Application of Robust and Inverse Optimization in Transportation

by

Thai Dung Nguyen

B.E. Electrical Engineering, National University of Singapore, 2009

Submitted to the School of Engineering

in partial fulfillment of the requirements for the degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .... [illegible]
School of Engineering
August 4, 2010

Certified by .... [illegible]
Dimitris J. Bertsimas
Boeing Professor of Operations Research
Thesis Supervisor

Certified by .... [illegible]
Georgia Perakis
William F. Pounds Professor of Operations Research
Thesis Supervisor

Accepted by .... [illegible]
Karen Willcox
Associate Professor of Aeronautics and Astronautics
Codirector, Computation for Design and Optimization Program

Application of Robust and Inverse Optimization in Transportation

by

Thai Dung Nguyen

Submitted to the School of Engineering
on August 4, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

We study the use of inverse and robust optimization to address two problems in transportation: finding the travel times and designing a transportation network. We assume that users choose the route selfishly and the flow will eventually reach an equilibrium state (User Equilibrium).

The first part of the thesis demonstrates how inverse and robust optimization can be used to find the actual travel times given a stable flow on the network and some noisy information on travel times from different users. We model the users' perception of travel times using three different sets and solve the robust inverse problem for all of them. We also extend the idea to find parametric functional forms for travel times given historical data. Our numerical results illustrate the significant improvement obtained by our models over a simple fitting model.

The second part of the thesis considers the network design problem under demand uncertainty. We show that for affine travel time functions, the deterministic problem can be formulated as a mixed integer programming problem with quadratic objective and linear constraints. For the robust network design problem, we propose a decomposition scheme: breaking a tri-level programming problem into two smaller problems and re-iterating until a good solution is obtained. To deal with the expensive computation required by large networks, we also propose a heuristic robust simulated annealing approach. The heuristic algorithm is computationally tractable and provides some encouragingly results in our simulations.

Thesis Supervisor: Dimitris J. Bertsimas
Title: Boeing Professor of Operations Research

Thesis Supervisor: Georgia Perakis
Title: William F. Pounds Professor of Operations Research

Acknowledgments

I would like to express my heartfelt gratitude to all the people who have helped make this project an enjoyable and fruitful experience.

First and foremost, I would like to thank my advisors, Prof. Dimitris Bertsimas and Prof. Georgia Perakis, for their expert guidance. It is my great honor to have an opportunity to work with two of the best professors in Operations Research today.

I would like to thank Dimitris for his mentorship. His creative ideas and insightful comments have always guided my research. Moreover, his enthusiasm, belief, and optimism has taught me a great lesson in life. Since my first day as his student, Dimitris has been always asking me to be self-confident. Living in a top intellectual environment in the world, this was not easy at first. However, I have come to realize that self-reliance and a positive attitude will help me to mature and to succeed in life.

I am deeply grateful to Georgia for her expertise, kindness and patience. Despite her busy schedule, she always spares time for me to present my ideas. She also explains things very carefully and makes sure that I fully understand her. Her invaluable knowledge in the area of variational inequalities and network equilibrium always helps me to enhance the horizon of my research and encourages me to explore new ideas and topics.

I would like to thank SMA program for creating such a wonderful opportunity. The SMA scholarship brings me to one of the places I have ever dreamed of. I would to thank Laura Koller for her sincere help in the CDO program.

In addtion, I feel very lucky to have many wonderful friends here, who have enriched my life and made it more enjoyable than I could ever expected. I would like to thank Kil Dong Kwak. Not only being a great friend, his persistence has taught me a lot. I will remember my discussions with Henry Chen and my other classmates. Besides, I do really enjoy my soccer games with my Redlobster gang and tennis games with Mr. Venu. I am also very grateful to my friends, who are not here, for their constant support and encouragement.

Last but not least, I am immensely indebted to my family for their unconditional love, encouragement and care. You are always the unlimited source of power for me to overcome any obstacles. I would not be able to come all my way today, without your strong support. I owe you more than I could express.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Uncertainty in Transportation Problems

Traditionally, transportation problems are studied using only deterministic information. For example, in the traffic assigment problem, all travel times are typically considered to be known accurately and the route choice of the users therefore can be modeled exactly. In the network design problem, researchers have focused on the case when the demands for each Origin-Destination (OD) pair of the network are fixed. However, in practical applications, decision makers and planners are often faced with uncertainty with respect to such information. Each user may have a different perception of the travel times of different links. Even if they travel on the same route, the reported travel times may still differ from one to another. Demands for the network also vary during different periods of a day, or different days of a week. Besides, the set of OD pairs may change significantly due to re-location or re-construction in the community.

In order to deal with such uncertainty, one crude approach is to ignore it and fix that information at "nominal" value. These "nominal" values may be the means, medians, or even the smallest or largest values of the given data. This approach; however, ignores the fact that some changes in the data may significantly affect the results. For example, in the User Equilibrium model (see Chapter 3), even the smallest change in travel times may make users not want to travel on a specific route.

Another way to deal with uncertainty is through scenario-based design, or schochastic programming. Stochastic programming considers a discrete representation of the uncertainty or assigns some probability to the realization of the data. This method requires a prior knowledge of the nature of the uncertainty, which might not be available. Besides, stochastic programming might become computationally expensive for a large number of scenarios. Moreover, the result of this approach may be sensitive to some realization of the data. In the network design problem with uncertainty in demands, a good design for a set of scenarios may lead to congestion in some other realization of the demands.

According to the U.S. Department of Transportation (see [40]), in 2009, \$48.6 billion was spent to rebuild the transportation network in cities and counties. Texas Transportation Institute estimated that in 2007, the congestion cost to the U.S economy was \$87 billion. Moreover, it wasted 3 billion gallons of gas and 4 billion hours of time (see [39]). One way to reduce this huge cost is to better deal with the uncertainty in the planning and designing phases.

In light of the recent developments in optimization, robust optimization stands out as a good solution to the issues discussed above. Without any assumption about the distribution, the uncertainty is modeled as some closed convex set. Being formulated as a minimax or maximin problem, the robust solution is protected against the least favorable data realization. It is, moreover, still capable of providing results with good mean and variance over different scenarios from the uncertainty set. Besides, robust otimization often leads to a computational tractable problem, and it is able to provide probabilistic bounds on the feasibility of the constraints and the objective values.

1.2 Contribution of the thesis

This thesis aims to employ the principles of robust optimization in order to deal with the uncertainty in two different problems in transportation: the inverse problem in order to find travel times and the network design problem.

The Inverse Optimization Problem to find Travel Times:

1. By repeatedly observing the flow of the network, we can obtain some stable pattern of the traffic flow. We also can obtain prior "nominal" values for the travel times at the stable flow. Assuming the stable flow is the User Equilibrium (UE) flow, we propose an inverse optimization problem to find the travel times that give rise to the flow under the UE principle, and that are closest to our prior information.
2. We model the uncertainty in the prior travel times using three different sets. We propose algorithms to solve the robust inverse problems for each of them.
3. We propose an algorithm using historical data to find the parametric functional forms for travel times.

The Network Design Problem (NDP) under User Equilibrium and Demand Uncertainty:

1. We revisit the mixed integer quadratic programming with linear constraints to solve the Deterministic Network Design Problem (DNDP) under UE with asymmetric affine travel time functions. We consider the discrete NDP where designers want to consider whether or not to build a link.
2. We model the uncertainty in demands as a polyhedron and propose a decomposition algorithm to solve the Robust Network Design Problem (RNDP).
3. We propose a heuristic algorithm using Robust Simulated Annealing to solve the RNDP for large networks.

1.3 Structure of the thesis

This thesis is structured as follows.

Chapter 2 gives a brief review of robust optimization. Chapter 3 revisits the User Equilibrium Principle, and introduces the Variational Inequality (VI) as well as different optimziation formulations for the UE problem. Chapter 4 developes the inverse problem to find travel times. We also test our algorithms using simulated data in this chapter. Chapter 5 is devoted to the Network Design Problem. Section

5.2 formulates the exact mixed integer programming model for the DNDP. Section 5.3 proposes algorithms to solve the RNDP with a polyhedron demand set. Finally, Chapter 6 concludes the thesis.

1.4 Notations

For the ease of the readers, we would like to introduce the notations that will be used throughout the thesis.

Consider a direct transportation network $G(V, A)$, where V and A are the sets of nodes and links. We use the following notation:

- P^w: set of all paths connects OD pair w
- $P = \cup_{w \in W} P^w$: Set of all paths
- h_p: Flow on a path $p \in P$
- $\mathbf{h}$: path flow vector
- $g_p(\mathbf{h})$: Travel time of path p given the path flow
- $\mathbf{g}(\mathbf{h})$: vector of travel time on all paths
- f_{ij}^w : the flow from on a link from node i to node j due to the OD pair w
- $\mathbf{f^w} \in \mathbb{R}^{|A|}$: the link flow vector to the network due to the OD pair w
- $\mathbf{f} = \sum_{\mathbf{w}} \mathbf{f^w}$: the link flow vector to the whole network
- $c_{ij}(\mathbf{f})$: travel time on link from node i to node j.
- $\mathbf{c}(\mathbf{f})$: vector of travel time on all links.

Let N be the link-node incident matrix ($N_{ki} = 1$ and $N_{kj} = -1$ if a link k^{th} start at node i and end at node j) and let W be the set of OD pairs and for $w \leq |W|$. Let d_w be the amount of flow to be routed from the orgin s_w to destination node t_w.

Define the demand vector $\mathbf{d^w} \in \mathbb{R}^{|V|}$ such that:

$$d_i^w = \begin{cases} d_w \text{ if } i = s_w \\ -d_w \text{ if } i = t_w \\ 0 \text{ otherwise} \end{cases}$$

A feasible flow $\mathbf{f}$ satisfies the equation: $N\mathbf{f^w} = \mathbf{d^w} \quad \forall w$

The total travel time of all users of the network $TC = \mathbf{g(h)^Th} = \mathbf{c(f)^Tf}$

Chapter 2

Review of Robust Optimization

2.1 Robust Linear Optimization

Robust optimization addresses the problem of data uncertainty by guaranteeing feasibility and optimizing the objective in the least favorable realization of the problem's data. With the uncertainty in parameters modeled lying in some convex, closed uncertainty sets (polyhedron, box, ellipsoid, etc..), the robust optimization problem usually takes the form of a minimax or maximin problem.

The first model in robust Linear Programming (LP) was proposed by Soyster [47]. In that model, all the parameters are set to satisfy the worst possible case. This method, however, produces a very conservative result. In addition, the model is only applicable to column-wise uncertainty. Consequently, the over-conservativeness was adrressed by Ben-Tal and Nemirovski [10, 11, 12], and El Ghaoui et al. [24]. They formulated the robust counterpart of the LP problem with an ellipsoidal uncertainy set as a quadratic conic programming problem. Their methods consider the polyhydral uncertainty set as a special case of the ellipsoidal set. The results are less conservative than Soyster's approach.

More recently, Bertsimas and Sim [16] proposed an approach based on duality trasformation to handle linear column-wise uncertainty set. While in approaches in [12, 24], the robust counterpart is more computationally expensive than the nominal problem, in the Bertsimas and Sim's approach, the robust counterpart is still a linear programming problem. Furthermore, both formulations yeild the same level of conservatism.

Bertsimas et al. [15] further extended this approach by considering an LP problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}'\mathbf{x} \\ \text{s.t:} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \in P^{\mathbf{x}} \\ & \mathbf{A} \in P^{A} \end{aligned} \tag{2.1}$$

with a more general uncertainty set:

$$P^{A} = \{\mathbf{A} \in \mathbb{R}^{\mathbf{mxn}} | \ ||\mathbf{\Sigma}^{-\frac{1}{2}}(\mathbf{vec}(\mathbf{A}) - \mathbf{vec}(\bar{\mathbf{A}}))||_{\mathbf{1}} \leq \mathbf{\Gamma}\} \tag{2.2}$$

They proved that if P^{x} is defined by r LP constraints and $\mathbf{b} \in \mathbb{R}^{l}$ then the robust LP problem equals a deterministic LP with $n + ml$ variables and $m^2n + m + ml + r$ constraints.

Ben-Tal et al. [9] also introduced the concept of adjustability in robust optimization. They considered the problem:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & \mathbf{c}'\mathbf{x} \\ \text{s.t:} \quad & \mathbf{Ax} + \mathbf{By} \leq \mathbf{b} \\ & \mathbf{A} \in \mathbb{Z}^{\mathbf{nxm}}, \mathbf{B} \in \mathbb{Z}^{\mathbf{lxm}}, \mathbf{m} \in \mathbb{Z}^{\mathbf{m}} \end{aligned} \tag{2.3}$$

where $\mathbf{x}$ are non-adjustable variables and $\mathbf{y}$ are adjustable variables. The Ajustable Robust Counterpart (ARC) is:

$$\begin{aligned} & \min_{\mathbf{x}} \quad \mathbf{c}'\mathbf{x} \\ & \forall \ \zeta = [\mathbf{A}, \mathbf{B}, \mathbf{b}] \ \exists y \quad \mathbf{Ax} + \mathbf{By} \leq \mathbf{b} \end{aligned} \tag{2.4}$$

The ARC approach, however, can be computationally intractable. In order to resolve this problem, they proposed an Affine Adjustable Robust Counterpart approach, where the adjustable variable $\mathbf{y}$ is modeled as: $\mathbf{y} = \mathbf{y_0} + \mathbf{W}\zeta$. Then, the robust counterpart can be formulated in a tractable form.

2.2 Robust Simulated Annealing

Bertsimas and Nohadani [14] considered the problem:

$$\min_{\mathbf{x}} g(\mathbf{x}) = \min_{\mathbf{x}} \max_{\Delta\mathbf{x} \in U} f(\mathbf{x} + \Delta\mathbf{x}) \tag{2.5}$$

The objective function f can be non-convex and even not given in explicit form. The variable $\mathbf{x}$ is continous. In order to solve the problem, [14] proposed a Robust Simulated Annealing approach, which is essentially similar to the normal Simulated Annealing method except for the following changes:

- At each iteration, we need to assembly a set $\mathcal{M}(\mathbf{x_k})$ of local maxima for the problem $\max_{\Delta\mathbf{x} \in U} f(\mathbf{x_k} + \Delta\mathbf{x})$
- The objective function is now the energy function of the set $\mathcal{M}(\mathbf{x_k})$:

$$W(\mathbf{x_k}) = \frac{1}{\beta} \log \sum_{\hat{\mathbf{x}} \in \mathcal{M}(\mathbf{x_k})} e^{\beta f(\hat{\mathbf{x}})} \tag{2.6}$$

  where β is the inverse temperature.

Note that we have $\lim_{\beta\to\infty} W(\mathbf{x_k}) = \max_{\hat{\mathbf{x}} \in \mathcal{M}(\mathbf{x_k})} f(\hat{\mathbf{x}})$.

With a proper cooling schedule and neighbour selection, [14] showed that the Robust Simulated Annealing method is able to converge to a global optimum. In Chapter 5, we will apply this algorithm to our discrete network design problem.

Chapter 3

User Equilibrium, VI and Optimization Formulations

3.1 User Equilibrium

For any transportation problem, it is essential to know how traffic will flow. In our work, we assume the flow will stabilize according to the User Equilibrium principle, or Wardrop Equilibrium principle (see [50]).

If users know exactly the travel times on each path, they choose the path that selfishly optimizes their own travel times. Then, when equilibrium is reached, any path carrying strictly positive flow between a given OD pair is a minimum travel time path for that OD pair. This implies that:

$$g_{p_i}(\mathbf{h}) = \mathbf{g_{p_j}}(\mathbf{h}) \quad \forall \mathbf{p_i}, \mathbf{p_j} \in \mathbf{P^w} \text{ s.t } \mathbf{h_{p_i}}, \mathbf{h_{p_j}} > \mathbf{0} \tag{3.1}$$

$$g_{p_i}(\mathbf{h}) > \mathbf{g_{p_j}}(\mathbf{h}) \quad \forall \mathbf{p_i}, \mathbf{p_j} \in \mathbf{P^w} \text{ s.t } \mathbf{h_{p_i}} = \mathbf{0}, \mathbf{h_{p_j}} > \mathbf{0}. \tag{3.2}$$

With such a flow, no user can reduce his travel time by changing his path. The following theorem in the literature states the existence and uniqueness of the User Equilibrium flow:

Definition A vector function $\mathbf{c} : \mathbf{X} \to \mathbb{R}^{\mathbf{n}}$ is monotone on $X \subset \mathbb{R}^n$ if it satisfies:

$$(\mathbf{c}(\mathbf{f_1}) - \mathbf{c}(\mathbf{f_2}))^{\mathbf{T}}(\mathbf{f_1} - \mathbf{f_2}) \geq \mathbf{0} \quad \forall \mathbf{f_1}, \mathbf{f_2} \in \mathbf{X} \text{ and } \mathbf{f_1} \neq \mathbf{f_2} \tag{3.3}$$

Moreover, if the inequality is strict, then the vector function is strictly monotone.

Theorem 3.1 *[22]: Consider a network $G(V, A)$ with continuous cost function $\mathbf{c}(\mathbf{f})$. There exists a user equilibirum flow for this network. This flow is unique if $\mathbf{c}(\mathbf{f})$ is strictly monotone function.*

This thesis adopts this principle to model the flow on a given network.

3.2 VI and optimization formulation

It is well known in the literature that we can find the User Equilibrium flow by solving the following VI problem:

Theorem 3.2 *[20]: Given a network $G(V, A)$ and a set of OD pairs with fixed demands. The UE flow $\bar{\mathbf{h}}$ satisfies:*

$$\mathbf{g}(\bar{\mathbf{h}})^{\mathbf{T}}(\mathbf{h} - \bar{\mathbf{h}}) \geq \mathbf{0} \tag{3.4}$$

$$\forall \text{ feasible flow } \mathbf{h} : \sum_{p \in P^w} h_p = d_w.$$

When the travel time is symmetric (the Hessian matrix of the travel time vector is symmetric) and separable, we can think of the VI formulation as the first order optimality condition for a minimization problem over a simplex set. Therefore, the following theorem applies:

Theorem 3.3 *With symmetric, separable path travel times, the equilibrium flow is* *the solution to the following optimization problem:*

$$\min_{\mathbf{h}} \sum_{p} \int_{0}^{h_p} g_p(s)ds \tag{3.5}$$
$$\textit{s.t:} \sum_{p \in P^w} h_p = d_w.$$

However, in reality, the travel time of each link and path is affected by the flow on other links and paths. Thus, we typically cannot make use of Theorem 3.3 in practice. In such case, we can employ the result by Aghassi, Bertsimas and Perakis [2] to formulate a general VI problem with asymmetric travel times as an optimization problem. The following theorem summarizes their result:

Theorem 3.4 *[2] Given a network $G(V, A)$ with travel time function $\mathbf{c}(\mathbf{f})$ on all links, the User Equilibrium flow is the solution to the following problem:*

$$\min_{\mathbf{f^w}, \lambda_\mathbf{w}} \mathbf{c}(\mathbf{f})^\mathbf{T}\mathbf{f} - \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{d}^{\mathbf{w}\mathbf{T}}\lambda^\mathbf{w} \tag{3.6}$$
$$\textit{s.t:} \; \mathbf{N}\mathbf{f^w} = \mathbf{d^w} \quad \forall \mathbf{w}$$
$$\mathbf{N}^\mathbf{T}\lambda^\mathbf{w} \leq \mathbf{c}(\mathbf{f}) \quad \forall \mathbf{w}$$
$$\mathbf{f^w} \geq \mathbf{0}, \; \mathbf{f} = \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{f^w}.$$

[2] shows that the optimal value of problem (3.6) is 0.
Theorem 3.4 implies that when $\mathbf{c}(\mathbf{f})^\mathbf{T}\mathbf{f}$ is a convex function and $\mathbf{c}(\mathbf{f})$ is a concave function, we can find the UE flow using a convex optimization problem.
If the travel time function is affine, i.e., $\mathbf{c}(\mathbf{f}) = \mathbf{G}\mathbf{f} + \mathbf{c_o}$, we can have the following corollary:

Corollary 3.5 *With affine travel time functions, the UE flow can be found by solving the following quadratic objective, linearly constrained problem:*

$$\min_{\mathbf{f^w}, \lambda_\mathbf{w}} (\mathbf{G}\sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{f^w} + \mathbf{c_o})^\mathbf{T} \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{f^w} - \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{d}^{\mathbf{w}\mathbf{T}}\lambda^\mathbf{w} \tag{3.7}$$

$$\begin{aligned} s.t.\text{: } & \mathbf{N}\mathbf{f}^{\mathbf{w}} = \mathbf{d}^{\mathbf{w}} \quad \forall \mathbf{w} \\ & \mathbf{N}^{\mathbf{T}}\lambda^{\mathbf{w}} \leq \mathbf{G}\sum_{\mathbf{w}=1}^{|\mathbf{W}|}\mathbf{f}^{\mathbf{w}} + \mathbf{c}_{\mathbf{o}} \quad \forall \mathbf{w} \\ & \mathbf{f}^{\mathbf{w}} \geq \mathbf{0}. \end{aligned}$$

The problem in Corollary 3.5 can be solved easily using commercial solvers. In Chapter 5, we will use this formulation to find the UE flow given the network design and fixed demands.

Chapter 4

The Inverse Optimization Problem to Find Travel Times

4.1 Introduction

Any transportation problem is strongly influenced by the travel times underlying the transportation network. In traffic assignment problems, the behavior of users is decided by their perception of travel times of different routes to their destinations. In the network design and routing problem, planners aim to minimize the total travel time of all users by controlling the flows and structures of the network. Therefore, understanding the relationship between the flow and the travel times is critical.

In the literature, there are many papers trying to model this relationship. Amongst those papers, researchers approach the problem in two different ways. One of them uses an analytical approach, modelling travel times as outputs of a dynamic flow model. Kachani and Perakis [25], Perakis and Roels [43] proposed a first and second order model using the hydro-dynamic model in Lighthill and Withham [28]. Using partial differential equation for wave propagation, those models describe the travel times better during the peak period. The second approach, which is a more traditional approach, models the travel times as some explicit functions of the traffic flow. Some of the commonly used functions are the Bureau of the Public Roads (BPR) function (see [38]), exponential travel times (see [35]) and polynomial-type functions (see [4, 5]). These functions are normally determined through statistical analysis. Even though they might neither be able to model the dynamic behavior of the traffic flow, nor be able to describe the peak period, they are simple and can describe fairly well the travel times when the flow does not reach the congested threshold of the network. Thus, they are widely used to study the traffic assigment and design problem in practice.

In line with the second approach, this chapter proposes a statistical, data-driven method to find travel times, given a stable flow and data on users' perception of the travel times. In real life, we can always observe some stable pattern of the flow, which can be modeled according to the User Equilibrium Principle.

The first part of this chapter proposes an inverse optimization problem to find the actual travel times at a stable flow. Given that the data are subjected to noise and uncertainty (users may have different perceptions, or we may have incorrect measurements of travel times), we model them in three different sets. We solve the robust inverse optimization problem for those sets. The second part of this chapter is a data mining problem. Given data on the historical flow, we try to find parametric functions to model the travel times.

4.2 Formulation

Non-parametric travel times

Deterministic problem

From day to day observations, we can observe some stable patterns of the flow $\bar{\mathbf{f}}$ on a network. Moreover, some prior approximation of the travel times $\bar{\mathbf{c}}$ for each link can also be obtained by surveying different users.

We assume each user chooses the route selfishly to minimize his own travel time. Then, the stable flow $\bar{\mathbf{f}}$ is the User Equilibrium flow, satisfying the following VI formulation:

$$\mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}}(\mathbf{f}-\bar{\mathbf{f}}) \geq \mathbf{0} \tag{4.1}$$
$$\text{s.t: } \mathbf{N}\mathbf{f}^{\mathbf{w}} = \mathbf{d}^{\mathbf{w}} \quad \forall \mathbf{w}$$
$$\mathbf{f}^{\mathbf{w}} \geq \mathbf{0}.$$

Due to uncertainty, $\bar{\mathbf{c}}$ might not be the actual travel time at flow $\bar{\mathbf{f}}$. We are interested in determining the value for the travel times $\mathbf{c}$ at the UE flow $\bar{\mathbf{f}}$, which represents the travel time at UE flow $\bar{\mathbf{f}}$ and is closest to our initial approximation $\bar{\mathbf{c}}$.

Using the idea of inverse optimization by Ahuja and Orlin [3], we can formulate the following problem to find the actual travel times.

$$\text{DInv:} \min_{\mathbf{c}(\bar{\mathbf{f}})} ||\mathbf{c}-\bar{\mathbf{c}}|| \tag{4.2}$$
$$\text{st: } \bar{\mathbf{f}} \text{ is UE flow with coresonding to } \mathbf{c}$$

In this section, we will use the L^1 norm ($||\mathbf{c}-\bar{\mathbf{c}}||_1 = \sum_k |c_k - \bar{c}_k|$) in our formulation of the inverse problem.

Robust problem

Problem (4.2) considers only a nominal approximation $\bar{\mathbf{c}}$. In reality, each user will have a slightly different perception of the travel times. Besides, our data may be limited, and $\bar{\mathbf{c}}$ may not be reliable.

We assume the prior travel times can be modeled as: $\tilde{c}_k = \bar{c}_k + \epsilon_k$ where $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_{|A|})$ belongs to an uncertainty set E. The robust version of this inverse optimization problem, thus, will become:

$$\text{RInv:} \min_{\mathbf{c}} \max_{\epsilon} ||\mathbf{c}-\bar{\mathbf{c}}-\boldsymbol{\epsilon}|| \tag{4.3}$$
$$\text{st: } \bar{\mathbf{f}} \text{ is the UE flow with coresonding to } \mathbf{c}$$
$$\boldsymbol{\epsilon} \in E.$$

There are several forms for set E. We will study the following uncertainty sets:

- Set 1: $E_1 = \{\boldsymbol{\epsilon} | l_i \leq \epsilon_i \leq u_i \quad \forall i\}$.
- Set 2: $E_2 = \{\boldsymbol{\epsilon} | \epsilon_i \geq 0 \, \forall i, \sum_i \epsilon_i \leq B\}$.
- Set 3: $E_3 = \{\boldsymbol{\epsilon} | l_i \leq \epsilon_i \leq u_i \, \forall i, \ \sum_i \epsilon_i \leq B\}$.

Parametric functions for travel times

In the previous part of this section, we try to find the travel times at a specific flow $\bar{\mathbf{f}}$. Assume the demands between each OD pair for the network varies through time, and we can obtain a set of historical stable flows $\bar{\mathbf{f}}_\mathbf{1}, \bar{\mathbf{f}}_\mathbf{2}, \ldots, \bar{\mathbf{f}}_\mathbf{H}$ corresponding to different demands.

We can therefore, solve H instances of the non-parametric travel time problem to obtain H different travel time vectors $\mathbf{c}(\bar{\mathbf{f}}_\mathbf{1}), \mathbf{c}(\bar{\mathbf{f}}_\mathbf{2}), \ldots, \mathbf{c}(\bar{\mathbf{f}}_\mathbf{H})$.

By solving a data mining problem on each link: fitting a curve to H data points, we then can obtain a parametric function for the travel times.

4.3 Algorithms

Non-parametric travel times

a) Deterministic problem

Parallel links: In this case we assume that there are N parralel links ($|A| = N$), one OD pair. We present the case of a single OD pair for the sake of simplicity of exposition (the case of multiple O-D pairs is very similar).

From the UE condition, let $I = \{i | \bar{f}_i > 0\}$ then $\forall i \in I \;\; c_i(\bar{\mathbf{f}}) = \lambda$ and $\forall i \in I^C \;\; c_i(\bar{\mathbf{f}}) \geq \lambda$. f_i denotes the flow on link i^{th} $(i = 1, \ldots, N)$

If we consider the L^1 norm, (4.2) becomes:

$$\min_{\lambda, \mathbf{c}} \sum_{i \in I} |\lambda - \bar{c}_i| + \sum_{i \in I^C} |c_i - \bar{c}_i| \tag{4.4}$$
$$\text{st: } c_i \geq \lambda \geq 0 \qquad \forall i \in I^C.$$

(4.4) can be solved by the following algorithm:

Algorithm A :

1. *Let $I' = \{i | \bar{f}_i > 0\}$ ($I' = I$ initially).*
2. *Sort I' in an ascending order in term of $\bar{c}_i$, let $\lambda^* = \bar{c}_{\lfloor \frac{|I'|+1}{2} \rfloor}$.*
3. *Pick $j = \arg\min_{i \in I'^C} \bar{c}_i$.*
4. *If $\bar{c}_j \geq \lambda^*$: Let $c_i^* = \lambda^* \forall i \in I'$ and $c_i^* = \bar{c}_i$ otherwise. Then, $\mathbf{c}^*$ denotes the solution.*
5. *Else let $I' = I' \cup j$. Repeat step 2.*

We have the following theorem:

Theorem 4.1 *Algorithm A converges to the optimal solution of Problem (4.4).*

Proof We first convert the problem into a linear optimization problem:

$$\begin{aligned}
&\min_{\theta,\lambda,\mathbf{c}} \sum_i \theta_i && (4.5)\\
\text{st: } &\theta_i \geq \lambda - \bar{c}_i \quad \forall i \in I\\
&\theta_i \geq -\lambda + \bar{c}_i \quad \forall i \in I\\
&\theta_i \geq c_i - \bar{c}_i \quad \forall i \in I^C\\
&\theta_i \geq -c_i + \bar{c}_i \quad \forall i \in I^C\\
&c_i \geq \lambda \geq 0 \qquad \forall i \in I^C\\
&\theta_i \geq 0 \quad \forall i.
\end{aligned}$$

Without loss of generality, let $I = \{1, 2, .., |I|\}$

From duality and complimentary slackness, λ^* and $\mathbf{c}^*$ represent the optimal solution if and only if:

$$1 \geq z_i^+ + z_i^- \quad \forall i \tag{4.6}$$

$$0 \geq \sum_{i=1}^{|I|} (-z_i^+ + z_i^-) - \sum_{i=|I|+1}^{|A|} \eta_i \tag{4.7}$$

$$0 \geq -z_i^+ + z_i^- + \eta_i \quad \forall i = |I|+1, \ldots, |A| \tag{4.8}$$

$$\eta_i(c_i^* - \lambda^*) = 0 \quad \forall i = |I|+1, \ldots, |A| \tag{4.9}$$

$$\theta_i^* = |\lambda^* - \bar{c}_i| \quad \forall i = |I|+1, \ldots, |A| \tag{4.10}$$

$$\theta_i^* = |c_i^* - \bar{c}_i| \quad \forall i = |I|+1, \ldots, |A| \tag{4.11}$$

$$z_i^+(\theta_i^* - \lambda^* + \bar{c}_i) = 0 \quad \forall i = 1, \ldots, |I| \tag{4.12}$$

$$z_i^-(\theta_i^* + \lambda^* - \bar{c}_i) = 0 \quad \forall i = |I|+1, \ldots, |A| \tag{4.13}$$

$$z_i^+(\theta_i^* - c_i^* + \bar{c}_i) = 0 \quad \forall i = |I|+1, \ldots, |A| \tag{4.14}$$

$$z_i^-(\theta_i^* + c_i^* - \bar{c}_i) = 0 \quad \forall i = |I|+1, \ldots, |A| \tag{4.15}$$

$$z_i^+, z_i^- \geq 0, \eta_i \geq 0. \tag{4.16}$$

Because the number of links is finite, the algorithm finds a solution in a finite number of steps and it produces a final set I'. Note that $I \subseteq I'$.

For $i \in I'$:

-If $\lambda_i^* - \bar{c}_i > 0$, let $z_i^+ = 1, z_i^- = 0$.

-If $\lambda_i^* - \bar{c}_i < 0$, let $z_i^+ = 0, z_1^- = 1$.

-If $\lambda_i^* - \bar{c}_i = 0$: If $|I'|$ odd, let $z_i^+ = z_1^- = 0$, else let $z_i^+ = 1, z_1^- = 0$.

For $i \in I'^C$ (we have $c_i^* = \bar{c}_i$): Let $z_i^+ = 1, z_1^- = 0$.

Those choices will satisfy the complimentary slackness conditions (4.12)-(4.15).

If $i \in I' \backslash I (c_i^* = \lambda^* \geq \bar{c}_i)$ and $z_i^+ = 1$, let $\eta_i = 1$. Else let $\eta_i = 0$. Those choices of η_i satisfy (4.8) and (4.9).

(4.6) follows as at most one of the z_i^+ and z_i^- can be 1.

With the above choices, the sum in the RHS of (4.7) will have $\lfloor \frac{|I'|}{2} \rfloor$ components equal to 1, $\lfloor \frac{|I'|}{2} \rfloor$ components equal to -1, and all other components equal to 0 . (4.7) satisfies with a strict equality.

Therefore, the algorithm produces an optimal solution of problem (4.4). □

General network: Consider a general network $\mathbf{G}(V, A)$ with a set of OD pairs W. The notations used below were defined in chapter 1. We will solve the inverse problem in order to find the travel time on each link.
We have the following lemma:

Lemma 4.2 *For a general network with the UE flow* $\bar{\mathbf{f}}$, *the link travel times satisfy the following conditions:*

$$\begin{aligned} c_{ij}(\bar{\mathbf{f}}) &= \lambda_i^w - \lambda_j^w, && \text{if } \bar{f}_{ij}^w > 0, \\ c_{ij}(\bar{\mathbf{f}}) &\geq \lambda_i^w - \lambda_j^w, && \text{if } \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.17}$$

Proof For $|W| = 1$:
The UE flow satisfy the VI condition:

$$\begin{aligned} &\bar{\mathbf{f}} = \arg\min_{\mathbf{f}} \mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}}\mathbf{f} \\ &\text{st: } \mathbf{N}\mathbf{f} = \mathbf{d} \\ &\quad f_{ij} \geq 0 \quad \forall (i,j) \in A. \end{aligned}$$

From duality of the above linear optimization problem, we have that:

$$\begin{aligned} &\mathbf{c}(\bar{\mathbf{f}}) - \mathbf{N}^{\mathbf{T}}\boldsymbol{\lambda} - \mathbf{I}\boldsymbol{\theta} = \mathbf{0} \\ &\bar{f}_{ij}\theta_{ij} = 0; \theta_{ij} \geq 0 \quad \forall (i,j) \in A. \end{aligned}$$

Note that a column of matrix $\mathbf{N}$ only has entry 1 at row i^{th} and entry -1 at row j^{th}. Thus we can rewrite the above condition as:

$$\begin{aligned} c_{ij}(\bar{\mathbf{f}}) &= \lambda_i - \lambda_j, && \text{if } \bar{f}_{ij} > 0, \\ c_{ij}(\bar{\mathbf{f}}) &\geq \lambda_i - \lambda_j, && \text{if } \bar{f}_{ij} = 0. \end{aligned}$$

For $|W| > 1$, we have that $\mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}}\mathbf{f} = \sum_{\mathbf{w}} \mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}}\mathbf{f}^{\mathbf{w}}$. □

With the above lemma, we have the following theorem:

Theorem 4.3 *The inverse UE problem (4.2) can be rewritten as a linear optimization problem; that is:*

$$\begin{aligned}
&\min_{\mathbf{c},\lambda} \sum_{(i,j)\in A} |c_{ij} - \bar{c}_{ij}| && (4.18)\\
\textit{st: } & c_{ij} = \lambda_i^w - \lambda_j^w \quad \textit{if } \bar{f}_{ij}^w > 0\\
& c_{ij} \geq \lambda_i^w - \lambda_j^w \quad \textit{if } \bar{f}_{ij}^w = 0\\
& c_{ij} \geq 0.
\end{aligned}$$

Problem (4.18) can be converted into a linear optimization problem by introducing new variables and new linear constraints to represent the absolute value function.

b) Robust problem:

This part considers a general network with different uncertainty sets.

Set E_1:

The robust inverse problem is:

$$\begin{aligned}
&\min_{\mathbf{c},\lambda} \max_{\epsilon} \sum |c_{ij} - \bar{c}_{ij} - \epsilon_{ij}| && (4.19)\\
\text{st: } & l_{ij} \leq \epsilon_{ij} \leq u_{ij} \quad \forall (i,j) \in A\\
& c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0,\\
& c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0.
\end{aligned}$$

The inner problem is separable. Let $t_{ij} = c_{ij} - \bar{c}_{ij}$ and consider the following problem:

$$\begin{aligned}
Adv_k = &\max_{\epsilon_k} |t_k - \epsilon_k| && (4.20)\\
&\text{st: } l_k \leq \epsilon_k \leq u_k.
\end{aligned}$$

Here, we use k to denote the k^{th} component in the vector. The solution to the above problem is:

- If $\frac{u_k+l_k}{2} > t_k$ then $\epsilon_i^* = l_k$,
- If $\frac{u_k+l_k}{2} \leq t_k$ then $\epsilon_i^* = u_k$.

We have that $Adv_k = |t_k - \frac{u_k+l_k}{2}| + \frac{u_k-l_k}{2}$. Therefore, we have the following theorem:

Theorem 4.4 *Problem (4.19) is equivalent to:*

$$\begin{aligned} \min_{\mathbf{c}} \sum_{(i,j)\in A} \left| c_{ij} - \bar{c}_{ij} - \frac{u_{ij}+l_{ij}}{2} \right| + \sum \frac{u_{ij}-l_{ij}}{2} \\ st\text{: } c_{ij} = \lambda_i^w - \lambda_j^w, \quad if\ \bar{f}_{ij}^w > 0, \\ c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad if\ \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.21}$$

Note that if $\frac{u_{ij}+l_{ij}}{2} = 0\ \forall (i,j) \in A$, the solution to robust problem (4.21) is the same as the solution to the deterministic problem (4.18).

Set E_2:

The robust problem in this case becomes:

$$\begin{aligned} \min_{\mathbf{c},\lambda} \max_{\epsilon} \sum |c_{ij} - \bar{c}_{ij} - \epsilon_{ij}| \\ \text{st: } \sum_{ij} \epsilon_{ij} \leq B, \quad \epsilon_{ij} \geq 0 \quad \forall (i,j) \in A \\ c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0, \\ c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.22}$$

Consider the inner problem:

$$\begin{aligned} Adv(\mathbf{t}) = \max_{\epsilon} \sum_{(i,j)} |t_{ij} - \epsilon_{ij}| \\ \text{st: } \sum_{(i,j)\in A} \epsilon_{ij} \leq B, \epsilon_{ij} \geq 0. \end{aligned} \tag{4.23}$$

This is a convex maximization problem. The maximum is obtained at an extreme point of the set.

The extreme points of the feasible set in problem (4.23) are:

$$(0,0,\ldots,0),(B,0,0,\ldots,0),\ldots,(0,0,\ldots,B)$$

Thus,

$$\begin{aligned} Adv(\mathbf{t}) = \max(&|t_1|+|t_2|+\ldots+|t_{|A|}|, |t_1 - B|+|t_2|+\ldots+|t_{|A|}|,\ldots \\ &\ldots,|t_1|+|t_2|+\ldots+|t_{|A|}-B|). \end{aligned} \tag{4.24}$$

(4.24) and (4.22) lead to the following theorem:

Theorem 4.5 *The robust inverse UE problem (4.22) can be rewritten as the following linear optimization problem:*

$$\min_{\mathbf{c},\theta,\lambda} \theta \tag{4.25}$$

$$\begin{aligned} st\text{: } &\theta \geq \sum_{(i,j)\in A} |c_{ij}-\bar{c}_{ij}| \\ &\theta \geq |c_{ij}-\bar{c}_{ij}-B| + \sum_{A\backslash(i,j)} |c_{pq}-\bar{c}_{pq}| \quad \forall (i,j)\in A \\ &c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0, \\ &c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0. \end{aligned}$$

We have that:

$$\max(|t_i|,|t_i-B|) = \begin{cases} |t_i|+B, & \text{if } t_i \leq 0, \\ B-|t_i|, & \text{if } 0 < t_i < B/2, \\ |t_i|, & \text{if } B/2 \leq t_i. \end{cases} \tag{4.26}$$

Therefore:

$$Adv(\mathbf{t}) = \begin{cases} B + \sum |t_i|, & \text{if } \exists i : t_i < 0, \\ \sum |t_i|, & \text{if } B/2 \leq t_i \ \ \forall i, \\ \sum |t_i| + (B - 2|t_{min}|), & \text{otherwise.} \end{cases} \tag{4.27}$$

(4.27) shows that if only one link in the network satisfies $t_{ij} = c_{ij} - \bar{c}_{ij} \leq 0$, then problem (4.22) is:

$$\begin{aligned} \min_{\mathbf{c},\lambda} & \sum |c_{ij} - \bar{c}_{ij}| + B \\ \text{st: } & c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0, \\ & c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.28}$$

This gives us the same solution as the deterministic problem (4.18). In general network, where we have a huge number of links, this is very likely to happen.

Set E_3:

The robust inverse problem in this case is:

$$\begin{aligned} & \min_{\mathbf{c},\lambda} \max_{\epsilon} \sum |c_{ij} - \bar{c}_{ij} - \epsilon_{ij}| \\ \text{st: } & \sum_{(i,j)\in A} \epsilon_{ij} \leq B \\ & l_{ij} \leq \epsilon_{ij} \leq u_{ij} \\ & c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0, \\ & c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.29}$$

In this case the inner problem is not separatable, and it is difficult to enumerate all the extreme points of this set. [18] showed that the norm maximization problem over a linear set is an NP hard problem. Therefore, we are unable to rewrite the min max problem as one-stage optimization problem.

However, the two stage optimization algorithm proposed by Bienstock and Özbay [17] can be used to solve problem (4.29). Instead of considering the whole set E, we only consider a discrete set E_s of all vectors $\boldsymbol{\epsilon}$ of interest. The decision problem, which is the outer minimization problem, solves the min max problem restricted to our sample set E_s to find a solution vector $\mathbf{c}^*$ and to obtain a lower bound for (4.29). Then, after fixing $\mathbf{c} = \mathbf{c}^*$, we solve the inner maximization problem (adversary problem) over the whole uncertainty set E to find the worst uncertainty vector $\boldsymbol{\epsilon}^*$ and to obtain an upper bound for (4.29). We then add ϵ^* to E_s and re-iterate until the lower bound is close enough to the upper bound. The whole algorithm can also be considered as one form of Bender's decomposition [13].

Initally let $E_s = \emptyset$.

Decision (outer) problem:

$$\begin{aligned} De = &\min_{\mathbf{c},\lambda,\theta} \theta \\ \text{st: } &\theta \geq ||\mathbf{c} - \bar{\mathbf{c}}||_1 \\ &\theta \geq ||\mathbf{c} - \bar{\mathbf{c}} - \boldsymbol{\epsilon}||_1 \quad \forall \boldsymbol{\epsilon} \in E_s \\ &c_{ij} = \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w > 0, \\ &c_{ij} \geq \lambda_i^w - \lambda_j^w, \quad \text{if } \bar{f}_{ij}^w = 0. \end{aligned} \tag{4.30}$$

Let $lb = De$. Let $\mathbf{c}^*$ be the solution of above problem, then:

Adversary (inner) problem:

$$\begin{aligned} Adv = &\max_{\epsilon} \sum_k |c_k^* - \bar{c}_k - \epsilon_k| \\ \text{st: } &\sum_k \epsilon_k \leq B \\ &l_k \leq \epsilon_k \leq u_k. \end{aligned} \tag{4.31}$$

In (4.31), we use the index k to denote the k^{th} component of the vector instead of using a pair (i, j) for each arc.

Let $ub = \min(ub, Adv)$. We stop when $ub - lb$ is small enough.

The decision problem is a linear optimization problem. However, problem (4.31) is a non-separable convex maximization problem over a convex set. In the rest of this section, we will solve problem (4.31).

Let $\mathbf{t} = \mathbf{c}^* - \bar{\mathbf{c}}$. Consider the adversary problem:

$$
\begin{aligned}
Adv(\mathbf{t}) = &\max_{\epsilon} \sum_k |t_k - \epsilon_k| \\
\text{st:} &\sum_k \epsilon_k \leq B \\
&l_k \leq \epsilon_k \leq u_k.
\end{aligned} \tag{4.32}
$$

Lemma 4.6 *There exists an optimal solution $\boldsymbol{\epsilon}^*$ with the following properties:*

(a) ϵ^* can have at most one component such that $l_k < \epsilon_k^* < u_k$. When we have such a component, $\sum_k \epsilon_k = B$. All other components have to be either at the upper bound or the lower bound.

(b) If $l_k < \epsilon_k^* < u_k$ then $\epsilon_k^* \geq t_k$.

Proof The problem is a convex maximization problem over a convex set. Therefore, an optimal solution exists at an extreme point. Since extreme points of the feasible set satisfies part (a), the theorem follows.
If $l_k < \epsilon_k^* < u_k$ and $t_k > \epsilon_k^*$ then $|t_k - \epsilon_k^*| = t_k - \epsilon_k^* < t_k - l_k = |t_k - l_k|$. We can set $\epsilon_k^* = l_k$ and increase the objective. Thus, $\epsilon_k^* \geq t_k$ if $l_k < \epsilon_k^* < u_k$. Part (b) follows. □

Let $B_s = B - \sum l_k$, $r_k = |u_k - t_k| - |l_k - t_k|$ and $w_k = u_k - l_k$, the following theorem applies:

Theorem 4.7 *The optimal solution of (4.32) can be found by solving the following mixed integer optimization problem:*

$$\max_{\mathbf{x},\mathbf{z},\mathbf{y}} \sum_k r_k x_k + y_k \tag{4.33}$$

$$s.t.\text{:} \sum_k w_k x_k \leq B_s \tag{4.34}$$

$$x_k + z_k \leq 1 \tag{4.35}$$

$$\sum_k z_k \leq 1 \tag{4.36}$$

$$y_k \leq M z_k \tag{4.37}$$

$$y_k \leq l_k + B_s - \sum w_p x_p - t_k - |l_k - t_k| + M(1 - z_k) \tag{4.38}$$

$$l_k + B_s - \sum w_p x_p - t_k \geq -M(1 - z_k) \tag{4.39}$$

$$l_k + B_s - \sum w_p x_p \leq u_k + M(1 - z_k) \tag{4.40}$$

$$x_k, z_k \in \{0, 1\}.$$

Let $\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*$ *to be the solution of the above mixed integer optimization problem, The solution to problem (4.32) is:* $\epsilon_k^* = l_k + w_k x_k^* + z_k^*(B_s - \sum w_p x_p^*)\ \forall k$.

Proof In the above formulation, x_k denotes whether ϵ_k at the lower bound or upper bound. z_k denotes whether ϵ_k lies in the interior of two bounds, i.e $l_k < \epsilon_k < u_k$. From Lemma 4.6a, we have that:

$$\epsilon_k = l_k + w_k x_k + z_k(B - \sum_p w_p x_p) \tag{4.41}$$

When $z_k = 0\ \forall k$, condition (4.34) makes sure we do not violate the budget.
When we have a component lies in the interior of two bounds, ie: $l_k < \epsilon_k < u_k$ for some k, conditions (4.35) and (4.36) guarantee that we only have at most one component. Conditions (4.39) and (4.40) guarantee $t_k \leq e_k \leq u_k$ for such component (Lemma 4.6b). Condition (4.34) and the way we represent the solution in (4.41) make sure that we have $\sum_k \epsilon_k = B$ in this case.
We use the point such that $\epsilon_k = l_k\ \forall k$ as the level of reference for the objective function

in (4.32). The objective function in (4.33) calculates the gain of other extreme points with respect to that reference point.
$r_k = |u_k - t_k| - |l_k - t_k|$ denotes the gain with respect to the reference point when $\epsilon_k = u_k$.
y_k denotes the gain if the component k^{th} lies in the interior of the bounds, ie: $l_k < \epsilon_k < u_k$. Because at optimality, we have that $\epsilon_k \geq t_k$ (Lemma 4.6b), thus:

$$\begin{aligned} y_k &= |\epsilon_k - t_k| - |t_k - l_k| \\ &= \epsilon_k - t_k - |t_k - l_k| \\ &= l_k + w_k x_k + z_k(B_s - \sum_p w_p x_p) - t_k - |t_k - l_k| \end{aligned}$$

We can rewrite it in linear form using conditions (4.37) and (4.38) for a sufficiently big number M. Those conditions make sure $y_k = 0$ when $z_k = 0$ and y_k has the appropriate value when $z_k = 1$.
Therefore, by solving the mixed integer optimization problem, we can find the extreme points with the largest objective value, which is the solution to our adversary problem.□

Note that from above theorem, if we disregard all the extreme points that has a component $l_k < \epsilon_k^* < u_k$, the problem is a knapsack problem with real weights and rewards (an NP hard problem). Besides, our simulation shows that we can choose a relatively large constant M without affecting the running time of the solver (CPLEX).

Parametric functions for travel times

The rest of this section shows how we generate the data and solve the problem to find patrametric functions for the travel times. Because we don't have the required data for this model, we will instead use the following algorithm to simulate the data $\bar{\mathbf{f}}_1, \bar{\mathbf{f}}_2, \dots, \bar{\mathbf{f}}_H$ and the corresponding prior information on travel time $\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2, \dots, \bar{\mathbf{c}}_H$:

1. Assume a parametric form for the real travel time (for example, a separable linear form: $c^r_{ij}(\mathbf{f}) = a^r_{ij} f_{ij} + b^r_{ij}$).

2. Randomly generate H different demands for the network. Solve the UE problem with each demand and the above parametric function to obtain $\bar{\mathbf{f}}_{\mathbf{1}}, \bar{\mathbf{f}}_{\mathbf{2}}, \ldots, \bar{\mathbf{f}}_{\mathbf{H}}$.
3. Let $\bar{\mathbf{c}}_{\mathbf{k}} = \mathbf{c}^{\mathbf{r}}(\bar{\mathbf{f}}_{\mathbf{k}}) + \boldsymbol{\epsilon}_{\mathbf{k}}$ with $\boldsymbol{\epsilon}_{\mathbf{k}}$ is a uncertainty vector $(k = 1, \ldots, H)$.

Then, we can solve H instances of the non-parametric travel time problem to obtain H different travel time vectors $\mathbf{c}(\bar{\mathbf{f}}_{\mathbf{1}}), \mathbf{c}(\bar{\mathbf{f}}_{\mathbf{2}}), \ldots, \mathbf{c}(\bar{\mathbf{f}}_{\mathbf{H}})$.
By solving a data mining problem on each link: fitting a curve to H data points, we then can obtain a parametric function for the travel time.

4.4 Computational Results

In this section, we will apply the above algorithm to the classic Sioux Falls network. This is a test network is well-known in the literature. The Sioux Falls network has 24 nodes, 76 arcs (see Figure 5-2 for more details).

Non-parametric travel times

Deterministic problem

Figure 4-1 shows the solution to the deterministic problem and the corresponding prior travel time on each link. The total error, or the L^1 distance from the solution to the prior information $(||\mathbf{c}^* - \bar{\mathbf{c}}||_1)$, is very small.

Robust problem

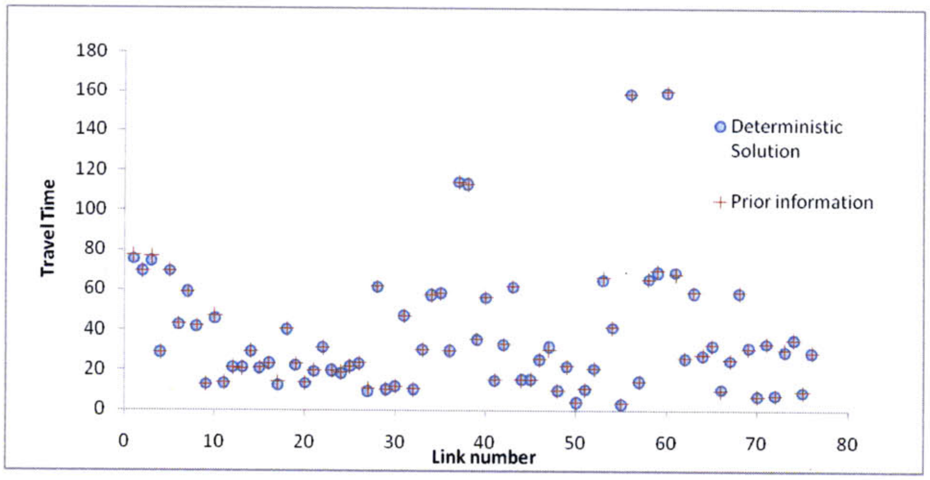

Figure 4-1: Deterministic solution. Total error = 20.

-Set E_1:

Consider the case when $\mathbf{u} = -0.05\bar{\mathbf{c}}$ and $\mathbf{l} = 0.1\bar{\mathbf{c}}$ (Note that when $\mathbf{l} + \mathbf{u} = \mathbf{0}$, the deterministic and robust solutions are the same).

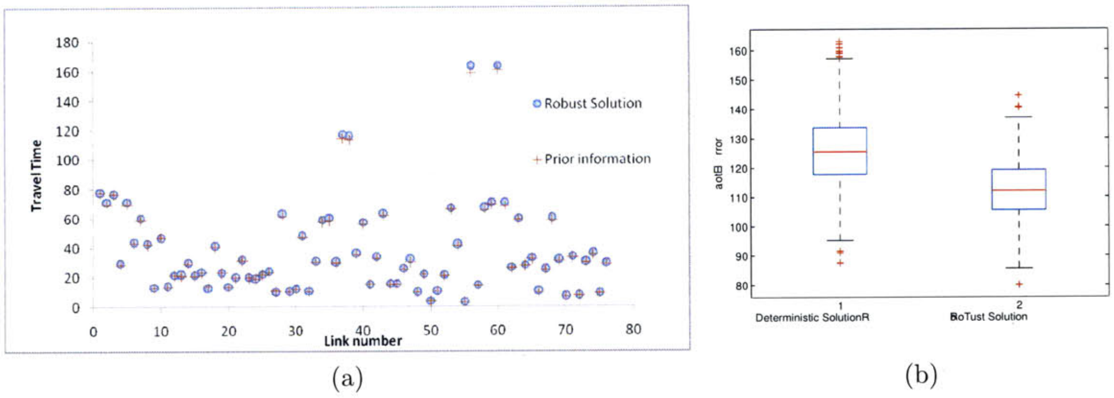

Figure 4-2: Robust solution (a) for Set E_1 and box plot (b) for 1000 cases

Figure 4-2a shows the result for the robust problem. We also sample 1000 points $\boldsymbol{\epsilon} \in E_1$ from the uncertainty set and plot the box-plot for the total errors of sampled scenarios ($||\mathbf{c}^* - \bar{\mathbf{c}} - \boldsymbol{\epsilon}||_1$) for the deterministic and robust solutions respectively in figure 4-2b. For the deterministic solution, with some uncertainty in the prior travel times, total errors of sampled scenarios increase significantly compared to the nominal total error ($||\mathbf{c}^* - \bar{\mathbf{c}}||_1$) (from 20 to around 130). The robust solution shows significant improvement in this case compared to the deterministic solution.

-Set E_2:

Figure 4-3 shows the solution for the uncertainty set E_2.

Interestingly, all of our simulation results for the Sioux Falls network suggest that the result for the robust problem is the same as the result for the deterministic problem. Only for networks with a small number of links, we can observe the differences between the two solutions.

-Set E_3:

The stopping criteria for the decomposition algorithm is $ub - lb \leq 0.5\% lb$.

Figure 4-4 shows the solution and the box-plot for 1000 samples when $B = 140$. Clearly, the robust solution provides a better mean and variance for the sampled

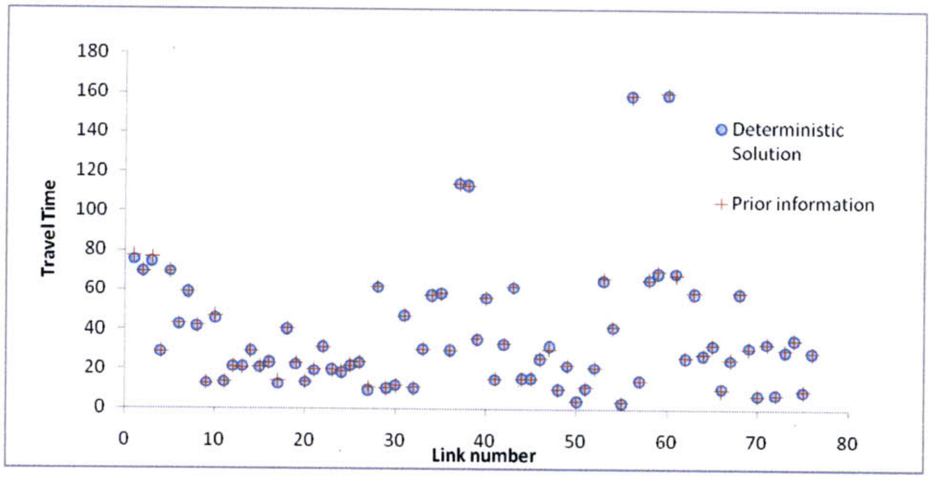

Figure 4-3: Robust solution for Set E_2 when B = 50. Total error = 20.

scenarios.

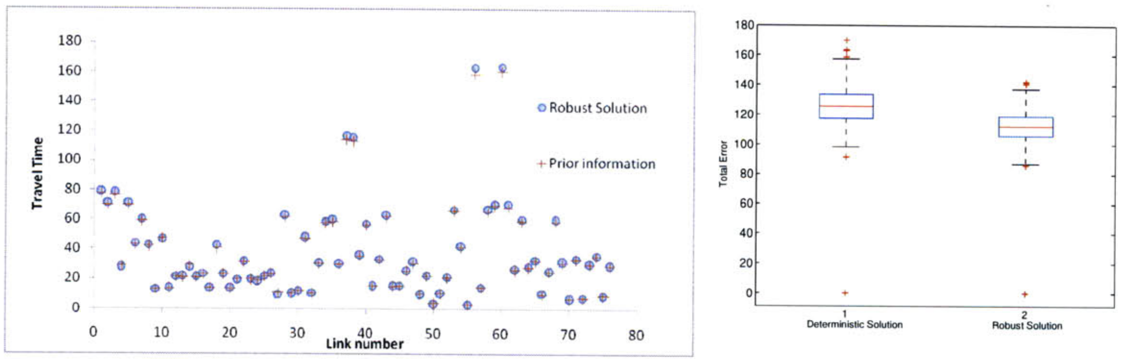

Figure 4-4: Robust solution (a) for Set E_3 and box plot (b) for 1000 cases when $B = 140$

Figure 4-5 shows the total error of the worst case and the nominal total error for the deterministic and robust solutions and when the uncertainty budget change. The worst case total error for the deterministic solution increases as the budget increases. On the other hand, the worst case total error for the robust solution increases with a decreasing rate and stables after a while. The nominal error for the robust solution decreases and then stables for the robust solution. This is an attractive property of the robust solution.

Figure 4-6 gives the number of iterations needed for our decomposition algorithm.

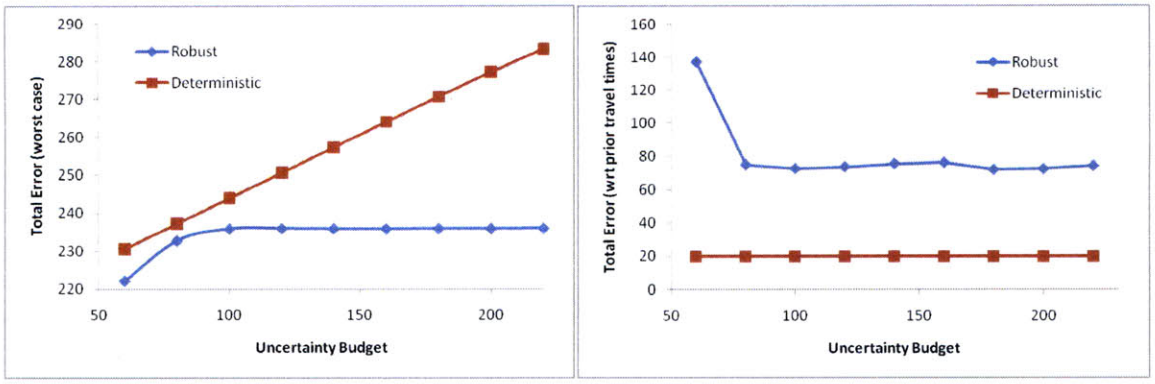

Figure 4-5: Worst case error (a) and nominal case error (b) with varying uncertainty budget

For this network, the dimension of the uncertainty set is 76. In order to solve the problem to optimality, we need to visit all extreme points of this uncertainty set. However, the decomposition algorithm stops after visiting only around 400 extreme points. This is rather efficient for large-scale problems.

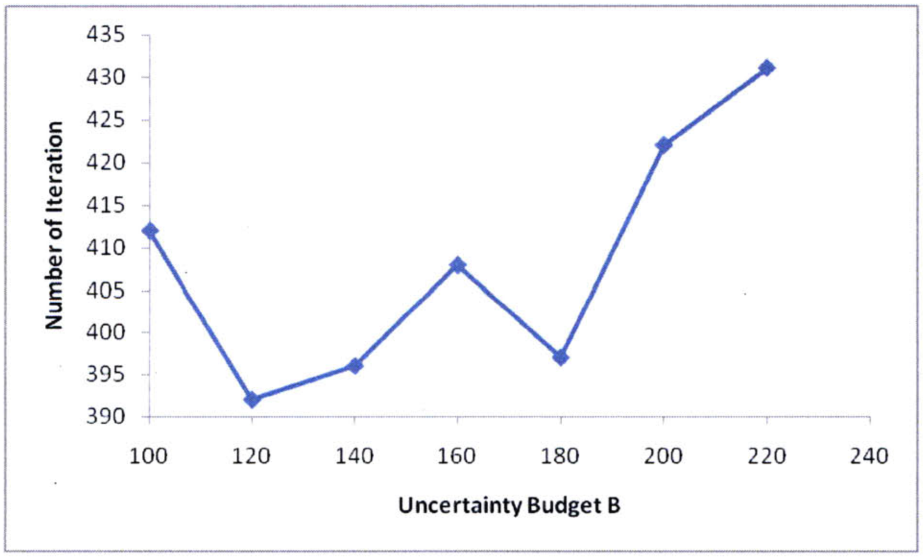

Figure 4-6: Number of iteration with varying budget B

Parametric travel times

In this section, we use 50 historical data points ($H = 50$).

Linear separable travel times :

In this part, to run our simulation, we assume the travel times have the form: $c_{ij}^r(\mathbf{f}) = a_{ij}^r f_{ij} + b_{ij}^r$ (see Table 5.3). We generate the uncertainty vector $\boldsymbol{\epsilon}$ such that: $|\epsilon_{ij}| \leq Lc_{ij}^r$. L can be thought of as an uncertainty level.

Figures 4-7 and 4-8 show how to fit parameters a and b for each link when $L = 20\%$. Apparently, the linear term a is fitted closely to the real value. However, there are some differences in fitting the constant term b. This is understandable, as one property of UE is that if we shift the whole travel times on all paths by the same constant, the UE flow does not change.

We also compare the results of our method with the results when we directly fit

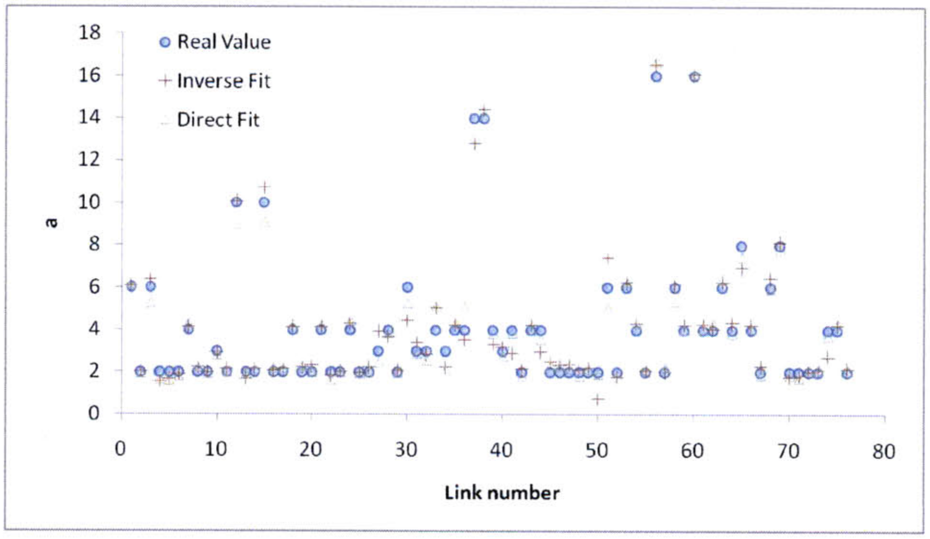

Figure 4-7: Parameter a when L = 20%

a curve to the given data. Figure 4-9 shows the total squared error of the fitted travel times and real travel times $\mathbf{c^r}$ over all H given flow ($\sum_{i=1}^{\mathbf{H}} ||\mathbf{c}^* - \mathbf{c_i^r}||_{\mathbf{2}}^{\mathbf{2}}$) against the uncertainty level we added to the real travel times. It shows that by solving the inverse optimization before carrying out the fitting process, we can improve the accuracy.

BPR travel times :

In this case, the actual travel time function for the simulation will be: $c_{ij}^r = a_{ij}^r f_{ij}^4 + b_{ij}^r$. The coefficients are taken from [8]. From the data, we will try to fit a parametric

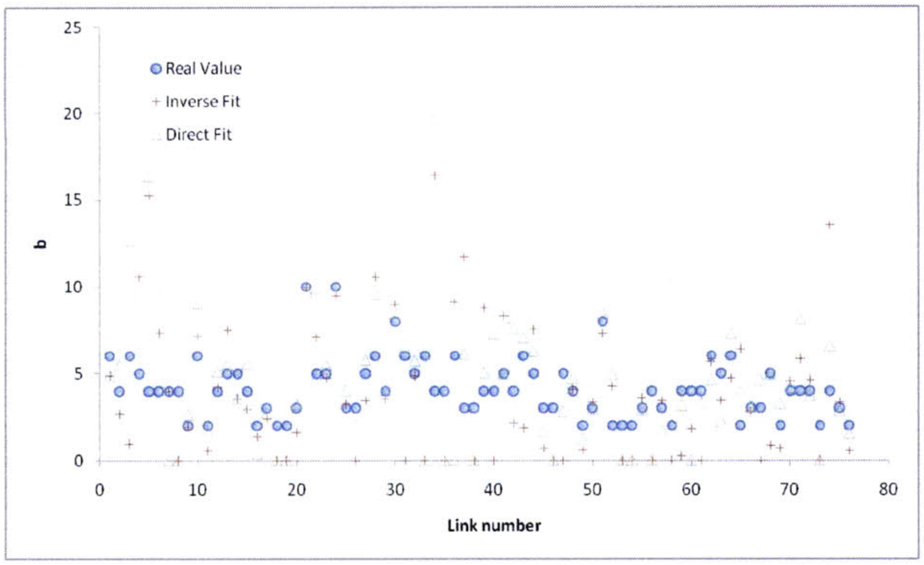

Figure 4-8: Parameter b when L = 20%

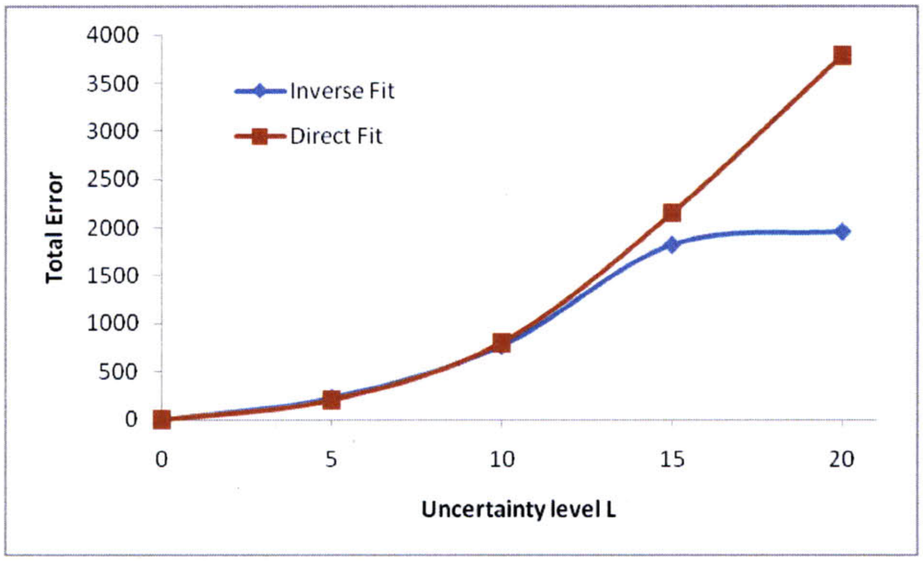

Figure 4-9: Fitting total squared error with varying uncertainty level(%)

travel time with the BPR general form:

$$c_{ij} = a_{ij} f_{ij}^{p} + b_{ij} \tag{4.42}$$

where a_{ij}, b_{ij}, and p are unknown parameters.

First, we will fix p. Then, a linear regression will be employed in order to find a_{ij}, b_{ij} and the total squared error to the given data. The value of p with the smallest total error will be chosen. By scanning p in [0.5 5], step size 0.5, the result in Figure 4-10

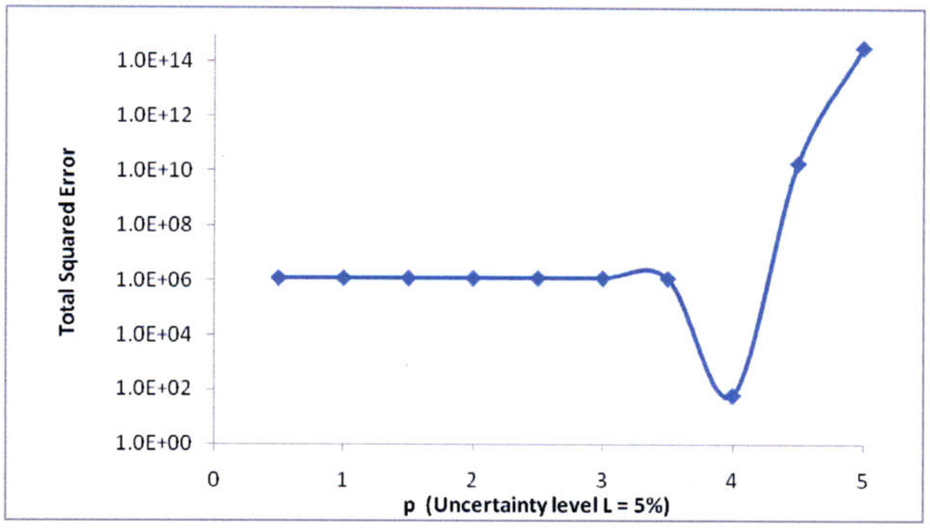

Figure 4-10: Fitting total squared error with varying p(L = 5%)

shows that $p = 4$ is the best value.

Figures 4-11 and 4-12 show the parameter **a** and **b** when the uncertainty level $L =$ 10%. It shows that we can recover pretty well the parameters using our inverse algorithm.

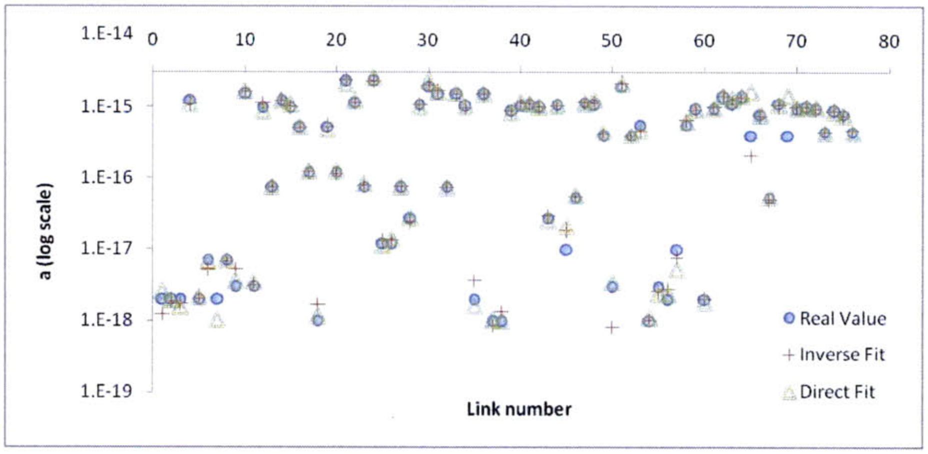

Figure 4-11: Fitting parameter a when L=10%

Figure 4-13 shows the total squared error of the fitted travel times and real travel times $\mathbf{c^r}$ over all $\mathbf{H}$ given flow with varying uncertainty level. It shows that when the uncertainty level is small, the direct fitting and the inverse fitting is quite similar. However, when the uncertainty level gets bigger, the inverse algorithm provides much

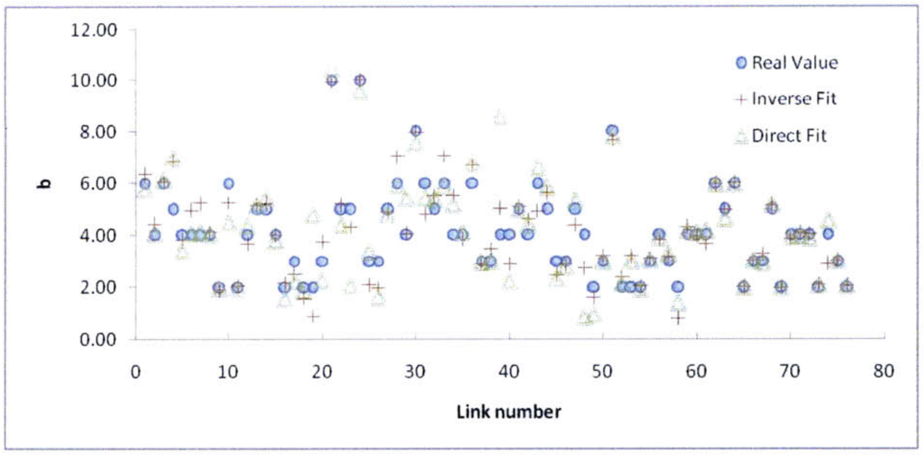

Figure 4-12: Fitting parameter b when L=10%

better results.

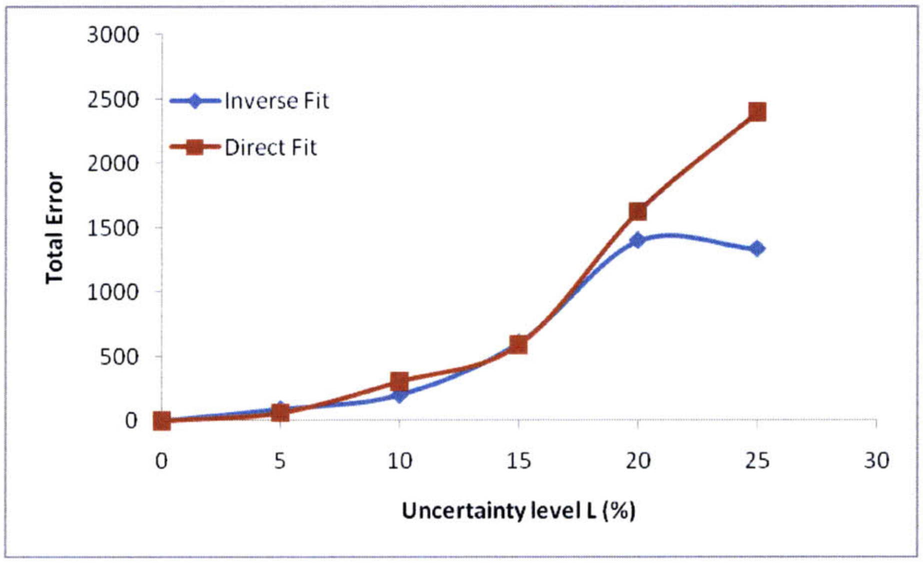

Figure 4-13: Total squared error with varying uncertainty level L (%)

Asymmetric travel times

We further apply our approach to a network (Network 1) with asymmetric travel times (see Figure 5-1 and Table 5.2). Table 4.1 shows the functional travel times found by the inverse optimization method. We can recover quite well the linear terms in most of the links.

In this case, we cannot recover the correct travel times for some of the links. When the uncertainty level increases, it is really hard to recover the underlying travel time

$c_1(\mathbf{f}) = 5.17f_1 + 2.88f_2 + 10.44$	$c_{15}(\mathbf{f}) = 9.83f_{15} + 2.66f_{14} + 1.314$
$c_2(\mathbf{f}) = 3.42f_2 + 1.21f_1 + 36.64$	$c_{16}(\mathbf{f}) = 8.16f_{16} + 6.35f_{12} + 8.27$
$c_3(\mathbf{f}) = 3.67f_3 + 1.05f_4 + 0$	$c_{17}(\mathbf{f}) = 6.41f_{17} + 0.01f_{15} + 44.71$
$c_4(\mathbf{f}) = 9.23f_4 + 0.02f_5 + 0$	$c_{18}(\mathbf{f}) = 6.31f_{18} + 3.38f_{16} + 0$
$c_5(\mathbf{f}) = 6.39f_5 + 3.98f_6 + 65.16$	$c_{19}(\mathbf{f}) = 9.53f_{19} + 1.74f_{17} + 27.54$
$c_6(\mathbf{f}) = 7f_6 + 3f_7 + 38.92$	$c_{20}(\mathbf{f}) = 7.45f_{20} + 0.01f_{21} + 32.67$
$c_7(\mathbf{f}) = 9.64f_7 + 0.01f_8 + 45.72$	$c_{21}(\mathbf{f}) = 3.45f_{21} + 1.29f_{22} + 47.41$
$c_8(\mathbf{f}) = 7.33f_8 + 0.56f_9 + 156.41$	$c_{22}(\mathbf{f}) = 6.27f_{22} + 1.45f_{23} + 26.39$
$c_9(\mathbf{f}) = 3.82f_9 + 2.04f_{10} + 109.78$	$c_{23}(\mathbf{f}) = 10.91f_{23} + 1.42f_{24} + 0$
$c_{10}(\mathbf{f}) = 3.52f_{10} + 2.11f_{12} + 70.80$	$c_{24}(\mathbf{f}) = 8.10f_{24} + 1.38f_{25} + 7.29$
$c_{11}(\mathbf{f}) = 4.84f_{11} + 3.95f_{12} + 95.92$	$c_{25}(\mathbf{f}) = 9.74f_{25} + 2.11f_{26} + 37.97$
$c_{12}(\mathbf{f}) = 6.11f_{12} + 3.21f_{13} + 85.95$	$c_{26}(\mathbf{f}) = 12.05f_{26} + 2.58f_{27} + 66.41$
$c_{13}(\mathbf{f}) = 8.07f_{13} + 2.08f_{18} + 65.11$	$c_{27}(\mathbf{f}) = 7.28f_{27} + 3.87f_{28} + 56.05$
$c_{14}(\mathbf{f}) = 4.41f_{14} + 3.24f_{15} + 102.85$	$c_{28}(\mathbf{f}) = 6.67f_{28} + 2.04f_{27} + 107.94$

Table 4.1: Link travel times by the inverse optimization approach when L = 10%
.

functions (in contrast, in our simulations with the Sioux Falls network, even when the uncertainty level is around 20%, we still can recover the parametric form quite accurately).

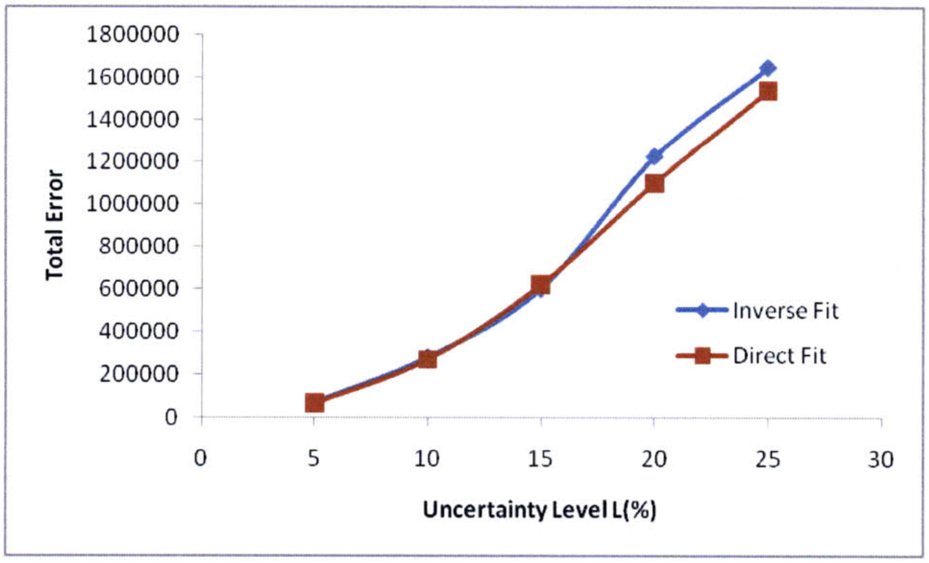

Figure 4-14: Total Fitting Error with varying uncertainty level

Figure 4-14 shows the total errors for the inverse fitting and the direct fitting with different uncertainty levels. It shows that the total fitting error diverges for both

1	7.5485	11	3.82478	21	3.84297	31	7.22923	41	8.94077
2	2.30665	12	8.78401	22	7.14774	32	4.51	42	5.99369
3	0.485115	13	5.52063	23	1.92086	33	7.36483	43	8.64232
4	2.63142	14	5.60239	24	3.09926	34	6.47613	44	5.60672
5	9.72079	15	2.01232	25	5.93617	35	7.23411	45	2.56693
6	5.84619	16	2.01421	26	3.83302	36	2.41407	46	0.0274235
7	2.15589	17	6.9382	27	1.8068	37	5.22737	47	2.48626
8	4.98802	18	3.80901	28	5.75866	38	8.00048	48	2.40043
9	3.80157	19	7.15673	29	3.36697	39	7.66997	49	5.08181
10	3.12428	20	6.34602	30	5.27685	40	6.19081	50	8.77143

Table 4.2: A sample of flow data on a link of the Sioux Fall network

.

1	4.58269	11	3.01202	21	2.99963	31	4.25552	41	3.98166
2	3.57032	12	3.17436	22	4.2842	32	3.45696	42	3.05507
3	3.53377	13	3.71733	23	2.75233	33	3.85384	43	4.48105
4	3.43239	14	3.22236	24	4.02023	34	3.41401	44	4.16973
5	3.68555	15	3.07308	25	3.27738	35	4.5044	45	3.9536
6	4.04595	16	4.33606	26	3.25645	36	3.90714	46	3.85052
7	2.62363	17	3.21006	27	3.35584	37	3.80329	47	3.02087
8	4.51399	18	4.0764	28	3.58138	38	3.52565	48	3.7795
9	4.23625	19	4.19781	29	3.32201	39	3.6257	49	4.08467
10	3.31667	20	3.5865	30	4.00193	40	3.80842	50	3.88628

Table 4.3: A sample of flow data on a link of Network 1

.

methods when the uncertainty level increases.

Looking at the data in detail, we find an interesting behavior for this network. When we vary the demands in the network, the UE flow on some of the links of Network 1 does not change a lot (because it has a limited number of paths), while the flow of the Sioux Falls varies more. Thus, when we employ the least square method to fit the data, the fitting results are not so good for Network 1.

4.5 Conclusions

This chapter studies the application of inverse and robust optimization in finding the travel times assuming the flow is a User Equilibrium flow. We propose algorithms to solve the robust inverse problem under uncertainty in the data that we have on the travel times. Our numerical results show that the robust algorithms provide better results for our problems. We also formulate a data mining problem to find parametric functions for travel times using historical data. Using computer-generated data, we illustrate that by first solving inverse problems for travel times, the fitting results can be much better than by directly using the given data.

Chapter 5

A Network Design Problem under User Equilibrium and Demand Uncertainty

5.1 Introduction

The Network Desgin Problem (NDP) (see, for example, [32]) has been a long researched area. Most of the studies consider the case where designers know the demands accurately, and are able to control the network as well as the traffic flow. This assumption, however, may not be relevent in practice as users will tend to choose the route themselves. Unfortunately, NDP under User Equilibrium behavior and demand uncertainty, is not well developed.

In general, the Deterministic Network Design Problem (DNDP) under UE is a bi-level programming problem. The outer level searches for the optimal setting of the network that minimizes the total travel time of all users at the equilibrium flow, while the inner level finds the UE flow at a specific setting. The NDP under UE is a hard problem to solve, especially when the outer problem is an integer programming problem (since one considers whether or not to build a link in the network).

The first paper studying the discrete NDP under UE was published by LeBlanc

[27]. He proposed a branch and bound algorithm, in which for each branch, we need to solve the UE problem for that specific design to obtain the upper bound, and a system optimization problem (where controllers have full control of the flow), to find the lower bound. However, this method is only applicable to a very small number of discrete varibles (the paper reported a case with only 5 discrete variables). Furthermore, the lower bound is not tight. (In [42], Perakis provided a formulation to calculate this bound compared to the UE total travel time.

Poorzahedy and Turnquist [45] proposed a method where instead of using the total travel time as the objective for the outer problem, they used the "average total cost" $\sum_p \int_0^{h_p} g_p(s)ds$. For symmetric separable travel time functions, this method makes the outer objective and inner objective the same. Then, they reduced the problem into a one-stage optimization problem. However, the objective is just a crude approximation.

There are several papers using heuristic algorithms. For example, [51] tried to apply genetic algorithms, while [44] employed simulated ant system. However, in all these papers, they only reported an NDP with a small number of discrete variables (around 10). The approaches they used to solve the inner UE problem are not efficient.

Recently, Gao et al. [23] described a heuristic, approximation algortihm to solve the discrete network design problem using the BPR cost function (see (4.42). It extended the idea of Lagragrian duality gap and shadow prices to discrete variables in order to find a new solution in the next iteration. The algorithm does not guarantee global convergence. They reported a result for a network with only 6 discrete variables. Zhang et al. [52] formulated the discrete NDP as an optimization problem with complementary constraints and proposed an active set algorithm to solve it. The algorithm makes use of KKT multipliers to adjust the discrete variables of the network. However, this approach is an approximation method, and the complimentary formulation is not computationally efficient for large networks.

In order to overcome the difficulites caused by integer varibles, some papers only consider the continuous NDP problem where they try to increase the capacity of the links. Those papers use the BPR travel time, where the capacity affects the link travel times. Abdulaal and LeBlanc [1] used non-convex line search in the Hook-Jeeves' heuristic algorithm to find the direction of improvement for the network. This approach was reported to be computational expensive. LeBlanc and Boyce [26] also proposed a bilevel optimization approach to solve the continuous NDP.

Marcotte [33] proposed an iterative approach. Initially, the system optimization problem is found. Then, the algorithm determines a path which violates the VI formulation the most and adds some constraints to the system optimization problem. However, as in his formulation, the NDP under UE will have many finite non-convex constraints. Because of the non-convexity, the algorithm is not computationally efficient. In his another paper [34], he proposed a heuristic approach. At each iteration, first, the capacity is fixed and the algorithm solves for the UE flow. Then, the flow is fixed, and the algorithm finds the optimal capacity. With this decomposition approach, the bi-level problem is reduced to solving iteratively two simpler optimization problems.

The literature for the NDP under uncertainty in demands is also very limited. Without considering the UE principle, Riis and Andersen [46], Lisser et al. [29] and Andrade et al. [6], modeled demand uncertainty via probability distribution and solved the NDP with the system optimal flow. Ordóñez and Zhao [41], as well as Atamturk and Zhang [7] proposed robust optimization models to find the best link capacities for the system optimal flow. Mudchanatongsuk et al. [36] considered the discrete network design problem under demand and travel time uncertainty when planners have full control of the flow.

To study the effect of demand uncertainty on the UE flow, there are several papers [21, 48, 49] working on sensitivity analysis. However, those papers only assume an infinitesimal change in demands, which does not change the current path choice.

To the best of our knowledge, there is only one paper dealing with discrete NDP under UE and demand uncertainty. In their paper, Lou et al. [30] proposed an iterative cutting plane scheme to solve the problem. In each iteration, they solved two subproblems: a relaxed discrete NDP with a subset of demand scenarios, and a problem generating the worst case scenario. They solve each problem using the active set algorithm in [52]. The reported test network only consists of 11 discrete variables.

5.2 Deterministic Network Design Problem

Given a network $G(V, A)$ where A is the set of possible links. We assume the demands for the network are fixed and the travel times are affine functions. Due to some constraints, we cannot build all possible links. In the network design problem, we are interested in finding a set of links to build in order to minimize the total travel time on the network subject to UE constraints.

Let $\mathbf{y} \in \{\mathbf{0}, \mathbf{1}\}^{|\mathbf{A}|}$ be our decision vector. $y_i = 1$ if link i^{th} is built and $y_i = 0$ otherwise. Assume at most B links (budget-type constraint) can be built. Then, the deterministic network design problem (DNDP) is:

$$\min_{\mathbf{y}|\mathbf{e}'\mathbf{y} \leq B} \mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}}\bar{\mathbf{f}} \tag{5.1}$$

where $\bar{\mathbf{f}}$ is the UE flow for the network $G(V, A, \mathbf{y})$.

From Theorem 3.4, $\bar{\mathbf{f}}$ can be solved by:

$$\bar{\mathbf{f}} = \arg\min_{\mathbf{f}^{\mathbf{w}}, \lambda_{\mathbf{w}}} \mathbf{c}(\mathbf{f})^{\mathbf{T}}\mathbf{f} - \sum_{\mathbf{w}=\mathbf{1}}^{\mathbf{W}} \mathbf{d}^{\mathbf{w}\mathbf{T}}\boldsymbol{\lambda}^{\mathbf{w}} \tag{5.2}$$

$$\text{s.t: } \mathbf{N}\mathbf{f}^{\mathbf{w}} = \mathbf{d}^{\mathbf{w}} \quad \forall \mathbf{w} \tag{5.3}$$

$$\sum_{w=1}^{W} \mathbf{f}^{\mathbf{w}} \leq K\mathbf{y} \tag{5.4}$$

$$\mathbf{N}^{\mathbf{T}}\boldsymbol{\lambda}^{\mathbf{w}} \leq \mathbf{c}(\mathbf{f}) + M(\mathbf{1} - \mathbf{y}) \quad \forall \mathbf{w} \tag{5.5}$$

$$\mathbf{f}^{\mathbf{w}} \geq \mathbf{0}, \;\; \mathbf{f} = \sum_{\mathbf{w}} \mathbf{f}^{\mathbf{w}}. \tag{5.6}$$

M and K are some sufficiently big numbers. We add constraint (5.4) to make sure the flow on links that are not built is equal to 0. The additional part in constraint (5.5) makes sure that it only applies to built links.

When $\bar{\mathbf{f}}$ exits, the optimal value of (5.2) is 0. Therefore, we can solve the DNDP by following theorem [31].

Theorem 5.1 *For some sufficiently big θ, the Deterministic Network Design Problem is equivalent to the following mixed integer optimization problem:*

$$\min_{\mathbf{f^w},\mathbf{y},\lambda_\mathbf{w}} \mathbf{c(f)^T f} + \theta(\mathbf{c(f)^T} \sum_{\mathbf{w=1}}^{\mathbf{W}} \mathbf{f} - \sum_{\mathbf{w=1}}^{\mathbf{W}} \mathbf{d^{wT}} \boldsymbol{\lambda}^\mathbf{w}) \tag{5.7}$$

$$\begin{aligned} s.t.\text{: } & \mathbf{Nf^w} = \mathbf{d^w} \quad \forall \mathbf{w} \\ & \sum_{w=1}^{W} \mathbf{f^w} \leq K\mathbf{y} \\ & \mathbf{N^T}\boldsymbol{\lambda}^\mathbf{w} \leq \mathbf{c(f)} + M(\mathbf{1} - \mathbf{y}) \quad \forall \mathbf{w} \\ & \mathbf{f^w} \geq \mathbf{0}, \ \ \mathbf{f} = \sum_{\mathbf{w}} \mathbf{f^w} \\ & \mathbf{e'y} \leq B. \end{aligned}$$

Proof For simplicity, assuming $|W| = 1$ (when $|W| > 1$, the proof is similar).
We have $\mathbf{c(f)^T f} - \mathbf{d^T}\boldsymbol{\lambda} = \mathbf{c(f)^T f} - (\mathbf{Nf})^\mathbf{T}\boldsymbol{\lambda} = (\mathbf{c(f)} - \mathbf{N^T}\boldsymbol{\lambda})^\mathbf{T}\mathbf{f}$
If $y_k = 0$ then $f_k = 0$, $(c_k(f) - (N^T\lambda)_k)f_k = 0$.
If $y_k > 0$ then from constraint (5.5) we have $(c_k(f) - (N^T\lambda)_k)f_k \geq 0$
Thus, $\mathbf{c(f)^T f} - \mathbf{d^T}\boldsymbol{\lambda} \geq \mathbf{0}$. With sufficiently big θ, the optimal solution $\bar{\mathbf{f}}$ to (5.7) will satisfy: $\mathbf{c}(\bar{\mathbf{f}})^\mathbf{T}\bar{\mathbf{f}} - \mathbf{d^T}\boldsymbol{\lambda} = \mathbf{0}$, i.e, $\bar{\mathbf{f}}$ is the UE flow to $G(V, A, \mathbf{y})$. The objective is then only the total cost of the network under the equilibrium flow.
Therefore, the DNDP is correctly formulated as problem (5.1). □

In Theorem 5.1 we need to choose appropriate constants θ, K, and M. We solve the mixed integer optimization with CPLEX, and the algorithm of the solver for mixed integer optimization problem is branch and bound.
- We have $f_{ij} \leq \sum_{w=1}^{|W|} d_w$. Therefore, we can choose $K = \sum_{w=1}^{|W|} d_w$. Choosing a bigger K does not really affect the speed of the whole algorithm.
- We can start with some θ, and increase θ until the optimal value converges. Choosing a bigger θ actually does not increase the running time significantly. Thus, we can

start with a relatively large value of θ.

- Choosing M however, not only affects the running time, but also affects the correctness of the algorithm. Constraint (5.5) can be rewritten as follows (where (i, j) denotes a link connecting node i to node j):

$$\lambda_i - \lambda_j \leq c_{ij}(\mathbf{f}) + M(1 - y_{ij}).$$

If $y_{ij} = 0$ then $\lambda_i - \lambda_j \leq c_{ij}(\mathbf{f}) + M$. Assuming there exists a node k such that $y_{ik} = 1$ and $y_{kj} = 1$, then :

$$\lambda_i - \lambda_k \leq c_{ik}(\mathbf{f})$$
$$\lambda_k - \lambda_j \leq c_{kj}(\mathbf{f}).$$

Thus, $\lambda_i - \lambda_j \leq c_{ik} + c_{kj}$. An appropriate value of M has to satisfy $M \geq c_{ik} + c_{kj}$. When there is a longer path connecting nodes i and j, M has to be even larger. With small M, we might not be able to obtain the equilibrium flow while solving the mixed integer optimization problem. Unfortunately, a bigger M can slow down the algorithm significantly (from several minutes to hours to run the DNDP, due to more complicated branching).

Therefore, we have to start with some small value of M, and slowly increase it until the optimal value does not decrease anymore.

5.3 Robust Network Design Problem

In reality, the demands for the network are not known accurately. People may vary their choice of origin destination day by day. Moreover, the network we build will need to serve the community for several years until we can make any adjustment. During that period, the demands for each OD pair may change significantly. Therefore, we need to find a design that is robust against all those changes.

Let $\mathbf{d} \in \mathbb{R}^{|W|}$ denote a vector consistings of all demands in the network.

Assumming the number of users does not change significantly. The demands thus can be modeled as a set:

$$E = \{\mathbf{d} | \sum_{w} d_w = TD, D_w^{min} \leq d_w \leq D_w^{max}\}. \tag{5.8}$$

Let $G(V, A, \mathbf{y}, \mathbf{d})$ be the network built with design vector $\mathbf{y}$ with fixed demand vector $\mathbf{d}$ (A and V are the sets of possible links and nodes). The robust network design problem (RNDP) is:

$$\min_{\mathbf{y}|\mathbf{e}'\mathbf{y} \leq B} \max_{\mathbf{d} \in E} \mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}} \bar{\mathbf{f}} \tag{5.9}$$

where $\bar{\mathbf{f}}$ is the UE flow for the network $G(V, A, \mathbf{y}, \mathbf{d})$.

For affine travel times, $\bar{\mathbf{f}}$ is uniquely defined for $G(V, A, \mathbf{y}, \mathbf{d})$.
This is a tri-level optimization problem, which is very complicated to solve directly. Given a design vector $\mathbf{y}$, let consider the inner maximization problem:

$$\tau(\mathbf{y}) = \max_{d_w \in E} TC(\mathbf{y}, \mathbf{d}) \tag{5.10}$$

$$\text{where } TC(\mathbf{y}, \mathbf{d}) = \mathbf{c}(\bar{\mathbf{f}})^{\mathbf{T}} \bar{\mathbf{f}}$$

$$\text{and } \bar{\mathbf{f}}(\mathbf{y}, \mathbf{d}) = \arg \min_{\mathbf{f^w}, \lambda_\mathbf{w}} \mathbf{c}(\mathbf{f})^{\mathbf{T}} \mathbf{f} - \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{d}^{\mathbf{wT}} \lambda^{\mathbf{w}} \tag{5.11}$$

$$\text{s.t } : \mathbf{N}\mathbf{f}^{\mathbf{w}} = \mathbf{d}^{\mathbf{w}} \quad \forall \mathbf{w}$$

$$\sum_{w=1}^{W} \mathbf{f}^{\mathbf{w}} \leq K\mathbf{y}$$

$$\mathbf{N}^{\mathbf{T}} \lambda^{\mathbf{w}} \leq \mathbf{c}(\mathbf{f}) + M(\mathbf{1} - \mathbf{y}) \quad \forall \mathbf{w}$$

$$\mathbf{f}^{\mathbf{w}} \geq \mathbf{0}, \;\; \mathbf{f} = \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{f}^{\mathbf{w}}.$$

With affine travel time functions, $TC(\mathbf{y}, \mathbf{d})$ is a quadratic function with respect to $\bar{\mathbf{f}}$. The region defined by constraints of problem (5.11) is a convex region and E is a convex set. Therefore, in order to find $\tau(\mathbf{y})$, we can test all extreme points of E. However for a realistic network, the number of OD pairs may be very big. Thus, instead of finding all extreme points, we can use local search with multiple starting points to find a local maximum.

Algorithm B *:*

- *Start with an extreme point* $\mathbf{d}$ *of* E*. Solving (5.11) and find* $TC(\mathbf{y}, \mathbf{d})$*.*
- *Generate a neighbour extreme point* $\mathbf{d_n}$ *of* $\mathbf{d}$*. If* $TC(\mathbf{y}, \mathbf{d}) \leq TC(\mathbf{y}, \mathbf{d_n})$ *then set* $\mathbf{d} = \mathbf{d_n}$
- *Repeat until we cannot find any improved neighbour.*

We can use multiple starting points to improve the quality of the algorithm and to obtain a set of local minima.An extreme point $\mathbf{d}$ for the uncertainty set we consider will have at least $|W| - 1$ components equal to either D_w^{max} or D_w^{min} and satisfy equality (5.8).

For fixed $\mathbf{y}$ and $\mathbf{d}$, solving problem (5.11) is very fast regardless of value of M. With Algorithm B we can find near optimal points for the inner maximization problem.

In order to solve the RNDP, we can use a decomposition approach. Consider only a discrete set $E_s \subset E$. Solve the minimax problem for E_s to get a lower bound and a decision vector $\mathbf{y}$. Fix $\mathbf{y}$, then solve (5.10) to get new demand scenarios for E_s and an upper bound to the problem:

Decision problem:

Let $\mathbf{d_1}$, $\mathbf{d_2}$ ....., $\mathbf{d_{|E_s|}} \in E_s$. With each demand vector, let $\mathbf{f_1}$, $\mathbf{f_2}$, ..., $\mathbf{f_{|E_s|}}$ be the corresponding flow. The decision problem solves :

$$\min_{\eta, \mathbf{y}, \mathbf{f_1}, \mathbf{f_2} .., \lambda_1, \lambda_2} \eta \tag{5.12}$$

$$\text{s.t}: \eta \geq \mathbf{c}(\mathbf{f_1})^{\mathbf{T}}\mathbf{f_1} + \theta(\mathbf{c}(\mathbf{f_1})^{\mathbf{T}}\mathbf{f_1} - \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{d}_1^{\mathbf{wT}}\lambda_1^{\mathbf{w}})$$

$$\eta \geq \mathbf{c}(\mathbf{f_2})^{\mathbf{T}}\mathbf{f_2} + \theta(\mathbf{c}(\mathbf{f_2})^{\mathbf{T}}\mathbf{f_2} - \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{d}_2^{\mathbf{wT}}\lambda_2^{\mathbf{w}})$$

$$\ldots$$

$$\mathbf{N}\mathbf{f}_1^{\mathbf{w}} = \mathbf{d}^{\mathbf{w}} \quad \forall \mathbf{w}$$

$$\sum_{w=1}^{W} \mathbf{f}_1^{\mathbf{w}} \leq K\mathbf{y}$$

$$\mathbf{N}^{\mathbf{T}}\lambda_1^{\mathbf{w}} \leq \mathbf{c}(\mathbf{f_1}) + M(\mathbf{1} - \mathbf{y}) \quad \forall \mathbf{w}$$

$$\mathbf{f}_1^{\mathbf{w}} \geq \mathbf{0},\ \mathbf{f_1} = \sum_{\mathbf{w}=1}^{|\mathbf{W}|} \mathbf{f}_1^{\mathbf{w}}$$

$$\ldots$$

$$\mathbf{e}'\mathbf{y} \leq B.$$

Adversary problem:

Let $\mathbf{y}^*$ be the solution to the decision problem. Use Algorithm B to find near optimal points to $\tau(\mathbf{y}^*)$ in (5.10). Add those points to E_s. Reiterate.

As we could not solve problem (5.10) to optimality, we cannot get an upper bound. Thus, the algorithm stops when after several iterations, we cannot obtain any better solution.

With affine travel times, the decision problem is a mixed integer optimization problem, with a linear objective problem and convex quadratic constraints. The resulting flow will be the UE flow, and η is the worst case objective in E_s. However, in computational experiment, we have found that problem (5.12) is not efficiently solvable. Thus, in what follows, we develop a heuristic algorithm using Robust Simulated Annealing (RSA) [14] to solve the RNDP problem.

RSA algorithm for RNDP:

Step 1: Initialization.

- Solve the DNDP using mixed integer optimization to get a starting design $\mathbf{y_0}$.
- Set iteration index $k = 0$, acceptance number $h = 0$ and annealing index $n = 0$.
- Use Algorithm B to assembly a set $\mathcal{M}(\mathbf{y_0})$ of local maxima demand vectors.
- Set the initial inverse temperature

$$\beta = \beta_0 = \frac{1}{\max_{\mathbf{d} \in \mathcal{M}(\mathbf{y_0})} TC(\mathbf{y_0}, \mathbf{d})} \tag{5.13}$$

- Compute the energy function of $\mathcal{M}(\mathbf{y_0})$ as:

$$W(\mathbf{y_0}) = \frac{1}{\beta_0} \log \sum_{\mathbf{d} \in \mathcal{M}(\mathbf{y_0})} e^{\beta_0 TC(\mathbf{y_0}, \mathbf{d})} \tag{5.14}$$

Step 2: Trial design at iteration k:

- Generate a neighbour design vector $\mathbf{y_t}$.
- Assemble $\mathcal{M}(\mathbf{y_t})$.
- Using (5.14), calculate $W(\mathbf{y_t})$
- Caculate the acceptance probability: $P(\mathbf{y_k} \rightarrow \mathbf{y_t}) = \min(1, e^{\beta(W(\mathbf{y_k}) - W(\mathbf{y_t}))})$

Step 3: Generate a random number $R \in [0, 1]$

- If $P \geq R$: Let $\mathbf{y_{k+1}} = \mathbf{y_t}$, increase h. Else let $\mathbf{y_{k+1}} = \mathbf{y_k}$
- If $k \mod 100 = 0$ then: Increase n. Let $\beta = (1 + \epsilon)^n \beta_0$. Recalculate $W(\mathbf{y_{k+1}})$
- Back to step 2 and re-iterate

We can choose a small ϵ (from 0.02 to 0.05). The algorithm stops when the inverse temperature gets big enough. Also note that $\lim_{\beta \rightarrow \infty} W(\mathbf{y_k}) = \max_{\mathbf{d} \in \mathcal{M}(\mathbf{y_k})} TC(\mathbf{y_k}, \mathbf{d})$

In the above algorithm, the choice of a neighbour $\mathbf{y_t}$ is important to the algorithm. We need to make sure that all possible designs can be visited. In our work, we generate $\mathbf{y_t}$ using following steps:

- Randomly choose 2 components i and j of vector $\mathbf{y_k}$ such that $y_{k,i} = 1$ and $y_{k,j} = 0$

- We can either flip both of them, or one of them with equal probability. Always make sure that at most B links are built.
- The new neighbour will thus, has at most two components different from the current design vector.

This is somewhat similar to the 2-OPT neighbour in the travelling salesman problem. As a result, we make sure that all possible designs could be visited.

5.4 Computational Results

5.4.1 Deterministic Network Desgin problem

In this case, we use 2 test networks. The first network (Network 1) is from [37] with 20 nodes, 28 links. The second network is Sioux Falls, a big network with 76 links, 28 nodes. Sioux Falls network is a model representing a real transporation network. This is a fairly large network. There are around 2000 paths connecting each OD pair.

Network 1:

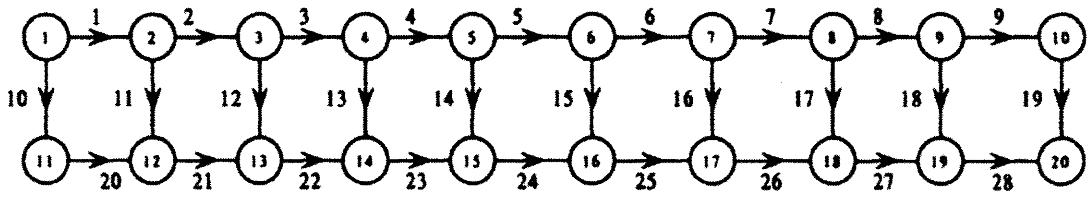

Figure 5-1: Network 1

First, we fix $\theta = 1000$ and vary M (we choose the same value for all links). We solve the mixed integer optimization problem and obtain the optimal design $\mathbf{y}^*$ and $\mathbf{f}^*$. Then, we find the UE flow $\bar{\mathbf{f}}$ for the network designed with $\mathbf{y}^*$ (As noted above, $\bar{\mathbf{f}}$ and $\mathbf{f}^*$ may not be the same for a small M). Figures 5-3 and 5-4 show that for small M, $\bar{\mathbf{f}}$ and $\mathbf{f}^*$ are not the same (the total travel times are different). Note that when $B = 20$ and $M = 600 - 1000$, the solution to the mixed integer optimization happens to be the optimal design under UE, but the flow found by mixed integer optimization

No.	OD pair	Demand
1	(1,19)	60
2	(1,20)	50
3	(2,13)	100
4	(2,17)	100
5	(2,20)	100
6	(3,14)	50
7	(4,20)	100
8	(6,19)	100

Table 5.1: OD table for Network 1

$c_1(\mathbf{f}) = 5f_1 + 2f_2 + 50$	$c_{15}(\mathbf{f}) = 9f_{15} + 2f_{14} + 20$
$c_2(\mathbf{f}) = 4f_2 + f_1 + 20$	$c_{16}(\mathbf{f}) = 8f_{16} + 5f_{12} + 30$
$c_3(\mathbf{f}) = 3f_3 + f_4 + 35$	$c_{17}(\mathbf{f}) = 7f_{17} + 2f_{15} + 45$
$c_4(\mathbf{f}) = 6f_4 + 3f_5 + 40$	$c_{18}(\mathbf{f}) = 5f_{18} + f_{16} + 30$
$c_5(\mathbf{f}) = 6f_5 + 4f_6 + 60$	$c_{19}(\mathbf{f}) = 8f_{19} + 3f_{17} + 60$
$c_6(\mathbf{f}) = 7f_6 + 3f_7 + 50$	$c_{20}(\mathbf{f}) = 6f_{20} + f_{21} + 30$
$c_7(\mathbf{f}) = 8f_7 + 2f_8 + 40$	$c_{21}(\mathbf{f}) = 4f_{21} + f_{22} + 40$
$c_8(\mathbf{f}) = 5f_8 + 2f_9 + 65$	$c_{22}(\mathbf{f}) = 6f_{22} + f_{23} + 50$
$c_9(\mathbf{f}) = 6f_9 + 2f_{10} + 70$	$c_{23}(\mathbf{f}) = 9f_{23} + 2f_{24} + 35$
$c_{10}(\mathbf{f}) = 4f_{10} + f_{12} + 80$	$c_{24}(\mathbf{f}) = 8f_{24} + f_{25} + 40$
$c_{11}(\mathbf{f}) = 7f_{11} + 4f_{12} + 65$	$c_{25}(\mathbf{f}) = 9f_{25} + 3f_{26} + 45$
$c_{12}(\mathbf{f}) = 8f_{12} + 2f_{13} + 70$	$c_{26}(\mathbf{f}) = 7f_{26} + 8f_{27} + 30$
$c_{13}(\mathbf{f}) = 7f_{13} + 3f_{18} + 60$	$c_{27}(\mathbf{f}) = 8f_{27} + 3f_{28} + 50$
$c_{14}(\mathbf{f}) = 6f_{14} + 3f_{15} + 50$	$c_{28}(\mathbf{f}) = 7f_{28} + 3f_{27} + 65$

Table 5.2: Link travel times for Network 1

is not the UE flow. Only when $M = 1800$, the mixed integer optimization obtains the correct design and the correct flow.

Figure 5-4 shows that after M is big enough that we are able to get a correct solution from the mixed integer optimization, when we increase M, the computation time increases. We will observe the same behavior if we extend the range of M in figure 5-3.

Next, we examine the effect of θ with fix M. Figure 5-5 shows that θ does not strongly affect the computation time (the difference in computation time between $\theta = 100$ and $\theta = 1000$ is not much). We thus can pick θ suffciently big to ensure the

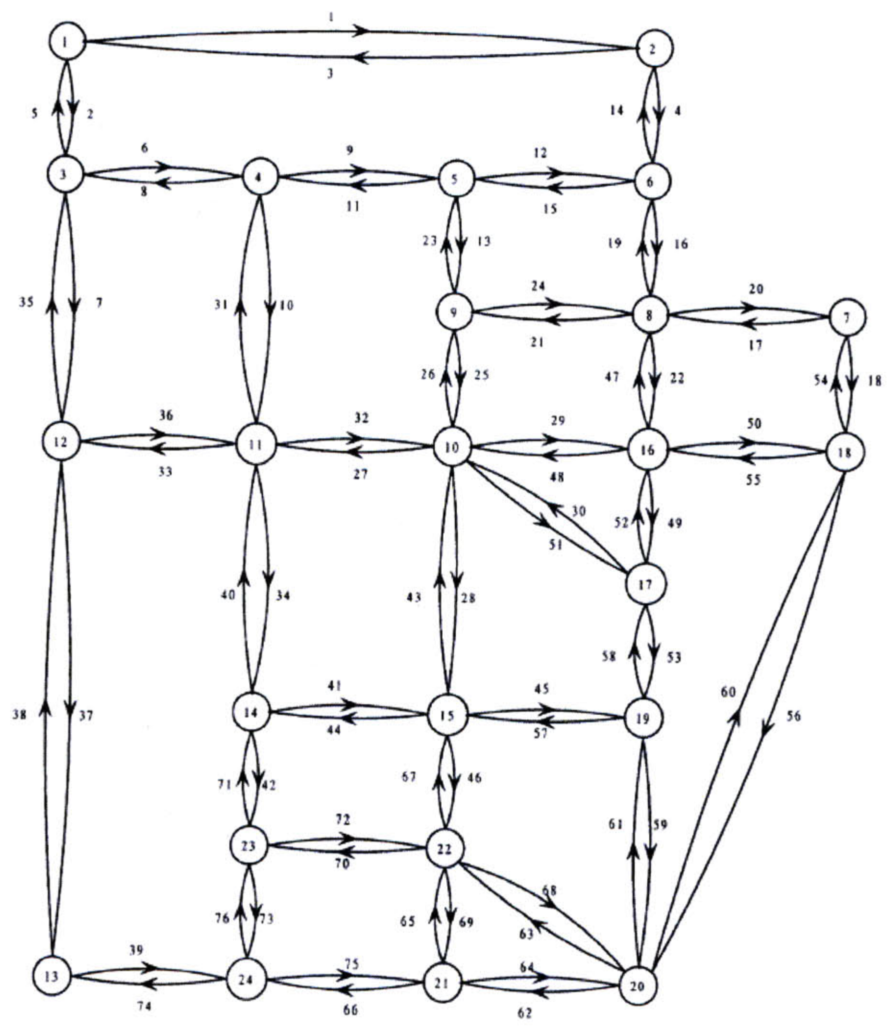

Figure 5-2: Sioux Falls network

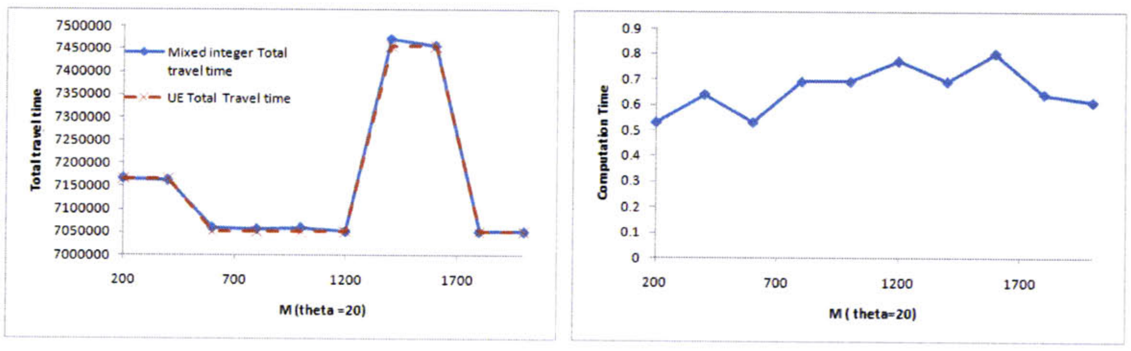

Figure 5-3: Total Cost and computation time (sec) with different M. Total Budget $B = 20$

$c_1(f_1) = 6f_1 + 6$	$c_{27}(f_{27}) = 3f_{27} + 5$	$c_{53}(f_{53}) = 6f_{53} + 2$
$c_2(f_2) = 2f_2 + 4$	$c_{28}(f_{28}) = 4f_{28} + 6$	$c_{54}(f_{54}) = 4f_{54} + 2$
$c_3(f_3) = 6f_3 + 6$	$c_{29}(f_{29}) = 2f_{29} + 4$	$c_{55}(f_{55}) = 2f_{55} + 3$
$c_4(f_4) = 2f_4 + 5$	$c_{30}(f_{30}) = 6f_{30} + 8$	$c_{56}(f_{56}) = 16f_{56} + 4$
$c_5(f_5) = 2f_5 + 4$	$c_{31}(f_{31}) = 3f_{31} + 6$	$c_{57}(f_{57}) = 2f_{57} + 3$
$c_6(f_6) = 2f_6 + 4$	$c_{32}(f_{32}) = 3f_{32} + 5$	$c_{58}(f_{58}) = 6f_{58} + 2$
$c_7(f_7) = 4f_7 + 4$	$c_{33}(f_{33}) = 4f_{33} + 6$	$c_{59}(f_{59}) = 4f_{59} + 4$
$c_8(f_8) = 2f_8 + 4$	$c_{34}(f_{34}) = 3f_{34} + 5$	$c_{60}(f_{60}) = 16f_{60} + 4$
$c_9(f_9) = 2f_9 + 2$	$c_{35}(f_{35}) = 4f_{35} + 4$	$c_{61}(f_{61}) = 4f_{61} + 4$
$c_{10}(f_{10}) = 3f_{10} + 6$	$c_{36}(f_{36}) = 4f_{36} + 6$	$c_{62}(f_{62}) = 4f_{62} + 6$
$c_{11}(f_{11}) = 2f_{11} + 2$	$c_{37}(f_{37}) = 14f_{37} + 3$	$c_{63}(f_{63}) = 6f_{63} + 5$
$c_{12}(f_{12}) = 10f_{12} + 4$	$c_{38}(f_{38}) = 14f_{38} + 3$	$c_{64}(f_{64}) = 4f_{64} + 6$
$c_{13}(f_{13}) = 2f_{13} + 5$	$c_{39}(f_{39}) = 4f_{39} + 4$	$c_{65}(f_{65}) = 8f_{65} + 2$
$c_{14}(f_{14}) = 2f_{14} + 5$	$c_{40}(f_{40}) = 3f_{40} + 4$	$c_{66}(f_{66}) = 4f_{66} + 3$
$c_{15}(f_{15}) = 10f_{15} + 4$	$c_{41}(f_{41}) = 4f_{41} + 5$	$c_{67}(f_{67}) = 2f_{67} + 3$
$c_{16}(f_{16}) = 2f_{16} + 2$	$c_{42}(f_{42}) = 2f_{42} + 4$	$c_{68}(f_{68}) = 6f_{68} + 5$
$c_{17}(f_{17}) = 2f_{17} + 3$	$c_{43}(f_{43}) = 4f_{43} + 6$	$c_{69}(f_{69}) = 8f_{69} + 2$
$c_{18}(f_{18}) = 4f_{18} + 2$	$c_{44}(f_{44}) = 4f_{44} + 5$	$c_{70}(f_{70}) = 2f_{70} + 4$
$c_{19}(f_{19}) = 2f_{19} + 2$	$c_{45}(f_{45}) = 2f_{45} + 3$	$c_{71}(f_{71}) = 2f_{71} + 4$
$c_{20}(f_{20}) = 2f_{20} + 3$	$c_{46}(f_{46}) = 2f_{46} + 3$	$c_{72}(f_{72}) = 2f_{72} + 4$
$c_{21}(f_{21}) = 4f_{21} + 10$	$c_{47}(f_{47}) = 2f_{47} + 5$	$c_{73}(f_{73}) = 2f_{73} + 2$
$c_{22}(f_{22}) = 2f_{22} + 5$	$c_{48}(f_{48}) = 2f_{48} + 4$	$c_{74}(f_{74}) = 4f_{74} + 4$
$c_{23}(f_{23}) = 2f_{23} + 5$	$c_{49}(f_{49}) = 2f_{49} + 2$	$c_{75}(f_{75}) = 4f_{75} + 3$
$c_{24}(f_{24}) = 4f_{24} + 10$	$c_{50}(f_{50}) = 2f_{50} + 3$	$c_{76}(f_{76}) = 2f_{76} + 2$
$c_{25}(f_{25}) = 2f_{25} + 3$	$c_{51}(f_{51}) = 6f_{51} + 8$	
$c_{26}(f_{26}) = 2f_{26} + 3$	$c_{52}(f_{52}) = 2f_{52} + 2$	

Table 5.3: Link travel times for Sioux Fall

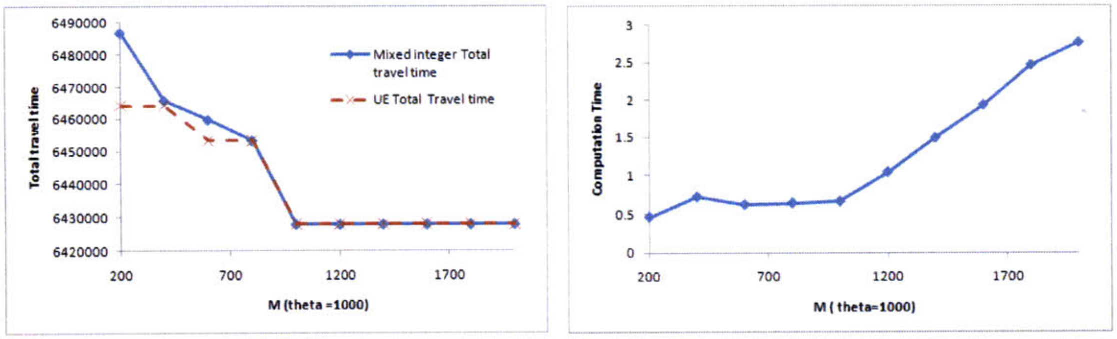

Figure 5-4: Total Cost and computation time (sec) with total Budget $B = 24$

correctness of the algorithm.

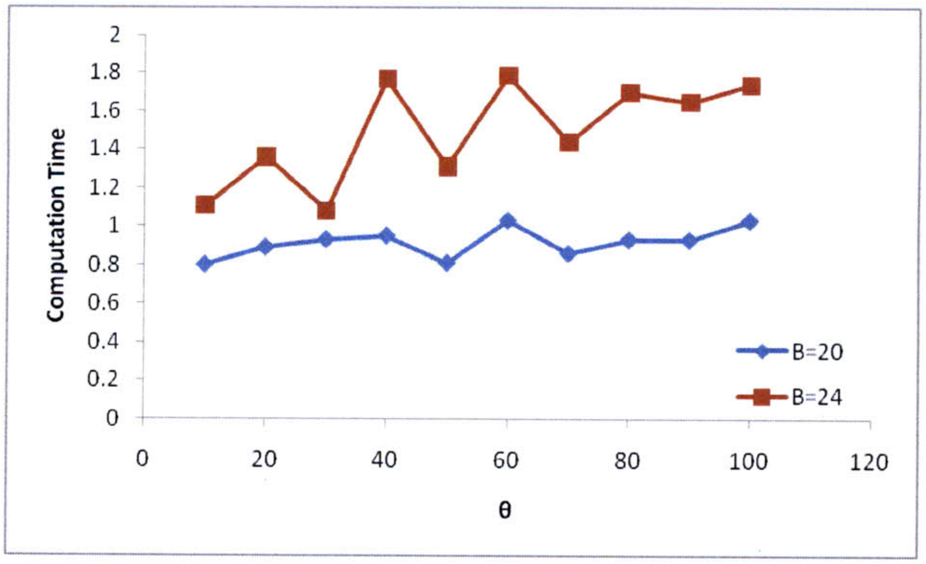

Figure 5-5: Computation time (sec) with varying θ, $M = 1600$

The optimal design vectors $\mathbf{y}*$ for each case are listed in table 5.4.

Sioux Falls Network with 6 OD pairs:

B	Links are not built
20	1,9,11,13,15,16,17,19
24	1,12,15,17

Table 5.4: Design vector for Network 1

In this case, we use a simplified OD pair table with only 6 OD pairs. The OD pairs and demands are listed in Table 5.5.

For this network, we let $\theta = 20$. The budget in this case is set to $B = 60$. This is a

No.	OD pair	Demand
1	(1,20)	25
2	(20,1)	25
3	(1,24)	20
4	(24,1)	20
5	(7,20)	15
6	(20,7)	15

Table 5.5: OD Table(case 1) for Sioux Falls

huge design problem, as in total we have $\binom{76}{60}$ (around 10^{39}) different possible designs.

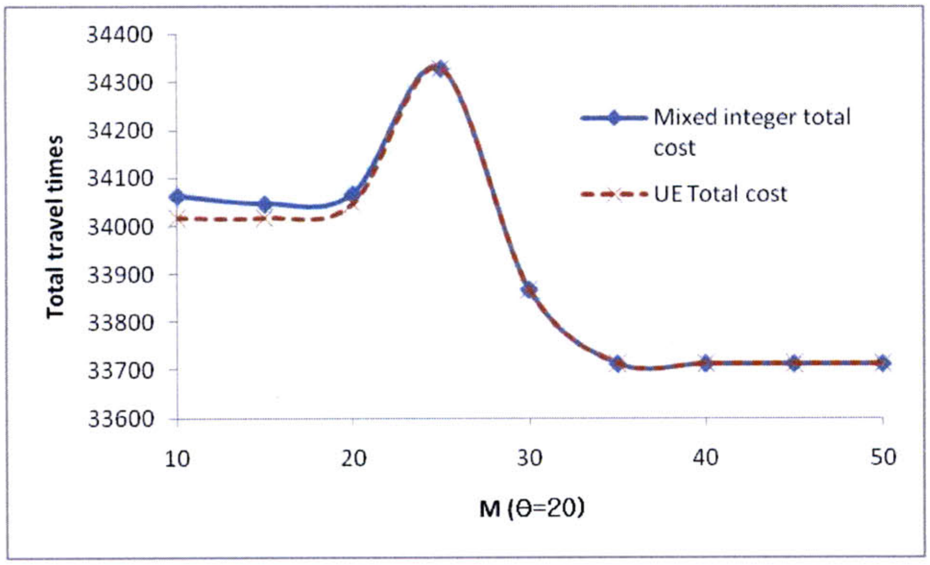

Figure 5-6: Total travel time with varying M

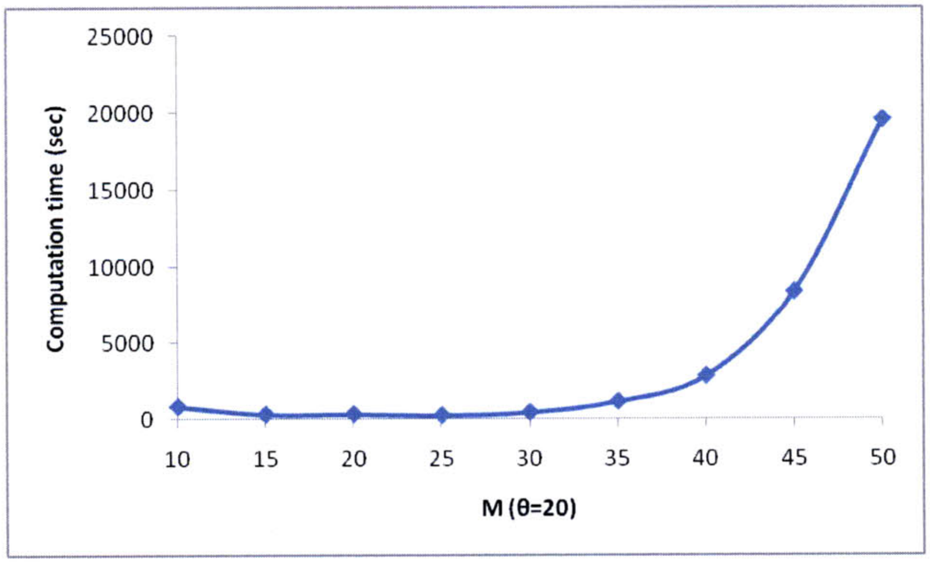

Figure 5-7: Computation time (sec) with varying M

Figures 5-6 and 5-7 give us the similar conclusion as we have from the previous test network. For small M, the mixed integer optimization will have incorrect solutions. The computation time also increases very fast as M increases.

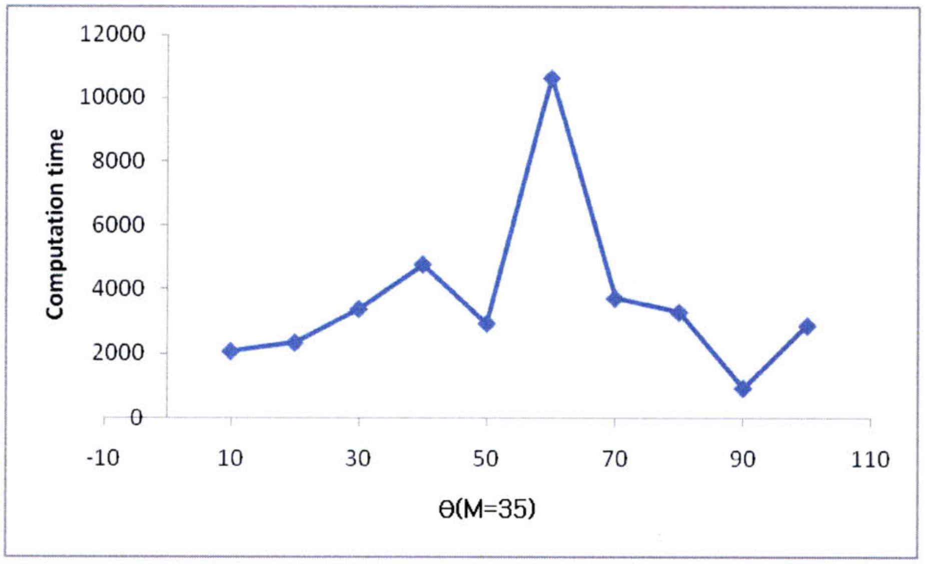

Figure 5-8: Computation time (sec) with varying θ

Figure 5-8 reports the computation time for fix M with different values of θ. The computation time actually fluctuates as θ increases.

Sioux Falls and Braess paradox:

We consider Sioux Falls with only 3 OD pairs. The demands and OD table are listed in Table 5.6. The total budget is $B = 70$. We choose to report this case because the DNDP is subjected to Braess paradox [19]: under UE, building extra links does not decrease the total travel time of the network. As shown in Table 5.7, we only need 68 links for the optimal design.

No.	OD pair	Demand
1	(1,20)	20
2	(24,1)	25
5	(7,20)	15

Table 5.6: OD Table(case 2) for Sioux Falls

y_1	1	y_{11}	1	y_{21}	1	y_{31}	1	y_{41}	1	y_{51}	1	y_{61}	1	y_{71}	1
y_2	1	y_{12}	1	y_{22}	1	y_{32}	1	y_{42}	1	y_{52}	1	y_{62}	0	y_{72}	1
y_3	1	y_{13}	1	y_{23}	1	y_{33}	1	y_{43}	1	y_{53}	1	y_{63}	0	y_{73}	0
y_4	1	y_{14}	1	y_{24}	1	y_{34}	1	y_{44}	0	y_{54}	1	y_{64}	1	y_{74}	1
y_5	1	y_{15}	1	y_{25}	1	y_{35}	1	y_{45}	1	y_{55}	1	y_{65}	1	y_{75}	1
y_6	1	y_{16}	1	y_{26}	1	y_{36}	1	y_{46}	1	y_{56}	1	y_{66}	0	y_{76}	1
y_7	1	y_{17}	1	y_{27}	0	y_{37}	1	y_{47}	1	y_{57}	0	y_{67}	1		
y_8	1	y_{18}	1	y_{28}	1	y_{38}	1	y_{48}	1	y_{58}	1	y_{68}	1		
y_9	1	y_{19}	1	y_{29}	1	y_{39}	1	y_{49}	1	y_{59}	1	y_{69}	1		
y_{10}	1	y_{20}	0	y_{30}	1	y_{40}	1	y_{50}	1	y_{60}	1	y_{70}	0		

Table 5.7: DNDP solution for Sioux Falls with 3 OD pairs

5.4.2 Robust Network Design Problem

In this section, we denote the uncertainty level as: $L = \frac{\sum_w (D_w^{max} - D_w^{min})}{TD}$.
L can be large in reality, even it can approach 100% (for example: the demands during weekdays and weekend).
In what follows, we will test the robust simulated annealing for RNDP with different uncertainty levels.
Sioux Falls with 3-OD pairs:
Table 5.8 shows the worst total travel time (found by Algorithm B) and nominal travel time (with the demands from Table 5.6). For the first two cases, the robust solutions are the same as the deterministic solution. However, when $L = 52.72\%$ and $L = 81.82\%$, the robust solutions are different from the deterministic solution. They have better worst total travel time. Note that in both of these two cases, we construct a total of 70 links (see Table 5.9) while the deterministic solution only builds 68 links.

We also sample 500 demand scenarios from the uncertainty set. With each of them, we find the total travel time under UE for the network designed by the robust and deterministic solution.
Figures 5-9 and 5-10 show the box plot when $L = 52.72\%$ and $L = 81.82\%$. They show that the robust solutions have better worst case travel times, means and variances. The robust simulated annealing performs well in those two cases.

	Worst total travel time		Nominal total travel time	
L(%)	Robust	Deterministic	Robust	Deterministic
34.55	10526.6	10526.6	9795.12	9795.12
41.18	10526.6	10526.6	9795.12	9795.12
52.72	11823.7	11869.5	9804.19	9795.12
81.82	12929.4	12959.1	9839.56	9795.12

Table 5.8: Worst total travel time and nominal travel time with varying L for Sioux Falls with 3 OD pairs

L(%)	Links are not built
52.72	15,27,44,62,63,70
81.82	15,51,62,63,69,70

Table 5.9: Solution to RNDP for Sioux Falls with 3 OD pairs

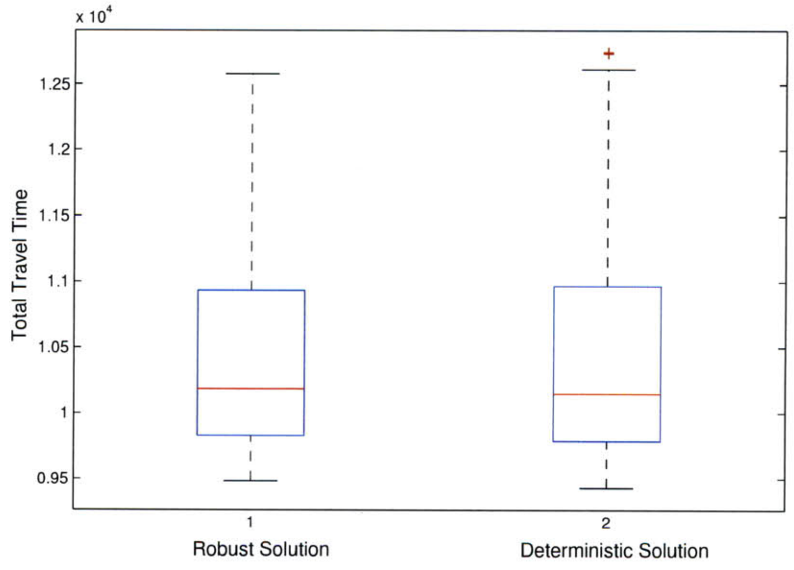

Figure 5-9: Box plot for 500 scenarios when L = 81.82%

Sioux Falls with 6-OD pairs:

In this case, we let $B = 70$.

For L from 20% to 60%. the robust solution is the same as the deterministic solution. Table 5.10 shows the solutions when $L = 80\%$ and $L = 91.6\%$. It also shows the worst total travel times and the nominal total travel times.

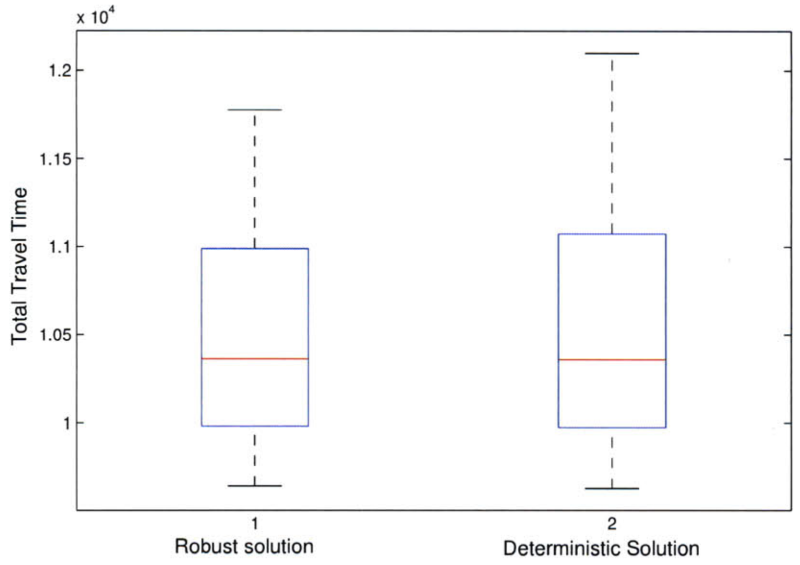

Figure 5-10: Box plot for 500 scenarios when L = 52.72%

	Worst total travel time		Nominal total travel time		Links are not built
L(%)	Robust	Deterministic	Robust	Deterministic	Robust solution
80	39053.8	39084.3	33838.1	33010.4	20,30,32,50,55,70
91.6	40998.9	41062.8	33104.2	33010.4	21,24,30,51,70,72

Table 5.10: Worst case and nominal case total travel time for Sioux Falls with 6 OD pairs

We also sample 500 demand scenarios from the uncertainty set, calculate the total travel time under the UE flow for each of them and plot the box plot in figures 5-11 and 5-12. Our robust solutions have better worst total travel times than the deterministic solution. Unfortunately, when $L = 80\%$, the performance of the robust solution on the sample scenarios is worse than the deterministic one (because the RSA algorithm aims to find the best design for worst case scenario). When $L = 91.6\%$, they perform almost the same.

5.5 Conclusions

The first part of this chapter revisits the exact mixed integer programming approach for the DNDP. We carefully study the effect of the choice of M and θ in the model.

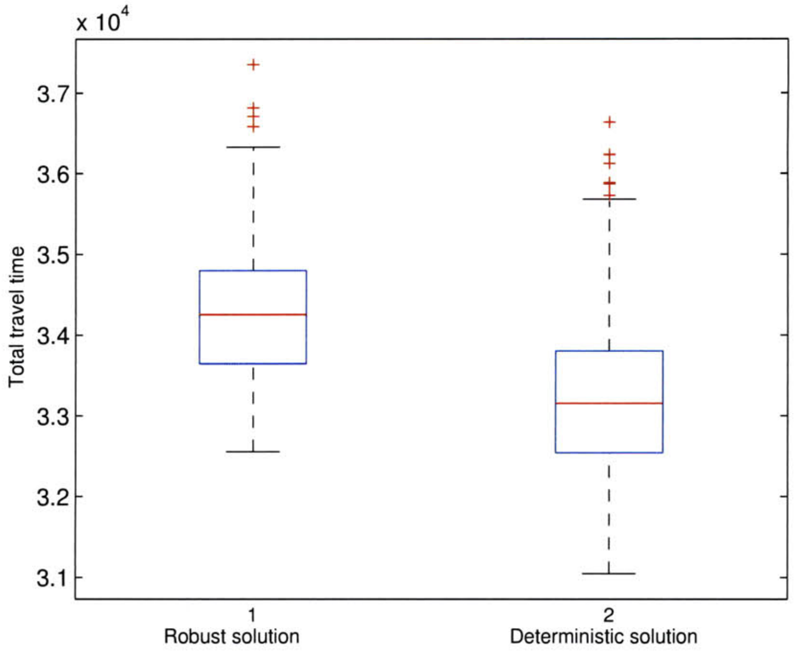

Figure 5-11: Box plot for 500 scenarios when L = 80%

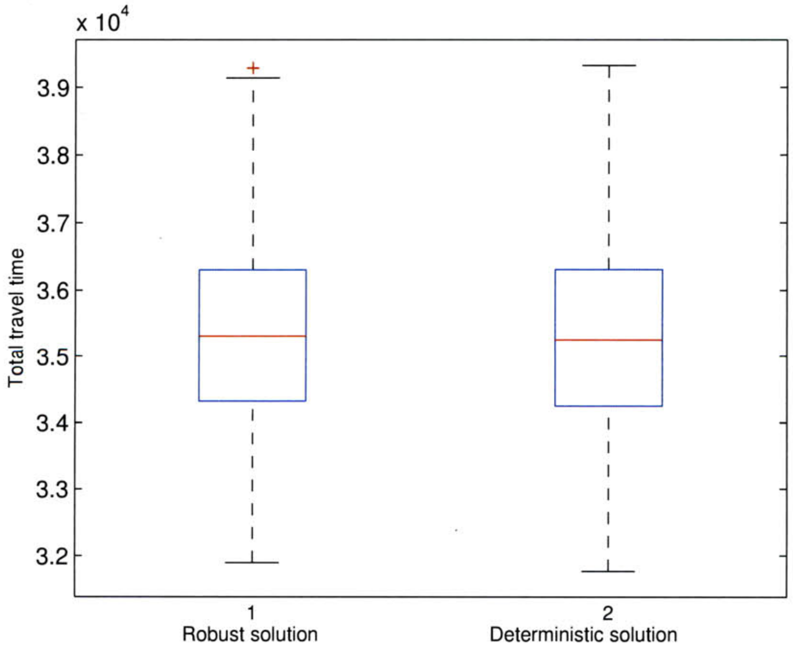

Figure 5-12: Box plot for 500 scenarios when L = 91.6%

The second part of this chapter studies the RNDP. For the RNDP, we propose a decompostion method and a heuristic method to solve the RNDP where the demands lie within a polyhedron. Our numerical examples use the Sioux Falls network to show that the robust simulated annealing method is able to solve the RNDP for large networks with many integer variables. However, there are still many rooms for improvement. First, our algorithm to find the worst demand scenario for the inner max problem is not an exact one. Second, it is important to improve the computation time for the decision problem in our decomposition algorithm, so we can use it to deal with real network design problems.

Chapter 6

Conclusions

This thesis offers a robust optimization approach to two different equilibrium-related problems in transportation.

The first part of the thesis proposes a novel inverse optimization method to find link travel times given the stable flow and a prior perception on the travel times. Under the L^1 norm objective fucntion, the inverse problem is a linear programming problem and can be solved easily. Using the robust optimization paradigm, we propose 3 distribution-free models for the uncertainty in the prior information on the travel times. We show that for the first two models, the robust problem can be solved deterministically. The first box-uncertainty model can be considered as a deterministic inverse problem with modified prior travel times. The second model is also a linear optimization problem with additional $|A|$ constraints. For the third model, the minimax problem is a hard problem. We propose an algorithm based on Bender's decomposition scheme. Instead of solving one big problem, we iteratively solve two simpler problem: one LP programming problem and one mixed integer LP problem. Both problems can be solved efficiently using CPLEX solver. We test our algorithm for the Sioux Falls network and illustrate that the decomposition algorithm converges after 400 iterations, which is very fast compared to the number of links on the network. We further extend the idea, using inverse optimziation for the historical stable flow to find the parametric forms of the travel times. Using simulated data, we show that by solving the inverse optimization problem before fitting the parameters, the fitting functions are more robust against the noise level in the prior data.

The second part of the thesis considers a binary choice network design problem under the User Equilibrium flow and uncertainty in demands. For the deterministic problem, we show that the problem can be formulated as a mixed-integer programming problem with a quadratic objective function and linear constraints. We give some insights and study the parameter choices of the formulation so that it can be solved efficiently and correctly using the CPLEX solver. For the robust problem, we model the uncertainty in demands using a linear set. We propose a decomposition algorithm, solving the robust problem iteratively through solving two smaller problems: a mixed integer programming problem with convex quadratic constraints and a convex maximization problem. Unfortunately, both of these problems are NP hard, and are not efficiently solvable for large network problem. For large network problems, we offer a heuristic approach using robust simulated annealing. Using a LP solver to find the UE flow given the design and demands of the network, the robust simulated annealing can give good performance for practical design problems. We illustrate our approach using the Nagurney's network and the Sioux Falls network.

The two problems addressed in this thesis are not well developed in the literature. Thus, there are still many interesting issues to address in consequent research.

First of all, the user choice is modeled based on Wardrop's principle. The principle assumes users know the travel information of the network accurately. This, however, is not a good assumption. Thus, it might be interesting to consider a stochastic or robust model rather than deterministic user equilibrium.

For the robust inverse problem, we only consider the uncertainty in the prior travel times. In a further study, we would like to consider the uncertainty in the flow. Besides, this robust inverse problem can be further extended to model users' behavior given the historical data and prior travel times. Users can be based on the information they have to make their route choices. This might lead to a new equilibrium behavior.

For the network design problem, we could not solve the inner problem to optimality. For a network with many OD pairs, it is critical to have a good algorithm to find the worst demand scenario. This is one of the major points for future developement. Besides, despite the fact that the robust simulated annealing approach is able to provide some good results, it is really sensitive to the choice of parameters and does not always converge to a good solution. Thus, it will be preferable to have a deterministic solution for big network problems, where we can evaluate how good our solution is.

Bibliography

[1] M. Abdulaal and J. LeBlanc. Continuous equilibrium network design models. *Transportation Research Part B: Methodological*, 13(1):19–32, 1979.

[2] M. Aghassi, D.Bertsimas, and G Perakis. Solving asymmetric variational inequalities via convex optimization. *Operations Research Letters*, 34(5):481–490, 2006.

[3] R.K. Ahuja and J.B Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

[4] R. Akcelik. Capacity of a shared lane. *Autralian Road Research Board Proceedings*, 14(2):228–241, 1988.

[5] R. Akcelik. Travel time functions for transportation planning purposes: Davidson's function, its time-dependent form and an alternative travel time function. *Autralian Road Research Board Proceedings*, 21(3):49–59, 1991.

[6] R. Andrade, A.Lisser, N.Maculan, and G. Plateau. Telecommunication network capacity design for uncertainty demand. *Computation Optimization and Applications*, 29:127–146, 2004.

[7] A. Atamturk and M.Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[8] Hillel Bar-Gera. Transportation network test problems, May 2007. `http://www.bgu.ac.il/ bargera/tntp/`.

[9] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(A):351–376, 2004.

[10] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[11] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1 13, 1999.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[13] J.F. Bendes. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[14] D. Bertsimas and O.Nohadani. Robust optimization with simulated annealing. *Journal of Global Optimization*, 2009.

[15] D. Bertsimas, D. Pachamanova, and M. Sim. Robust linear optimization under general norms. *Operations Research Letters*, 32:510–516, 2004.

[16] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[17] D. Bienstock and N. Özbay. Computing robust basestock levels. *Discrete Optimization*, 5:389–414, 2008.

[18] H.L. Bodlander, P. Gritzmann, V. Lee, and J.V. Leeuwen. Computational complexity of norm-maximization. *Combinatorica*, 10(2):203–225, 1990.

[19] D. Braess. Uber ein paradoxon der verkehrsplanung. *Unternehmensforschun*, 12:258–268, 1962.

[20] S. Dafermos. Traffic equilibria and variational inequalities. *Transportation Science*, 14:42–54, 1980.

[21] S. Dafermos and A. Nagurney. Sensitivity analysis for the asymmetric network equilibrium problem. *Mathematical Programming*, 28:174–184, 1984.

[22] M. Florian and D.Hearn. Network equilibrium models and algorithms. In *Network Routing*, pages 485–550. Elsevier B.V., 1995.

[23] Z. Gao, J.Wu, and H. Sun. Solution algorithm for bi-level discrete network design problem. *Trasportation Research Part B*, 39(2005):479–495, 2004.

[24] L. El Ghaoui, F.Oustry, and H.Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9:33–52, 1998.

[25] S. Kachani and G.Perakis. Fluid dynamics models and their applications in trasportation and pricing. *European Journal of Operational Research*, 170(2006):496–517, 2004.

[26] J. LeBlanc and D. Boyce. Bilevel programming algorithm for exact solutions of the network design problem with user-optimal flows. *Transportation Research*, 20B:259–265, 1986.

[27] L.J. LeBlanc. An algorithm for the discrete network design problem. *Transportation Science*, pages 183–199, 1975.

[28] M.J Lighthill and G.B Whitham. On kinematic waves ii. atheory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London, Series A*, 229:317–345, 1955.

[29] A. Lisser, A. Ouorou, J. Vial, and J. Gondzio. Capacitated planning under uncertain demand in telecommunication networks. 1999.

[30] Y. Lou, Y. Yin, and S. Lawphongpanich. Robust approach to discrete netwokr designs with demand uncertainty. *Trasportation Research Record 2090*, pages 86–94, 2009.

[31] Y. Lu. Robust transportation network design under user equilibrium. Master's thesis, Massachusetts Institute of Technology, 2007.

[32] T. Magnati and R. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18:1–55, 1984.

[33] P. Marcotte. Network design with continuous control parameters. *Transportation Science*, pages 181–197, 1983.

[34] P. Marcotte. Network design problem with congestion effects: a case of bilevel programming. *Mathematical Programming*, 34:142–162, 1986.

[35] C. Meneguzzer, D.E Boyce, N.Roupail, and A.Sen. Implementation and evaluation of an asymetric equilibrium route choice model incorporating intersection related travel times. Technical report, 1990.

[36] S. Mudchanatongsuk, F. Ordóñez, and J. Liu. Robust solutions for network design under transportation cost and demand uncertainty. *Journal of the Operational Research Society*, 59:652–662, 1986.

[37] A. Nagurney. Comparative tests of multimodel traffic equilibrium methods. *Transportation Research Part B*, 18B(6):469–485, 1984.

[38] Bureau of Public Roads. *Traffic assigment manual.* US Dep. Of Commerce, Washington, DC, 1964.

[39] U.S. Department of Transportation. 2009 urban mobility report and appendices, 2009. `http://mobility.tamu.edu/ums/report/`.

[40] U.S. Department of Transportation. Fy 2009 dot summary of performance and financial information, 2009. `http://www.dot.gov/budget/2010/fy2009parsummary.pdf`.

[41] F. Ordónez and J.Zhao. Robust capacity expansion of network flows. *Networks*, 50(2):136–145, 2007.

[42] G. Perakis. The price of anarchy when costs are non-separable and asymmetric. *Mathematics of Operations Research*, 32:614–628, 2007.

[43] G. Perakis and G.Roels. An analytical model for traffic delays and the dynamic user equilibrium problem. *Operations Research*, 54(6):1151–1171, 2006.

[44] H. Poorzahedy and F. Abulghasemi. Application of ant system to network design problem. *Trasportation*, 32:251–273, 2005.

[45] H. Poorzahedy and M.A. Turnquist. Approximate algorithms for the discrete network design problem. *Transportation Research Part B: Methodological*, 16(1):45–55, 1982.

[46] M. Riss and K.Andersen. Capacitated network design with uncertain demand. *INFORMS Journal on Computing*, 14:56–70, 2002.

[47] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[48] R.L Tobin. Sensivity anaylsis for variational inequalities. *Journal of Optimization Theory and Application*, 48(1):191–204, 1986.

[49] R.L Tobin and T.L. Friesz. Sensivity anaylsis for equilibrium network flow. *Transportation Science*, 22(4):242–250, 1986.

[50] J.G. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of the Institue of Civil Enginerings*, pages 325–378. 1952.

[51] Y. Yin. Genetic algorithms based approach for bilevel programming models. *Journal of Transporation Engineering*, 126(2):115–120, 2000.

[52] L. Zhang, S. Lawphongpanich, and Y.Yin. An active-set algorithm for discrete network design problems. In *Transportation and Traffic Theory 2009: Golden Jubilee*, pages 283–300. Springer Science+ Business Media, 2009.