Network Design for Integrated Vehicle-Sharing and Public Transportation Service

by

Tianli Zhou

B.Eng. Industrial Engineering
Tsinghua University, 2013

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Author .... Signature redacted ....
Department of Civil and Environmental Engineering
May 20, 2015

Certified by .... Signature redacted ....
Cynthia Barnhart
Chancellor and Ford Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by .... Signature redacted ....
Carolina Osorio
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .... Signature redacted ....
Heidi Nepf
Donald and Martha Harleman Professor of Civil and Environmental Engineering
Chair, Graduate Program Committee

Network Design for Integrated Vehicle-Sharing and Public Transportation Service

by

Tianli Zhou

Submitted to the Department of Civil and Environmental Engineering
on May 20, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

Vehicle sharing services have become a major urban transportation mode. One-way vehicle sharing service facilitates access to public transportation systems, thereby addressing the first and last mile challenges and creating an integrated vehicle-sharing and public transportation network providing origin-to-destination service. In this thesis we provide models and methods for design one-way vehicle-sharing networks. The location of one-way vehicle sharing stations strongly influence the level of travel time savings achieved by the users of the system. Our goal, then, is to select station locations so as to maximize the connectivity with the public transportation system, increase the accessibility to the urban area, reduce travel times, reduce congestion, and reduce emissions. We select a certain number of stations to install from a set of candidates whose locations are predetermined.

In Chapter 2, we review existing literature in which the objective is to minimize total user travel cost. In Chapter 3, we propose a new model with the objective to design a network such that more users experience travel time savings that are sufficiently large to elicit mode shifts to the integrated public transportation option. We develop a decomposition procedure to solve our model and propose cut generation methods to expedite the solution process. Computational results in Chapter 4 show that our algorithm reduces solution times, while increasing the number of travelers who can experience travel time savings of significance by using our newly designed network. In Chapter 5, we propose a heuristic method to generate a network design with (near-) minimal total travel cost. Our decomposition method that searches in a neighborhood around the known best design, and changes the neighborhood center when improved solution are identified or expands the neighborhood if no better solution is found. Computational results show that our algorithm finds improved solutions, compared to existing approaches, for large-scale networks with imposed limits on computation time. In Chapter 6, we conclude the thesis and provide future research guidance.

Thesis Supervisor: Cynthia Barnhart
Title: Chancellor and Ford Professor of Civil and Environmental Engineering

Thesis Supervisor: Carolina Osorio
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisors, Professor Cynthia Barnhart and Professor Carolina Osorio, for everything you have done for me. Without doubt, your guidance, expertise and patience support my journey at MIT and contribute greatly to this thesis. You teach me how to be a good researcher. You are my role models. It is my honor to know you.

I would like to thank Dr. Virot Chiraphadhanakul. My research could not have been done without your help. Your patience and insights open my door to the world of data science and sharing economy. You are amazing.

I would like to thank Maria Marangiello, who always tries hard to find me a good time slot in Cindy's busy schedule. I would also like to appreciate the Ford-MIT Alliance for sponsoring this research.

Additionally, many thanks go to Yin Wang, Linsen Chong, Yan Zhao, Chiwei Yan, Hai Wang and He Sun. You are like brothers and sisters to me. Your sincere care and suggestions help me survive from the MIT life. You are incredible.

Many thanks go to my roommates and friends Haizheng Zhang and Chao Zhang. You make my life at MIT an enjoyable and unforgettable one. And I also like to thank all my MST fellow students who make it a diverse and vibrant community.

I would like to thank Professor Hai Jiang, who taught me in Tsinghua University and brought me into the world of operations research and transportation. You make me believe that I can use my knowledge and skills to make the world a better place. Your suggestions about the future life and career choosing are really valuable for me.

Finally, I would like to thank my family, especially my father Xiaorui Zhou and my mother Yueming Liu. Your love and support are, and will always be, my strongest power to move forward.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Motivation

Vehicle-sharing (VS) has become one of the mainstream urban transportation modes for over one million users in the past decades. It has been deployed in about 1100 cities of 26 nations. These services had at least 31,665 vehicles and 1,251,504 members in 2010 worldwide, bringing the world lower greenhouse gas emissions and reducing the vehicle miles traveled, thereby benefiting both the environment and traffic congestion (Shaheen and Cohen, 2013). Well-known car-sharing programs include Car2Go and ZipCar. For bike-sharing, by 2011, over 236,700 bikes and 13,500 stations have been installed in 135 bike-sharing programs worldwide, these include Hubway in Boston, Citibike in NYC and Verlib in Paris (Shaheen et al., 2012).

Most major VS services nowadays require round-trip travel, that is, users must return the vehicle back to the same station where it is picked up. However, one-way sharing services, where users can drop off the rented vehicle at any station with capacity to accept the vehicle, provide them flexibility. These services also provide users with better access to existing public transit systems. Such an advantage is particularly impactful in areas where travel demand is low or where adding transit services, in terms of both frequency and new routes, is not a practical option.

Chiraphadhanakul (2013) presents an optimization model and solution approach to minimize total user travel time by designing a network that integrates the one-way

VS system and the existing transit system. While effective for smaller networks, this approach encounters tractability issues when real-world, large-scale input networks are unsolved. Meanwhile, maximum travel time savings can be achieved by small amounts the travel time for many origin-destination (OD) pairs. Our concern is that these travel time savings might not be sufficient to change traveler's mode choice and hence, these savings will not be realized.

In this thesis, we address these two issues: tractability of the VS network design for large-scale problems and use of sufficient travel time savings to induce use of the integrated VS-public transportation networks. We first provide a new model and its corresponding solution method for the design of one-way VS service, with the explicit goal of integrating it with the public transportation system and achieving travel time savings of significance to travelers. Then we provide heuristic method aimed at expediting the solution of the model proposed by Chiraphadhanakul (2013), particularly for large-scale, real-world sized urban networks.

1.2 Related Work

Previous researches on one-way VS systems mainly focused on vehicle re-distribution, demand estimation, user behavior, and VS market and benefits evaluation. Jorge and Correia (2013) provide a thorough literature review about these topics. The literature on VS network design is very limited. To our knowledge, this work is one of the first that takes a passenger-centric approach to framing this problem. Lin and Yang (2011) use a nonlinear optimization model to find the best locations for bike-sharing stations, taking into consideration service levels. Their work takes operational issues into account, but makes strong assumptions, such as constant lead time. Kumar and Bierlaire (2012) use a two-step model to analyze the performance of the electric car-sharing service. They use regression methods to identifying factors that impact demand, then use nonlinear integer programming optimization to select station locations. Correia and Antunes (2012) propose a mixed-integer programming (MIP) model for one-way car-sharing system design. Their model selects locations to install VS stations by considering demand, temporal issues, revenue and cost. They consider the car-sharing information system but do not take the existing transit system into account. Correia et al. (2014) propose a MIP model to design the VS network. Their model allows users to choose a nearby station when they find no vehicle is available at the nearest one, with design objective to maximize profit for the service provider. Nair and Miller-Hooks (2014) provide a bi-level MIP model with the objective to maximize the profit of the bike-sharing system. They consider station locations, commuter flows, bike re-distribution and major transit service. They report that tractability issues are encountered in solving relatively large problems. García-Palomares et al. (2012) use a GIS-based approach to design a one-way VS network. They calculate the demand for trips and use location-allocation models to determine station locations. In their work, they compare two objective functions: minimizing impedance and maximizing coverage. Chiraphadhanakul (2013) propose several MIP models aimed at minimizing total user travel cost. The author solves these models using Benders decomposition. and also proposes a cut strengthening algorithm to accelerate the solution process. In this thesis, we build upon this work.

1.3 Contribution and Thesis Outline

The contributions of this work are two folds. First, we provide a new formulation for the VS network design problem that can provide a solution designed to induce changes in travel behavior. Specifically to lead travelers to shift their modes of travel to the newly integrated VS and public transportation networks. Specifically, we consider the bike-sharing network design problem, and we build off the work of Chiraphadhanakul (2013). In particular, we consider a set of candidate VS stations and optimize the selection of a subset of them. The design goal is for travelers to combine the bike trips and the transit trips so their travel times are reduced, compared with using the transit system only. We assume that number of the stations to install is predetermined. We also assume that given the selected candidate stations, all users will be wise enough to choose their best route over the integrated VS and public transportation network.

The new model is a mixed-integer program (MIP) with the objective to increase the number of travelers with enough travel time savings, that is, savings that are sufficiently large for passengers to shift their travel patterns to use the integrated VS and public transportation network. To expedite the solution process, we decompose our model and develop special cut generation methods.

In Chapter 2 we review the network design model and the solution method proposed in Chiraphadhanakul (2013). In Chapter 3, we present our new network design model formulation, the model decomposition and cut generation methods. In Chapter 4, we present computational results that show the efficiency of our solution method and its effectiveness in generating travel time savings.

In this thesis, we also contribute by providing methods to speed-up the solution of the model presented in Chiraphadhanakul (2013). Our methods achieve improved total travel cost results for real-world, large-scale networks. In Chapter 5, we show that solving the Chiraphadhanakul (2013) model for large-scale networks is difficult. We provide a decomposition heuristic to find quality solutions within limited time. Computational results on random networks and a real-world Boston-based network are then presented.

Finally, we conclude this thesis in Chapter 6.

Chapter 2

Minimizing Total Travel Cost

2.1 Introduction

In this chapter, we review the vehicle-sharing (VS) network design model proposed in Chapter 4 of Chiraphadhanakul (2013). Our philosophy of design is passenger-centric. We consider the VS system and the existing transit system at the same time, and find a design such that the total travel cost of all travelers, referred to as demand, is minimized. Here cost and the travel time are synonymous. The problem we want to solve is a facility location problem in nature: we assume that there is a set of predefined candidate stations whose locations have been selected in advance. The problem is formulated as a mixed-integer program (MIP). We denote the formulation in this chapter as the min-cost model because the objective here is to minimize the total travel cost. Two formulations, OD-based and tree-based, are presented and discussed.

In the following sections, we first present the notation of the min-cost model and the corresponding two formulations. Then we briefly show the decomposition and cut generation methods to solve the model.

2.2 Model Formulation

In this section we present the notation and formulations of the min-cost VS network design problem. We will present both the OD-based formulation and the tree-based formulation. The connection and difference between the two formulations will be elaborated later.

We model the whole system as a directed graph. The candidate stations and transit stops form the set of nodes. The set of arcs includes transit/walking connections between two bus/subway/trolley/ferry stops, walking trips between a VS station and a transit stop and VS trips between two VS stations. One thing to note is that for the one-way VS sharing system, the vehicles can only travel between two existing stations. So a VS trip arc can be considered as non-existing if at least one of its end VS stations is not installed.

2.2.1 Notation

Parameters:

- G: a directed graph $G = (\mathcal{N}, \mathcal{A})$;
- $\mathcal{N}$: set of nodes, including transit stops and candidate; stations. Locations are predetermined;
- $\tilde{\mathcal{N}}$: $\tilde{\mathcal{N}} \subseteq \mathcal{N}$, set of candidate stations;
- $\mathcal{A}$: set of arcs, each $(i, j) \in \mathcal{A}$ incurs a positive cost of c_{ij};
- $\tilde{\mathcal{A}}$: $\tilde{\mathcal{A}} \subseteq \mathcal{A}$, set of VS arcs. We use b-arc for short as we mainly consider bike sharing service in the following chapters. A b-arc is available only if both of its end stations are installed;
- $\mathcal{H}$: set of all OD-pairs, $\mathcal{H} \subseteq \mathcal{N} \times \mathcal{N}$;
- h: one OD pair, $h \in \mathcal{H}$;

- $\mathcal{L}$: set of all origin nodes;
- l: one origin, $l \in \mathcal{L}$;
- w^h: the travel demand from the origin $o(h)$ to the destination $d(h)$, $\forall h \in \mathcal{H}$;
- W^l: the travel demand originating from the origin l, $W^l = \sum_{h \in \mathcal{H}^l} w^h$;
- $\mathcal{H}^l$: all OD pairs that have l as their origin;
- K: the number of stations to install.

Decision variables:

- y_i: binary variable indicating whether a candidate $i \in \tilde{\mathcal{N}}$ is selected in a solution;
- x_{ij}^h: continuous variable used in the OD-based formulation and indicating flow traveling between OD-pair $h \in \mathcal{H}$ on arc $(i,j) \in \mathcal{A}$;
- x_{ij}^l: continuous variable used in the tree-based formulation and indicating flow originated from the origin $l \in \mathcal{L}$ on arc $(i,j) \in \mathcal{A}$.

In the computational experiments of the following chapters, we will briefly discuss the data we use and how to generate the input network. For more details, see Chapter 3 of Chiraphadhanakul (2013).

2.2.2 OD-based Formulation

In this part of the thesis, we present the OD-based formulation of the VS network design problem with an objective to minimize total travel time. The formulation is the following:

$$\text{minimize} \quad \sum_{h \in \mathcal{H}} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}^h \tag{2.1}$$

subject to

$$\sum_{(i,j) \in \mathcal{A}} x_{ij}^h - \sum_{(j,i) \in \mathcal{A}} x_{ji}^h = \begin{cases} w^h & i = o(h) \\ -w^h & i = d(h) \\ 0 & \text{otherwise} \end{cases} \qquad \forall h \in \mathcal{H},\ \forall i \in \mathcal{N} \tag{2.2}$$

$$x_{ij}^h \le w^h y_i \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.3}$$

$$x_{ij}^h \le w^h y_j \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.4}$$

$$\sum_{i \in \tilde{\mathcal{N}}} y_i = K \tag{2.5}$$

$$x_{ij}^h \ge 0 \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \mathcal{A} \tag{2.6}$$

$$y_i \in \{0,1\} \qquad \forall i \in \tilde{\mathcal{N}}. \tag{2.7}$$

In the objective function Equation (2.1), we want to minimize the sum of the travel cost for the flow from all OD pairs and on all arcs. Equation (2.2) represents the flow conservation requirement of a network problem. For any OD pair, the flow out of the origin node is the demand of this OD pair, the flow into the destination is also the demand, and for other nodes in the network, the in-flow and out-flow should be the same. Equation (2.3) and (2.4) guarantee that the flows on b-arcs can be larger than zero only if both of the b-arc's end stations are selected to be installed. We denote them as the b-arc availability constraints. Equation (2.5) is the restriction that the total number of stations to install is K.

As this formulation is based on each OD pair, there will be $|\mathcal{H}| \times |\mathcal{N}|$ flow conservation constraints and $2 \times |\mathcal{H}| \times |\tilde{\mathcal{A}}|$ b-arc availability constraints. The problem size increases linearly with the number of OD pairs.

2.2.3 Tree-based Formulation

As in the OD-based formulation, the tree-based formulation objective is to minimize the total travel cost of all users over the entire network. The formulation is the following:

$$\text{minimize} \qquad \sum_{l \in \mathcal{L}} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}^l \tag{2.8}$$

subject to

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^l - \sum_{(j,i)\in\mathcal{A}} x_{ji}^l = \begin{cases} W^l & i = l \\ -w^h & (l,i) = h \in \mathcal{H} \\ 0 & \text{otherwise} \end{cases} \qquad \forall l \in \mathcal{L},\ \forall i \in \mathcal{N} \tag{2.9}$$

$$x_{ij}^l \le W^l y_i \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.10}$$

$$x_{ij}^l \le W^l y_j \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.11}$$

$$\sum_{i\in\tilde{\mathcal{N}}} y_i \le K \tag{2.12}$$

$$x_{ij}^l \ge 0 \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \mathcal{A} \tag{2.13}$$

$$y_i \in \{0,1\} \qquad \forall i \in \tilde{\mathcal{N}}. \tag{2.14}$$

Similar to the OD-based formulation, Equation (2.9) represents the flow conservation constraints. Equation (2.10) and (2.11) represent the b-arc availability constraints. In the objective function, we sum the cost of all flows originating from each of the OD pairs. As we have flow conservation constraints, the total flow out of or into a origin l is $W^l = \sum_{h\in\mathcal{H}^l} w^h$. This indicates that the flows of any feasible OD-based solution can be aggregated into the flow of a feasible tree-based solution. Thus, the objective function Equation (2.8) minimizes total travel cost. The optimal objective values of the two formulations are the same.

For the tree-based formulation, there will be $|\mathcal{L}| \times |\mathcal{N}|$ flow conservation constraints and $2 \times |\mathcal{L}| \times |\tilde{\mathcal{A}}|$ b-arc availability constraints. As the flow between the OD pairs are aggregated into the flow from the origins, there are fewer constraints in the tree-based formulation than in the OD-based formulation.

It can be shown that in terms of the linear programming (LP) relaxation, the OD-based formulation is stronger than the tree-based one. But when the input network size and number of OD pairs are large, both formulations may be hard to solve. In the next section, we will review the methods that can solve the tree-based formulation in a fast way.

2.3 Model Decomposition and Cut Generation

In this section we present the methods for model decomposition and cut generation proposed in Chapter 4 of Chiraphadhanakul (2013). These methods aim at expediting the solution process of the tree-based min-cost model. The decomposition procedure is based on Benders decomposition. The cut improving methods are based on the idea of Pareto optimal cut and the network structure of our problems.

2.3.1 Benders Decomposition of the Tree-based Model

Benders decomposition (Benders, 1962) is a method that is often applied to large-scale MIP problems with special structure. Consider a MIP problem:

$$\begin{aligned} \text{minimize} \quad & cx + dy \\ \text{subject to} \quad & Ax + Dy \geq b \\ & Fy \geq f \\ & x \in \mathbb{R}_+^{n_1},\ y \in \mathbb{Z}_+^{n_2}, \end{aligned} \tag{OP}$$

where all c, d, A, D, b, F and f are in the corresponding dimensions. Denote it as original problem (OP). We assume OP is finite optimal. Let $Y = \{y \in \mathbb{Z}_+^{n_2} \mid Fy \geq f\}$. For a given $\bar{y} \in Y$, OP becomes

$$\begin{aligned} z(\bar{y}) \quad = \quad \text{minimize} \quad & cx \\ \text{subject to} \quad & Ax \geq b - D\bar{y} \\ & x \in \mathbb{R}_+^{n_1}. \end{aligned} \tag{PS}$$

Denote it as primal subproblem (PS). PS is a linear program, and its dual form is

$$z(\bar{y}) = \max_{\pi \geq 0}\{\pi(b - D\bar{y}) \mid \pi A \leq c\}, \tag{DS}$$

where π is the vector of dual variables associated with the constraints $Ax \geq b - D\bar{y}$. As we assume problem (OP) is finite optimal, the dual subproblem (DS) is also finite optimal. In such cases, the strong duality theory ensures that the optimal values of both PS and DS are the same. Let Q be the set of extreme rays and P be the extreme points of the polyhedron $\{\pi \mid \pi A \leq c\}$, respectively. It can be shown that problem (OP) is equivalent to the master problem (MP)

$$\begin{array}{lll} \text{minimize} & z + dy & \\ \text{subject to} & q(b - Dy) \leq 0 & \forall q \in Q \\ & p(b - Dy) \leq z & \forall p \in P \\ & y \in Y. & \end{array} \tag{MP}$$

$q(b - Dy) \leq 0$ is called a feasibility cut and $p(b - Dy) \leq z$ is called an optimality cut. We can solve problem (OP) iteratively. In each iteration, we first solve the relaxed master problem (RMP) and get a lower bound (LB) of the optimal value of problem (OP) and a first-stage solution $\bar{y}$. Then we use $\bar{y}$ to solve a DS and get an upper bound (UB) of the optimal value of problem (OP), and an extreme ray q or an extreme point p. And finally, we add the feasibility cut or optimality cut parameterized by q or p to (RMP) and resolve it. When the largest LB is close enough to the smallest UB, we stop with the near optimal solution of y corresponding to the smallest UB.

For the VS network design problem, as the cut improving method we will review later results in the tree-based formulation outperforming the OD-based approach, we present the Benders decomposition for the tree-based formulation only. The tree-based PS for a given VS network design $\bar{y}$ is formulated as

$$\text{minimize} \quad \sum_{l \in \mathcal{L}} \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij}^{l} \tag{2.15}$$

subject to

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^l - \sum_{(j,i)\in\mathcal{A}} x_{ji}^l = \begin{cases} W^l & i = l \\ -w^h & (l,i) = h \in \mathcal{H} \\ 0 & \text{otherwise} \end{cases} \qquad \forall l \in \mathcal{L},\ \forall i \in \mathcal{N} \tag{2.16}$$

$$x_{ij}^l \le W^l \bar{y}_i \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.17}$$

$$x_{ij}^l \le W^l \bar{y}_j \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.18}$$

$$x_{ij}^l \ge 0 \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \mathcal{A}. \tag{2.19}$$

Let p_i^l, u_{ij}^l, and v_{ij}^l denote the negatives of the dual variables associated with constraints (2.16), (2.17), and (2.18), respectively. The corresponding tree-based DS is given by:

$$\text{maximize} \qquad z(\bar{y}) = \sum_{l\in L}\Bigg[\sum_{h\in\mathcal{H}^l} w^h\left(p_{d(h)}^l - p_{o(h)}^l\right) - \sum_{i\in\tilde{\mathcal{N}}} W^l\left(\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^l + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^l\right)\bar{y}_i\Bigg] \tag{2.20}$$

subject to

$$p_j^l - p_i^l \le c_{ij} \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \mathcal{A} \setminus \tilde{\mathcal{A}} \tag{2.21}$$

$$p_j^l - p_i^l \le c_{ij} + u_{ij}^l + v_{ij}^l \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{2.22}$$

$$u_{ij}^l, v_{ij}^l \ge 0 \qquad \forall l \in \mathcal{L},\ \forall (i,j) \in \tilde{\mathcal{A}}, \tag{2.23}$$

where $\mathcal{H}^l$ denotes the set of O-D pairs originating from node $l \in \mathcal{L}$. We decompose any subproblem into $|\mathcal{L}|$ subproblems, each of which corresponds to an origin point $l \in \mathcal{L}$. Then the Benders cut corresponding to l is

$$z^l \ge \sum_{h\in\mathcal{H}^l} w^h\left(p_{d(h)}^l - p_{o(h)}^l\right) - \sum_{i\in\tilde{\mathcal{N}}} W^l\left(\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^l + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^l\right)\bar{y}_i, \tag{2.24}$$

where $o(h) = l$.

2.3.2 Cut Generation

Benders decomposition is guaranteed to converge to the optimal solution. However, the solution process can be very time-consuming. Solving RMP is usually the bottleneck because it is an integer program whose number of cuts is increasing.

Pareto Optimal Cuts

To reduce the number of iterations, Chiraphadhanakul (2013) uses the idea of Pareto optimal cuts from Magnanti and Wong (1981):

1. The cut $\pi^1(b - Dy) \le z$ generated from a dual solution $\pi^1 \in \Pi$ dominates or stronger than the cut $\pi^2(b - Dy) \le z$ generated from $\pi^2 \in \Pi$ if $\pi^1(b - Dy) \ge \pi^2(b - Dy)$ for all $y \in Y$, with a strict inequality for at least one point.

2. A cut is Pareto optimal (PO) if no cut dominates it.

Here we use the notation of the PS proposed in Section 2.3.2. To obtain a PO cut, we need to solve an auxiliary problem (AP). Let y^0 be a point in the relative interior of the convex hull of Y (core point). AP is

$$\begin{aligned} \text{maximize} \quad & \pi(b - Dy^0) \\ \text{subject to} \quad & \pi(b - D\bar{y}) = z(\bar{y}) \\ & \pi A \le c \\ & \pi \ge 0. \end{aligned} \tag{AP}$$

For the min-cost VS network design problem, we can decompose the AP of the tree-based formulation into $|\mathcal{L}|$ different ones. Then we generate a PO cut for the sub-problem of any origin $l \in \mathcal{L}$ by solving the corresponding sub-AP.

Cut Improving

For an origin $l \in \mathcal{L}$, let (p^l, u^l, v^l) be an optimal solution to the subproblem of the tree-based DS. Given a feasible network design $\bar{y}$, the following two propositions are proved in Chapter 4 of Chiraphadhanakul (2013).

Proposition 1. An optimal solution (p^l, u^l, v^l) is also an optimal solution to the subproblem of the O-D-based DS associated with any O-D pair $h \in \mathcal{H}^l$.

Proposition 2. Consider the O-D-based DS associated with O-D pair $h \in \mathcal{H}$ for a given $\bar{y} \in Y$ with an optimal cost of $z^h(\bar{y})$. Let (p^1, u^1, v^1) be an optimal solution and (p^2, u^2, v^2) be another feasible solution. If $p^2_{o(h)} = p^1_{o(h)}, p^2_{d(h)} = p^1_{d(h)}, u^2_{ij} \leq u^1_{ij}$ for all $(i,j) \in \tilde{\mathcal{A}}$, and $v^2_{ij} \leq v^1_{ij}$ for all $(i,j) \in \tilde{\mathcal{A}}$, the (p^2, u^2, v^2) is also optimal.

Based on the two propositions, a procedure to improve a valid cut is proposed. For each origin $l \in \mathcal{L}$, consider each $h \in \mathcal{H}^l$. The Basic algorithm will find a sequence of all nodes except $o(h)$ and $d(h)$, update p^h_i by letting $p^h_i = \max_{j:(i,j)\in\mathcal{A}}\{p^h_j - c_{ij} - u^h_{ij} - v^h_{ij}\}$ ($\forall i \in \mathcal{N}$, $i \neq o(h)$ and $d(h)$), then update the value of u^h_{ij} and v^h_{ij} by letting

$$u^h_{ji} = \frac{u^l_{ji}}{u^l_{ji} + v^l_{ji}}\delta_{ji} \quad \text{and} \quad v^h_{ji} = \frac{v^l_{ji}}{u^l_{ji} + v^l_{ji}}\delta_{ji},$$

where $\delta_{ji} = \max(0, p^h_i - p^h_j - c_{ji})$. It is shown by Chiraphadhanakul (2013) that computation complexity of the Basic algorithm is $O(|\mathcal{N}|\log|\mathcal{N}| + |\mathcal{H}^l|(|\mathcal{A}| + |\tilde{\mathcal{A}}|))$.

In the Improved algorithm, Chiraphadhanakul (2013) execute the Basic algorithm for *pass* times (*pass* is a parameter). The insight is that after executing the Basic algorithm once there may still be possibilities to improve cut strength. But executing the Basic algorithm may be time-consuming. So we may need to determine in advance how many times we need to do so.

2.4 Conclusion

We do not present other details of the Basic algorithm and the Improved algorithm here due to their complexity. The computational results in Chiraphadhanakul (2013) show that the combination of the decomposition of the tree-based formulation, the PO cut strategy and the cut improving method manage to reduce solution time, compared with solving either the OD-based formulation or the tree-based formulation by classical Benders decomposition. Generally speaking, the larger the network size is, the more *pass* we need to shorten solution times.

Chapter 3

Maximizing the Number of Passengers with Time Savings above a Minimal Threshold

3.1 Introduction

In this chapter, we provide an alternative model formulation for the one-way VS system design. We still consider the VS system and the existing transit system at the same time, and find a design such that the two systems can complement each other. Unlike the previous model of minimizing total travel cost introduced by Chiraphadhanakul (2013), we propose a new model focusing on increasing the number of demands with satisfactory travel time savings. To solve it, we present a decomposition procedure and two cut generation methods.

3.1.1 Motivation: Drawback of Minimizing Total Travel Cost

For the model proposed in the previous chapter, solution quality is evaluated by the reduction in total travel cost over the whole network and for all OD pairs. As we examine the travel cost savings of each OD pair for solutions from the Chiraphadhanakul (2013) model, many OD pair savings reduce travel time by very small amounts even though these OD pairs have the potential to produce larger travel time savings. In other words, the objective to minimize total travel cost may not give sufficient savings to entice people to alter their travel patterns. We illustrate this point in the following example.

For an input network with 704 nodes($\mathcal{N}$), 3,008 arcs($\mathcal{A}$), 30 b-nodes($\tilde{\mathcal{N}}$), 742 b-arcs($\tilde{\mathcal{A}}$), 5,103 OD($\mathcal{H}$), 30 origins($\mathcal{L}$), and total demand of 248,192, we solve the Chiraphadhanakul (2013) model with $K = 15$, which means that we need to choose 15 out of 30 candidate stations to install.

Table 3.1: Time Savings Analysis for OD Pairs

Savings (min)	Demand				Comparison	
	All-Open		$K = 15$		Ratio	Percentage Diff
≥ 50	1390	0.56%	968	0.39%	69.64%	0.17%
≥ 45	3279	1.32%	2596	1.05%	79.17%	0.28%
≥ 40	6955	2.80%	4235	1.71%	60.89%	1.10%
≥ 35	13632	5.49%	8189	3.30%	60.07%	2.19%
≥ 30	24448	9.85%	17277	6.96%	70.67%	2.89%
≥ 25	37869	15.26%	24524	9.88%	64.76%	5.38%
≥ 20	57995	23.37%	35084	14.14%	60.49%	9.23%
≥ 15	92715	37.36%	67497	27.20%	72.80%	10.16%
$0 < Savings < 15$	86950	35.03%	73034	29.43%	84.00%	5.61%

We analyze the travel time savings of each OD pair for the best solution. The result is summarized in Table 3.1. We consider two cases: one where all 30 candidate stations are open and one where the 15 candidate stations in the best solution are open. We compute the sum of the demand of the OD pairs that have at least $50, 45, \ldots, 15$ minute savings, and present the values in columns 2 and 4 for the two cases, respectively. Demand of OD pairs that have positive but fewer than 15 minutes time saving is presented in the last line. For a row with the first column as $savings \geq s$ minutes, column 3 and 5 are the ratios between the demand that achieves at least s minutes savings and the total demand. Column 6 is the ratio between values in column 2 and values in column 4. Column 7 is the difference between column 3 and column 5. From this table, we see that the demand that realizes more than $15, 20, \ldots, 50$ minute savings for the $K = 15$ solution is around 60% to 70% of the demand that realizes the same level of savings for the all-open case (the ≥ 45 case is about 80%, but the demand is small even for the all open case). The demand with savings fewer than 15 minutes, however, is 84%. This indicates that the solution fails to achieve the savings potential for more demands.

3.1.2 Chapter Structure

In the following sections we review literature on two-stage stochastic programs with recourse. Then we present the notation and propose our new model formulation, an instance of a two-stage stochastic program with recourse. In Section 3.3 we present our decomposition procedure and the cut generation methods to solve our model. Computational results are presented in the next chapter.

3.2 Literature Review

The main contribution of this chapter is the max-demand model for the VS network design problem and the corresponding cut generation methods. We view the max-demand model as an instance of two-stage stochastic programs with recourse. In this section we will briefly review the definition of stochastic programming and some of its solution methods.

3.2.1 Two-Stage Stochastic Program with Recourse

There are two types of stochastic programs: stochastic programs with recourse models and stochastic programs with chance/probability-constraints (Zhu, 2006). Our focus is on the first type with two stages of decisions. After the first stage decisions have been made, the decision maker will be faced with different scenarios with different probabilities. In each scenario, a specific second-stage decision should be made. For a stochastic program with probability-constraints, in-feasibility is allowed with some probability.

If there are reasonable schema to divide the uncertainty of the second stage problem into a limited number of scenarios, according to Bertsimas and Tsitsiklis (1997), a two-stage stochastic program with recourse can be formulated as

$$\begin{aligned}
\text{maximize} \quad & cy + \alpha_1 f x_1 + \alpha_2 f y_2 + \ldots + \alpha_K f x_K & \text{(TSSP)}\\
\text{subject to} \quad & Ax = b & \\
& B_1 y + D x_1 = d_1 & \\
& B_2 y + D x_2 = d_2 & \\
& \vdots & \\
& B_K y + D x_K = d_K & \\
& y \in \mathbb{R}_+^{n1}; x_1, x_2, \cdots x_K \in \mathbb{R}_+^{n2}, &
\end{aligned}$$

where c, f, D, d_i and B_i ($\forall i \in \{1, 2, \cdots, K\}$) have their corresponding dimensions. Denote it as TSSP. TSSP has K scenarios. The y is the first stage decision variables and the x's are the second stage decision variables of different scenarios. D is called recourse matrix, and it may be scenario specific. Each scenario $i \in \{1, 2, \ldots, K\}$ has a weight α_i in the objective function. Because the objective function can be viewed as to minimize the sum of the first-stage cost and the expected second-stage cost, a typical stochastic programming problem will have α_i defined as the probability of scenario i occurring. Hence $\sum_i^K \alpha_i = 1$. But this can be relaxed. If we let some of the components of the vector x_i ($\forall i \in \{1, 2, \cdots, K\}$) be integer, then the problem becomes a stochastic mixed-integer programming (SMIP) problem.

SMIP has been studied for decades. To our best knowledge, there are some general purpose SMIP solving algorithms available now, but they are not efficient enough to solve large-scale problems. A typical large scale problem can be an airline fleet assignment problem with 900 flight legs and 50 scenarios (Zhu, 2006). SMIP is viewed as one of the most challenging optimization problems (Sherali and Zhu, 2006).

3.2.2 Solving Two-Stage Stochastic Programs with Recourse

Laporte and Louveaux (1993) present an L-shape based method of generating cuts for two-stage SMIP with recourse. If a problem has a set of optimality cuts and a set of feasibility cuts, the L-shape method will guarantee to find the optimal solution within a finite number of iterations. Their problem had only binary variables in the first-stage problem and binary and continuous variables in the second-stage problem. This method requires the original problem to be bounded. For a minimization problem, the authors propose the following cut as Equation (3.1) given a current first-stage decision variable x^r ($x^r \in \{0,1\}^n$) at the rth iteration:

$$\theta \geq (\theta_r - L)\left(\sum_{i\in S_r} x_i - \sum_{i\notin S_r} x_i\right) - (\theta_r - L)(|S_r| - 1) + L, \tag{3.1}$$

where θ is the upper bound for the objective value related to the second-stage decision variables, θ_r is the second-stage objective value of the current iteration r, L is a lower bound of the original minimization problem and $S_r = \{i : x_i^r = 1 \text{ and } i = 1, 2, \cdots, n\}$. The intuition of the cut is that any change of the first-stage decision variables will allow the value of θ to be less than θ_r. Thus when we minimize the objective function, this type of cut will help us eliminate the visited feasible first-stage solutions. This method only requires some simple assumptions and can be applied in many SMIP problems. It is a general purpose method, but not computationally efficient.

Besides L-shape methods, the Reformulation-Linearizion Technique (RLT) (Sherali and Adams, 1998) and disjunctive programming (Balas, 1979) are often used to develop solution methods for two-stage SMIP with recourse.

Sherali and Fraticelli (2002) propose a decomposition method based on some modifications for the Benders method for two-stage SMIP with integer recourse. The proposed method uses the idea of RTL and lift-and-project cuts and develops a finitely converging decomposition algorithm. If the original problem has a dual-angular structure, this method finds the convex hull representation of the feasible region and sequentially finds partial descriptions of this convex hull. The Benders cuts generated from these partial convex hulls are valid, and can be viewed as functions of first-stage solutions. Storing these cuts can save time: changing the value of first-stage variables can generate new cuts quickly by using a specialized algorithm.

Sen and Higle (2005) convexify the second stage problem and use a decomposition method to solve the two-stage SMIP with recourse. The key insight is that the proposed Common Cut Coefficients (C^3) theorem states that valid inequalities from one first-stage solution for one scenario can be used to derive valid cut for other first-stage solutions for the same scenario. The proposed Disjunctive Decomposition (D^2) algorithm uses the sequential convexification of the disjunction of the subproblem's constrained region to approximate the subproblems objective value. For each scenario, the algorithm updates the cut generated in the previous iterations by solving an auxiliary LP and adds the cut to the master problem. The whole process can be proved to be finitely converging.

Sen and Sherali (2006) develop extensions of decomposition-based cutting plane methods and apply them to a branch-and-cut algorithm to solve two-stage SMIP with recourse. The method combines the results of Sherali and Fraticelli (2002) and Sen and Higle (2005). The proposed D^2-BAC methods partially solve one MIP subproblem using branch-and-bound, derive special form constraints for all scenarios and aggregate these constraints to generate cuts and update the approximation of the subproblem objective value. This method reuses the disjunctive cuts. But this method requires that all subproblems are feasible when we branch on the subproblem's binary variables, which is not suitable for our case.

There is also extensive research about modeling practical problems as two-stage SMIP problems. Here we present two problems related to our case; their cut generating methods inspired our expand-cut decomposition.

Penuel et al. (2010) model a facility location problem with second-stage activation cost as a two-stage SMIP problem. Whether a station should be installed is the first-stage binary decision variable, whether an installed station should be activated is the second-stage binary decision variable, and the flow amount between facilities are second-stage continuous decision variables. In the decomposition method proposed, given a first-stage solution and a scenario's second-stage solution, the authors add and activate another facility, and use the augmented flow on the current so-called residual network generated by considering the new facility to obtain a valid cut.

Shen and Smith (2013) solve an optimization problem aimed at minimizing the cost of a broadcast domination network on an undirected graph. The first-stage decision variable is selection of arcs. The authors use Benders-like decomposition methods. They develop a cut generation method by adding an additional arc to the given graph. This cut generation, plus cut improving methods based on covering cut bundle, allow the authors to improve the solution time significantly.

To the best of our knowledge, no literature applies the SMIP to the VS network design problem and no problem-specific algorithm has been developed yet.

3.3 Model Formulation

To address the problem discussed in Section 3.1.1, we propose a new objective: design a network such that more demands will have attractive travel time savings. Here if the difference between the travel cost with all candidate stations installed (all-open cost) and travel cost with no station installed (all-close cost) is large, we say such saving is *attractive*. For simplicity, we assume the savings threshold to be considered attractive to be the same for all OD pairs.

3.3.1 Problem Formulation

Parameters:

- G: a directed graph $G = (\mathcal{N}, \mathcal{A})$
- $\mathcal{N}$: set of nodes, including transit stops and candidate stations. Locations are predetermined
- $\tilde{\mathcal{N}}$: $\tilde{\mathcal{N}} \subseteq \mathcal{N}$, set of candidate stations
- $\mathcal{A}$: set of arcs, each $(i,j) \in \mathcal{A}$ incurs a positive cost of c_{ij}
- $\tilde{\mathcal{A}}$: $\tilde{\mathcal{A}} \subseteq \mathcal{A}$, set of bike arcs (b-arc). A b-arc is available only if both end stations are installed
- $\mathcal{H}$: set of all OD-pairs

- h: one OD pair, $h \in \mathcal{H}$
- w^h: the travel demand from the origin $o(h)$ to the destination $d(h)$, $\forall h \in \mathcal{H}$
- K: the number of stations to install
- t^h_{none}: the shortest travel time between OD pair h when no station is installed
- s: threshold, the criterion of time saving that we want OD pairs to meet

Decision variables:

- y_i: binary variable indicating whether a candidate $i \in \tilde{\mathcal{N}}$ is selected in a solution.
- x^h_{ij}: continuous variable indicating flow traveling between OD-pair $h \in \mathcal{H}$ on arc $(i,j) \in \mathcal{A}$.

Auxiliary Variables:

- z^h: binary variable indicating whether an OD-pair $h \in \mathcal{H}$ can meet the criterion. If the OD pair h can achieve a cost saving of at least s unit of time then $z^h = 1$, otherwise $z^h = 0$.

The network design model is formulated as

$$\text{maximize} \quad \sum_{h \in \mathcal{H}} w^h z^h \tag{3.2}$$

subject to

$$\sum_{(i,j)\in\mathcal{A}} x^h_{ij} - \sum_{(j,i)\in\mathcal{A}} x^h_{ji} = \begin{cases} w^h & i = o(h) \\ -w^h & i = d(h) \\ 0 & \text{otherwise} \end{cases} \qquad \forall h \in \mathcal{H},\ \forall i \in \mathcal{N} \tag{3.3}$$

$$x^h_{ij} \le w^h y_i \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.4}$$

$$x^h_{ij} \le w^h y_j \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.5}$$

$$\sum_{(i,j)\in\mathcal{A}} c_{ij} x^h_{ij} + s w^h z^h \le w^h t^h_{none} \qquad \forall h \in \mathcal{H} \tag{3.6}$$

$$\sum_{i \in \tilde{\mathcal{N}}} y_i = K \tag{3.7}$$

$$x_{ij}^h \geq 0 \qquad \forall h \in \mathcal{H},\ \forall (i,j) \in \mathcal{A} \tag{3.8}$$

$$y_i \in \{0,1\} \qquad \forall i \in \tilde{\mathcal{N}} \tag{3.9}$$

$$z^h \in \{0,1\} \qquad \forall h \in \mathcal{H} \tag{3.10}$$

In the objective function Equation (3.2), we assign the demand w^h as the weight for the binary indicator z^h. Thus we maximize the number of demands that can meet the criterion. Equation (3.6) allows z^h to be one only if the OD pair h satisfies the s saving criterion. Given a network design, if the travel time on shortest path for h fails to meet the s saving criterion, adding s units of time will always make the trip travel time longer than t^h_{none}, no matter what route the travelers choose. Constraints for flow conservation (Equation (3.3)), b-arc availability (Equation (3.4) and (3.5)) and the total number of stations to install (Equation (3.7)) are all previously defined in Section 2.2.2. We call this formulation the *max-demand* model.

For the max-demand VS network design MIP formulation presented above, if we view each OD pair $h \in \mathcal{H}$ as a scenario, and the demand w^h as the weight of this scenario, then our formulation has the form of a two-stage stochastic program with recourse. The first stage is to find the location to install the facilities (y_i's), and the second is to route over the network (x^h_{ij}'s). Still we assume the users will use the shortest path between the corresponding OD pair. Whether each OD pair meets the s saving criterion (z^h's) can be determined in the second stage simultaneously with the x^h_{ij}'s. Equation (3.7) corresponds to the constraint of $Ax = b$ in (TSSP). Equation (3.3) - (3.6) correspond to the remaining constraints in TSSP. Note for the constraints the coefficient of the x's are the same for all scenarios (OD pairs) while the coefficient of the z's may be different for all scenarios, as suggested by Equation (3.6). This indicates that in our problem the recourse matrices are scenario specific. As the weight of each scenario is the demand of the corresponding OD pair, the sum of the weights is not 1, and we are not minimizing the expected costs.

3.3.2 Model Decomposition

In the max-demand model we consider whether an OD pair can meet a saving criterion, and thus, it is natural to formulate it in an OD-based way. Note that for each OD pair and each node, there is a flow conservation constraint. For each OD pair and each b-arc, there are two b-arc availability constraints. And for each OD pair, there is a criterion satisfying constraint. Thus the total number of constraints is $|\mathcal{H}| \times (|\mathcal{N}| + 2|\tilde{\mathcal{A}}| + 1) + 1$. The number of constraints will be very large as the network size or the number of OD pairs increases. Note the s saving criterion allows us to eliminate OD pairs that cannot meet such criterion when all stations are installed. This may reduce the problem size. Preliminary computational results show that for even a small scale problem, the insufficient memory issues can result, requiring decomposition methods to be used.

Once we decide which stations are to be installed, the problem can be decomposed into $|\mathcal{H}|$ shortest-path problems. They can be solved by solving $|\mathcal{L}|$ shortest-path-tree problems. Given a network design $y = \bar{y}$ and modifying the problem slightly, the subproblem corresponding to OD-pair $h \in \mathcal{H}$ can be written as:

$$\delta^h(\bar{y}) \quad = \quad \text{maximize } z^h \tag{3.11}$$

subject to

$$\sum_{(i,j)\in\mathcal{A}} x_{ij}^h - \sum_{(j,i)\in\mathcal{A}} x_{ji}^h = \begin{cases} 1 & i = o(h) \\ -1 & i = d(h) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{N} \tag{3.12}$$

$$x_{ij}^h \le \bar{y}_i \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.13}$$

$$x_{ij}^h \le \bar{y}_j \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.14}$$

$$\sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}^h + s z^h \le t_{none}^h \tag{3.15}$$

$$x_{ij}^h \ge 0 \qquad \forall (i,j) \in \mathcal{A} \tag{3.16}$$

$$z^h \in \{0,1\} \qquad \forall h \in \mathcal{H}. \tag{3.17}$$

Let Y_E be the set of all solutions of first stage variables y in previous iterations and θ^h be the second stage objective value of subproblem h. The master problem will be

$$\text{maximize} \quad \sum_{h\in\mathcal{H}} w^h\theta^h \tag{3.18}$$

subject to

$$\sum_{i\in\tilde{\mathcal{N}}} y_i = K \tag{3.19}$$

$$\theta^h \le f(h,\hat{y}) \qquad \forall h\in\mathcal{H}, \forall \hat{y}\in Y_E \tag{3.20}$$

$$0 \le \theta^h \le 1 \qquad \forall h\in\mathcal{H} \tag{3.21}$$

$$y_i \in \{0,1\} \qquad \forall i\in\tilde{\mathcal{N}}. \tag{3.22}$$

Equation (3.20) represents the cuts that we generate from previous iterations, with the right-hand-side depending on the network design and the specific OD pair. Unlike the tree-formulation in the Chiraphadhanakul (2013) problem, Benders decomposition cannot be applied here due to the fact that the z^h's are second-stage binary variables. This fact requires us to find new methods to generate valid cuts for the master problem.

3.3.3 Cut Generation Method

Similar to the idea from Penuel et al. (2010) and Shen and Smith (2013), with given values of the first-stage decision variable $y = \bar{y}$, we want a cut with the generic form of

$$\theta^h \le \delta^h(\bar{y}) + \sum_{i\in\tilde{\mathcal{N}}\setminus I^+(\bar{y})} \alpha_i^h(\bar{y})y_i \qquad \forall h\in\mathcal{H}, \tag{3.23}$$

where $\delta^h(\bar{y})$ is the objective value of the subproblem $h \in H$ for the current first-stage decision variable $\bar{y}$, $\alpha_i^h(\bar{y})$ is the parameter we want, and $I^+(\bar{y})$ is the index set $\{i\in\tilde{\mathcal{N}} \mid \bar{y}_i = 1\}$.

Even though our VS network design problem is a deterministic one, the connection with stochastic programming and the form of our problem inspires us to use algorithms for solving SMIP. In the following sections we propose a method to solve the max-demand formulation of the VS network design problem.

For Satisfied Subproblem: Modified Benders Cut

For any given $\bar{y}$, a subproblem $h \in \mathcal{H}$ that can meet the s saving criterion will has $z^h = 1$, as the subproblem has objective function as $\max z^h$. When $z^h = 1$, the linear programming (LP) relaxation of the subproblem h will still lead to the same result. But if the shortest path between an OD pair h has a travel time shorter than the all-closed travel time but cannot meet the s saving criterion, the integral subproblem will have $z^h = 0$ while the LP relaxation may give a fractional value for z^h .

For an OD pair h that has $z^h = 1$ given the current $\bar{y}$, we formulate the dual of the subproblem's LP relaxation. Let p_i^h, u_{ij}^h, v_{ij}^h and q^h be the dual variables of the subproblem constraints Equation (3.12), (3.13), (3.14) and (3.15) respectively. And let r^h be the dual variables corresponding to the constraints $z^h \leq 1$. The dual formulation of the subproblem's LP relaxation (DSL) is

$$\text{minimize} \quad \left(p_{o(h)}^h - p_{d(h)}^h\right) + \sum_{i \in \tilde{\mathcal{N}}} \left(\sum_{(i,j) \in \tilde{\mathcal{A}}} u_{ij}^h + \sum_{(j,i) \in \tilde{\mathcal{A}}} v_{ji}^h \right) \bar{y}_i + t_{none}^h q^h + r^h \tag{3.24}$$

subject to

$$p_i^h - p_j^h + c_{ij} q^h \geq 0 \qquad \forall (i,j) \in \mathcal{A} \setminus \tilde{\mathcal{A}} \tag{3.25}$$

$$p_i^h - p_j^h + u_{ij}^h + v_{ij}^h + c_{ij} q^h \geq 0 \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.26}$$

$$s q^h + r^h \geq 1 \tag{3.27}$$

$$u_{ij}^h, v_{ij}^h \geq 0 \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.28}$$

$$q^h \geq 0, \ r^h \geq 0. \tag{3.29}$$

For a general MIP problem, to generate Benders cut we should aggregate all the subproblems. As indicated in Section 2.3.1, we can decompose the subproblem of the VS network design problem into $|\mathcal{H}|$ different ones and each corresponds to an OD pair. Here we just consider OD pairs that can meet the s saving criterion with the given $\bar{y}$.

If we solve the dual LP problem directly, we will get an extreme point or an extreme ray of the dual problem polyhedron. In our case, the problem is bounded (the LP relaxation has the constraint $0 \le z^h \le 1$ and the objective function is z^h), the solution will only be an extreme point. The Benders cut of DSL can be written as

$$\theta^h \le \left(p^h_{o(h)} - p^h_{d(h)}\right) + \sum_{i \in \tilde{\mathcal{N}}} \left(\sum_{(i,j) \in \tilde{\mathcal{A}}} u^h_{ij} + \sum_{(j,i) \in \tilde{\mathcal{A}}} v^h_{ji} \right) y_i + t^h_{none} q + r^h. \tag{3.30}$$

As the subproblem h has $z^h = 1$, the objective of the LP relaxation's dual problem will also be 1.

For a MIP problem solved with Benders decomposition, any dual feasible solution for the subproblem can be used to generate a valid cut. Note in our problem we relax the integrality requirement for the MIP subproblems. In such cases, any dual solution can still be used to generate a valid cut for the master problem. Preliminary computational results show that solving the DSL, for any OD pair (using the solver ILOG CPLEX 12.5), will give a solution with all u^h_{ij}'s and v^h_{ij}'s, for each $(i,j) \in \tilde{\mathcal{A}}$, equal to zero and $\delta^h(\bar{y}) = \left(p^h_{o(h)} - p^h_{d(h)}\right) + t^h_{none} q + r^h = 1$. However, such a solution will lead to useless cuts for the master problem. By substituting the values of the variables into Equation (3.30), we get the cut $\theta^h \le 1$. Note that in the master problem, for each $h \in \mathcal{H}$, we have $0 \le \theta_h \le 1$. Namely, the new cut is a repeat of an existing constraint.

A strong cut in our problem is one with a near zero constant part, and only a few positive coefficients of the y_i's. That is, we want a solution to the dual of the subproblem LP relaxation with a small value of $\delta^h(\bar{y}) = \left(p^h_{o(h)} - p^h_{d(h)}\right) + t^h_{none} q + r^h$. And for each $i \in \tilde{\mathcal{N}}$, we also hope the solution will have a small number of positive coefficients $\alpha^h_i = \left(\sum_{(i,j) \in \tilde{\mathcal{A}}} u^h_{ij} + \sum_{(j,i) \in \tilde{\mathcal{A}}} v^h_{ji}\right)$. To achieve this goal, we need to adjust the dual extreme point solution.

To make $\delta^h(\bar{y})$ as close to zero as possible, we add to the DSL the constraint

$$\left(p^h_{o(h)} - p^h_{d(h)}\right) + t^h_{none} q^h + r^h \le \eta, \tag{3.31}$$

where η is a parameter to control the value of $\delta^h(\bar{y})$. Denote the problem as DSL2. The most ideal case is $\delta^h(\bar{y}) = 0$. That is, the dual subproblem has a feasible solution that generates a cut with no constant part. The parameter η can be set at 0 at the beginning. If infeasible, we can increase its value gradually until it becomes feasible.

Once we get a feasible dual solution for DSL with $\eta = \eta^*$, we can adjust the values of the dual decision variables so as to have a fewer number of positive coefficients for the y_i's in the cut. Suppose the objective value of the DSL2 is ϕ. For each $i \in \tilde{\mathcal{N}}$, let $l_i^h \in \{0,1\}$ indicate if the coefficient of y_i, namely $\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^h + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^h$, is zero. We solve the following optimization problem

$$\text{maximize} \qquad \sum_{i\in\tilde{\mathcal{N}}} l_i^h \tag{3.32}$$

subject to

$$p_i^h - p_j^h + c_{ij}q^h \geq 0 \qquad \forall (i,j) \in \mathcal{A} \setminus \tilde{\mathcal{A}} \tag{3.33}$$

$$p_i^h - p_j^h + u_{ij}^h + v_{ij}^h + c_{ij}q^h \geq 0 \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.34}$$

$$sq^h + r^h \geq 1 \tag{3.35}$$

$$\left(p_{o(h)}^h - p_{d(h)}^h\right) + t_{none}^h q^h + r^h = \eta^* \tag{3.36}$$

$$\sum_{i\in\tilde{\mathcal{N}}} \left(\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^h + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^h \right) \bar{y}_i = \phi - \eta^* \tag{3.37}$$

$$\left(\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^h + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^h \right) + M l_i^h \leq M, \qquad \forall i \in \tilde{\mathcal{N}} \tag{3.38}$$

$$l_i^h \in \{0,1\}, \qquad \forall i \in \tilde{\mathcal{N}} \tag{3.39}$$

$$u_{ij}^h, v_{ij}^h \geq 0 \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.40}$$

$$q^h \geq 0, \;\; r^h \geq 0, \tag{3.41}$$

where M is a large positive number. Denote this optimization problem as the DSL-Adjust (DSLA) problem. The objective function Equation (3.32) is to maximize the number of coefficients that can be zero. Equation (3.36) ensures that $\delta^h(\bar{y})$ will have the same value as the DSL problem. Equation (3.37), together with Equation (3.36), serves the function that we will get the value of ϕ if we substitute the solution of DSLA into the objective function of DSL2. Equation (3.38) ensures that for each $i \in \tilde{\mathcal{N}}$, l_i^h can be 1 only when $\left(\sum_{(i,j)\in\tilde{\mathcal{A}}} u_{ij}^h + \sum_{(j,i)\in\tilde{\mathcal{A}}} v_{ji}^h\right) = 0$.

Note that DSLA is an integer program. When the input network size is large, DSLA will have a large number of constraints. And when the problem has a large number of OD pairs, we need to solve a large number of DSLAs. In either case, it will be difficult to generate strong valid cuts for the master problem.

For Unsatisfied Subproblem: Expand Cut

Consider an unsatisfied subproblem whose optimal solution is $z^h = 0$ under a given first-stage solution $\bar{y}$. Subproblem $h \in \mathcal{H}$ cannot meet the s saving criterion. We ask the question: how many more stations do we need to open to ensure that the subproblem $h \in \mathcal{H}$ has the objective value $z^h = 1$. Namely, we want to expand the network and know how many more stations are needed to allow OD pair h to meet the s saving criterion. We use the following optimization problem to find this value:

$$M \quad = \quad \text{minimize} \sum_{i\in\tilde{\mathcal{N}}} y_i - K \tag{3.42}$$

subject to

$$\sum_{(i,j)\in\mathcal{A}} x_{ij} - \sum_{(j,i)\in\mathcal{A}} x_{ji} = \begin{cases} 1 & i = o(h) \\ -1 & i = d(h) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{N} \tag{3.43}$$

$$x_{ij} \le y_i \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.44}$$

$$x_{ij} \le y_j \qquad \forall (i,j) \in \tilde{\mathcal{A}} \tag{3.45}$$

$$\sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} + s \le t_{none}^h \tag{3.46}$$

$$y_i = 1 \qquad \forall i \in I^+(\bar{y}) \tag{3.47}$$

$$x_{ij} \ge 0 \qquad \forall (i,j) \in \mathcal{A} \tag{3.48}$$

$$y_i \in \{0,1\} \qquad \forall i \in \tilde{\mathcal{N}}. \tag{3.49}$$

Denote it as *expand problem* (EP). Equation (3.47) enforces that any station that is selected to be installed in the current first-stage solution $\bar{y}$ will be kept open. If the EP is feasible, we have $M \geq 1$. We can guarantee this problem to be feasible by eliminating OD pairs that cannot meet the s saving criterion even when all $|\tilde{\mathcal{N}}|$ candidate stations are installed. Note that this may lead to a significant reduction in the number of OD pairs.

Another thing to note is that EP is an integer program. Solving it can be time-consuming. Here we propose a method to find the value of M in a gradual way. Let M' be a parameter indicating how many more stations we choose to install. Add the constraint $\sum_{i\in\tilde{\mathcal{N}}} y_i - K = M'$ to EP and change the objective to $\min 0$. Set $M' = 0$ at the beginning, and we increase its value by one if we do not find a feasible solution. The smallest M' for which the modified EP is feasible is the final value of M that we are seeking. The intuition behind this is that by fixing the number of stations to install, we limited the solution space. If most OD pairs just need a small value of M, then the process to find the smallest value of M can be sped up.

Proposition 3. If a first stage solution $y \neq \bar{y}$ can allow OD pair h to meet the s saving criterion, it must have at least M selected stations different from $\bar{y}$.

Proof. Suppose there is a solution $\tilde{y}$ that has $M - k$ $(0 < k < M)$ stations different from $\bar{y}$ and OD-pair h satisfies the s saving criterion. Then there exists a path that uses at most K stations, with at most $M - k$ different from $\bar{y}$, that help h satisfy the criterion. Thus if we open the K stations given by $\bar{y}$ plus the $M - k$ different stations, we can still find a path with h meeting the criterion. Then the solution of the EP will be less than M (since the K stations in $\bar{y}$ plus the $M - k$ additional stations will be feasible, but the objective value is only $M - k$). Hence any first stage solution with OD-pair h satisfying the s saving criterion must have at least M selected stations different from $\bar{y}$. □

Based on Proposition 3, we generate the cut

$$\theta^h \leq \sum_{i\in\tilde{\mathcal{N}}\setminus I^+(\bar{y})} \frac{1}{M} y_i. \tag{3.50}$$

Equation (3.50) is a valid cut. As it is generated from solving the EP, we call it the *expand cut.* For a subproblem h that cannot satisfy the s saving criterion under the current network design $\bar{y}$, the cut indicates that a y will leave $\theta^h = 1$ only when it contains at least M installed stations that are different from those currently in $\bar{y}$.

As indicated earlier in this section, a strong cut in our VS network design problem should have a near zero constant part and only a few positive coefficients of the y_i's. Equation (3.50) has a zero constant part ($\delta^h(\bar{y}) = 0$). Note $|I^+(\bar{y})| = K$, thus there will be $|\tilde{\mathcal{N}}| - K$ terms with non-zero coefficients. If K is much smaller than $|\tilde{\mathcal{N}}|$, Equation (3.50) will have a large number of non-zero coefficient. This will lead to a relatively weak cut. And if for OD pair h and the current $\bar{y}$, the value of M is small, then it is very likely for a solution y to have the right-hand-side of Equation (3.50) larger than one. However, in the master problem θ^h is the second stage objective value and $\theta^h \leq 1$. A right-hand-side of Equation (3.50) larger than one renders the generated expand cut inconsequential.

Changing M will not be a good choice. The value of M is determined by the current solution $\bar{y}$ and the underlying network structure. To have a stronger expand cut, we need to limit the number of coefficients of y_i's that are positive.

Proposition 4. When all candidate stations are installed, for one candidate station $i^* \in \tilde{\mathcal{N}}$, if there is no path that can connect the OD pair h through i^* and satisfy the s saving criterion, then the expand cut Equation (3.50) will still be valid if we remove the term $\frac{1}{M} y_{i^*}$ from the right-hand-side of it.

Proof. When all candidate stations are installed, all b-arcs are available. Under this condition, if there is no path that connects the OD pair h, passes through i^* and meets the s saving criterion for h, any path that can satisfy the s saving criterion will not pass through i^*. Then for h, whether it can satisfy the s saving criterion is not dependent on the installation of i^*. Namely, $z^h = 0$ or $z^h = 1$ is not related to the value of y_{i^*}. To have $z^h = 1$ (in the master problem it is $\theta^h = 1$), we need a first-stage solution with at least M stations from $\left(\tilde{\mathcal{N}} \setminus I^+(\bar{y})\right) \setminus \{i^*\}$ installed. Thus for a $\tilde{y}$ ($\tilde{y} \neq \bar{y}$) with $z^h = 1$, the constraint $\theta^h \leq \sum_{i \in (\tilde{\mathcal{N}} \setminus I^+(\bar{y})) \setminus \{i^*\}} \frac{1}{M} y_i$ will not eliminate the master problem solution $(\theta^h, y) = (1, \tilde{y})$. □

Proposition 4 gives us a method to remove items with non-zero coefficients from the expand cut. Here we propose a stronger version:

$$\theta^h \leq \sum_{i \in \tilde{\mathcal{N}}(h) \setminus I^+(\tilde{y})} \frac{1}{M} y_i, \tag{3.51}$$

where $\tilde{\mathcal{N}}(h)$ is the set of candidate stations that may appear in a path that meets the s saving criterion for OD pair h. By using $\tilde{\mathcal{N}}(h)$ instead of $\tilde{\mathcal{N}}$ for h, we only keep those candidates that may help h to save at least s units of time. If the cardinality of $\tilde{\mathcal{N}}(h)$ is much smaller than the cardinality of $\tilde{\mathcal{N}}$, we can improve the quality of the cut greatly. Note the $\tilde{\mathcal{N}}(h)$ depends on the value of s.

To find $\tilde{\mathcal{N}}(h)$ for each $h \in \mathcal{H}$, start with $\tilde{\mathcal{N}}(h) = \varnothing$. For each $i \in \tilde{\mathcal{N}}$ ($i \neq o(h)$, $i \neq d(h)$), we perform the following steps:

1. Find the shortest path from $o(h)$ to i, and denote it as $path_{i1}$.
2. Temporarily remove nodes in $path_{i1}$ and all arcs coming into or out of these nodes.
3. Find the shortest path from i to $o(d)$, record as $path_{i2}$.
4. If the combination of $path_{i1}$ and $path_{i2}$ forms a path that meets the s saving criterion, then $\tilde{\mathcal{N}}(h) \leftarrow \tilde{\mathcal{N}}(h) \cup \{i\}$.
5. Recover nodes in $path_{i1}$ and all arcs coming into or out of these nodes.

This procedure is summarized in Algorithm 1. For each $h \in \mathcal{H}$, we find $\tilde{\mathcal{N}}(h)$ before executing it. We call the procedure expand-cut decomposition.

One thing to note is that the expand cut can be applied to situations where the criterion for "attractive" is different. For instance, if an OD pair can saving more than 30% of its travel time, we saving such saving is "attractive". For each scenario, it considers which candidate location is helpful. And for each $h \in \mathcal{H}$, $\tilde{\mathcal{N}}(h)$ can be found based on the problem-specific criterion before the expand-cut decomposition is executed.

Algorithm 1 Expand-cut Decomposition

```
1: Initialize $BestLB \leftarrow 0$
2: while remaining time > 0 do
3:     Solve the master problem, get the objective value $Obj$ and solution $y = \bar{y}$. Solve all subproblems using shortest-path-tree method. Evaluate the travel cost for each $h \in \mathcal{H}$. Get $LB(\bar{y})$, the amount of demand satisfying the $s$ saving criterion
4:     For each $h \in \mathcal{H}$ that meets the $s$ saving criterion, generate Benders cut for the LP relaxation of the subproblem, modify it and add them to the master problem
5:     For each $h \in \mathcal{H}$ that fails to meet the $s$ saving criterion, use $\bar{y}$ and $\tilde{\mathcal{N}}(h)$ to generate cut and add it to the master problem
6:     if $Obj - BestLB < 10^{-6}$ then
7:         Break
8:     else if $LB(\bar{y}) > BestLB$ then
9:         $BestLB \leftarrow LB(\bar{y})$
10:        $y^* \leftarrow \bar{y}$
11:    end if
12: end while
13: return $y^*$
```

3.4 Conclusions

In this chapter we proposed an optimization model (max-demand model) that maximizes demand that meets certain travel time savings criteria. We develop because the existing min-cost model (Chiraphadhanakul, 2013) minimizes the total travel cost of all demand and overemphasizes the OD pairs with small savings and large demands. With a predetermined value of the parameter s, we can delete the OD pairs that cannot save at least s minutes when all candidate stations are installed. This can reduce the input size to a great extent.

We formulate the max-demand model as a two-stage stochastic program. We develop a decomposition algorithm and the corresponding method to generate special cuts to solve the model. Two types of cuts are used: for subproblems that meet the s saving criterion under the current first-stage network design, a modified Benders cut is generated; and for subproblems that fail to meet the criterion, a special expand cut is generated. To generate a valid cut, we solve different optimization models for the two types of subproblems. To generate an expand cut, we need to find the candidate set that may help an OD pair meet the s saving criterion before the decomposition algorithm starts.

Chapter 4

Computational Results of the Max-Demand Model

4.1 Introduction

In this chapter, we present the computational results for solving the max-demand model of the VS network design system. The VS system is based on Hubway, a bike sharing system in Boston. The transit system is based on the Massachusetts Bay Transportation Authority (MBTA). We first explain briefly how the input networks are generated. Then we show that our decomposition algorithm can reduce the solution time of the max-demand formulation. At last, we compare the solutions of the min-cost model and the max-demand model under the same network input. The results show that the max-demand model is able to increase the demand that meets a certain s saving value without increasing the total travel cost significantly.

All algorithms and models are implemented in Java 1.6 under the IDE Eclipse Kepler Service Release 1 and all optimization models are solved by the IBM ILOG CPLEX 12.5 solver. We run all computational experiments on a MacBook Pro with 2.6 GHz Intel Core i7 CPU and 1 GB of RAM allocated to the programs.

4.2 Input Data

The input networks are based on the MBTA General Transit Feed Specification (GTFS) data[1] and the publicly accessible Hubway location data[2]. We use existing Hubway stations as our candidate stations. In our problem, we consider a situation where Hubway decides to open fewer stations, or where a new agency wants to start a bike sharing business from scratch with preselected station candidates. In this chapter our goal is to maximize demand of OD pairs whose travel time savings are attractive, that is, significant enough to affect changes in travel pattern and mode choice.

When generating the network, we use the idea of a transfer tree from Section 3.5.2 of Chiraphadhanakul (2013). The basic idea of a transfer tree is to find the shortest paths from an origin node to all other transit stops that can be reached on the given public transportation network and VS network. In a transfer tree, every node other than the origin is a transfer node: a transit stop where people need to make a transfer for the bus/subway/trolley/ferry service along the path of the OD pair. The arc between two nodes on the transfer tree is the shortest path between them. For each origin, we generate one transfer tree. The trees will share many transfer nodes. B-arcs, i.e. arcs connecting two bike candidate stations, are also added to the network. Together they form a network.

Bike-sharing may be used as the first or last segment of a trip in a city. Hence as in Chiraphadhanakul (2013), we assume that two trips using the same path but having opposite directions will have the same benefit on a given network. This assumption simplifies the problem so we can reduce the number of OD pairs. For travel demand, we consider the possible trips originating from a bike candidate station to a transit node, and aggregate the demand between this OD pair based on the transit points that can be reached if a trip goes beyond this destination transit node. We use the open API of Mapquest[3] to find the biking time between any pair of bike candidate stations and add the b-arcs that take less than a specified time to the network we

[1]`http://www.mbta.com/rider_tools/developers/default.asp?id=21895`
[2]`http://hubwaydatachallenge.org`
[3]`http://open.mapquestapi.com/directions`

generate. We set the biking time threshold to 30 minutes. The current Hubway maximum bike usage time is also 30 minutes.

4.3 The Efficiency of Decomposition

First, we run experiments to evaluate the performance of the proposed expand-cut decomposition method. We test the algorithm on a network with $|\mathcal{N}| = 597$, $|\tilde{\mathcal{N}}| = 19$, $|\mathcal{A}| = 2024$ and $|\tilde{\mathcal{A}}| = 312$. We set $K = 9$, $s = 25$ min, and compare the performance of the expand-cut decomposition with the performance of solving the model directly in ILOG CPLEX. For the decomposition method, in one experiment we use both modified Benders cut and expand cut and in another only expand cut is used. To study the efficiency of the decomposition algorithm, we solve the max-demand model with different numbers of OD pairs. The OD pairs are uniformly chosen from those with $s = 25$ minutes saving potential. The results are summarized in Table 4.1.

Table 4.1: Solving Power of CPLEX and Expand Decomposition

$\|\mathcal{H}\|$	CPLEX		Benders + Expand Cut			Expand Cut		
	Time(sec)	Obj	Time(sec)	Iterate	Obj	Time(sec)	Iterate	Obj
10	3.3	109	75.9	120	109	9.5	156	109
50	73.0	918	46.3	18	918	5.5	18	918
100	243.7	3712	186.5	48	3712	45.1	59	3712
150	788.4	4238	401.1	63	4238	65.0	68	4238
200	1066.3	5844	278.9	32	5844	44.8	36	5844
250	20 min	6702*	484.4	43	7040	59.4	43	7040

* the value of objective is not optimal.

In this table, we present the running time (Time), number of iterations (Iterate, if applicable) and objective value (Obj). These results show that the run time of the CPLEX solver rises dramatically as the number of OD pairs increases. As the number of OD pairs has a linear relationship to the number of constraints, the run time is superlinear compared to the number of constraints. This indicates that when the number of OD pairs is large, solving the model directly becomes slow. When $|\mathcal{H}| = 250$, the CPLEX solver cannot solve the problem within a 20 minute time limit, while both decomposition methods successfully find the optimal solution within a relatively short time.

Our decomposition procedure solves all problems within the 20 minute time limit. One thing to note is that the run time of using both modified Benders cut and expand cut is longer than using only expand cut. Further analysis of the computation process shows that most of the solution time is spent on cut generation in these cases. As we pointed out in Section 3.3.3, generating a relatively strong Benders cut for the LP relaxation of a satisfied subproblem requires solution of an integer program. It will take a long time if a large number of subproblems meet the s saving criterion. If we only generate expand cuts and not modified Benders cuts, only subproblems that fail to satisfy the s saving criterion will be solved by the decomposition algorithm. When we use both types of cuts, the two decomposition procedures have a similar number of iterations. These results indicate that the modified Benders cuts are still not very strong. According to this insight, we use only expand cut for the decomposition method in the following computational experiments.

Another thing to note is that both the run time and the number of iterations of our decomposition method are not strictly non-decreasing with the number of OD pairs. This is because when we have fewer OD pairs, the number of expand cuts that can be generated in each iteration is also small. Thus even though solving each iteration becomes faster, more iterations may be needed.

4.4 The Quality of Max-Demand Solutions

In this section we compare the best solutions we have from both the min-cost and max-demand models. We first investigate how much saving each OD pair realizes under the two solutions respectively. Then we exam two secondary attributes: achieved saving potential of each OD pair and number of transfers per OD pair.

4.4.1 Demand Meeting the s Saving Criterion

In Section 3.1.1, we pointed out that the min-cost solution may overemphasize OD pairs that have small savings and large demand. To examine to what extent the max-demand model can resolve the drawback of the min-cost model, we solve both min-cost and max-demand models on the same network, for different numbers of open stations (K) and different saving criteria (s). For different values of K and s, we calculate the demand that satisfy the saving criteria. The network we present here has $|\mathcal{N}| = 704$, $|\mathcal{A}| = 3008$, $|\tilde{\mathcal{N}}|=30$, $|\tilde{\mathcal{A}}| = 742$, $|\mathcal{L}| = 30$, $|\mathcal{H}| = 5103$ and the total demand is 248192.

Table 4.2: Number of OD Pairs with Potential Saving of s Minute, $K = 15$

s (min)	0	> 0	15	20	25	30	35	40	45	50
Number of OD Pairs	5103	3982	2095	1431	976	637	346	212	93	35
Demand	248192	179665	92715	57995	37869	24448	13632	6955	6955	1390

For the cases of $K = 15$, we test values for $s = 15, 20, \ldots, 50$ minutes. The number of OD pairs with potential to save at least s minutes when all candidate stations are installed, and the corresponding amount of demand, are presented in Table 4.2. Note that the larger the value of s, the smaller the number of OD pairs and smaller the demand that meets the saving criterion. As we mentioned in Section 3.3.3, we only need to take the OD pairs that have the potential to meet the s saving criterion into account. Thus solving a problem with a large s value is easier than with a small s value.

Table 4.3: Number of Demands Meeting s Saving Criterion, $K = 15$

Savings (min)	Min Cost	Value of s (min)							
		50	45	40	35	30	25	20	15
≥ 50	968	1390	1388	632	1207	1217	1387	1106	968
≥ 45	2596	1645	3271	3024	2781	2871	1790	2761	3047
≥ 40	4235	5802	6311	6905	5688	5711	5627	4952	5353
≥ 35	8189	8661	9810	9878	12849	11448	9703	11755	9769
≥ 30	17277	18070	20777	19443	22100	22848	19414	21999	20721
≥ 25	24524	25036	28132	27323	30922	31711	34229	30549	28992
≥ 20	35084	30446	37777	42500	43537	44595	41494	46524	38046
≥ 15	67497	55604	63607	61827	59313	64140	61234	61777	76457

In Table 4.3, for different values of s, we calculate the demand that realizes a s minute savings under the best min-cost solution with $K = 15$, and calculate the same for the best max-demand solution with $K = 15$ and different values of s. For instance, in the column with $s = 20$ minutes and the row with *savings* ≥ 25 minutes, the 30,549 means that 30,549 of the demand realizes at least a 25 minute savings among all 5103

OD pairs in the network design given by the best solution of the max-demand model of $K = 15$ and $s = 20$ minutes. The cells with red background contain the number of demands that satisfies the target s saving criteria.

Table 4.4: Improvement of Number of Demands Satisfying s Saving Criterion, $K = 15$

	Value of s (min)							
Saving (min)	50	45	40	35	30	25	20	15
≥ 50	43.6%	43.4%	-34.7%	24.7%	25.7%	43.3%	14.3%	0.0%
≥ 45	-36.6%	26.0%	16.5%	7.1%	10.6%	-31.1%	6.4%	17.4%
≥ 40	37.0%	49.0%	63.1%	34.3%	34.9%	32.9%	16.9%	26.4%
≥ 35	5.8%	19.8%	20.6%	56.9%	39.8%	18.5%	43.6%	19.3%
≥ 30	4.6%	20.3%	12.5%	27.9%	32.3%	12.4%	27.3%	19.9%
≥ 25	2.1%	14.7%	11.4%	26.1%	29.3%	39.6%	24.6%	18.2%
≥ 20	-13.2%	7.7%	21.1%	24.1%	27.1%	18.3%	32.6%	8.4%
≥ 15	-17.6%	-5.8%	-8.4%	-12.1%	-5.0%	-9.3%	-8.5%	13.3%

In Table 4.4, we calculate the improvement in the number of demands that satisfy the s saving criterion with different s values. This comparison is between the best min-cost model solution and the best max-demand model parameterized with different s values. For instance, in the column with $s = 20$ minutes and the row with $savings \geq 25$ min, the 24.57% means that when we compare the solution of the max-demand model with $s = 25$ minutes and the best min-cost model solution, the demand that meets a 25 minute savings increased by $(30549 - 24524)/24524 = 24.57\%$. From this table, for reasonable values of s, we can see that there is a quite large improvement in the number of demands that satisfy the s saving criterion. Similar to Table 4.3, the cells with red background contain the percentage of demand improvement corresponding to the target value of s. When the parameter s is set to be one value, we can see that the demand that satisfies another saving criterion might increase. For example, in Table 4.4, when $s = 15$ minutes, the demand that satisfied the savings criteria of $20, 25, \ldots, 45$ all increased. However, such increases may not exist when s is large. In this case with $K = 15$, the demand that saves more than 15 minutes with the max-demand solution is less than the demand that meets the same criterion with the min-cost solution when $s = 20, 25, \ldots, 50$.

For the same input network, computational results with $K = 12$, 18 and 21 are presented in Appendix A. When $K = 12$, solving the min-cost model is very difficult, and the gap between the Benders lower and upper bound is still 12.1% after five hours of solution time. Solving the max-demand model is also difficult when s is small (namely, the number of OD pairs is large). In this case, the difference of the numbers of demands satisfying the s saving criterion between the two models is large. For the min-cost model with $K = 21$, the gap between the upper and low bound is 0.0029%. This shows that the best solution we have is very close to the optimal one. When K is large, the saving criterion is easier to be satisfied for all OD pairs. This explains that the difference between the numbers of demands satisfying the s saving criterion between the two models in general is smaller than for the cases with smaller K.

4.4.2 Total Travel Cost

The total travel time, or costs, of all demand on the network are summarized in Table 4.5 for the solutions obtained in Section 4.4.1. In the table, the row of *total cost* is the sum of travel time of all demand over all OD pairs, and the row of *increase* is the increase of total travel cost of a max-demand solution compared with the total travel cost of the min-cost solution. Compared with the min-cost solution, the total travel cost for all demand does not vary much if we use the max-demand solution when s is not large. This indicates that using the max-demand solution does not have much negative impact on the objective of the min-cost model.

Table 4.5: Comparison of Total Travel Cost with $K = 15$

Objective	Min-Cost	Max-Demand (for Different Value of s)							
		15	20	25	30	35	40	45	50
Total Cost ($\times 10^6$ min)	9.42	9.39	9.47	9.57	9.46	9.46	9.52	9.59	9.89
Increase	0.00%	-0.27%	0.47%	1.63%	0.44%	0.53%	1.09%	1.81%	4.97%

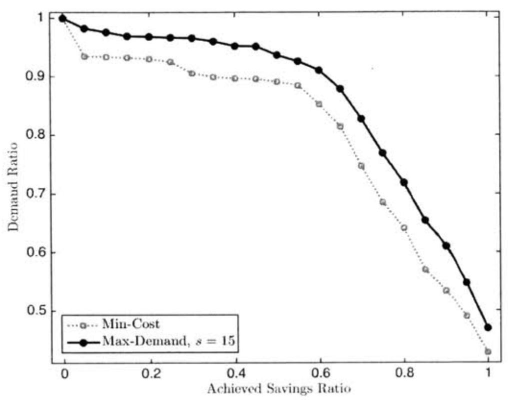
4.4.3 Other Performance Metrics

In addition to the number of demands that satisfy the savings criterion, we investigate other secondary criteria to thoroughly compare the two models. The two metrics we investigate here are the achieved savings ratio and the number of transfer arcs. Their definitions are in the following sections.

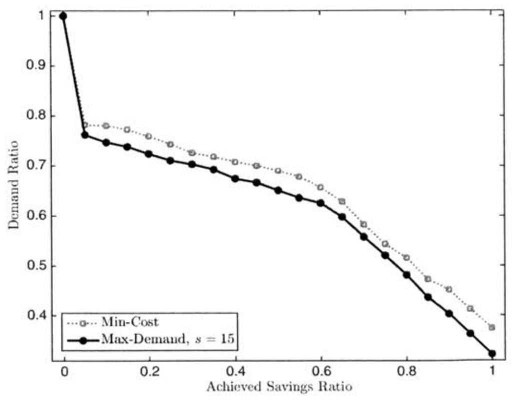
Achieved Savings Ratio

For each $h \in \mathcal{H}$, let s_{all}^{h} be the all-open time savings of OD pair h (all-closed shortest travel time of h minus the all-open shortest travel time of h) and $s^{h}(\bar{y})$ be the largest possible saving time of h given a first-stage solution $\bar{y}$ (all-closed shortest travel time of h minus the shortest travel time given the network design $\bar{y}$). Define $s^{h}(\bar{y})/s_{all}^{h}$, the actual time savings divided by the potential maximum time savings, as the *achieved savings ratio* of h, given $\bar{y}$. For the same network in Section 4.4.1 and with $K = 15$, we compare the two types of solutions in terms of the achieved savings ratio for each OD pair.

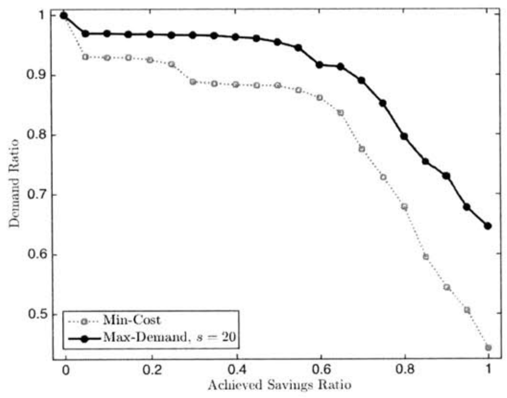
The comparison results are plotted in Figure 4-1a to 4-1p. The horizontal axis is the achieved savings ratio, and the vertical axis is the demand ratio. Let D^{s} be the demand that has s savings potential. Let $D_{a}^{s}(\bar{y})$ be the the number of demands that have s savings potential and also has at least $a\%$ of the achieved savings ratio under $\bar{y}$. Also let D^{+} be the total demand that has positive savings potential. Let $D_{a}^{+}(\bar{y})$ be the demand that has savings potential greater than 0 and also has at least $a\%$ of the achieved savings ratio under $\bar{y}$. Then in the left figures, the vertical axis is $D_{a}^{s}(\bar{y})/D^{s}$, and in the right figures the vertical axis is $D_{a}^{+}(\bar{y})/D^{+}$. The black solid line represents the result from the max-demand solution parameterized with s and the gray dashed line represents the result from the min-cost model. Here the figures on the left count only demand with potential to save at least s minutes, and the figures on the right count all demand that have positive all-open savings. For example, on the red line of Figure 4-1a the point $(0.8, 0.72)$ on the black line means that about 72% of the demand that has the potential to save at least 15 minutes achieves at least 80% of the saving ratio. And on the gray line of Figure 4-1a the point $(0.8, 0.64)$ on the gray line means that about 64% of the demand that has positive all-open saving potential achieves at least 80% of the savings ratio.



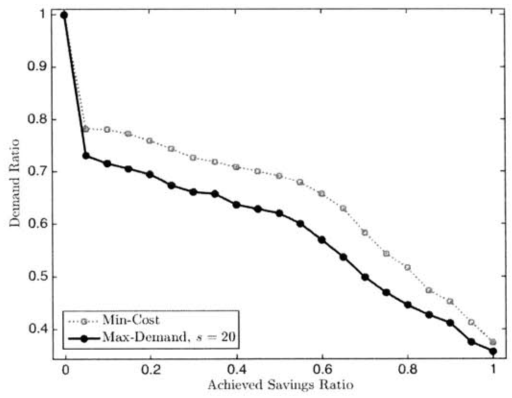
(a) Demand Having s min Savings Potential



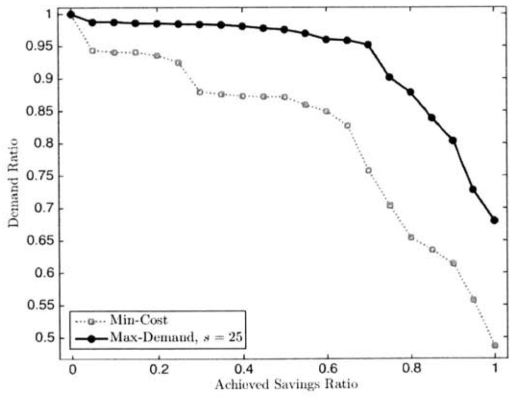
(b) Demand Having Positive All-Open Savings



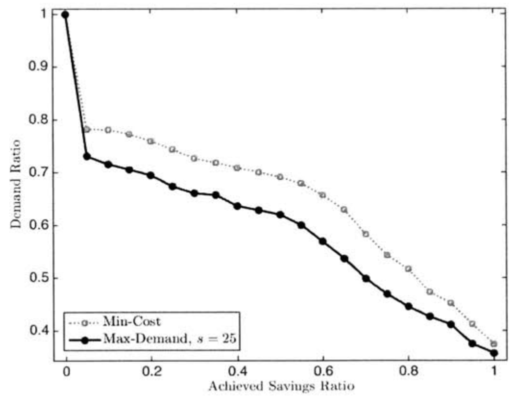
(c) Demand Having s min Savings Potential



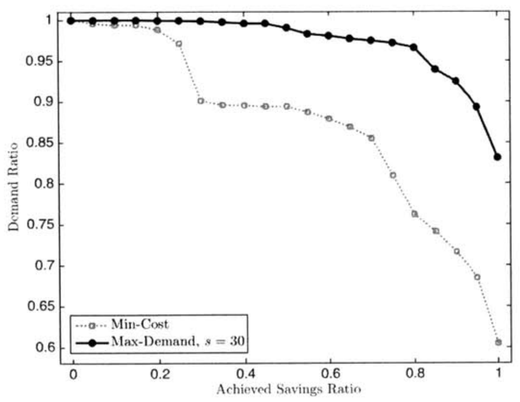
(d) Demand Having Positive All-Open Savings



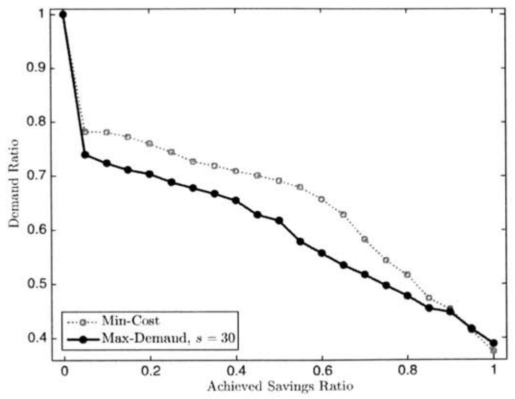
(e) Demand Having s min Savings Potential



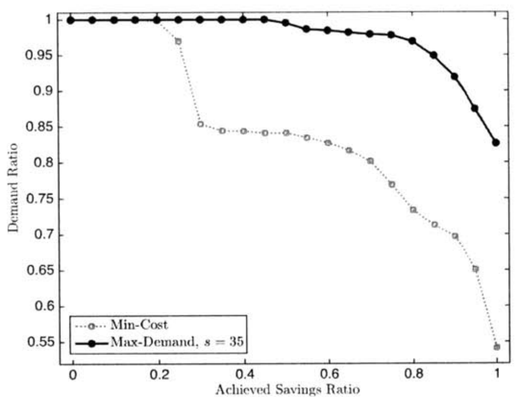
(f) Demand Having Positive All-Open Savings



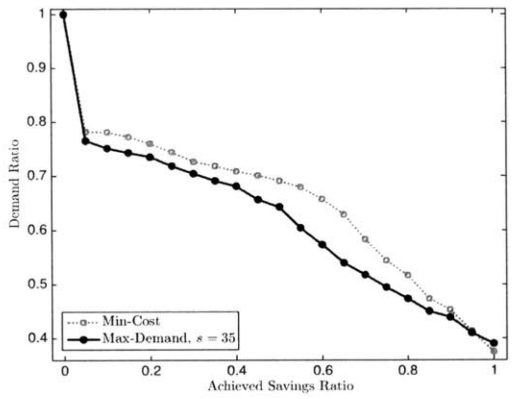
(g) Demand Having s min Savings Potential



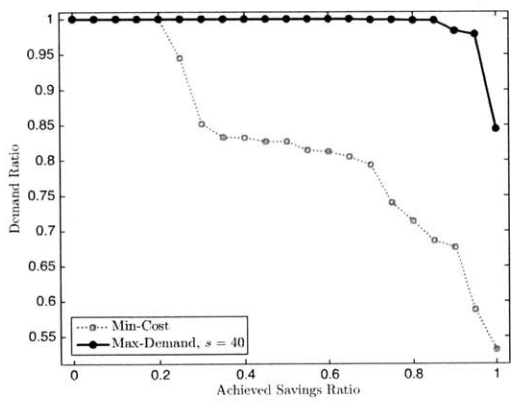
(h) Demand Having Positive All-Open Savings



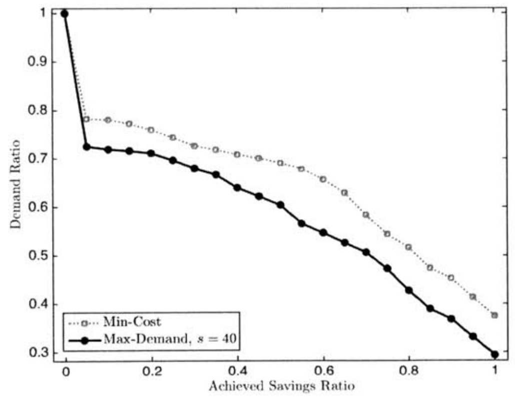
(i) Demand Having s min Savings Potential



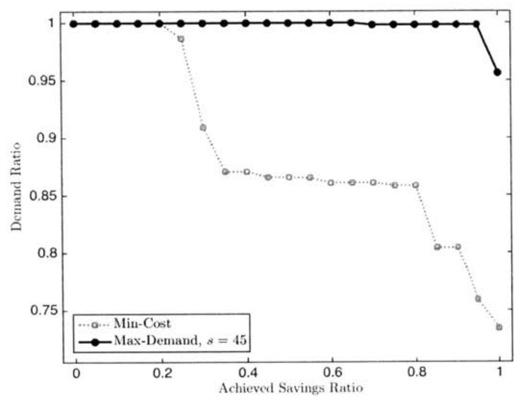
(j) Demand Having Positive All-Open Savings



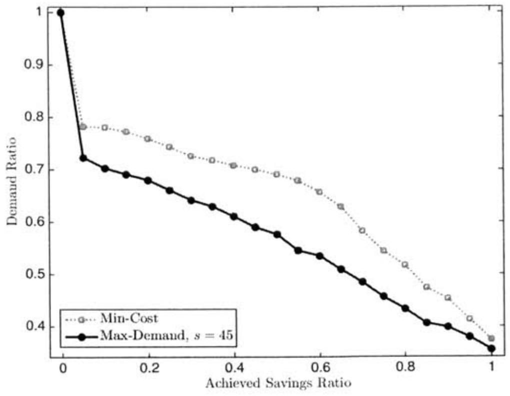
(k) Demand Having s min Savings Potential



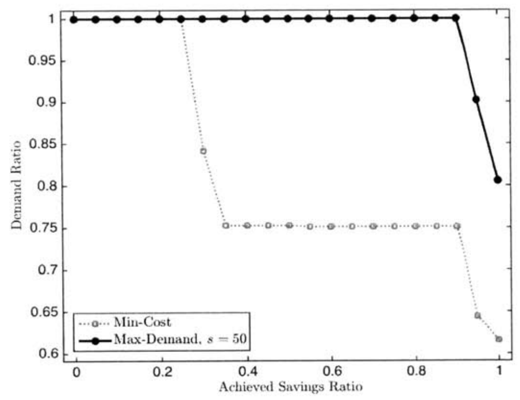
(l) Demand Having Positive All-Open Savings



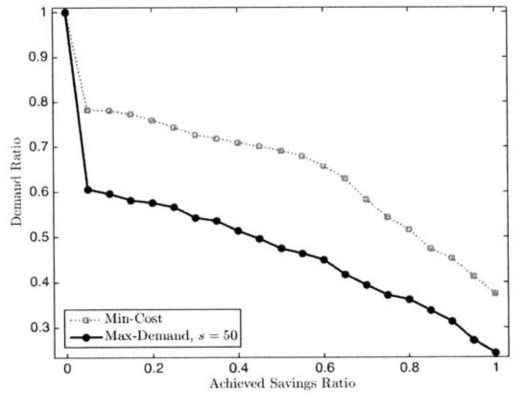
(m) Demand Having s min Savings Potential



(n) Demand Having Positive All-Open Savings



(o) Demand Having s min Savings Potential



(p) Demand Having Positive All-Open Savings

Figure 4-1: Percentage of Demand that Achieved Savings Ratio

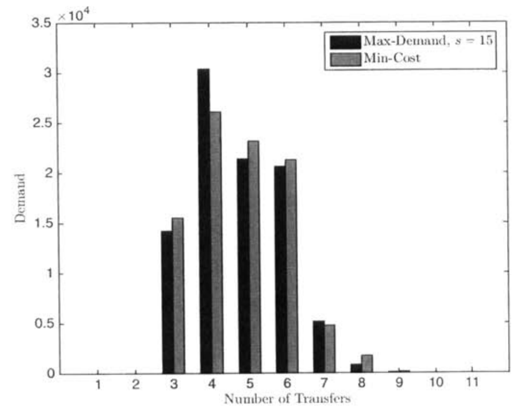
At any given achieved ratio (horizontal axis), the higher the curve is, the more demands that satisfy the savings ratio. The max-demand solution curve is higher than the min-cost one in all figures on the left side. This indicates that for the targeted demands that have at least s minutes savings potential, the max-demand solution better achieves the savings potential of the demand compared to the min-cost solution. It also indicates that the max-demand model will give a design that concentrates more on the benefit of that targeted demand, as we expect. Another thing to note is that the gaps between the two lines in the left graphs tend to increase as s becomes large. This is because the demand with large savings potential is small compared to the entire demand in this network. And the performance of the min-cost model is consistent with the analysis at the beginning of this chapter: it may ignore small demand OD pairs.

The figures on the right side compare the solutions' effects on all OD pairs that have a positive all-open savings potential. The higher the curve at a given achieved ratio, the more demand that achieves that savings ratio. Note that s is a parameter of the max-demand model, so the curve for the min-cost model does not change with the value of s. We can see that in 6 out of the 8 figures (except for $s = 30$ and 35), the min-cost model curve is higher than the max-demand model curve even for large achieved savings ratio, but the gap is not very large (at least not larger than their counterparts on the left) except for the case where $s = 50$. This is understandable because there are many demands in the network ignored by the max-demand model. With larger s, more demand is ignored. And this explains why the gap in Figure 4-1p ($s = 50$) is larger.

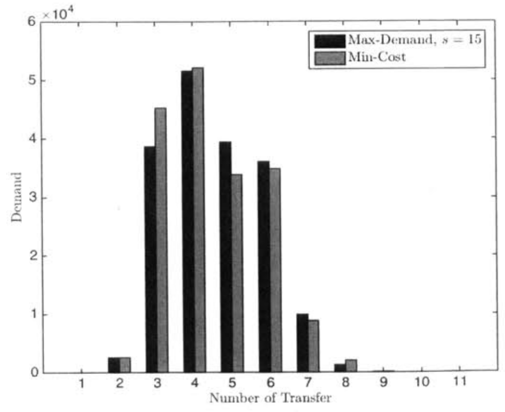
Figure 4-1a to 4-1p compare the solutions of the two models in terms of the savings potential of each OD pair. In general, the max-demand model is more effective in realizing savings potential than the min-cost model. The larger the value of s, the more obvious is this effect. The min-cost model, however, does better for demands with positive all-open saving, but we should note that there are many large-demand OD pairs that do not have large time savings even when all candidate stations are installed.

Number of Transfer Arcs

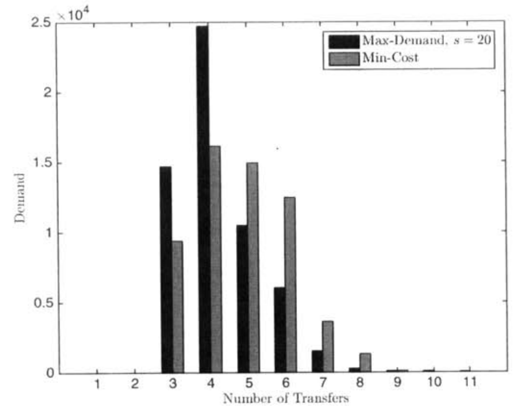
Another metric of interest is the number of transfers each OD pair requires. Within the transfer tree network, each node is a transfer node and each arc is a bike or transit trip. Thus, the number of arcs between one OD pair, given a network design, is the number of bike/transit trips we need on the path. We refer to call such arc as a transfer arc. Again, we assume the shortest path on a given network will always be used.



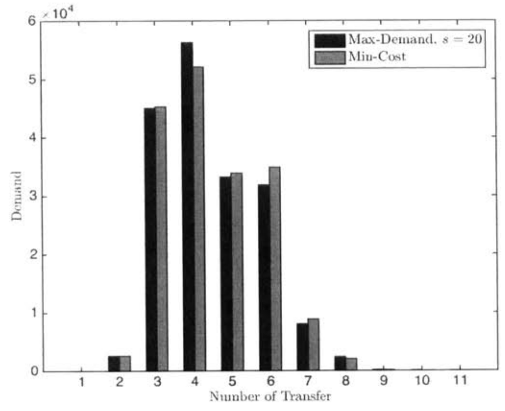
(a) Demand Having s min Savings Potential



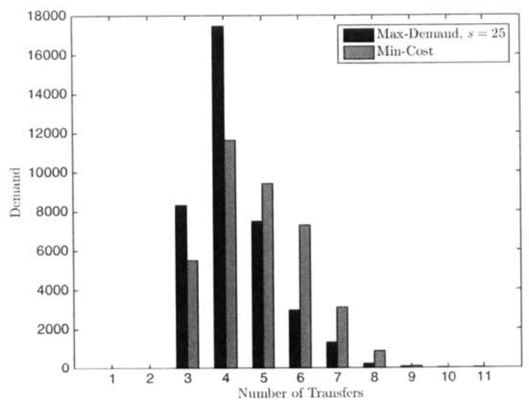
(b) Demand Having Positive All-Open Savings



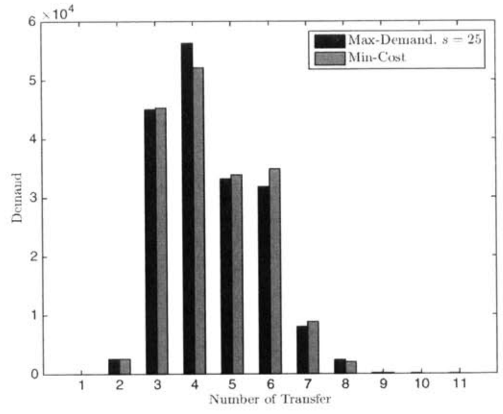
(c) Demand Having s min Savings Potential



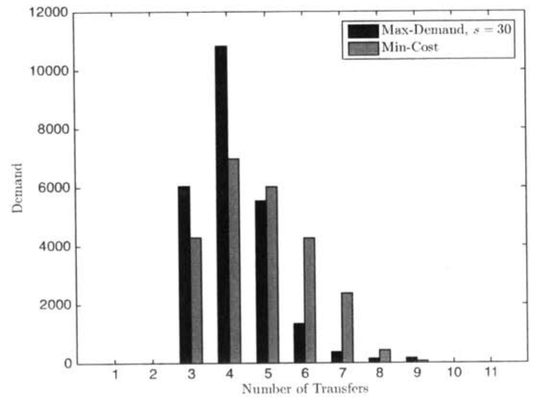
(d) Demand Having Positive All-Open Savings



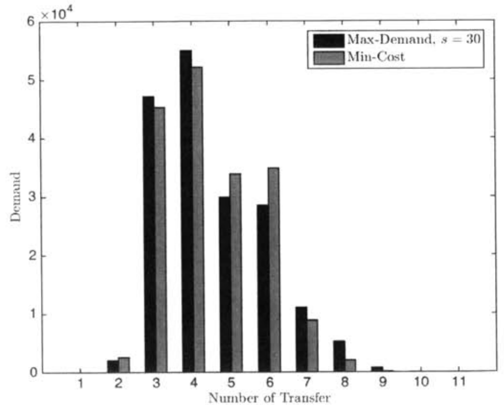
(e) Demand Having s min Savings Potential



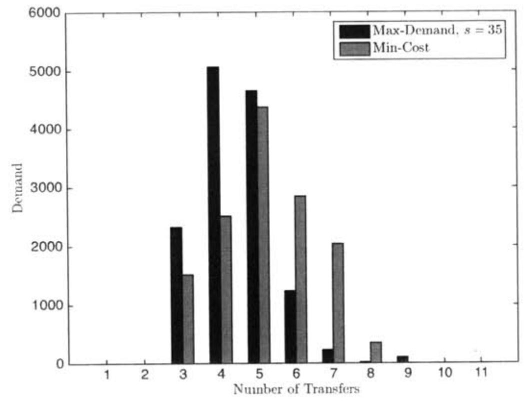
(f) Demand Having Positive All-Open Savings



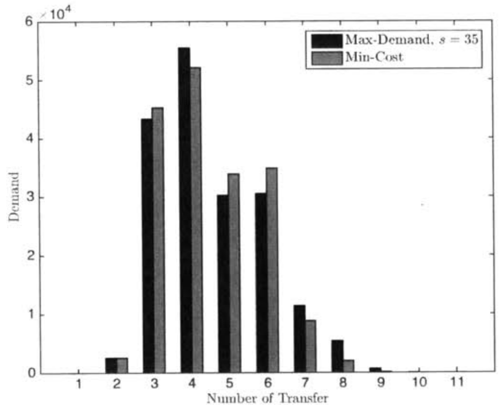
(g) Demand Having s min Savings Potential



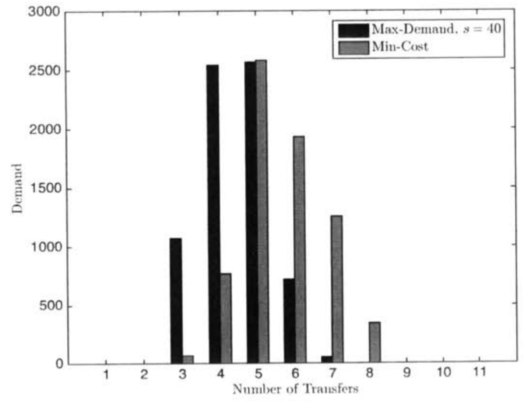
(h) Demand Having Positive All-Open Savings



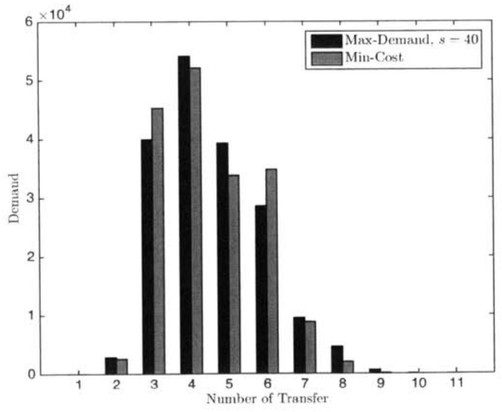
(i) Demand Having s min Savings Potential



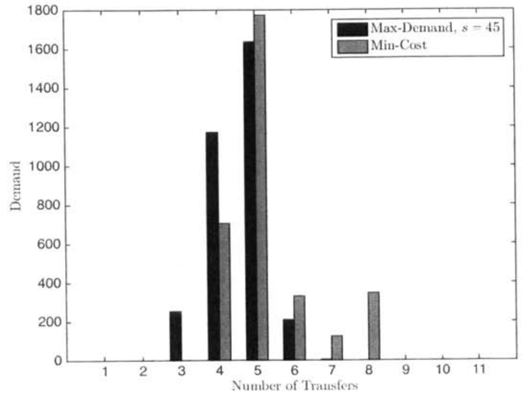
(j) Demand Having Positive All-Open Savings



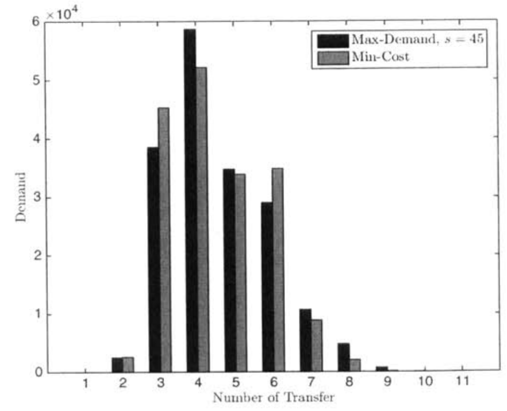
(k) Demand Having s min Savings Potential



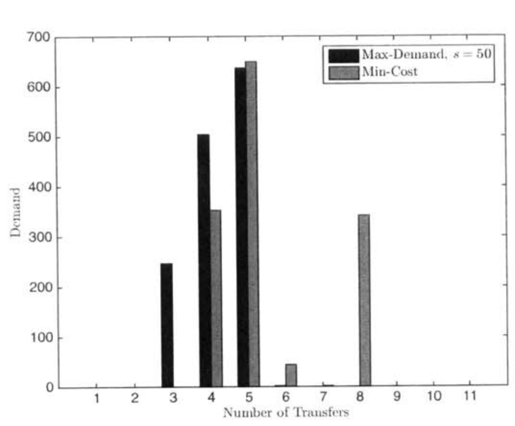
(l) Demand Having Positive All-Open Savings



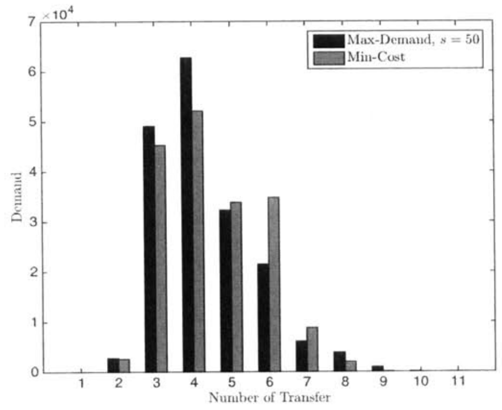
(m) Demand Having s min Savings Potential



(n) Demand Having Positive All-Open Savings



(o) Demand Having s min Savings Potential



(p) Demand Having Positive All-Open Savings

Figure 4-2: Count of Demand that Uses Transfer Arcs

From the Figure 4-2a to 4-2p, we show the relationship between the number of transfer arcs and the demand. We count the demand that uses $1, 2, \ldots$ arcs on the shortest path for each OD pair. Note no demand, or no OD pair, uses only one arc on its shortest path. This is because in the network generating process described in the Chapter 3 of Chiraphadhanakul (2013), only bike candidate stations serve as origins and no OD pair has transit stops within walking distance as a destination node.

For the solutions of the two type of models, the figures on the left side only count the demand that has s-minute savings potential. We can see the max-demand solution helps more targeted demand use fewer arcs. This effect is more obvious when

s is large. On the right side, the figures count the demand having positive all-open savings. The min-cost model is not influenced by the value of s, so the demand count for any number of arcs for the min-cost solution in all the graphs on the right are the same. Considering all such demand, the solutions from the two types of models seem not to make much difference in the number of arcs on the shortest path for each OD pair. Just as the figures on the left show that the targeted demand in general contains fewer arcs in their shortest paths, non-targeted demand generally has fewer arcs in the min-cost solution.

In Figure 4-3, we plot the cumulative number of demands (with positive all-open savings) that have a specified number of transfers. All the curves are close to each other except for the one corresponding to the max-demand model with $s = 50$. This indicate that the max-demand model generally does not lead to solutions with more transfer arcs.

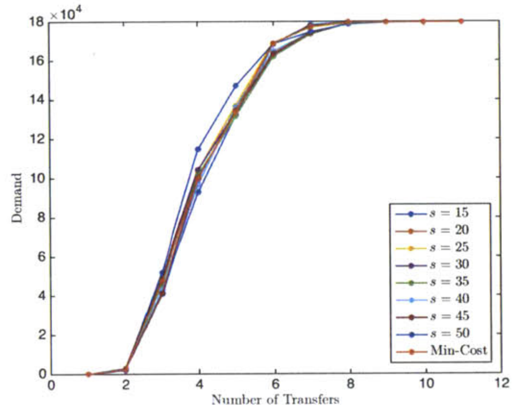


Figure 4-3: Cumulative Number of Demands Having a Specified Number of Transfers

4.5 Conclusions

The computational results in this chapter show that our decomposition method is effective in reducing the run time for the max-demand model. On one hand, using the modified Benders cuts can help reduce the number of iterations. On the other hand, these reductions are achieved at the expense of taking a longer time to generate them.

Using only expand cut is effective in solving the max-demand model quickly. OD-wise analysis of the achieved savings shows that the max-demand solution increases the demand that meets an s savings criterion. The improvement is large when the number of stations to install is not large. For the max-demand solution, the targeted demands also have large relative time savings and fewer transfers. However, when s is large, the number of targeted demands is small. The max-demand model only considers such demands that have the potential to save at least s minutes. Thus when we consider all demands having positive savings potential, the achieved savings ratios are smaller under the min-cost solution that under the max-demand solution. This occurs because we only focus on the targeted demand and ignore the rest.

For future work, other formulations may be considered. For example, each OD pair can have its own savings criterion. For the expand cut decomposition converges, we know the solution must be optimal for expand cuts to be valid for the original problem. Because convergence is not known to be guaranteed, another area for future research is to prove convergence or generate some other cuts to ensure the convergence of the process.

Chapter 5

Enhancing Model Scalability

5.1 Introduction and Motivation

In this chapter, we propose a heuristic-based decomposition method to solve the min-cost model of the VS network design problem. This method is aimed at finding a feasible, quality network design within a reasonable time. Experiments reveal that tractability issues result using the decomposition methods proposed in previous chapters when we attempt to solve large-scale network design problems.

The entire network generated from the MBTA GTFS and Hubway location data has $|\mathcal{N}| = 1234$, $|\tilde{\mathcal{N}}| = 131$, $|\mathcal{A}| = 21123$, $|\tilde{\mathcal{A}}| = 14816$ and $|\mathcal{H}| = 22819$. The computational experiments in the previous chapter use only part of the entire network as input. We solve the min-cost model for $K = 94$ over the entire network. We use the tree-based model with the Bender-like decomposition method presented in Section 2.3.1. We generate PO cuts and set $pass = 35$ when improving each cut. The time spent in each iteration during the solution process is recorded and plotted in Figure 5-1. The run time limit is 10 hours and only 19 iterations are executed. We can see that the solution time increases dramatically after the first several iterations. Detailed analysis shows that the time increase is due to solving the master problem. This is understandable as we mentioned in Section 2.3.2: the master problem is a MIP and the number of constraints keeps increasing. These results indicate that we need to go through more iterations to find a good solution.

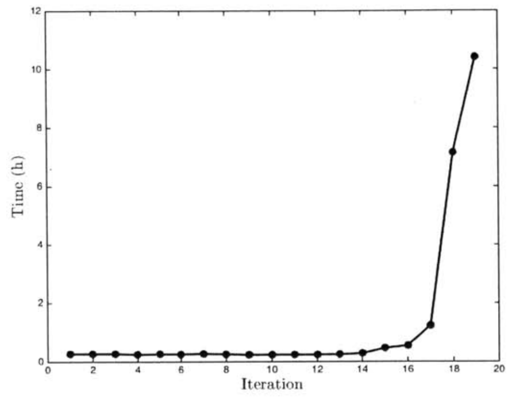


Figure 5-1: Min-Cost Model Iteration: Solution Time for the Entire Network

In the following sections of this chapter, we present a heuristic that can find a good network design solution in a very efficient way. Then we present a decomposition method based on a variable neighborhood search method. Finally, we present the computational results of these models for several randomly generated networks and a real-world, large-scale network.

5.2 Algorithm

The heuristic to find a good feasible network design is based on an estimate of how much savings each candidate station can bring to the entire network. We rank the stations according to that estimate. Finally we use rank to control the installation of each candidate station in order to limit our search space for the master problem. Another way to limit the search space of the master problem is variable neighborhood search. Using these two types of constraints in the decomposition procedure, we can find additional feasible solutions in the search space around the heuristic result, thus increasing the chance to find better solutions within specified time limits.

5.2.1 Low-High Savings Constraints

To estimate the potential savings of each candidate station, we estimate the potential savings of each b-arc. For each b-arc $(i,j) \in \tilde{\mathcal{A}}$, let g_{ij} be the potential savings of (i,j)

and initialize $g_{ij} = 0$. We first force all candidates to be installed. For each OD pair $h \in \mathcal{H}$, find its all-closed shortest path. Denote the path as p_h^1 and its distance as c_h^1. Then we force all candidate stations to be closed, i.e. no b-arc is available. Then, for each $h \in \mathcal{H}$, we find its shortest path. Let p_h^2 be the all-open shortest path of h and c_h^2 be the distance. The generation process described in Chapter 3 of Chiraphadhanakul (2013) ensures that for each $h \in \mathcal{H}$, both p_h^1 and p_h^2 exist. Next, for each OD-pair $h \in \mathcal{H}$, we compute the cost savings as $s_h = c_h^2 - c_h^1$, and find all b-arcs on p_h^1. Denote the set of such b-arcs as $\tilde{\mathcal{A}}_h$. For each $(i,j) \in \tilde{\mathcal{A}}_h$, we calculate $g_{ij} = g_{ij} + s_h w^h / |\tilde{\mathcal{A}}_h|$. We repeat this calculation for all $h \in \mathcal{H}$ and all $(i,j) \in \tilde{\mathcal{A}}_h$.

The following method of ranking the stations using the estimated savings potential is from communication with Dr. Virot Chiraphadhanakul. We generate another network G' which only contains the candidate stations and the b-arcs. We have $G' = (\tilde{\mathcal{N}}, \tilde{\mathcal{A}})$, with the weight of each $(i,j) \in \tilde{\mathcal{A}}$ set equal to g_{ij}. An arc can only exist when both end b-nodes are installed. For each $i \in \tilde{\mathcal{N}}$, let y_i be a binary variable indicating whether candidate station i is selected to be installed. For each $(i,j) \in \tilde{\mathcal{A}}$, let x_{ij} be a binary variable indicating whether the b-arc (i,j) exists. Our goal is to find K candidate stations to install such that the value of $\sum_{(i,j)\in\mathcal{A}} g_{ij}x_{ij}$ is maximized. The intuition is that we want to find b-arcs that can bring large savings to the network. As g_{ij} is the estimated potential savings that b-arc (i,j) can bring to the network, we want to maximize these savings given the constraints. The problem formulation is

$$\text{maximize} \quad \sum_{(i,j)\in\mathcal{A}} g_{ij}x_{ij} \tag{5.1}$$

subject to

$$x_{ij} \le y_i, \forall (i,j) \in \tilde{\mathcal{A}} \tag{5.2}$$

$$x_{ij} \le y_j, \forall (i,j) \in \tilde{\mathcal{A}} \tag{5.3}$$

$$\sum_{i\in\tilde{\mathcal{N}}} y_i = K \tag{5.4}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{A} \tag{5.5}$$

$$y_i \in \{0,1\}, \forall i \in \mathcal{H}, \tag{5.6}$$

where Equation (5.2) and (5.2) make sure that a b-arc can exist only when both of its end stations exist.

The formulation above turns out to be an example of the so called *densest K-subgraph* (DkS) problem. This problem also has many other names such as the maximum edge-weighted subgraph, p-dispersion, remote clique and so on (Martí et al., 2013). It has been used to model facility location problems (Pisinger, 2006), polymerase chain reaction (PCR) primer design in biology (Fernandes and Skiena, 2007), construction of genomic prediction (Maenhout et al., 2010), selecting codewords for memoryless low-power bus coding (Gustafsson, 2004) and optical communication (Sau, 2009).

DkS is an NP-hard combinatorial optimization problem, but Asahiro et al. (2000) provides an approximation method to solve it quickly. The algorithm is the following.

1. Set $\mathcal{X} = \tilde{\mathcal{N}}$.

2. If $|\mathcal{X}| = K$, stop. Otherwise, find $i \in \tilde{\mathcal{N}}$ whose weighted degree is the smallest among all candidate stations and set $\mathcal{X} = \mathcal{X} \setminus \{i\}$. Repeat step 2.

3. Return $\mathcal{X}$ as the design.

This algorithm is called *Greedy*. The weighted degree of i is defined as $r_i = \sum_{(i,j)\in\tilde{\mathcal{A}}} g_{ij}x_{ij} + \sum_{(j,i)\in\tilde{\mathcal{A}}} g_{ji}x_{ji}$. In step 2, when a candidate station i is removed, all arcs connected to i, namely those in $\{(i,j) \mid \exists j, (i,j) \in \tilde{\mathcal{A}}\}$ or $\{(j,i) \mid \exists j, (j,i) \in \tilde{\mathcal{A}}\}$, are also eliminated.

The approximation ratio R of the Greedy algorithm is defined as the ratio between the optimal value of Equation (5.1) and the value of Equation (5.1) given by the Greedy solution. It is proved in Asahiro et al. (2000) that the worst-case R satisfies

$$\left(\frac{1}{2} + \frac{|\tilde{\mathcal{N}}|}{2K}\right)^2 - O\left(|\tilde{\mathcal{N}}|^{-1/3}\right) \le R \le \left(\frac{1}{2} + \frac{|\tilde{\mathcal{N}}|}{2K}\right)^2 + O\left(1/|\tilde{\mathcal{N}}|\right), |\tilde{\mathcal{N}}|/3 < K \le |\tilde{\mathcal{N}}|, \tag{5.7}$$

and

$$2\left(\frac{|\tilde{\mathcal{N}}|}{K}\right) - O\left(1/K\right) \le R \le 2\left(\frac{|\tilde{\mathcal{N}}|}{K}\right) + O\left(|\tilde{\mathcal{N}}|/K^2\right), K \le |\tilde{\mathcal{N}}|/3. \tag{5.8}$$

In our case, the weight of any arc in G' is estimated from the original VS and transit network. Denote R' as the ratio between the optimal objective value of the min-cost model and the objective value of the min-cost model given by the Greedy solution. Note R' is different from the R we defined earlier. Thus, Equation (5.7) and (5.8) are not the range of R', and the range of R' is not clear now.

If we do not stop the candidate station removing process in step 2 of the Greedy algorithm until $\mathcal{X} = \varnothing$, we will have a sequence of all the candidates. It naturally leads to a rank, from 1 to $|\tilde{\mathcal{N}}|$, of these stations. If we predefine two parameters k_1 and k_2, we can add two types of constraints:

$$y_{(i)} = 1, \forall i \in \{1, 2, \ldots, N_1\}, \tag{5.9}$$

and

$$y_{(j)} = 0, \forall j \in \{|\tilde{\mathcal{N}}| - N_2 + 1, |\tilde{\mathcal{N}}| - N_2 + 2, \ldots, |\tilde{\mathcal{N}}|\}. \tag{5.10}$$

$y_{(i)}$ represents the station ranked at the ith place. Here Equation (5.9) is the *high-savings constraint* and Equation (5.10) is the *low-savings constraint.* Together we call them *low-high savings constraints* (LHSCs). It is easy to see that LHSCs fix the value of $N_1 + N_2$ binary variables. Adding them to the master problem of the decomposition described in Section 2.3.1 helps limit the searching space.

5.2.2 Variable Neighborhood Search

Mladenović and Hansen (1997) propose a new local search method called *variable neighborhood search* (VNS) for MIP problems. VNS defines a neighborhood around a current solution, and searches a better solution in this neighborhood. VNS does not follow any trajectory. It changes the neighborhood center if a better solution is found and increases the neighborhood size if no better solution is found after a certain time.

To use VNS in our VS network design problem, we need to first define the distance between the solutions. Suppose we already have the known-best solution $\bar{y}$, and call it the *incumbent.* The *Hamming distance* between $\bar{y}$ and y, another solution, is defined as

$$d(y,\bar{y}) = \sum_{j \in \bar{S}} (1 - y_j) + \sum_{j \in \tilde{\mathcal{N}} \setminus \bar{S}} y_j, \tag{5.11}$$

where $\bar{S} = \{j \in \tilde{\mathcal{N}} \mid \bar{y}_j = 1\}$ (Hansen et al., 2006). The Hamming distance measures the number of binary variables having values different from the incumbent solution.

Denote this constraint $d(y,\bar{y}) \leq k$ as the VNS constraint, or $VNS(\bar{y},k)$. It limits the number of binary variables with values different from the incumbent one to be not larger than k. Set $k = k_{min}$ whenever we find a new incumbent, add this constraint to the Benders master problem and remove the last local search cut . If after several iterations no better solution is found, we increase the neighborhood size by increasing the value of k. VNS cut is another method to limit the search space of the master problem in the decomposition process.

5.2.3 LHSC and VNS Decomposition

We propose a decomposition procedure in this section. We combine the Benders decomposition, the cut improving algorithm in Section 2.3.2, the LHSCs and the VNS cut. The algorithm is summarized in Algorithm 2. We call it *LHSC and VNS Decomposition.* To make it easier to understand, we also draw a flow diagram to show the process in Figure 5-2. g, g' and g'' are thresholds controlling when we stop the search process. k_1 and k_2 are used to control the number of LHSC constraints.

One thing to note is that in Algorithm 2, we relax the VNS cut and LHSC cuts and let the algorithm find the best solution or prove the incumbent is optimal. But as the limit on the search space is removed, solving it will become extremely difficult as we add a large number of improved Benders cuts to the master problem after many iterations. This is illustrated by our computational experiments.

5.3 Computational Results

In this section we first examine the quality of the solution from the Greedy algorithm by using some randomly generated network. To generate the bus-bus arcs and bike-

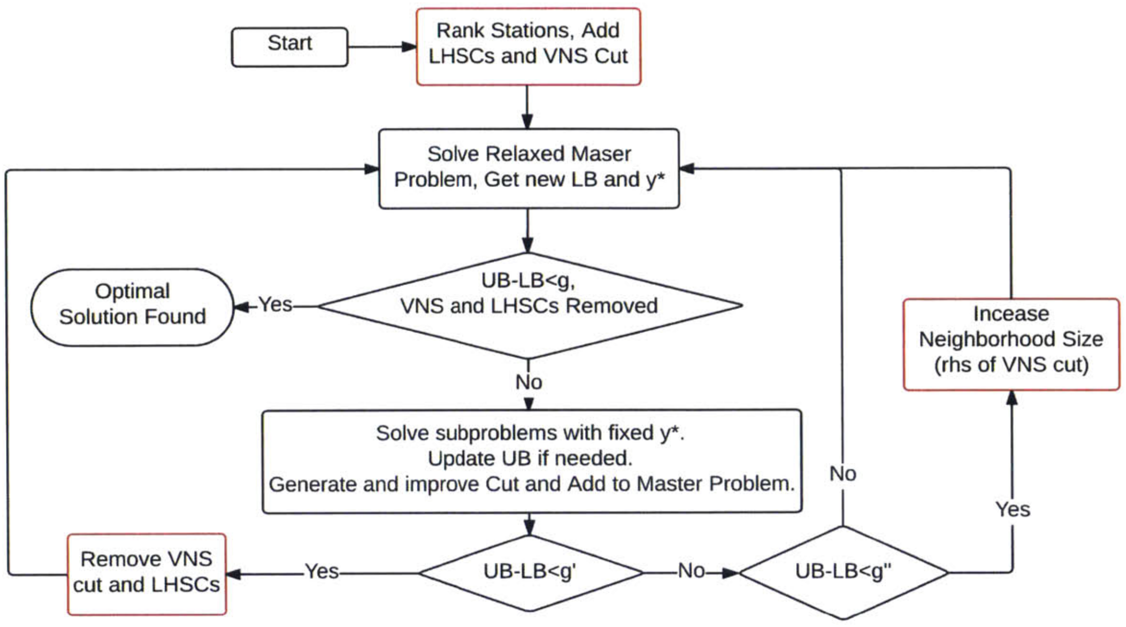

Start
Rank Stations, Add LHSCs and VNS Cut
Solve Relaxed Maser Problem, Get new LB and y*
Optimal Solution Found
Yes
UB-LB<g, VNS and LHSCs Removed
No
Incease Neighborhood Size (rhs of VNS cut)
Solve subproblems with fixed y*. Update UB if needed. Generate and improve Cut and Add to Master Problem.
No
Yes
Remove VNS cut and LHSCs
Yes
UB-LB<g'
No
UB-LB<g"

Figure 5-2: Algorithm: LHSC and VNS Decomposition

bus arcs, the open source software Cytoscape[1] and its plug-in Random Network[2] are used to form the links. For the bike-bike arcs (b-arcs), we connect b-nodes whose distances are within a certain limit. Then we present the case study of the entire Boston network to see the performance of LHSC and VNS decomposition.

5.3.1 LHSC-VNS Solution vs. Greedy Solution

We compare the difference of station selection and the corresponding total cost between the Greedy solution and the LHSC- VNS solutions. In total, three groups of random networks are used, and each group has five different networks. The network files used are summarized in Table 5.1. Net410 - net414 have similar arc density to the MBTA-Hubway network generated by the method in Chapter 3 of Chiraphadhanakul (2013) with 60 bike candidates. We call them the normal density networks. Net420 - net424 are sparse networks, and net430 - net434 are dense networks.

We define two metrics. For a network and a value for K, let s_G^K be the savings of total travel cost achieved by the Greedy solution and s_L^K be the savings from the

[1]`http://www.cytoscape.org/`
[2]`https://sites.google.com/site/randomnetworkplugin/`

Table 5.1: Network Summary

	B-Nodes($\tilde{\mathcal{N}}$)	Nodes($\mathcal{N}$)	Arcs($\mathcal{A}$)	B-Arcs($\tilde{\mathcal{A}}$)	ODs($\mathcal{H}$)
net410	60	760	6141	3100	9950
net411	60	760	6141	3100	9950
net412	60	760	6106	3100	9950
net413	60	760	6111	3100	9950
net414	60	760	6225	3100	9950
net420	60	760	3513	1500	9950
net421	60	760	3526	1500	9950
net422	60	760	3536	1500	9950
net423	60	760	3521	1500	9950
net424	60	760	3418	1500	9950
net430	60	760	11004	3100	9950
net431	60	760	11080	3100	9950
net432	60	760	11062	3100	9950
net433	60	760	11054	3100	9950
net434	60	760	11004	3100	9950

LHSC-VNS solution. Define $I^K = (s_L^K - s_G^K)/s_G^K$ as the percentage improvement in savings of the LHSC-VNS solution's compared to the Greedy solution's. Another metric is the number of different selected stations. We count how many stations are in the LHSC-VNS solution but not in the Greedy solution, and denote this value as N_d^K, given that K candidate stations should be installed. Let $N^K = N_d^K / \min\{(1-k_1)K, (1-k_2)(N-K))\}$ be the ratio of the number of stations changed, where the denominator $\min\{(1-k_1)K, (1-k_2)(N-K))\}$ is the largest number of stations that can be different between a Greedy solution and a LHSC-VNS solution.

Set $k_1 = k_2 = 0.5$, so about half of the candidate stations are selected to be opened or closed. Set $g = 10^{-6}$, $g' = 0.0005$ and $g'' = 0.005$. Note in our case we need to close at least one candidate and install at least one to find a solution different from the current solution, so we set $k_{min} = 2$, and each time we increase k by 2. For each network, we run the LHSC and VNS decomposition for 1.5 hours. The results are summarized in Table 5.2.

For each type of network, we present both the mean and standard deviation of I^K and N^K for different values of K. We can see that I^K decreases as K increases. N^K indicates how different the LHSC-VNS solution and Greedy solution can be in terms of the selected stations. For all three network types, the mean value of N^K follows a U-shaped curve. Note that the closer K is to 0 or $|\tilde{\mathcal{N}}|$, the smaller is the denominator

Table 5.2: LHSC and VNS Decomposition vs. Greedy Algorithm

K	Normal (410-414)				Spare (420-424)				Dense (430-434)			
	I^K		N^K		I^K		N^K		I^K		N^K	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
8	9.7%	5.0%	45%	21%	31.0%	15.1%	80%	11%	11.8%	6.7%	50%	25%
16	9.9%	5.3%	33%	7%	16.8%	9.9%	60%	10%	7.6%	4.0%	45%	21%
24	5.1%	2.6%	32%	11%	8.9%	6.2%	48%	9%	4.6%	0.7%	27%	7%
32	3.8%	1.6%	39%	8%	7.0%	3.8%	46%	11%	3.1%	1.0%	30%	6%
40	2.9%	1.1%	56%	9%	6.6%	2.6%	60%	19%	1.8%	0.9%	38%	13%
48	1.3%	0.4%	87%	7%	2.9%	1.4%	97%	25%	1.2%	0.8%	73%	19%

$\min\{(1-k_1)K, (1-k_2)(N-K))\}$. For most cases, the mean value of N^K is larger than 30%. Combining the two metrics, we can see that when K becomes large, the total cost from the Greedy solution is hard to improve, but the network design may be quite different from the LHSC-VNS solutions that provide smaller total travel cost. Considering the network density, we see that the Greedy solution can be improved for sparse networks in terms of both metrics. The trend is that the quality of the Greedy solution increases with network density.

5.3.2 Case Study: Entire Boston Network

We compare the performance of LHSC and VNS Decomposition and the method presented in Chapter 2. Both use the cut-improving strategy presented in Section 2.3.2. The input network we use here is the entire network generated by MBTA GTFS and Hubway data. The network has $|\mathcal{N}| = 1234$, $|\tilde{\mathcal{N}}| = 131$, $|\mathcal{A}| = 21123$, $|\tilde{\mathcal{A}}| = 14816$ and $|\mathcal{H}| = 22819$. We set a time limit of 5 hours, and let $pass = 15$ for both methods. Other parameter settings are $k_1 = k_2 = 0.5$, $g = 10^{-6}$, $g' = 0.0005$, $g'' = 0.005$ and $k_{min} = 2$.

Table 5.3: LHSC and VNS Decomposition vs. Benders Decomposition

K	Improved Benders				LHSC and VNS			
	LB	UB	Gap	Iterate	LB	UB	Gap	Iterate
20	1.344×10^9	1.755×10^9	30.55%	16	$1.695\times10^9*$	1.740×10^9	29.45%	36
40	1.395×10^9	1.715×10^9	22.88%	8	$1.665\times10^9*$	1.687×10^9	20.89%	32
60	1.538×10^9	1.681×10^9	9.29%	11	$1.634\times10^9*$	1.647×10^9	7.08%	57
80	1.567×10^9	1.657×10^9	5.72%	16	$1.601\times10^9*$	1.615×10^9	3.11%	47
100	1.572×10^9	1.609×10^9	2.26%	28	1.569×10^9	1.587×10^9	1.02%	44

* LB is invalid because the LHSCs and VNS have not been removed.

The results are summarized in Table 5.3. LB and UB stands for the largest lower bound and the smallest valid upper bound on the optimal objective value of the min-cost model, respectively. Gap reflects the percentage difference between UB and the largest valid LB. In our case, the largest valid LB for all K values are from the Improved Benders method. Iterate is the number of iterations within the time limit.

We can see that for all K values, LHSC and VNS decomposition provides a smaller UB, where UB is the total cost of the network design from the solution. The LHSC and VNS decomposition method finds better solutions within limited running times. However, LHSC and VNS decomposition may not provide a valid LB within the run time limit, Gaps between the best UB and the valid LB decrease as K increases, which indicates that the VS network design problem with the objective to minimize total cost is hard to solve when we have fewer stations to install. When K is large, the Gap column shows that the total cost value from the LHSC-VNS solution is closer to the best valid LB.

5.4 Conclusions

In this chapter, we focus on the VS network design with the objective to minimize total cost. We introduce a heuristic called Greedy algorithm to find a feasible solution quickly. We provide the Greedy solution to the LHSC and VNS decomposition method, which forces some stations to open and some to close (LHSCs) and uses VNS constraints to limit the search space of the master problem of the decomposition. Computational results on random networks show that the quality of the Greedy solutions is high when the number of stations to open is large, and LHSC and VNS decomposition can help find better solutions. For the Boston network case study, we learned that LHSC and VNS decomposition can help find better solutions than using the decomposition method from Chiraphadhanakul (2013). For both decomposition methods, the VS network design problem is harder to solve when the number of stations to install is small, and when the number is large, LHSC and VNS decomposition more clearly outperforms.

Algorithm 2 LHSC and VNS Decomposition

```
1: Initialize $g$, $g'$, $g''$, $k_1$, $k_2$, $k_{min}$ and $\lambda$
2: Run *Greedy*, get sorted sequence of stations
3: Let $y_G$ be the last $K$ stations in the sequence. Set the $y_G$ as the MIP start point,
   and the incumbent solution $y^* \leftarrow y_G$. Best upper bound $BestUB \leftarrow Cost(y_G)$
4: $k \leftarrow k_{min}$ and add VNS cut $VNS(y^*, k)$
5: $N_1 \leftarrow K - K_e$, $N_2 \leftarrow N - K$ ($K_e$: number of existing stations)
6: Force the $k_1 \times N_1$ candidates at the end of the sequence to be open (high-saving)
   and $k_2 \times N_2$ candidates at the start of the sequence to be closed (low-saving).
   They are LHSCs.
7: while remaining time $> 0$ do
8:     Solve the Benders master problem, get master objective value $Obj$ and current
       solution $\bar{y}$. Solve all Benders subproblems, using $\bar{y}$ and cut improving algorithms
       to generate cuts and get its corresponding cost $Cost(\bar{y})$
9:     if $BestUB - Obj < g$ then
10:        if LHSCs have not been removed then
11:            Remove $VNS(y^*, k)$
12:            $k \leftarrow k + 2$ and add $VNS(y^*, k)$
13:        else
14:            Break
15:        end if
16:    end if
17:    if $Cost(\bar{y}) < BestUB$ then
18:        $BestUB \leftarrow Cost(\bar{y})$
19:        $y^* \leftarrow \bar{y}$
20:        Remove $VNS(y^*, k)$
21:        $k \leftarrow k_{min}$ and add $VNS(y^*, k)$
22:    else
23:        if $(BestUB - Obj)/BestUB < g''$ then
24:            Remove $VNS(y^*, k)$
25:            $k \leftarrow k + 2$ and add $VNS(y^*, k)$
26:        end if
27:    end if
28:    if $(BestUB - Obj)/BestUB < g'$ then
29:        Remove LHSCs and $VNS(y^*, k)$
30:    end if
31: end while
32: return $y^*$
```

Chapter 6

Conclusions and Future Work

In this thesis, we address the issue of network design for integrated vehicle-sharing and public transportation service in order to reduce people's travel time. We focus on the problem of selecting from a predefined set of candidate VS stations those to be optimal.

We propose a new model with the objective to provide more travelers with significant travel time savings. To expedite the model solution process, we develop a decomposition procedure and special cut generation method. Computational results shows that our efforts are effective. The expand cut we proposed can reduce the solution time significantly. OD-wise analysis of the achieved savings shows that the max-demand solution increases the number of demands that satisfy a savings criterion. The targeted demands also have large relative time savings and fewer transfers with a max-demand solution. We also propose a heuristic-based decomposition method that can produce solutions to the Chiraphadhanakul (2013) model with reduced total cost within specified time limit. Computational results on random networks show that our LHSC and VNS decomposition can help find solutions better than the solutions from the Greedy method. For the Boston network case study, we learned that LHSC and VNS decomposition can help find better solutions than using the decomposition method from Chiraphadhanakul (2013).

For future work, one thing we need is to prove that the decomposition and cut generating process of the new model will converge. As the cuts we generate are valid, the decomposition schema will give an optimal solution if it converges. But we have not proved that such convergence will always happen. Another thing to note is that currently in the model, we assume all OD pairs have the same criterion for satisfactory travel time savings. We could, in the future, take into account more complicated situations. For instance, both absolute savings and relative savings could be taken into account. This may require a new objective function, a new corresponding model, and new cut generating methods. Also, larger and more realistic instances should be used to test our new model and algorithm. The heuristic-based method may also be modified to adapt the possible new models. Now the network data we use are from the MBTA schedule and the biking time between points are queried from Mapquest. In the future, we may use real-time traffic data to better calibrate arc travel times.

Appendix A

Improvement in Total Demand Meeting Different Savings Criterion and Number of Open Stations

Table A.1: Number of Demands Meeting s Saving Criterion, $K = 12$

Saving (min)	Min Cost	Value of s							
		50	45	40	35	30	25	20	15
≥ 50	199	1388	609	316	823	955	1204	819	159
≥ 45	514	1643	3194	2663	2291	2526	1346	2303	533
≥ 40	2366	5126	5631	6675	3797	3897	4959	3940	1054
≥ 35	5943	7675	8615	9181	11903	8696	9186	7527	2564
≥ 30	12555	17276	15949	15969	20135	21172	18261	17192	6971
≥ 25	15293	24311	19317	22126	27398	25917	30927	25541	14299
≥ 20	24975	27189	30669	32055	34927	37824	38331	37777	21729
≥ 15	45971	51231	47211	47511	42244	53822	49410	48217	56785

Table A.2: Improvement of Number of Demands Satisfying s Saving Criterion, $K = 12$

	Value of s							
Saving (min)	50	45	40	35	30	25	20	15
≥ 50	597.5%	206.0%	58.8%	313.6%	379.9%	505.0%	311.6%	-20.1%
≥ 45	219.7%	521.4%	418.1%	345.7%	391.4%	161.9%	348.1%	3.7%
≥ 40	116.7%	138.00%	182.1%	60.5%	64.7%	109.6%	66.5%	-55.5%
≥ 35	29.1%	45.0%	54.5%	100.3%	46.3%	54.6%	26.6%	-56.7%
≥ 30	37.6%	27.0%	27.2%	60.4%	68.6%	45.5%	36.9%	-44.5%
≥ 25	59.0%	26.3%	44.7%	79.2%	69.5%	102.2%	67.0%	-6.5%
≥ 20	8.9%	22.8%	28.4%	39.9%	51.5%	53.5%	51.3%	-13.0%
≥ 15	11.4%	2.7%	3.5%	-8.1%	17.1%	7.5%	4.9%	13.3%

Table A.3: Number of Demands Meeting s Saving Criterion, K = 18

Saving (min)	Min Cost	Value of s							
		50	45	40	35	30	25	20	15
≥ 50	1008	1390	1388	644	1383	1383	1389	1123	1385
≥ 45	2649	1794	3279	3120	3247	3253	1798	2857	3132
≥ 40	5163	6056	6317	6952	5881	5887	6306	4988	6187
≥ 35	9571	9520	10878	10954	13291	12627	11516	12154	9853
≥ 30	19684	19372	21134	21244	23465	23528	21552	23113	21041
≥ 25	31281	28973	29711	30065	33083	33296	36293	33278	30573
≥ 20	41607	37018	40497	46299	46945	47574	46086	50104	40531
≥ 15	75646	67529	66380	68639	69053	69777	71280	71587	82231

Table A.4: Improvement of Number of Demands Satisfying s Saving Criterion, K = 18

	Value of s							
Saving (min)	50	45	40	35	30	25	20	15
≥ 50	37.9%	37.7%	58.8%	-36.1%	37.2%	37.8%	11.4%	37.4%
≥ 45	-32.3%	23.8%	17.8%	22.6%	22.8%	-32.1%	7.9%	18.2%
≥ 40	17.3%	22.6%	34.7%	13.9%	14.0%	22.1%	-3.4%	19.8%
≥ 35	-0.5%	13.7%	14.5%	38.9%	31.9%	20.3%	27.0%	3.0%
≥ 30	-1.6%	7.4%	7.9%	19.2%	19.5%	9.5%	17.4%	6.9%
≥ 25	-7.4%	-5.0%	-3.9%	5.8%	6.4%	16.0%	6.4%	-2.3%
≥ 20	-11.0%	-2.7%	11.3%	12.8%	14.3%	10.8%	20.4%	-2.6%
≥ 15	-10.7%	-12.3%	-9.3%	-8.7%	-7.8%	-5.8%	-5.8%	8.7%

Table A.5: Number of Demands Meeting s Saving Criterion, K = 21

Saving (min)	Min Cost	Value of s							
		50	45	40	35	30	25	20	15
≥ 50	1230	1390	1388	1388	653	1383	1390	1130	1387
≥ 45	3091	3073	3279	3144	3255	3255	1851	2888	3138
≥ 40	6312	6069	6370	6955	6893	5887	6529	5675	6202
≥ 35	12697	11954	10894	12018	13590	12723	12547	12332	11384
≥ 30	22539	22145	21462	23751	22808	23989	23040	23500	21614
≥ 25	36443	32763	30721	35491	33042	34919	37458	35735	32067
≥ 20	48211	46319	44504	49909	49627	51820	48629	53378	45367
≥ 15	82291	75575	76042	77103	72828	77393	77274	77959	86834

Table A.6: Improvement of Number of Demands Satisfying s Saving Criterion, K = 21

	Value of s							
Saving (min)	50	45	40	35	30	25	20	15
≥ 50	13.0%	12.9%	12.9%	-46.9%	12.4%	13.0%	-8.1%	12.8%
≥ 45	-0.6%	6.1%	1.7%	5.3%	5.3%	-40.1%	-6.6%	1.5%
≥ 40	-3.9%	0.9%	10.2%	9.2%	-6.7%	3.4%	-10.1%	-1.7%
≥ 35	-5.85%	-14.20%	-5.35%	7.0%	0.2%	-1.2%	-2.9%	-10.3%
≥ 30	-1.8%	-4.8%	5.4%	1.2%	6.4%	2.2%	4.3%	-4.1%
≥ 25	-10.1%	-15.7%	-2.6%	-9.3%	-4.2%	2.8%	-1.9%	-12.0%
≥ 20	-3.9%	-7.7%	3.5%	2.9%	7.5%	0.9%	10.7%	-5.9%
≥ 15	-8.2%	-7.6%	-6.3%	-11.5%	-6.0%	-6.1%	-5.3%	8.7%

Bibliography

Asahiro, Y., Iwama, K., Tamaki, H. and Tokuyama, T. (2000). Greedily finding a dense subgraph, *Journal of Algorithms* **34**(2): 203–221.

Balas, E. (1979). Disjunctive programming, *Annals of Discrete Mathematics* **5**: 3–51.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems, *Numerische mathematik* **4**(1): 238–252.

Bertsimas, D. and Tsitsiklis, J. N. (1997). Large scale optimization, *Introduction to linear optimization*, Athena Scientific, Belmont, MA, Co-published by Dynamic Ideas, LLC.

Chiraphadhanakul, V. (2013). *Large-scale analytics and optimization in urban transportation: improving public transit and its integration with vehicle-sharing services*, PhD thesis, Massachusetts Institute of Technology.

Correia, G. H. d. A. and Antunes, A. P. (2012). Optimization approach to depot location and trip selection in one-way carsharing systems, *Transportation Research Part E: Logistics and Transportation Review* **48**(1): 233–247.

Correia, G. H. D. A., Jorge, D. R. and Antunes, D. M. (2014). The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system: An application in lisbon, portugal, *Journal of Intelligent Transportation Systems* **18**(3): 299–308.

Fernandes, R. and Skiena, S. (2007). Multiprimer, *PCR Primer Design*, Springer, pp. 305–314.

García-Palomares, J. C., Gutiérrez, J. and Latorre, M. (2012). Optimizing the location of stations in bike-sharing programs: a gis approach, *Applied Geography* **35**(1): 235–246.

Gustafsson, O. (2004). Graph-based codeword selection for memoryless low-power bus coding, *Electronics Letters* **40**(24): 1531–1532.

Hansen, P., Mladenović, N. and Urošević, D. (2006). Variable neighborhood search and local branching, *Computers & Operations Research* **33**(10): 3034–3045.

Jorge, D. and Correia, G. (2013). Carsharing systems demand estimation and defined operations: a literature review, *EJTIR* **13**(3): 201–220.

Kumar, V. P. and Bierlaire, M. (2012). Optimizing locations for a vehicle sharing system, *Swiss Transport Research Conference. http://www. strc. ch/conferences/2012/Kumar_ Bierlaire. pdf.*

Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse, *Operations research letters* **13**(3): 133–142.

Lin, J.-R. and Yang, T.-H. (2011). Strategic design of public bicycle sharing systems with service level constraints, *Transportation research part E: logistics and transportation review* **47**(2): 284–294.

Maenhout, S., De Baets, B. and Haesaert, G. (2010). Graph-based data selection for the construction of genomic prediction models, *Genetics* **185**(4): 1463–1475.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating benders decomposition: Algorithmic enhancement and model selection criteria, *Operations research* **29**(3): 464–484.

Martí, R., Gallego, M., Duarte, A. and Pardo, E. G. (2013). Heuristics and metaheuristics for the maximum diversity problem, *Journal of Heuristics* **19**(4): 591–615.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search, *Computers & Operations Research* **24**(11): 1097–1100.

Nair, R. and Miller-Hooks, E. (2014). Equilibrium network design of shared-vehicle systems, *European Journal of Operational Research* **235**(1): 47–61.

Penuel, J., Smith, J. C. and Yuan, Y. (2010). An integer decomposition algorithm for solving a two-stage facility location problem with second-stage activation costs, *Naval Research Logistics (NRL)* **57**(5): 391–402.

Pisinger, D. (2006). Upper bounds and exact algorithms for p-dispersion problems, *Computers & operations research* **33**(5): 1380–1398.

Sau, I. (2009). *Optimization in Graphs under Degree Constraints. Application to Telecommunication Networks*, PhD thesis, Université Nice Sophia Antipolis; Universitat Politécnicade Catalunya.

Sen, S. and Higle, J. L. (2005). The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification, *Mathematical Programming* **104**(1): 1–20.

Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming, *Mathematical Programming* **106**(2): 203–223.

Shaheen, S. A. and Cohen, A. P. (2013). Carsharing and personal vehicle services: worldwide market developments and emerging trends, *International Journal of Sustainable Transportation* **7**(1): 5–34.

Shaheen, S., Guzman, S., Zhang, H., Pucher, J. and Buehler, R. (2012). Bikesharing across the globe, *Pucher J, Buehler R. eds* pp. 183–209.

Shen, S. and Smith, J. C. (2013). A decomposition approach for solving a broadcast domination network design problem, *Annals of Operations Research* **210**(1): 333–360.

Sherali, H. D. and Adams, W. P. (1998). *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, Vol. 31, Springer Science & Business Media.

Sherali, H. D. and Fraticelli, B. M. (2002). A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse, *Journal of Global Optimization* **22**(1-4): 319–342.

Sherali, H. D. and Zhu, X. (2006). On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables, *Mathematical Programming* **108**(2-3): 597–616.

Zhu, X. (2006). *Discrete two-stage stochastic mixed-integer programs with applications to airline fleet assignment and workforce planning problems*, PhD thesis, Virginia Polytechnic Institute and State University.