.21
				BLTR-L	1	-10.00	-10.00	-10.00	-10.00	-10.00	-3.61	-5.25	-2.15	-4.15
					2	-9.00	-9.00	-9.00	-3.62	-1.34	-2.49	-1.15	-2.16	-9.00
43	LamparSagrat2017Ex23	O	-1.00	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-L	1	-1.00	-1.00	-1.00	-1.00	-1.00	-0.31	-0.31	-0.31	-0.31
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-0.72	-0.72	-0.72	-0.72
44	LamparSagrat2017Ex31	O	1.00	BLTR	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-Q	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-L	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
45	LamparSagrat2017Ex32	O	0.50	BLTR	1	0.63	0.56	0.52	0.51	0.50	0.50	0.50	0.49	0.50
					2	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49	0.50
				BLTR-Q	1	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49	0.50
					2	0.63	0.56	0.52	0.51	0.50	0.50	0.50	0.49	0.50
				BLTR-L	1	0.72	0.63	0.56	0.55	0.52	0.52	0.52	0.52	0.52
					2	0.71	0.63	0.82	0.82	0.82	0.82	0.81	0.81	0.81
46	LamparSagrat2017Ex33	O	0.50	BLTR	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-Q	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-L	1	0.50	0.50	0.50	0.50	0.51	0.52	0.50	0.50	0.50
					2	0.62	0.50	0.50	0.53	0.51	0.50	0.50	0.50	0.50
47	LamparSagrat2017Ex35	O	0.80	BLTR	1	1.00	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
					2	1.00	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
				BLTR-Q	1	1.00	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
					2	1.00	0.80	0.80	0.80	0.80	0.80	0.80	0.80	0.80
				BLTR-L	1	1.10	0.80	0.80	0.80	1.00	0.80	0.80	0.80	0.80
					2	0.96	0.96	0.80	0.80	0.80	0.80	0.80	0.80	1.00
48	LucchettiEtal1987	O	0.00	BLTR	1	0.00	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50
				BLTR-Q	1	0.00	0.50	0.50	0.50	0.50	0.50	0.50	0.00	0.00
					2	0.00	0.50	0.50	0.50	0.50	0.50	0.50	0.00	0.00
				BLTR-L	1	0.00	0.00	0.87	0.91	0.97	0.90	0.95	0.97	0.99
					2	0.66	0.88	0.88	0.91	0.80	0.90	0.95	0.99	0.99
49	LuDebSinha2016a	K	1.14	BLTR	1	1.14	1.14	1.14	1.14	1.14	1.14	1.14	1.14	0.98
					2	1.21	1.30	1.14	1.14	1.14	1.14	1.14	0.98	0.98
				BLTR-Q	1	1.14	1.65	1.64	1.64	1.14	1.69	1.53	1.14	1.15
					2	1.16	1.16	1.16	1.64	1.16	1.30	1.30	1.30	1.31
				BLTR-L	1	1.84	1.14	1.70	1.70	1.70	1.70	1.70	1.70	1.70
					2	1.64	1.64	1.79	1.80	1.80	1.81	1.81	1.81	1.81
50	LuDebSinha2016b	K	0.00	BLTR	1	0.05	0.05	0.05	0.07	0.05	0.05	0.06	0.06	0.06
					2	0.05	0.05	0.05	0.08	0.08	0.08	0.08	0.08	0.08
				BLTR-Q	1	0.05	0.05	0.05	0.05	0.05	0.06	0.06	0.06	0.05
					2	0.10	0.08	0.08	0.06	0.06	0.06	0.06	0.05	0.06
				BLTR-L	1	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
					2	0.11	0.13	0.17	0.22	0.23	0.24	0.24	0.24	0.24
51	LuDebSinha2016c	K	1.12	BLTR	1	1.12	1.12	1.12	1.12	1.12	1.12	1.12	1.12	1.12
					2	1.12	1.12	1.12	1.12	1.12	1.12	1.12	1.12	1.12
				BLTR-Q	1	1.12	1.12	1.14	1.12	1.12	1.12	1.12	1.12	1.12
					2	1.12	1.12	1.14	1.14	1.12	1.12	1.12	1.12	1.12
				BLTR-L	1	1.81	1.12	1.12	1.12	1.14	1.18	1.14	1.13	1.22
					2	1.12	1.12	1.12	1.12	1.18	1.12	1.12	1.15	1.15
52	LuDebSinha2016d	U		BLTR	1	-56.27	-56.27	-56.27	-56.27	-19.56	-19.56	-19.56	-19.56	-19.56
					2	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56
				BLTR-Q	1	-56.27	-56.27	-56.27	-56.27	-19.56	-19.56	-19.56	-19.56	-19.56
					2	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56
				BLTR-L	1	-56.27	-56.27	-56.27	-56.27	-19.56	-19.56	-19.56	-19.56	-19.56
					2	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56	-19.56
53	LuDebSinha2016e	U		BLTR	1	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
					2	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
				BLTR-Q	1	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
					2	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
				BLTR-L	1	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10

	Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
54	LuDebSinha2016f	U		BLTR	1	-19.10	-18.85	-18.71	-18.63	-18.59	-18.57	-18.55	-18.55	-18.55
					2	-87.50	-75.00	-68.75	-65.62	-64.06	-63.28	-62.89	-62.70	-62.60
				BLTR-Q	1	-19.10	-18.85	-18.71	-18.63	-18.59	-18.57	-18.55	-18.55	-18.55
					2	-87.50	-75.00	-68.75	-65.62	-64.06	-63.28	-62.89	-62.70	-62.60
				BLTR-L	1	-19.10	-0.78	-0.78	-0.80	-0.80	-0.78	-0.78	-0.78	-0.80
					2	-19.10	-0.78	-0.78	-0.80	-0.80	-0.78	-0.78	-0.78	-0.80
55	MacalHurter1997	O	81.33	BLTR	1	81.72	81.46	81.35	81.33	81.33	81.33	81.32	81.34	81.70
					2	81.81	81.44	81.35	81.34	81.33	81.35	81.32	81.40	81.64
				BLTR-Q	1	83.17	82.19	82.09	82.08	82.08	82.03	81.98	82.03	82.03
					2	83.55	82.46	82.30	82.13	82.05	82.00	82.00	81.98	82.00
				BLTR-L	1	124.2	152.7	156.8	156.7	170.0	155.6	156.2	156.2	156.9
					2	123.3	173.8	188.4	188.3	189.6	194.4	194.5	194.7	194.6
56	Mirrlees1999	O	1.00	BLTR	1	0.01	0.01	0.01	3.95	3.93	3.92	3.92	3.92	3.92
					2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.08
				BLTR-Q	1	3.92	0.01	0.01	3.92	3.92	3.92	3.92	3.92	3.92
					2	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.05
				BLTR-L	1	0.01	0.01	3.92	3.99	3.96	3.93	3.93	3.93	3.92
					2	0.25	0.25	0.25	0.25	0.25	0.20	0.46	0.64	0.80
57	MitsosBarton2006Ex38	O	0.00	BLTR	1	1.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-Q	1	1.00	1.00	1.00	1.00	1.00	1.00	0.01	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-L	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
58	MitsosBarton2006Ex39	O	-1.00	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	0.00	0.00	0.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	0.00	-1.00	0.76	0.76	0.76	0.76	0.76	0.76
				BLTR-L	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
59	MitsosBarton2006Ex310	O	0.50	BLTR	1	-0.50	-0.50	-0.50	-0.50	-0.50	-0.50	-0.50	-0.51	-0.50
					2	-0.50	-0.50	-0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-Q	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	-0.50	-0.50	-0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-L	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	-0.50	-0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
60	MitsosBarton2006Ex311	O	-0.80	BLTR	1	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80
					2	-0.80	-0.80	-0.80	-0.80	-0.80	-0.50	-0.09	-0.10	-0.10
				BLTR-Q	1	-0.50	-0.50	-0.50	-0.50	-0.50	-0.50	-0.39	-0.39	-0.50
					2	-0.80	-0.80	-0.80	-0.80	-0.80	-0.09	-0.09	-0.09	0.00
				BLTR-L	1	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80
					2	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80	-0.80
61	MitsosBarton2006Ex312		0.00	BLTR	1	7.99	7.99	5.13	0.00	0.00	0.00	0.00	0.00	0.00
					2	7.99	7.99	5.13	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	7.99	7.99	4.39	2.37	0.82	0.00	0.58	0.00	0.02
					2	7.99	7.99	4.39	1.87	1.87	0.00	0.00	0.00	0.00
				BLTR-L	1	-0.58	-0.58	-0.58	-0.58	-0.58	-0.58	-0.58	-0.58	-0.58
					2	0.00	0.00	0.00	0.00	-0.99	0.00	0.00	0.00	0.00
62	MitsosBarton2006Ex313	O	-1.00	BLTR	1	-2.00	-2.00	0.03	0.03	0.03	0.02	0.00	0.01	0.00
					2	-2.00	-2.00	-2.00	-2.00	0.03	0.02	0.03	0.03	-1.77
				BLTR-Q	1	-2.00	-2.00	-2.00	0.06	0.03	0.02	0.00	0.00	0.00
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-0.50
				BLTR-L	1	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
63	MitsosBarton2006Ex314	O	0.25	BLTR	1	0.18	0.14	0.10	0.08	0.07	0.06	0.19	1.00	1.00
					2	0.20	0.20	0.25	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-Q	1	0.21	0.33	0.06	0.25	0.06	0.10	0.09	0.16	0.15
					2	0.20	0.56	1.00	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-L	1	0.25	0.14	0.10	0.07	0.06	0.06	0.06	0.06	0.06
					2	0.56	1.02	1.04	1.00	1.00	1.00	1.00	1.00	1.00
64	MitsosBarton2006Ex315	O	0.00	BLTR	1	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	-2.00	-2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	-2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
				BLTR-L	1	-2.00	-2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	-2.00	-2.00	-1.00	-1.00	0.50	0.50	0.88	0.88	0.97
65	`MitsosBarton2006Ex316`	O	-2.00	BLTR	1	-2.00	-2.00	-2.00	-1.82	-3.00	-3.00	-2.00	-2.00	-1.82
					2	-3.00	-2.00	-2.00	-2.00	-2.00	-3.00	-2.00	-2.00	-2.82
				BLTR-Q	1	-3.00	-3.00	-2.00	-1.86	-3.00	-3.00	-1.82	-1.82	-1.82
					2	-3.00	-3.00	-2.00	-2.00	-2.00	-3.00	-2.82	-3.00	-3.00
				BLTR-L	1	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-0.30	-0.22
					2	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-3.00	-2.00
66	`MitsosBarton2006Ex317`	O	0.19	BLTR	1	0.00	0.00	0.00	0.00	0.19	0.19	0.25	0.24	0.25
					2	0.00	0.00	0.00	0.00	0.19	0.19	0.25	0.24	0.25
				BLTR-Q	1	0.23	0.00	0.00	0.19	0.19	0.18	0.19	0.20	0.22
					2	0.23	0.00	0.00	0.19	0.19	0.18	0.19	0.20	0.22
				BLTR-L	1	0.44	0.44	0.24	0.24	0.28	0.27	0.31	0.30	0.34
					2	0.44	0.44	0.24	0.24	0.28	0.27	0.31	0.35	0.34
67	`MitsosBarton2006Ex318`	O	-0.25	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-L	1	-1.00	-1.00	-1.00	-1.00	-1.00	-0.97	-0.23	-0.23	-0.25
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-0.97	-0.23	-0.23	-0.25
68	`MitsosBarton2006Ex319`	O	-0.26	BLTR	1	-0.01	-0.01	-0.01	-0.25	-0.25	-0.26	0.50	0.50	0.50
					2	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.26	-0.26	0.39
				BLTR-Q	1	0.00	0.00	0.00	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.00	0.00	0.00	-0.26	-0.26	-0.20	-0.26	-0.17	-0.26
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
69	`MitsosBarton2006Ex320`	O	0.31	BLTR	1	0.06	0.07	0.09	0.06	0.06	1.00	1.00	1.00	1.00
					2	0.06	0.07	0.09	0.06	1.00	0.06	0.02	0.03	0.03
				BLTR-Q	1	0.06	0.05	0.08	0.06	0.04	0.03	0.03	0.05	0.50
					2	0.05	0.05	0.06	1.00	1.00	1.00	1.00	1.00	1.00
				BLTR-L	1	0.31	0.32	0.06	0.06	0.06	0.10	0.30	0.31	0.31
					2	0.31	0.31	0.06	0.06	0.06	0.09	0.30	0.31	0.31
70	`MitsosBarton2006Ex321`	O	0.21	BLTR	1	0.21	0.21	0.21	0.31	0.31	0.31	0.31	0.30	0.31
					2	0.21	0.21	0.21	0.31	0.31	0.31	0.31	0.30	0.31
				BLTR-Q	1	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
					2	0.21	0.21	0.21	0.21	0.21	0.24	0.24	0.20	0.21
				BLTR-L	1	0.25	0.25	0.21	0.21	0.52	0.52	0.52	0.52	0.36
					2	0.25	0.25	0.25	0.25	0.21	0.21	0.21	0.21	0.21
71	`MitsosBarton2006Ex322`	O	0.21	BLTR	1	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.21
					2	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.21
				BLTR-Q	1	0.21	0.21	0.21	0.21	0.21	0.24	0.24	0.20	0.21
					2	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
				BLTR-L	1	0.21	0.21	0.25	0.21	0.21	0.26	0.45	0.45	0.56
					2	0.21	0.21	0.25	0.25	0.22	0.34	0.44	0.42	0.49
72	`MitsosBarton2006Ex323`	O	0.18	BLTR	1	0.08	0.08	0.07	0.07	0.18	0.18	0.18	0.18	0.18
					2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
				BLTR-Q	1	0.08	0.07	0.07	0.07	0.18	0.18	0.18	0.18	0.18
					2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
				BLTR-L	1	0.07	0.07	0.07	0.07	0.18	0.18	0.18	0.18	0.18
					2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
73	`MitsosBarton2006Ex324`	O	-1.75	BLTR	1	-0.22	-1.75	-0.29	-0.29	-0.29	-0.29	-0.29	-0.30	-0.29
					2	-0.22	-1.75	-0.29	-0.29	-0.29	-0.29	-0.29	-0.30	-0.29
				BLTR-Q	1	-0.22	-1.75	-1.75	0.90	0.90	0.90	0.90	0.90	0.90
					2	-0.22	-1.75	-1.75	-1.75	-1.10	-1.10	-1.10	-1.10	-1.10
				BLTR-L	1	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10
					2	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10	-1.10
74	`MitsosBarton2006Ex325`	K	-1.00	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

	Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	-1.00	-1.00	-1.00	0.00	-1.00	-1.00	-1.00	-1.00	0.00
				BLTR-L	1	-1.00	-1.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-0.78
					2	-1.00	-1.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-0.79
75	MitsosBarton2006Ex326	O	-2.35	BLTR	1	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
					2	-2.05	-2.35	-2.35	-2.00	-2.00	-2.00	-2.00	-2.00	-2.35
				BLTR-Q	1	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
					2	-2.05	-2.35	-2.35	-2.00	-2.00	-2.00	-2.00	-2.00	-1.32
				BLTR-L	1	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
					2	-2.00	-2.35	-2.93	-2.01	-1.32	-1.75	-1.32	-1.75	-2.62
76	MitsosBarton2006Ex327	K	2.00	BLTR	1	1.00	2.98	2.00	3.00	2.00	2.00	2.00	2.00	2.97
					2	1.00	2.00	2.97	2.00	3.00	2.00	3.00	2.00	3.00
				BLTR-Q	1	1.00	2.99	2.99	3.00	2.00	2.00	2.00	3.00	2.00
					2	1.10	2.99	1.00	2.00	2.00	2.12	3.00	2.99	3.39
				BLTR-L	1	2.01	2.97	2.00	2.00	2.00	2.24	2.57	2.07	2.32
					2	3.10	2.93	2.02	3.07	1.08	3.00	3.06	3.00	2.01
77	MitsosBarton2006Ex328	K	-10.00	BLTR	1	-9.00	-9.00	-10.00	-9.28	-10.00	-10.00	-6.00	-6.00	-6.00
					2	-9.00	-9.00	-10.00	-8.00	-6.00	-7.09	-6.00	-6.00	-10.00
				BLTR-Q	1	-9.00	-9.64	-9.31	-9.24	-6.09	-1.10	-4.25	-4.07	-2.91
					2	-9.00	-9.99	-10.00	-4.16	-4.14	-4.16	-4.13	-4.43	-3.44
				BLTR-L	1	-9.00	-8.00	-10.00	-9.00	-9.00	-4.06	-3.68	-5.04	-4.19
					2	-8.00	-8.22	-10.00	-2.28	-2.60	-2.12	-5.10	-4.15	-0.83
78	MorganPatrone2006a	O	-1.00	BLTR	1	0.79	-1.00	-1.00	-1.00	-1.00	-0.99	-0.98	-1.00	1.00
					2	0.79	1.00	-1.00	-1.00	-1.00	-0.99	-0.99	-1.00	-0.91
				BLTR-Q	1	0.79	1.00	-1.00	1.00	-0.99	-0.91	1.00	1.00	-0.02
					2	0.78	1.00	-1.00	1.00	-0.99	-0.91	-0.99	-0.91	-0.91
				BLTR-L	1	0.95	-1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
					2	0.95	1.00	-1.00	-0.99	-0.99	-0.99	-0.99	-1.00	-1.00
79	MorganPatrone2006b	O	-1.25	BLTR	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	-1.25	-1.25	-1.23	-0.50	-0.50	-1.20	-1.15	-1.15
				BLTR-Q	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	0.50	0.50	0.75	0.75	0.75	0.75	0.75	0.75
				BLTR-L	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	0.50	0.50	0.75	0.75	0.75	0.75	0.75	0.75
80	MorganPatrone2006c	O	-1.00	BLTR	1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-Q	1	-1.00	-1.00	-0.75	-0.75	-0.75	-0.75	-0.75	-0.75	-0.75
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				BLTR-L	1	-1.00	-1.00	-0.75	-0.75	-0.75	-0.75	-0.75	-0.75	-0.75
					2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
81	MuuQuy2003Ex1	K	-2.08	BLTR	1	-1.43	-1.83	-2.00	-2.05	-2.07	-2.08	-2.08	-2.06	-2.09
					2	-1.45	-1.84	-2.00	-2.05	-2.07	-2.07	-2.08	-2.05	-2.02
				BLTR-Q	1	-1.45	-1.83	-2.00	-2.05	-2.05	-2.00	-2.05	-1.81	-1.75
					2	-1.46	-1.83	-2.00	-2.05	-2.05	-2.00	-2.05	-2.00	-2.02
				BLTR-L	1	-0.94	-1.91	-2.00	-2.05	-2.07	-2.04	-1.81	-1.39	-1.35
					2	-0.94	-1.88	-1.70	-1.92	-1.92	-2.00	-2.00	-2.00	-2.00
82	MuuQuy2003Ex2	K	0.64	BLTR	1	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.80
					2	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64
				BLTR-Q	1	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.79
					2	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64
				BLTR-L	1	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64
					2	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64	0.64
83	NieWangYe2017Ex34	O	2.00	BLTR	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
				BLTR-Q	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
				BLTR-L	1	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.02	2.00
					2	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
84	NieWangYe2017Ex52	O	-1.71	BLTR	1	-1.71	-1.71	-1.71	-1.22	-1.71	-1.22	-1.22	-1.22	-1.22
					2	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.22	-1.22	1.41
				BLTR-Q	1	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.48
					2	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-1.33	0.30
				BLTR-L	1	-1.71	-1.71	-1.71	-1.71	-1.71	-1.71	-0.99	-1.22	-1.22

	Example	**Solution** Status	F^*	**Alg**	**SP**	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	-1.71	-1.71	-1.71	-1.71	-1.71	-1.64	-1.48	-1.48	-0.51
85	NieWangYe2017Ex54	O	-0.44	BLTR	1	-0.26	-0.40	-0.44	-0.44	-0.44	-0.03	-0.44	-0.43	-0.10
					2	-0.26	-0.39	-0.44	-0.43	0.13	0.05	0.30	0.44	0.49
				BLTR-Q	1	-0.18	-0.39	-0.44	-0.44	-0.44	-0.44	-0.19	0.16	-0.12
					2	-0.26	-0.39	-0.44	0.02	0.20	0.30	0.50	0.30	0.22
				BLTR-L	1	0.07	-0.07	-0.01	-0.01	0.00	-0.01	-0.02	-0.03	0.02
					2	0.22	0.01	-0.03	0.06	0.20	0.23	0.19	0.36	0.38
86	NieWangYe2017Ex57	K	-2.00	BLTR	1	-2.00	-2.00	-2.00	0.00	0.00	-2.00	-2.00	0.00	0.00
					2	-2.00	-2.00	-2.00	0.00	-2.00	-2.00	0.00	0.00	0.00
				BLTR-Q	1	-2.00	-2.00	-2.00	-2.00	0.00	-0.25	0.21	0.09	-0.25
					2	-2.00	-2.00	-2.00	0.01	-0.13	0.00	0.00	0.00	0.00
				BLTR-L	1	-2.00	-2.00	-1.01	-1.00	-1.00	-1.00	-1.00	-1.02	-1.02
					2	-2.00	-2.00	-1.01	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
87	NieWangYe2017Ex58	K	-3.49	BLTR	1	-3.49	-3.49	-3.49	-3.49	-2.14	-1.34	-2.60	-2.56	-2.81
					2	-3.49	-3.49	-3.49	-3.49	-2.25	-1.91	0.06	0.14	-0.10
				BLTR-Q	1	-3.49	-3.49	-3.49	-3.49	-3.49	-2.57	-2.27	0.11	-1.63
					2	-3.49	-3.49	-3.49	-3.49	-1.97	-3.28	-2.48	-0.77	0.05
				BLTR-L	1	-3.49	-3.49	-3.49	-3.49	-2.05	-2.51	-2.05	-1.91	-1.84
					2	-3.49	-3.49	-3.49	-3.48	-3.47	-3.22	-2.67	0.09	0.17
88	NieWangYe2017Ex61	K	-1.02	BLTR	1	-1.02	-1.02	-0.07	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02
					2	-1.02	-1.02	-1.02	-2.00	-1.02	-1.02	-1.02	-1.02	-1.02
				BLTR-Q	1	-1.02	-1.02	-1.02	-1.02	-1.02	-1.99	-1.02	-1.02	-1.02
					2	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02
				BLTR-L	1	-1.02	0.81	-0.07	-0.07	-0.07	0.00	-0.07	-0.07	0.58
					2	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02	-1.02
89	Outrata1990Ex1a	K	-8.92	BLTR	1	-8.78	-8.87	-8.90	-8.91	-8.91	-8.88	-8.68	-8.42	-8.68
					2	-8.78	-8.87	-8.90	-8.91	-8.03	-8.42	-8.42	-8.17	-8.42
				BLTR-Q	1	-8.77	-8.87	-8.91	-8.91	-8.90	-8.78	-8.54	-7.93	-8.39
					2	-8.79	-8.87	-8.91	-8.91	-3.19	-2.91	-2.66	-2.92	-2.10
				BLTR-L	1	-8.78	-8.87	-8.91	-8.23	-7.01	-6.62	-6.62	-6.62	-6.72
					2	-8.79	-8.87	-7.88	-5.80	-5.77	-6.01	-5.85	-5.64	-5.63
90	Outrata1990Ex1b	K	-7.56	BLTR	1	-6.35	-6.96	-7.57	-7.57	-7.56	-7.53	-7.53	-7.53	-7.57
					2	-6.34	-7.00	-7.55	-7.55	-7.33	-7.55	-7.55	-7.54	-7.57
				BLTR-Q	1	-6.60	-7.00	-7.57	-7.10	-7.13	-7.52	-7.53	-7.28	-6.87
					2	-6.58	-7.01	-7.40	-7.31	-7.53	-7.55	-7.55	-7.36	-6.41
				BLTR-L	1	-7.12	-7.55	-7.55	-7.56	-6.73	-6.60	-6.58	-5.94	-6.10
					2	-7.55	-6.92	-7.41	-5.90	-6.76	-7.11	-7.19	-7.28	-7.14
91	Outrata1990Ex1c	K	-12.00	BLTR	1	-12.00	-12.00	-12.00	-12.00	-12.00	-12.00	-11.38	-7.47	-4.00
					2	-12.00	-12.00	-4.00	-4.00	-4.00	-4.00	-4.00	-4.00	-4.00
				BLTR-Q	1	-12.00	-12.00	-12.00	-12.00	-6.00	-4.16	-6.00	-4.01	-3.08
					2	-12.00	-12.00	-4.00	-4.00	-4.00	-6.00	-4.73	-4.29	-1.51
				BLTR-L	1	-4.68	-4.94	-4.85	-4.84	-4.82	-4.85	-4.78	-4.84	-4.84
					2	-2.94	-2.95	-0.98	-1.03	-1.67	-1.06	-1.50	-1.39	-1.33
92	Outrata1990Ex1d	K	-3.60	BLTR	1	-3.11	-3.33	-3.48	-3.53	-3.57	-3.59	-3.60	-3.15	-3.60
					2	-2.40	-3.34	-3.47	-3.54	-3.57	-3.60	-2.60	-3.56	-2.30
				BLTR-Q	1	-3.12	-3.34	-3.47	-3.54	-3.56	-3.58	-3.59	-3.59	-3.60
					2	-3.10	-3.33	-3.47	-3.54	-3.57	-3.58	-0.38	-0.38	-0.39
				BLTR-L	1	-3.25	-2.29	-3.47	-3.46	-3.48	-3.52	-3.52	-3.48	-3.50
					2	-1.45	-0.85	-1.55	-0.11	-0.09	-0.07	-0.03	-0.14	0.02
93	Outrata1990Ex1e	K	-3.15	BLTR	1	-1.21	-3.78	-3.78	-3.38	-3.66	-1.22	-2.20	-1.92	-1.59
					2	-1.18	-3.77	-3.78	-3.79	-3.74	-1.22	-1.95	-1.61	-2.82
				BLTR-Q	1	-2.32	-3.88	-3.79	-3.02	-3.90	-2.67	-3.21	-1.51	-1.33
					2	-2.25	-3.80	-3.88	-3.84	-3.75	-3.57	-3.73	-3.20	-3.21
				BLTR-L	1	-1.99	-2.85	-2.13	-1.61	-1.85	-1.86	-1.87	-1.87	-1.87
					2	-1.19	-0.67	-0.82	0.35	0.43	0.08	0.19	0.34	0.36
94	Outrata1990Ex2a	K	0.50	BLTR	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	1.27
					2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-Q	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
					2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
				BLTR-L	1	0.50	0.50	0.50	0.50	0.50	0.50	1.21	1.65	1.88
					2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
95	Outrata1990Ex2b	K	0.50	BLTR	1	0.52	0.55	0.50	0.53	0.50	0.50	0.50	0.50	0.50

	Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
					2	0.54	0.55	0.50	0.51	0.50	0.50	0.50	1.48	0.50
				BLTR-Q	1	0.52	0.50	0.51	0.51	0.50	0.53	0.53	0.66	0.66
					2	0.52	0.50	0.50	0.51	0.51	0.50	0.54	0.62	0.92
				BLTR-L	1	0.76	0.76	0.50	0.75	0.78	0.75	0.74	0.74	0.74
					2	1.51	1.54	1.53	1.52	1.52	1.52	1.52	1.51	1.51
96	Outrata1990Ex2c	K	1.86	BLTR	1	2.25	2.01	1.92	1.88	1.87	1.86	1.86	1.85	1.87
					2	2.25	2.01	1.92	1.88	1.88	1.86	1.86	1.85	1.86
				BLTR-Q	1	2.20	2.01	1.92	1.88	1.88	1.86	1.86	1.85	1.86
					2	2.23	2.03	1.91	1.88	1.87	1.87	1.86	1.85	1.86
				BLTR-L	1	1.86	1.87	1.88	1.89	1.88	1.88	1.88	1.88	1.88
					2	1.90	1.88	1.89	1.89	1.89	1.89	1.89	1.89	1.89
97	Outrata1990Ex2d	K	0.92	BLTR	1	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
					2	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
				BLTR-Q	1	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.86
					2	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.86
				BLTR-L	1	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
					2	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85	0.85
98	Outrata1990Ex2e	K	0.90	BLTR	1	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.89	1.56
					2	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.89	0.90
				BLTR-Q	1	1.04	0.90	0.90	0.90	0.90	0.90	0.90	0.89	2.19
					2	0.98	0.90	0.90	0.90	0.90	0.90	0.90	0.89	2.08
				BLTR-L	1	0.90	0.90	0.91	1.11	1.42	1.79	1.80	1.73	1.80
					2	0.90	1.46	0.92	1.02	1.72	1.77	1.77	1.76	1.76
99	Outrata1993Ex31	K	1.56	BLTR	1	2.57	2.06	1.81	1.68	1.63	1.60	7.25	7.25	7.25
					2	2.59	2.05	1.81	1.67	1.62	1.60	1.58	7.25	1.60
				BLTR-Q	1	2.57	2.03	1.80	1.67	1.63	1.59	7.25	2.24	1.63
					2	2.57	2.07	1.80	1.65	1.63	1.60	1.59	7.25	1.63
				BLTR-L	1	2.57	2.04	1.80	1.61	1.63	1.59	1.58	1.58	1.56
					2	2.57	2.04	1.80	1.73	1.62	1.59	1.56	1.58	1.56
100	Outrata1993Ex32	K	3.21	BLTR	1	4.04	3.60	3.38	3.30	3.25	3.23	3.22	3.20	3.20
					2	4.00	3.59	3.38	3.29	3.25	3.23	3.20	3.22	3.20
				BLTR-Q	1	4.02	3.60	3.40	3.29	3.25	3.23	3.22	3.20	3.69
					2	4.02	3.58	3.40	3.30	3.25	3.23	3.22	3.20	3.20
				BLTR-L	1	4.00	3.59	3.39	3.29	3.25	3.23	3.22	4.41	7.33
					2	4.02	3.59	3.39	3.29	3.25	3.23	3.33	3.95	4.25
101	Outrata1994Ex31	K	3.21	BLTR	1	4.02	3.60	3.39	3.30	3.25	3.23	3.22	3.20	3.20
					2	4.04	3.58	3.38	3.30	3.25	3.23	3.22	3.20	3.20
				BLTR-Q	1	4.02	3.58	3.39	3.30	3.25	3.23	3.22	3.20	3.69
					2	3.99	3.59	3.38	3.30	3.25	3.23	3.22	3.20	3.20
				BLTR-L	1	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00	8.00
					2	4.01	3.59	3.39	3.30	3.25	3.23	3.20	3.67	4.91
102	OutrataCervinka2009	O	0.00	BLTR	1	0.01	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00
					2	0.01	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.00
				BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	1.09
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	1.09
				BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.73	0.74	0.75	0.75
					2	0.00	0.00	0.00	0.00	0.00	0.73	0.74	0.75	0.75
103	PaulaviciusEtal2017a	O	0.25	BLTR	1	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				BLTR-Q	1	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	0.01	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
				BLTR-L	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
					2	0.25	0.25	0.04	1.00	1.00	1.00	1.00	1.00	1.00
104	PaulaviciusEtal2017b	O	-2.00	BLTR	1	-2.00	-1.00	-2.00	-2.00	-2.00	-2.00	-1.00	-2.00	-2.00
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
				BLTR-Q	1	-2.00	-2.00	-2.00	-2.00	1.00	0.94	0.99	0.99	0.99
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-1.00
				BLTR-L	1	-1.00	-2.00	-2.00	-2.00	-1.00	-1.00	-1.00	-1.00	-2.00
					2	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00	-2.00
105	SahinCiric1998Ex2	O	5.00	BLTR	1	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.01	5.00
					2	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
				BLTR-Q	1	9.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00

Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
				2	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
			BLTR-L	1	9.00	5.00	5.00	5.00	5.00	5.00	5.00	9.03	9.18
				2	9.06	9.00	9.00	9.06	9.47	9.00	9.59	9.17	9.62
106 ShimizuAiyoshi1981Ex1	O	100.0	BLTR	1	72.19	82.85	90.31	94.82	97.32	98.64	99.31	99.65	99.83
				2	72.19	82.85	90.31	94.82	97.32	98.64	99.31	99.65	99.83
			BLTR-Q	1	72.19	82.85	90.31	94.82	97.32	98.64	99.31	99.65	99.83
				2	72.19	82.85	90.31	94.82	97.32	98.64	99.31	99.65	99.83
			BLTR-L	1	72.18	82.87	90.30	94.82	97.32	98.63	99.31	99.65	99.83
				2	72.18	82.85	90.27	95.00	97.31	98.64	99.31	99.65	99.83
107 ShimizuAiyoshi1981Ex2	O	225.0	BLTR	1	237.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0
				2	237.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0
			BLTR-Q	1	237.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0
				2	237.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0	225.0
			BLTR-L	1	237.0	225.0	225.0	225.0	225.0	225.0	226.5	225.0	225.0
				2	236.9	225.0	225.0	225.0	225.0	225.0	225.0	252.1	250.6
108 ShimizuEtal1997a	U		BLTR	1	49.31	16.89	16.89	16.89	16.89	16.89	16.89	16.89	16.89
				2	49.31	43.03	50.00	25.00	25.00	25.00	25.00	25.00	25.00
			BLTR-Q	1	49.31	16.89	16.89	16.89	16.89	16.89	16.89	16.89	16.89
				2	49.31	43.03	30.31	25.00	25.00	25.00	25.00	25.00	25.00
			BLTR-L	1	49.07	16.89	16.89	16.89	16.89	16.89	16.89	16.89	16.89
				2	49.07	62.61	31.02	25.00	25.00	25.00	25.00	25.00	25.00
109 ShimizuEtal1997b	O	2250.0	BLTR	1	2370.2	2347.5	2332.7	2323.1	2316.6	2312.6	2309.9	2308.1	2307.3
				2	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.3	2251.4
			BLTR-Q	1	2370.2	2347.5	2332.7	2323.1	2316.6	2312.6	2309.9	2308.1	2307.3
				2	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.0	2250.3	2251.5
			BLTR-L	1	2371.6	2351.7	2333.3	2321.7	2316.6	2312.4	2311.9	2310.9	2308.5
				2	2369.6	2250.0	2250.0	2321.3	2316.3	2313.9	2309.2	2250.0	2250.0
110 SinhaMaloDeb2014TP3	K	-18.68	BLTR	1	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68
				2	-18.68	-18.68	-18.68	-18.68	-18.68	-18.63	-18.62	-18.62	-18.61
			BLTR-Q	1	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68
				2	-18.68	-18.68	-18.68	-18.68	-18.68	-18.63	-18.62	-18.62	-18.61
			BLTR-L	1	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68
				2	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68	-18.68
111 SinhaMaloDeb2014TP6	K	-1.21	BLTR	1	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21
				2	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21
			BLTR-Q	1	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21
				2	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21
			BLTR-L	1	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	0.99	-1.21
				2	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	-1.21	0.92
112 SinhaMaloDeb2014TP7	K	-1.96	BLTR	1	-1.96	-1.60	-1.96	-1.96	-1.70	-1.12	-1.00	-1.00	-1.00
				2	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.00	-1.00
			BLTR-Q	1	-1.96	-1.65	-1.45	-1.58	-1.46	-1.42	-1.00	-1.00	-1.00
				2	-1.96	-1.96	-1.96	-1.96	-1.89	-1.96	-1.00	-1.00	-1.00
			BLTR-L	1	-1.61	-1.65	-1.65	-1.26	-1.12	-1.13	-1.03	-1.01	-1.00
				2	-1.96	-1.96	-1.96	-1.96	-1.96	-1.96	-1.00	-1.00	-1.00
113 SinhaMaloDeb2014TP8	O	0.00	BLTR	1	50.36	48.58	45.65	41.53	36.79	32.48	29.31	27.33	26.22
				2	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.53	1.86
			BLTR-Q	1	50.36	48.58	45.65	41.53	36.79	32.48	29.31	27.33	26.22
				2	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.47	1.74
			BLTR-L	1	279.1	252.8	249.1	42.03	100.9	32.26	29.32	27.12	33.47
				2	690.7	662.2	2.06	1.65	29.93	18.07	0.03	0.08	0.19
114 SinhaMaloDeb2014TP9	K	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			BLTR-Q	1	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
				2	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			BLTR-L	1	1.05	1.69	3.68	0.05	0.01	0.28	0.33	2.41	29.02
				2	1.82	0.11	0.25	0.09	0.08	0.83	0.23	1.70	1.85
115 SinhaMaloDeb2014TP10	K	0.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
			BLTR-Q	1	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01
				2	0.00	0.00	0.00	0.01	0.02	0.00	0.00	0.00	0.00
			BLTR-L	1	0.02	0.05	0.11	0.03	0.09	0.51	0.06	0.29	0.17

Example	Solution Status	F^*	Alg	SP	F $\gamma:$ 2^{-1}	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
				2	0.03	0.93	0.44	0.06	0.07	2.00	0.05	0.09	0.54
116 TuyEtal2007	O	22.50	BLTR	1	24.73	24.48	24.01	23.27	22.53	22.50	22.50	22.50	22.50
				2	24.73	24.48	24.01	22.50	22.50	22.50	22.50	22.50	22.50
			BLTR-Q	1	24.73	24.48	24.01	23.27	22.53	22.50	22.50	22.50	22.50
				2	24.73	24.48	24.01	22.50	22.50	22.50	22.50	22.50	22.50
			BLTR-L	1	24.07	24.75	22.85	24.07	22.50	22.50	22.50	22.50	22.50
				2	24.07	24.75	22.85	22.50	22.50	22.50	22.50	22.50	22.50
117 Vogel2002	O	1.00	BLTR	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				2	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.98
			BLTR-Q	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
				2	4.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.98
			BLTR-L	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	3.98
				2	0.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	3.98
118 WanWangLv2011	O	10.63	BLTR	1	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
				2	12.75	12.75	12.75	12.75	7.50	7.50	10.63	10.63	10.63
			BLTR-Q	1	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
				2	12.75	12.75	12.75	12.75	7.50	10.62	7.50	10.63	10.63
			BLTR-L	1	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00	17.00
				2	12.75	12.75	12.75	12.75	7.50	10.63	7.50	10.63	7.50
119 YeZhu2010Ex42	O	1.00	BLTR	1	1.09	1.05	1.02	1.01	1.00	1.00	1.00	0.99	1.13
				2	1.09	1.05	1.02	1.01	1.00	1.00	1.00	0.99	0.99
			BLTR-Q	1	1.09	1.05	1.02	1.01	1.00	1.00	1.00	0.99	1.00
				2	1.09	1.05	1.02	1.01	1.00	1.00	1.00	0.99	0.99
			BLTR-L	1	1.10	1.04	1.02	1.01	1.00	1.00	1.00	0.99	0.99
				2	1.10	1.04	1.02	1.19	1.02	1.23	4.40	1.24	3.80
120 YeZhu2010Ex43	O	1.25	BLTR	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
				2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
			BLTR-Q	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99
				2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
			BLTR-L	1	1.06	1.06	1.00	1.19	1.14	1.25	1.25	1.25	1.25
				2	1.01	1.04	1.00	1.00	1.00	1.00	1.00	0.99	0.99
121 Yezza1996Ex31	O	1.50	BLTR	1	3.76	1.52	1.50	1.50	1.60	1.52	1.50	3.50	3.50
				2	3.85	1.52	1.50	3.50	1.50	1.50	1.50	1.50	1.50
			BLTR-Q	1	3.76	1.50	1.50	1.50	3.00	3.00	3.00	3.00	3.00
				2	3.85	1.50	1.50	1.50	1.50	1.50	1.50	1.50	1.50
			BLTR-L	1	2.30	1.50	1.50	3.50	3.50	3.50	3.50	3.50	1.50
				2	3.58	1.50	1.50	1.50	1.50	1.50	1.50	1.50	1.50
122 Yezza1996Ex41	O	0.50	BLTR	1	0.72	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49
				2	0.72	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49
			BLTR-Q	1	0.72	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49
				2	0.72	0.62	0.56	0.52	0.51	0.50	0.50	0.50	0.49
			BLTR-L	1	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.49
				2	0.63	0.63	0.63	0.63	0.52	1.42	1.22	1.09	2.43
123 Zlobec2001a	O	-1.00	BLTR	1	0.00	0.00	-1.00	0.00	0.00	0.00	0.00	0.00	0.00
				2	0.00	0.00	-1.00	-0.91	0.00	0.00	0.00	0.00	0.00
			BLTR-Q	1	0.00	0.00	0.00	0.00	-0.99	-1.00	-1.00	-1.00	-0.89
				2	51.77	0.00	-1.00	-0.91	0.00	0.00	0.00	0.00	0.00
			BLTR-L	1	120.2	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
				2	68.00	0.00	-1.00	-0.88	-0.95	-1.00	-1.00	-1.00	-0.78
124 Zlobec2001b	N		BLTR	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
				2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
			BLTR-Q	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
				2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
			BLTR-L	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
				2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00

Appendix B

Data for network examples

Table B.1 details the data for the free-flow travel time t_a^0, the capacity C_a, and the length l_a for each link $a \in A$ for Example 3 in Section 4.5.4. This was the London network example used to test the performance of the trust-region algorithm on the road pricing model presented in Section 4.4. The capacity is given in vehicles per hour, the free-flow travel time is given in minutes, and the length of each road is given in metres.

Table B.1: Data for Example 3

Link	C_a	t_a^0	l_a	Link	C_a	t_a^0	l_a	Link	C_a	t_a^0	l_a
1	800	6	900	53	800	4	900	105	640	4	500
2	800	7	775	54	800	5	850	106	640	5	825
3	800	6	1200	55	800	4	900	107	640	4	375
4	800	7	400	56	800	5	850	108	640	5	625
5	800	6	350	57	800	4	450	109	640	4	350
6	800	7	400	58	800	5	900	110	640	5	1300
7	800	6	700	59	800	4	750	111	640	4	1100
8	800	7	1500	60	800	5	1400	112	640	5	925
9	800	6	750	61	800	4	1400	113	640	4	925
10	800	7	1100	62	800	5	400	114	640	5	525
11	800	6	450	63	800	4	950	115	640	4	850
12	800	7	1000	64	800	5	975	116	640	5	825
13	800	6	1000	65	800	4	1100	117	640	4	825
14	800	7	1000	66	800	5	1250	118	640	5	750
15	800	6	2000	67	800	4	1150	119	640	4	575
16	800	7	450	68	800	5	1200	120	640	5	450
17	800	6	1450	69	800	4	1200	121	640	4	600
18	800	7	800	70	800	5	1000	122	640	5	575
19	800	6	1300	71	800	4	800	123	640	4	1250
20	800	7	700	72	800	5	1600	124	640	5	1100
21	800	6	900	73	800	4	1500	125	640	4	300
22	800	7	775	74	800	5	1000	126	640	5	750
23	800	6	1200	75	800	4	900	127	640	4	425
24	800	7	400	76	800	5	850	128	640	5	1200
25	800	6	350	77	800	4	900	129	640	4	300
26	800	7	400	78	800	5	850	130	640	5	675
27	800	6	700	79	800	4	450	131	960	2	625
28	800	7	1500	80	800	5	900	132	960	3	750
29	800	6	750	81	800	4	750	133	960	2	725
30	800	7	1100	82	800	5	1400	134	960	3	600
31	800	6	450	83	800	4	1400	135	960	2	650
32	800	7	1000	84	800	5	400	136	960	3	500
33	800	6	1000	85	640	4	825	137	960	2	825
34	800	7	1000	86	640	5	825	138	960	3	375
35	800	6	2000	87	640	4	750	139	960	2	625
36	800	7	450	88	640	5	575	140	960	3	350
37	800	6	1450	89	640	4	450	141	960	2	1300
38	800	7	800	90	640	5	600	142	960	3	1100
39	800	6	1300	91	640	4	575	143	960	2	925
40	800	7	700	92	640	5	1250	144	960	3	925
41	800	4	950	93	640	4	1100	145	960	2	525
42	800	5	975	94	640	5	300	146	960	3	850
43	800	4	1100	95	640	4	750	147	960	2	1000
44	800	5	1250	96	640	5	425	148	960	3	800
45	800	4	1150	97	640	4	1200	149	960	2	700
46	800	5	1200	98	640	5	300	150	960	3	800
47	800	4	1200	99	640	4	675	151	960	2	1000
48	800	5	1000	100	640	5	625	152	960	3	800
49	800	4	800	101	640	4	750	153	960	2	700
50	800	5	1600	102	640	5	725	154	960	3	800
51	800	4	1500	103	640	4	600				
52	800	5	1000	104	640	5	650				

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Nework flows: Theory, algorithms, and applications*. Prentice-Hall, New Jersey, 1993.

[2] E. Aiyoshi and K Shimizu. A solution method for the static constrained stackelberg problem via penalty method. *IEEE Transactions on Automatic Control*, 29(12):1111–1114, 1984.

[3] G. Anandalingam, R. Mathieu, L. Pittard, and R. Sinha. Artificial intelligence based approaches for hierarchical optimization. In R. et al. Sharda, editor, *Impact of recent computer advances in operations research*. North-Holland, New York, 1989.

[4] G. Anandalingam and D.J. White. A solution method for the linear stackelberg problem using penalty functions. *IEEE Transactions on Automatic Control*, 35(10):1170–1173, 1990.

[5] P. Apkarian, D. Noll, and L. Ravanbod. Nonsmooth bundle trust-region algorithm with applications to robust stability. *Set-Valued and Variational Analysis*, 24:115–148, 2016.

[6] Asian Development Bank. The case for electronic road pricing, 2016. URL: https://development.asia/case-study/case-electronic-road-pricing.

[7] J. Bard. *Practical bilevel programming: Algorithms and applications*. Kluwer Academic Publishers, Dordrecht, 1999.

[8] J.F. Bard. Convex two-level optimization. *Mathematical Programming*, 40:15–27, 1988.

[9] J.F. Bard and J. Falk. An explicit solution to the multi-level programming problem. *Computers and Operations Research*, 9:27177–100, 1982.

[10] M. Beckmann, C. Mcquire, and C.B. Winsten. *Studies in the economics of transportation*. Yale University Press, New Haven, 1956.

[11] M.G.H. Bell. Stochastic user equilibrium assignment in networks with queues. *Transportation Research Part B*, 29(2):125–137, 1994.

[12] W. Bialas and M. Karwan. Two-level linear programming. *Management Science*, 30:1004–1020, 1984.

[13] J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. *Operational Research*, 21:37–44, 1973.

[14] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4):345–358, 2001.

[15] P. Calamai and L. Vicente. Generating linear and linear-quadratic bilevel programming problems. *SIAM Journal on Scientific and Statistical Computing*, 14:770–782, 1993.

[16] W. Candler and R. Norton. Multilevel programming. Technical Report 20, World Bank Development Research Center, Washington, DC, 1977.

[17] F.H. Clarke. *Optimization and nonsmooth analysis*. John Wiley & Sons, New York, 1983.

[18] B. Colson, P. Marcotte, and G. Savard. A trust-region method for nonlinear bilevel programming: Algorithm and computational experience. *Computational Opimization and Applications*, 30:211–227, 2005.

[19] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operational Research*, 153:235–256, 2007.

[20] A. Conn, N. Gould, and P. Toint. *Trust-region methods*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[21] A. Daniilidis and P. Georgiev. Approximate convexity and submonotonicity. *Journal of Mathematical Analysis and Applications*, 291:292–301, 2004.

[22] S. Dempe. A simple algorithm for the linear bilevel programming problem. *Optimization*, 18:373–385, 1987.

[23] S. Dempe. A bundle algorithm applied to bilevel programming problems with non-unique lower level solutions. *Computational Optimization and Applications*, 15:145–166, 2000.

[24] S. Dempe. *Foundations of bilevel programming*. Kluwer Academic Publishers, Dordrecht, 2002.

[25] S. Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 522(3):339–359, 2003.

[26] S. Dempe and J. Dutta. Is bilevel programming a special case of a mathematical program with complementarity constraints. *Mathematical Programming*, 131(1):37–48, 2012.

[27] S. Dempe, J. Dutta, and B.S. Mordukhovich. New necessary optimality conditions in optimistic bilevel programming. *Optimization*, 56:577–604, 2010.

[28] S. Dempe and S. Franke. An algorithm for solving a class of bilevel programming problems. Preprint 2011-04, TU Bergakademie, 2011.

[29] S. Dempe and S. Franke. Solution algorithm for an optimistic linear stackelberg problem. *Comuters and Operations Research*, 41:277–281, 2014.

[30] S. Dempe and S. Franke. The bilevel road pricing problem. *International Journal of Computing and Optimization*, 2(2):71–92, 2015.

[31] S. Dempe and S. Franke. On the solution of convex bilevel optimization problems. *Computational Optimization and Applications*, 63:685–703, 2016.

[32] S. Dempe, V. Kalashnikov, G. A. Perez-Valdes, and Kalashnykova N. *Bilevel programming problems: Theory, algorithms and applications to energy networks*. Springer-Verlag Berlin Heidelberg, 2015.

[33] S. Dempe, B. S. Mordukhovich, and A. B. Zemkoho. Necessary optimality conditions in pessimistic bilevel programming. *Optimization*, 63(4):505–533, 2014.

[34] S. Dempe and A. B. Zemkoho. The bilevel programming problem: reformulations, constraint qualifications and optimality conditions. *Mathematical Progamming*, 138:447–473, 2013.

[35] S. Dempe and A.B. Zemkoho. The generalizard mangasarian-fromowitz constraint qualification and optimality conditions for bilevel programs. *Journal of Optimization Theory and Application*, 148:46–68, 2011.

[36] S. Dempe and A.B. Zemkoho. Bilevel road pricing: theoretical analysis and optimality conditions. *Annals of Operational Research*, 196:223–240, 2012.

[37] S. Dempe and A.B. Zemkoho. On the kurush-kuhn tucker reformulation of the bilevel optimization problem. *Nonlinear Analysis Theory Methods and Applications*, 75(3):1202–1218, 2012.

[38] Stephan Dempe. An implicit function approach to bilevel programming problems. In A. Migdalas, P.M. Pardalos, and P. Värbrand, editors, *Multilevel optimization: Algorithms and applications*, pages 273–294. Kluwer Academic Publishers, Dordrecht, 1998.

[39] X. Deng. Complexity issues in bilevel linear programming. In A. Migdalas, P.M. Pardalos, and P. Värbrand, editors, *Multilevel optimization: Algorithms and applications*, pages 149–164. Kluwer Academic Publishers, Dordrecht, 1998.

[40] J. Dennis, S. Li, and R. Tapia. A unified approach to global convergence of trust region methods for nonsmooth optimization. *Mathematical Programming*, 68:319–346, 1995.

[41] E.D. Dolan and J.J. More. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[42] T.A. Edmunds and J.F. Bard. Algorithms for nonlinear bilevel mathematical programs. *IEEE Transactions on Systems*, 21:83–89, 1991.

[43] J. Eliasson. The stockholm congestion charges: an overview. Technical report, KTH Royal Institute of Technology, Centre for Transport Studies, 2014.

[44] A.W. Evans. Road congestion pricing: when is it a good policy? *Journal of Transport Economics and Policy*, 26:213–243, 1992.

[45] J.E. Falk and J. Liu. On bilevel programming, part i : general nonlinear cases. *Mathematical Programming*, 70:47–72, 1995.

[46] A.V. Fiacco and J. Kyparisis. Convexity and concavity properties of the optimal value function in parametric nonlinear programming. *Journal of Optimization Theory and Applications*, 48:95–126, 1986.

[47] A. Fischer, A. Zemkoho, and S. Zhou. Detailed numerical results for "Semismooth newton-type method for bilevel optimization: Global convergence and extensive numerical experiments". Technical report, School of Mathematics, University of Southampton, Southampton, UK, 2019.

[48] A. Fischer, A. Zemkoho, and S. Zhou. Semismooth newton-type method for bilevel optimization: Global convergence and extensive numerical experiments, 2019. arXiv:1912.07079.

[49] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65:1615–1637, 2017.

[50] Department for Transport. TAG unit M3.1: Highway assignment modelling, 2014. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/427124/webtag-tag-unit-m3-1-highway-assignment-modelling.pdf.

[51] Department for Transport. TAG data book, 2020. URL: https://www.gov.uk/government/publications/tag-data-book.

[52] J. Gauvin and F. Dubeau. Differential properties of the marginal function in mathematical programming. *Mathematical Programming Study*, 19:101–119, 1982.

[53] N. Gould and S. Leyffer. An introduction to algorithms for nonlinear optimization. In J.F. Blowey, A.W. Craig, and T. Shardlow, editors, *Frontiers in Numerical Analysis*, pages 109–197. Springer, Berlin, Heidelberg, 2003.

[54] N. Gould, D. Orban, A. Sartenaer, and P. Toint. Sensitivity of trust-region algorithms to their parameters. *4OR*, 3:227–241, 2005.

[55] R. Henrion and T. Surowiec. On calmness conditions in convex bilevel programming. *Applicable Analysis*, 90(6):951–970, 2011.

[56] R. Horst and N.V. Thoai. Dc programming: Overview. *Journal of optimization thoery and applications*, 103:1–43, 1999.

[57] N. Hoseini and S. Nobakhtian. A new trust region method for nonsmooth nonconvex optimization. *Optimization*, 67(8):1265–1286, 2018.

[58] Y. Ishizuka and E. Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, 1992.

[59] Y. Jiang, C. H. Li, and X. Wu. Application of particle swarm optimization based on chks smoothing function for solving nonlinear bilevel programming problem. *Applied Mathematics and Computation*, 219(9):4332–4339, 2013.

[60] J. J. Júdice and A. M. Faustino. A sequential lcp method for bilevel linear programming. *Annals of Operational Research*, 34:89–106, 1992.

[61] V.V. Kalashnikov, R.C. Herrera-Maldonado, J. Camacho-Vallejo, and N.I. Kalashnykova. A heuristic algorithm solving bilevel toll optimization problems. *The International Journal of Logistics Management*, 27(1):31–51, 2016.

[62] V.V. Kalashnikov, N.I. Kalashnykova, and R.C. Herrera-Maldonado. Bilevel optimal toll problems with nonlinear costs: A heuristic solution method. In Kosheleva O., Shary S., Xiang G., and Zapatrin R., editors, *Beyond traditional probabilistic data*

processing techniques: Interval, fuzzy etc. methods and their applications, pages 481–516. Springer, Cham, 2020.

[63] P.-M. Kleniati and C.S. Adjiman. Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems, part i: Theoretical development. *Journal of Global Optimization*, 60(3):425–458, 2014.

[64] P.-M. Kleniati and C.S. Adjiman. Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems, part ii: Convergence analysis and numerical results. *Journal of Global Optimization*, 60(3):459–481, 2014.

[65] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its applications to optimal highway pricing. *Management Science*, 44(12-part-1):1608–1622, 1998.

[66] L. Lampariello and S. Sagratella. A bridge between bilevel programs and nash games. *Journal of Optimization Theory and Applications*, 174(2):613–635, 2017.

[67] L. Lampariello and S. Sagratella. Numerically tractable optimistic bilevel problems. *Optimization-Online*, 2017.

[68] S. Lawphongpanich and Y. Yin. Nonlinear pricing on transportation networks. *Transportation Research Part C*, 20:218–235, 2012.

[69] X. Li, Y. Lv, W. Sun, and L. Zhou. Cordon- or link-based pricing: Environment-oriented toll design models development and application. *Sustainability, MDPI, Open Access Journal*, 11(1):1–16, 2019.

[70] G. Lin, M. Xu, and J. Ye. On solving simple bilevel programs with a nonconvex lower level program. *Mathematical Programming*, 144:277–305, 2014.

[71] T. Lipp and S. Boyd. Variations and extension of the convex-concave procedure. *Optimization and Engineering*, 17:263–287, 2016.

[72] G. Liu, J. Han, and S Wang. A trust region algorithm for bilevel programming problems. *Chinese Science Bulletin*, 43:820–824, 1998.

[73] Yu-Lan Liu and Hu-Nan Li. E-convexity of the optimal value function in parametric nonlinear programming. *The Ninth International Symposium on Operations Research and Its Applications*, pages 75–82, 2010.

[74] Z. Liu, S. Wang, and Q. Meng. Optimal joint distance and time toll for cordon-based congestion pricing. *Transportation Research Part B*, 69:81–97, 2014.

[75] Transport For London. Central london congestion charging impact monitoring: Second annual report, 2004. URL: http://content.tfl.gov.uk/impacts-monitoring-report-2.pdf.

[76] Transport For London. Central london congestion charging impact monitoring: Sixth annual report, 2008. URL: http://content.tfl.gov.uk/central-london-congestion-charging-impacts-monitoring-sixth-annual-report.pdf.

[77] Transport For London. Mayor's transport strategy, 2018. URL: https://www.london.gov.uk/sites/default/files/mayors-transport-strategy-2018.pdf.

[78] Transport For London. Phvs and the congestion charge, 2020. *Accessed Oct. 10, 2021.* URL: https://tfl.gov.uk/info-for/taxis-and-private-hire/phvs-and-the-congestion-charge.

[79] L. Lozano and J. Cole Smith. A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research*, 65(3):768–786, 2017.

[80] P. Marcotte, G. Savard, and D.L. Zhu. A trust region algorithm for nonlinear bilevel programming. *Operations Research Letters*, 29:171–179, 2001.

[81] R. Mathieu, L. Pittard, and G. Anandalingam. Genetic algorithm based approach to bi-level linear programming. *Operations Research*, 28(1):1–21, 1994.

[82] A.D. May and D.S. Milne. Effects of alternative road pricing systems on network performance. *Transportation Research*, 34A:407–436, 2000.

[83] A.C. McKinnon. A review of European truck tolling schemes and assessment of their possible impact on logistics systems. *International Journal of Logistics*, 9(3):191–205, 2011.

[84] P. Mehlitz and A. Zemkoho. Sufficient optimality conditions in bilevel programming. *Mathematics of Operations Research*, in press, 2020. arXiv:1911.01647.

[85] Q. Meng, Z. Liu, and S. Wang. Optimal distance tolls under congestion pricing and continuously distributed value of time. *Transportation Research Part E*, 48:937–957, 2012.

[86] Q. Meng, H. Yang, and M.G.H. Bell. An equivalent continuously differentiable model and a locally convergent algorithm for the continuous network design problem. *Transportation Research Part B*, 35:83–105, 2001.

[87] A. Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, 7:381–405, 1995.

[88] J.A. Mirrlees. The theory of moral hazard and unobservable behaviour: Part i. *Rev. Econ. Stud.*, 66:3–21, 1999.

[89] A. Mitsos and P.I. Barton. A test set for bilevel programs. Technical report, Massachusetts Institute of Technology, 2006.

[90] A. Mitsos, P. Lemonidis, and P.I. Barton. Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimisation*, 42(4):475–513, 2008.

[91] A. Namdeo and G. Mitchell. An empirical study of estimating vehicle emissions under cordon and distance based road user charging in leeds, uk. *Environmental Monitoring and Assessment*, 136:45–51, 2008.

[92] M. Netter. Equilibrium and marginal cost pricing a road network with several traffic flow types. In G. F. Newell, editor, *Traffic flow and transportation, proceedings of the 5th international symposium on the theory of traffic flow and transportation*, pages 155–163, American Elsevier, New York, NY, 1972.

[93] D. Noll, O. Prot, and A. Rondepierre. A proximity control algorithm to minimize nonsmooth and nonconvex functions. *Pacific Journal of Optimization*, 4(3):571–604, 2008.

[94] J.V. Outrata. On the numerical solution of a class of stackelberg problems. *Methods and Models of Operations Research*, 34:255–277, 1990.

[95] M. Patricksson. *The traffic assignment problem: Models and methods*. VSP BV, Utrecht, Netherlands, 1994.

[96] E. Paulavicius, J. Gao, P.M. Kleniati, and C.S. Adjiman. Basbl: Branch-and-sandwich bilevel solver. implementation and computational study with the basblib test set. *Computers & Chemical Engineering*, 132:1–23, 2020.

[97] A.C. Pigou. *The economics of welfare*. MacMillan, London, 1920.

[98] L. Qi and J. Sun. A trust region algorithm for minimization of locally lipschitzian functions. *Mathmatical Programming*, 66:25–43, 1994.

[99] G. Santos, K. Button, and R. Noll. London congestion pricing. *Brookings-Wharton Papers on Urban Affairs*, pages 177–234, 2008.

[100] G. Santos and G. Fraser. Road pricing: lessons from London. *Economic Policy*, 21(46):264–310, 2006.

[101] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15:265–272, 1994.

[102] H. Scheel and S. Scholtes. Mathematical programs with complementarity constraints: stationarity, optimality and sensitivity. *Mathematics of Operational Research*, 25:1–22, 2000.

[103] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM J. Optimization*, 2:121–152, 1992.

[104] K. Shimizu, Y. Ishizuka, and J.F. Bard. *Nondifferentiable and two-level mathematical programming*. Kluwer Academic Publishers, Dordrecht, 1997.

[105] K. Shimizu and M. Lu. A global optimisation method for the stackelberg problem with convex functions via problem transformation and concave programming. *IEEE Transactions on Systems, Man and Cybernetics*, 25(12):1635–1640, 1995.

[106] A. Sinha, K. Deb, and P. Malo. Bilevel optimization based on iterative approximation of multiple mappings. *Journal of Heuristics*, 26:151–185, 2020.

[107] A. Sinha, P. Malo, and K. Deb. Efficient evolutionary algorithm for single-objective bilevel optimization, 2013. arXiv:1303.3901.

[108] A. Sinha, P. Malo, and K. Deb. Solving optimistic bilevel programs by iteratively approximating lower level optimal value function. *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1877–1884, 2016.

[109] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[110] M.J Smith, A.D. May, M.B. Wisten, D.S. Milne, D. Van Vliet, and M.O. Ghali. A comparison of the network effects of four road-user charging systems. *Traffic Engineering and Control*, 35(5):311–315, 1994.

[111] J.E. Spingarn. Submonotone subdifferentials of lipschitz functions. *Transactions of the American Mathematical Society*, 264:77–89, 1981.

[112] B.K. Sriperumbudur and G.R.G. Lanckriet. On the convergence of the concave-convex procedure. *Advances in Neural Information Processing Systems*, 22:1759–1767, 2009.

[113] H. Stackelberg. *The theory of market economy*. Oxford University Press, Oxford, 1952.

[114] C. Suwansirikul, T. L. Friesz, and R. L. Tobin. Equilibrium decomposed optimization: A heuristic for the continuous equilibrium network design problem. *Transportation Science*, 21:254–263, 1987.

[115] J. Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966.

[116] London Assembly Transport Committee. London stalling: reducing congestion in London, 2017. URL: https://www.london.gov.uk/about-us/londonassembly/meetings/documents/s61576/Traffic%20congestion%20report.pdf.

[117] E.T. Verhoef. Second-best congestion pricing in general networks: heuristic algorithms for finding second-best optimal toll levels and toll points. *Transportation Research Part B*, 36(8):707–729, 2002.

[118] L.N. Vicente, G. Savard, and J.J. Júdice. Descent approaches for quadratic bilevel programming. *Journal of Optimisation Theory and Applications*, 81(2):379–399, 1994.

[119] G. Wang, Z. Wan, X. Wang, and Y. Lv. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications*, 56(10):2250–2555, 2008.

[120] J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of Institution of Civil Engineers Part 2*, 1:325–378, 1952.

[121] D.J. White and G. Anandalingam. A penalty function for solving bilevel linear programs. *Journal of Global Optimization*, 3:397–419, 1993.

[122] W. Wiesemann, A. Tsoukalas, P. Kleniati, and B. Rustem. Pessimistic bilevel optimization. *SIAM Journal on Optimization*, 23:353–380, 2013.

[123] M. Xu and J.J. Ye. A smoothing augmented lagrangian method for solving simple bilevel programs. *Computational optimization and Applications*, 59:353–377, 2014.

[124] M. Xu, J.J. Ye, and L. Zhang. Smoothing sqp methods for solving degenerate nonsmooth constrained optimization problems with applications to bilevel programs. *SIAM Journal of Optimization*, 25:1388–1410, 2015.

[125] H. Yan and W. Lam. Optimal road tolls under conditions of queuing and congestion. *Transportation Research Part A*, 30:319–332, 1996.

[126] H. Yang and M. G. H. Bell. Traffic restraint, road pricing and network equilibrium. *Transportation Research Part B*, 31(4):303–314, 1997.

[127] H. Yang and S. Yagar. Traffic assignment and signal control in saturated road networks. *Transpn Res.*, 29A:125–139, 1995.

[128] H. Yang, S. Yagar, Y Iida, and Y. Asakura. An algorithm for the inflow control problem on urban freeway networks with user-optimal flows. *Transportation Research Part B: Methodological*, 28(2):123–139, 1994.

[129] H. Yang, X. Zhang, and Q. Meng. Modeling private highways in networks with entry-exit based toll charges. *Transportation Research Part B*, 38:191–213, 2004.

[130] J. Ye, X. Yuan, S. Zeng, and J. Zhang. Difference of convex algorithms for bilevel programs with applications in hyperparameter selection, 2021. arXiv:2102.09006.

[131] J. J. Ye and D. L. Zhu. Optimality conditions for bilevel programming problems. *Optimization*, 33:9–27, 1995.

[132] J. J. Ye and D. L. Zhu. New necessary optimality conditions for bilevel programs by combining the mpec and value function approaches. *SIAM Journal on Optimization*, 20(4):1885–1905, 2010.

[133] Y. Yin. Genetic-algorithms-based approach for bilevel programming models. *Journal of Transportation Engineering*, 126(2):115–120, 2000.

[134] Y. Yin and S. Lawphongpanich. Internalizing emission externality on road networks. *Transportation Research Part D: Transport and Environment*, 11(4):292–301, 2006.

[135] Y. Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151:249–281, 2015.

[136] A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[137] X. Zhang and H. Yang. Multiclass network toll design problem with social and spatial equity constraints. *Journal of Transportation Engineering*, 238:420–428, 2002.

[138] X. Zhang and H. Yang. The optimal cordon-based network congestion pricing problem. *Transportation Research Part B*, 38:517–537, 2004.

[139] S. Zhou, A. Zemkoho, and A. Tin. Bolib: Bilevel optimization library of test problems. In S. Dempe and A. Zemkoho, editors, *Bilevel optimization: Advances and next challenges*, pages 563–580. Springer, Switzerland, 2020.

[140] S. Zhou and A.B. Zemkoho. Biopt toolbox, 2020. URL: https://biopt.github.io/.

[141] S. Zhou and A.B. Zemkoho. Theoretical and numerical comparison of the

karush-kuhn tucker and value function reformulations in bilevel optimization, 2020. arXiv:2004.10830.

[142] S. Zlobec. Bilevel programming: Optimality conditions and duality. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, pages 180–185. Springer, Boston, MA, 2001.