The Dissertation Committee for Can Gokalp
certifies that this is the approved version of the following dissertation:

Three Nonlinear Network Flow Problems

Committee:

Stephen D. Boyles, Supervisor

Avinash Unnikrishnan

Erhan Kutanoglu

John Hasenbein

Three Nonlinear Network Flow Problems

by

Can Gokalp

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2021

To *mom, great-aunt & great-uncle*

Acknowledgments

I would like to extend my gratitude and thanks to many people that have helped me in many ways – academically, socially, emotionally, and financially – with their guidance and support throughout my doctoral journey.

Firstly, many thanks to my advisor Steve Boyles! He has been a teacher, an advisor, and a friend. His expertise in the field, guidance, and edits on our work has been very valuable in developing this thesis. Without his inputs, this thesis wouldn't have been possible. I want to mention that his calm and encouraging approach helped me navigate the times I got stuck and kept me believing that I can do it. Besides academics, I would also like to thank him for creating a healthy research environment by putting together a research group that is very social and active, and also for sharing valuable/fun off-topic knowledge and life hacks!

Thanks to all of our faculty for their teaching. Special thanks to my committee members for their help, contributions, and guidance. Thanks to Erhan Kutanoglu for teaching me Integer Programming and having an open office door that is always welcoming. Thanks to John Hasenbein for teaching Stochastic Processes and Markov Decision Processes. I am going to miss his style of teaching. The most fun and enjoyable classes I have taken in UT! Thanks to Avinash Unnikrishnan for his comments and guidance on my thesis. Thanks to Grani Hanasusanto for advising me for a year.

I can not express how grateful I am for my family; my mom (Surreyya Bayrak)

my sister (Sila Gokalp), dad (Cengiz Gokalp), great-aunt (Mucella Yildiz Erkal), great-uncle (Ural Erkal), and uncle (Huseyin Bayrak). Special thanks to my mom. She was my first teacher through the 3rd grade. After that, we still went to the school every day together till I graduated high school - she was teaching elementary at the same school (Ted Istanbul Koleji). I appreciate all the hard work she put in to make my doctoral journey possible.

I would like to thank all my friends both in Turkey and in the states. Their friendships made me grow in many different directions. It would be infeasible to give a shoutout to all my friends here, but I would like to mention friends most relevant for their support and friendship during my studies at UT. Thanks to my long-time friends from Turkey – Oguz Zagra, Alper Aydin, Melis Aksoy, Ezgi Korba, Alican Ates, Dogakan Toka, Guchan Ozbilgin, and Ayhan Aydogan – for kept checking in with me and not avoiding their jokes about me being in school forever. Thanks to Kory Harb, Andrew Daw, Daniel Schra, and Mark Hettig for including me in their super close friend group, creating great memories during my Master's degree at The University of Florida. Thanks to Will Schievelbein, Dan Kinn, and Claire Herlin for their friendship that started on my first day at UT. It was a blast to work on classes and projects together while also exploring and enjoying Austin.

Lastly, I would like to thank National Science Foundation (NSF) for supporting the research work presented in this thesis under grant numbers; CMMI-121562109, CMMI-1254921, CMMI-1562291.

Three Nonlinear Network Flow Problems

Publication No. _____

Can Gokalp, Ph.D.
The University of Texas at Austin, 2021

Supervisor: Stephen D. Boyles

This dissertation is concerned with network flow problems for which some of the conventional assumptions are in violation. These violations in the assumptions disrupt the exploitable structure that exists in traditional cases. As a result, the resulting problems are then not amenable to the efficient solution methods developed for the traditional cases. They still allow convex programming methods. However, the networks considered often represent cities and states that can be large in size, and therefore there is a need to investigate more practically scalable approaches for the considered applications.

In particular, in each chapter, we study a different application, each challenging a different conventional assumption. We then present solution methods that are more scalable compared to the current approaches for the resulting problems. Described methods we develop rely on; characterizing the optimal solution and optimality conditions, analyzing solution sensitivities, and exploiting the remaining network structure.

The first problem is concerned with reliability in minimum cost flow problems. In many applications, especially logistics, travel time variation often has critical implications. As a result, it might be more preferable to use a route with

less variation but with a higher mean cost depending on the risk averseness of the decision-maker. One way to model this is by considering a weighted combination of the mean and standard deviation of the flow costs. Even though this choice has modeling benefits, the resulting optimization problem has an objective function that is non-separable by arcs and thus harder to solve than the separable case. We first prove that the solution for this problem coincides with the solution for a particular separable problem. We then leverage this result by using root-finding methods to find that particular separable problem. Essentially, this approach solves the original non-separable problem by solving many easier separable problems. We also discuss how the results could be extended for a more general case of non-separable convex problems. Our experiments show that the proposed methods provide significant advantages over solving the non-separable directly by using an off-the-shelf solver.

We then study how to solve for a system optimal assignment for a particular parking model. Identifying system optimal assignment is crucial as it can be a guide for the transportation planners for implementing policies/strategies that can reduce the congestion in the system. The parking model considered, captures the inherent circling for parking behavior of the users. As a result, it leads to nonlinearities in the flow conservation equations. To solve for a system optimal assignment, we first formulate the optimization problem. By representing decision variables in splitting fractions space, we get linear constraints. However, doing so shifts the complexity to the objective, resulting in a non-convex function. Nevertheless, we propose a descent approach. To find the required derivative information, we provide sensitivity analysis for the cyclic network. We showcase the value of identifying a system optimal assignment on a small toy network, where the planner can adjust the parking

prices to derive the user equilibrium towards the system optimal and therefore reducing the total system cost.

In the last chapter, we study a repair sequencing problem for road networks that are impaired due to natural disasters. Our focus here is on long-term recovery disaster relief. We investigate how to identify a (sequential) link repair sequence that minimizes total travel time over the repair horizon, given that at each repair stage road traffic distributes according to the principle of user equilibrium. Unlike the problems in the single machine scheduling literature, the project weights, in this case, are unknown a priori due to the nonlinear congestion effects coming from traffic equilibrium. We first derive an analogue of Bellman's optimality principle, allowing us to solve the problem using methods of dynamic programming. We then develop a bidirectional search heuristic with customized pruning and branching strategies that exploit specific properties of traffic assignment. Our experiments show that our method is scalable and performs well even on networks involving thousands of links.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Network flow problems are widely used to study transportation and other infrastructure and organizational systems. Classical network optimization problems have special structure which allows for efficient solutions even on very large networks. However, particular applications may modify the objective function or constraints in ways that disrupt these special structures, or embed network optimization as a subproblem in a larger optimization problem. In all of these cases, the classical methods fail. Nevertheless, the underlying network structure remains, and it is often still possible to design specialized algorithms which exploit it, even if such methods are more involved or less efficient than in the classical case.

This dissertation explores three such problems: (1) a minimum cost flow problem with stochastic link costs, where the objective is to minimize a weighted sum of the mean and standard deviation of the flow; (2) a minimum cost flow problem with nonlinear flow conservation constraints; and (3) a sequential repair problem on a network, where the flow attains an equilibrium state at each repair stage.

The **first problem**, discussed in Chapter 2, is motivated by the importance of reliability and risk management in many logistics applications. Link costs are subject to uncertainty due to weather, traffic congestion, and many other

reasons. We propose to represent risk aversion by finding a flow which minimizes a weighted sum of mean cost, and standard deviation of the flow cost. In contrast to the mean cost, the standard deviation of the flow cost involves a square root which cannot be separated by link. Classical minimum-cost flow algorithms rely crucially on this separability assumption.

We will show that this mean-standard deviation problem is closely related to the mean-variance problem (minimizing a weighted sum of mean and variance), whose objective function *is* separable by link. In particular, we show that optimal solutions to the mean-standard deviation problem are also optimal to the mean-variance problem, possibly with a different weighting between mean and variance. This allows us to solve the nonseparable mean-standard deviation problem by solving a sequence of easier, separable mean-variance problems.

We provide three algorithms of this type. The first is based on bisection, the second a Newton search, and the third is a hybrid of the two. The bisection method is simplest. The Newton method requires fewer iterations to converge, but requires solving a related sensitivity problem to calculate derivatives. The hybrid method is essentially the Newton method with a bisection "backstop" in case of pathological behavior. Our experiments show significant computational advantages over off-the-shelf solvers. We also explore generalizations of this approach to other non-separable parametric minimum cost flow problems.

The **second problem**, discussed in Chapter 3, is motivated by drivers searching for parking on urban road networks. This phenomenon contributes significantly to traffic congestion in certain neighborhods, and parking management is receiving increasing attention in urban planning. Prior research has developed a "user equilibrium" model of parking search: parking availability is

stochastic, drivers search for parking where it is likely to be found, but the likelihood of finding parking in any particular location depends on the searching patterns by everyone else. The resulting game-theoretic model yields equilibrium solutions to this mutual dependency. This dissertation focuses on the "system optimal" problem in this system, identifying the flows which would minimize total cost. Such flows are of interest to planners in designing parking policies (such as on- and off-street parking prices) that can shift the user equilibrium towards a lower-cost state.

The main complication in this problem is that the flow conservation constraints become nonlinear, because they reflect the cycling dynamics which occur at parking lots or at on-street spaces. As a result, even identifying the flows given a set of parking strategies is difficult, let alone identifying improving directions for reducing total cost. We formulate the parking strategies in terms of "splitting proportions" at each node, then derive the Jacobian of the flows on each link with respect to these proportions. We then calculate the gradient of the total cost, a propose a descent algorithm. Due to the nonlinearities in the constraints, the objective function is not convex in the splitting proportions (and indeed, even the feasible region may not be a convex set). We therefore content ourselves with identifying local optima, and in particular show how to improve upon a given initial solution (perhaps the user equilibrium obtaining in the field).

The **third problem**, discussed in Chapter 4, is a network repair problem, motivated by post-disaster scenarios where links must be reconstructed. We specifically investigate the scheduling problem of determining a sequence to rebuild damaged links in a way that minimizes total user cost over the repair horizon. We assume that repairs must proceed sequentially (and not in paral-

lel), and that at each stage the network flows reach a user equilibrium state. Unlike classical scheduling problems, the benefit of repairing a particular link depends heavily on where it lies in the sequence, because of dependencies between network components, as when damaged links are in series or in parallel. Nevertheless, we derive an analogue of Bellman's optimality principle, showing that any subsequence of an optimal repair sequence must be "locally optimal" in the sense that it minimizes cost no matter how the links in the other parts of the sequence are permuted.

We develop a bidirectional search heuristic exploiting this subsequence optimality condition, reformulating the problem using dynamic programming. To restrict the number of states which must be explored, we develop a large number of problem-specific branching and pruning rules. The most significant of these rules uses heuristic estimates of the optimal cost for repairing subsets of links, based on the observation that diminishing marginal returns are typically seen (repairing a link earlier in the sequence improves network conditions more than when the network is nearly rebuilt). We show that this algorithm is scalable (computationally feasible on networks with thousands of links), and performs favorably compared to simpler heuristics which ignore the dependencies between network links.

Finally, a brief conclusions chapter summarizes the main contributions and future directions for each problem.

Chapter 2

Mean-Standard Deviation Model for Minimum Cost Flow Problem

2.1 Introduction

The minimum cost flow (MCF) problem is to find the flow in a network that minimizes total cost while satisfying node demands and arc capacities. Many other flow and circulation problems are special cases of MCF, including the shortest path and maximum flow problems. Decision-making problems in a variety of industries — transportation, manufacturing, medicine, health care, energy, and defense, to name a few — can be formulated as MCF problems. In the traditional MCF formulation, the arc costs are assumed to be deterministic. This setting is well studied and several families of efficient algorithms have been developed for it [1].

When the arc costs are stochastic, the decision maker is often concerned with solution reliability in addition to minimizing the expected cost. Results in the travel choice literature show that travel time reliability is of comparable importance as mean travel costs [34, 33, 32, 25], motivating the incorporation of reliability-based objectives into specific network optimization problems with transportation applications. There is a rich body of literature on stochastic shortest path variants using different reliability specifications — minimizing variance or standard deviation in addition to expected travel times [104, 90,

94, 45, 92, 119, 117], maximizing probability of arrival or disutility associated with a pre-specified arrival time [70, 71, 29, 30, 73, 75, 23, 95, 108], percentiles [105, 108], risk aversion [76, 103, 111], and so forth. Reliability and risk related objectives have also been incorporated into traffic assignment models [22, 78, 110, 74, 77, 101, 91, 81].

There has been relatively less research on incorporating reliability objectives into other traditional minimum cost flow and max flow network problems. Boyles and Waller [20] study a specific instance of the convex MCF problem, with independent uncertain arc costs, where the aim is to minimize linear combination of the mean and the variance of the total flow costs - termed as the mean-variance minimum cost flow problem (MVMCF). In their model, the decision maker chooses a weight parameter indicating the relative importance of the mean and variance. They define arc marginal costs and use them to modify the generic cycle canceling algorithm. The objective is non-linear, but separable by arcs. This separability property is critical for their algorithm. The general case of MCF problems with costs that are strictly convex, differentiable and separable by arc is studied in [72]. They derive optimality conditions and provide a primal-dual algorithm. One other approach is to simply transform the problem to the traditional linear MCF problem by using piecewise linearization of the arc cost functions and use existing linear MCF solution methods [67, 48]. More recently, Végh [98] describes a strongly polynomial algorithm. In this chapter, we study the MCF problem with independent uncertain arc costs where the objective is to minimize the mean and the standard deviation of the total flow cost. While the resulting objective function is still convex, it is not separable by arc. Therefore, the approaches to the convex separable version of the problem mentioned above are are not

applicable. This type of convex nonseparable flow problems can still be solved in polynomial, although not strongly polynomial – the best running time reported for such problems $O(m^3L)$, where L is the total length of the input coefficients and m is the number of arcs [43]. In this chapter, we have a different approach. We prove that the solution to the mean-standard deviation minimum cost flow (MSDMCF) problem can be obtained by solving the MVMCF problem for an appropriate choice of weight parameter. We provide three root-finding-based algorithms (bisection, Newton-Raphson, and hybrid) to determine the appropriate weight parameter. A network flow sensitivity analysis procedure is developed to determine the derivatives for the Newton-Raphson and hybrid procedures. The MSDMCF problem is a special case of the more generalized non-separable parametric MCF (GNPMCF) problem where the objective consists of a linear additive function of flow and a weighted non-linear, non-separable function of flow. Our algorithms can be extended to the GNPMCF problem as long as the non-additive component of the objective function is differentiable, monotonically increasing, and convex function of an additive and differentiable criterion.

Other researchers have applied robust optimization approaches to account for uncertainties in network parameters such as demands [3], costs and capacity [13], and network structure [12]. In the robust optimization paradigm, the uncertain parameters are assumed to vary in a pre-specified uncertainty set. The aim is to arrive at the best solution which is feasible for all possible realization of the uncertain parameters from their pre-specified sets. The shape of the uncertainty set indicates the decision makers risk preference and affects the tractability of the model [9]. Birge [14] and Glockner [39] apply a multi-stage stochastic programming approach to model uncertainties in network parame-

ters in a stochastic and dynamic network flow setting.

Stochastic programming approaches require knowledge of the probability distribution of the uncertain parameters. In contrast, we assume that the decision maker knows the mean and standard deviation of arc costs and is interested in minimizing the mean and standard deviation of total network flow costs. We do not focus on worst-case scenarios which can lead to overly conservative solutions.

There has been a separate body of work focusing on the impact of node and arc disruptions on the ability of a network to sustain a specific amount of flow [58, 59, 61]. Lin et al. [60] study the stochastic maximum flow problem where the nodes and arcs have uncertain discrete capacities and develop an algorithm to compute the system reliability defined as the probability that the maximum flow is greater than the given demand. Lin [62] focuses on the multi-commodity variant of [60] and defines system reliability objective as the probability of upper bound of system capacity equals a given pattern subject to budget constraints on flows. Along similar lines, Lin [63] adopts a throughput style definition of system reliability as the probability of sending a pre-specified amount of flow through the network under a cost constraint. Kuipers [52] formulate two stochastic maximum flow models: (i) maximum flow in stochastic networks (MFSN) - where the bandwidth or capacity has a log-concave probability distribution, (ii) maximum delay constrained flow problem (MDCF) where an additional stochastic delay constraint is imposed on the flows. A convex formulation and polynomial time algorithm is provided for the MFSN problem. The MDCF formulation is shown to be NP-hard and solved using an approximation algorithm. The MSDMCF model presented in this chapter does not consider disruptions, failures, or uncertainties in capacity.

Our model has a cost minimization perspective, whereas the above studies are concerned with maximizing flows and require full knowledge of the probability distributions.

The remainder of the chapter is organized as follows. We introduce the problem formulation of the MSDMCF and show the relevance to the MVMCF in Section 2.2. Section 2.3 describes the algorithm developed for solving the MSDMCF. In Section 2.4, we extend the results to a more general class of GNPMCF problems. We demonstrate the efficiency of our methods on randomly generated networks in Section 2.5, and finally, we conclude and discuss future directions in Section 4.8.

2.2 Problem Statement

2.2.1 Problem formulation

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ represent a directed network with $\mathcal{N}$ and $\mathcal{A}$ denoting the set of nodes and arcs, respectively with $m = |\mathcal{A}|$ and $n = |\mathcal{N}|$. The arc costs, c_{ij}, are stochastic, but independent, with known means, $E[c_{ij}]$, and variances, $Var[c_{ij}]$. Nodes and arcs are assumed to have deterministic demands d_j and finite capacities u_{ij} respectively. Let x_{ij} denote the flow on arc (i, j) and $\mathbf{x}$ the vector of all flows. The MSDMCF problem considered in this chapter has the following form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \sum_{(i,j)\in\mathcal{A}} E[c_{ij}]x_{ij} + \bar{\lambda}\sqrt{\sum_{(i,j)\in\mathcal{A}} Var[c_{ij}]x_{ij}^2} & & \\ \text{s.t.} \quad & \sum_{(j,k)\in A} x_{jk} - \sum_{(i,j)\in\mathcal{A}} x_{ij} = d_j & \forall j \in \mathcal{N} & \qquad (\text{MSDMCF}(\bar{\lambda})) \\ & 0 \le x_{ij} \le u_{ij} & \forall (i,j) \in \mathcal{A} & \end{aligned}$$

or, more compactly,

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & \boldsymbol{\mu}^T\mathbf{x} + \bar{\lambda}\sqrt{\mathbf{x}^T\mathbf{V}\mathbf{x}} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{array} \qquad \text{(MSDMCF}(\bar{\lambda})\text{)}$$

with

$$\boldsymbol{\mu} = \begin{bmatrix} E_1 \\ \vdots \\ E_m \end{bmatrix}, \quad \mathbf{V} = diag(\mathbf{Var}) = \begin{bmatrix} Var_1 & 0 & 0 \\ & \ddots & \\ 0 & 0 & Var_m \end{bmatrix}, \quad \bar{\lambda} \geq 0,$$

and $\mathbf{A}\mathbf{x} = \mathbf{b}$ representing the flow conversation equations where $\mathbf{V}$ is a positive semi-definite matrix. Moreover, despite the square root, the objective is convex, as can be seen by writing

$$\sqrt{\mathbf{x}^T\mathbf{V}\mathbf{x}} = \left\| \mathbf{V}^{\frac{1}{2}}\mathbf{x} \right\|_2$$

and applying the triangle inequality.

The proposed algorithm exploits the relationship between the MSDMCF and the MVMCF, with the latter given by:

$$\begin{array}{lll} \min\limits_{\mathbf{x}} & \displaystyle\sum_{(i,j)\in\mathcal{A}} E[c_{ij}]x_{ij} + \lambda \sum_{(i,j)\in\mathcal{A}} Var[c_{ij}]x_{ij}^2 & \\ \text{s.t.} & \displaystyle\sum_{(j,k)\in A} x_{jk} - \sum_{(i,j)\in\mathcal{A}} x_{ij} = d_j & \forall j \in \mathcal{N} \\ & 0 \leq x_{ij} \leq u_{ij} & \forall (i,j) \in \mathcal{A}. \end{array} \qquad \text{(MVMCF}(\lambda)\text{)}$$

or, compactly,

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & \boldsymbol{\mu}^T\mathbf{x} + \lambda\mathbf{x}^T\mathbf{V}\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{array} \qquad \text{(MVMCF}(\lambda)\text{)}$$

with nonnegative λ. Unlike the MSMCF, the MVMCF problem is separable by arc. Both of the problems are convex; however, the separability structure is exploitable by the solver and also other solution algorithms e.g., [20].

2.2.2 Proposed approach

We adopt a parametric search method to solve the MSDMCF problem. Given $\bar{\lambda}$, we show that there exists some λ, for which optimal solutions for the mean-variance problem with λ are also optimal for the mean-standard deviation problem with $\bar{\lambda}$. In the remainder of the chapter, we will refer to this parameter that produces the optimal solution to MSDMCF($\bar{\lambda}$) as λ^*. Our approach is similar in spirit to methods that have previously been applied for the mean-standard deviation shortest path problem (MSSPP) [49], [118]. However, since the MSDMCF is a continuous optimization problem, rather than a combinatorial optimization problem, existing results on MSSPP do not directly apply to the problem studied in this chapter.

We first derive the relationship between the two weight parameters of these problems, which will guide the search. By applying root-finding methods to the function that defines this relationship, we find λ^* iteratively. To this end, we propose three algorithms, one based on bisection (BSC), one based on the Newton-Raphson (NR) method, and another using a combination of the two (NR-BSC). In order to obtain derivative information required for the NR algorithm, we perform sensitivity analysis on the solution of the MVMCF problem.

All of the results can be extended to a more general class of MCF problems — the generalized non-separable parametric MCF (GNPMCF), shown below:

$$\min\{\mu(\mathbf{x}) + \bar{\lambda} g(v(\mathbf{x})) : \mathbf{x} \in \mathcal{X}\}.$$

where $\mathcal{X}$ represents the MCF problem feasible set, μ and v are separable and differentiable functions, g is a strictly monotone, increasing and differentiable function, with the composition $g \circ v(\mathbf{x}) = g(v(\mathbf{x}))$ convex. In such cases, we can transform the function $g(\cdot)$ such that the problem becomes additive, therefore simpler. The same procedure proposed for the mean-standard deviation model can then be used to solve the GNPMCF problems with the stated assumptions above. Our main contributions are as follows:

1. We prove that the optimal solution to the MSDMCF problem is also optimal to the MVMCF problem for a particular chosen weight parameter. We also show that the converse of this claim is true, unlike the mean-standard deviation shortest path problem.

2. By deriving a key equation characterizing the relationship between the optimal solutions of the two problems, we develop three algorithms for finding the particular weight parameter λ^* to the MVMCF problem for which the optimal solution is also optimal to the MSDMCF problem for a given $\bar{\lambda}$.

3. We further show that all of our results can be extended to a more general class of MCF problems.

This model differs from the bi-objective MCF literature [55, 83, 89, 41, 86, 27, 28, 69, 87] in two aspects. The bi-objective MCF research mentioned above primarily focuses on two linear objectives whereas we have a non-separable non-linear component in our objective function. A key focus of the bi-objective

MCF literature is determining the non-dominated solution set. In our model, the two objectives can be collapsed into a single objective using a weight parameter, and we do not attempt to find the set of non-dominated solutions.

2.2.3 Relevance to the MVMCF

In this section, we will show that given an instance of the MSDMCF, there exists λ^* for which the optimal solution for the MVMCF(λ^*) is also optimal for the MSDMCF problem. The proof for this claim relies on the Karush-Kuhn-Tucker (KKT) necessary conditions. Therefore, we first derive these conditions for both problems below.

Let $\boldsymbol{\ell}(\mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{x}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{x} - \mathbf{u}$. The feasible solution sets for the two problems are identical since their constraints are the same. Then the complementary slackness, primal feasibility, and dual feasibility conditions for both problems are given by:

$$\begin{array}{l} \eta_{ij} h_{ij}(\mathbf{x}) = 0 \quad \forall (i,j) \in \mathcal{A} \\ h_{ij}(\mathbf{x}) \leq 0 \quad \forall (i,j) \in \mathcal{A} \\ \ell_i(\mathbf{x}) = 0 \quad \forall i \in \mathcal{N} \\ \eta_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A} \\ p_i \quad \text{free} \quad \forall i \in \mathcal{N} \end{array} \tag{2.1}$$

where η_{ij} and p_i are the dual variables for the capacity and flow balance constraints, respectively. Next, the stationary conditions are:

$$\mathbf{0} = \nabla_{\mathbf{x}} \left(\boldsymbol{\mu}^T \mathbf{x} + \bar{\lambda} \sqrt{\mathbf{x}^T \mathbf{V} \mathbf{x}} + \sum_{i \in \mathcal{N}} p_i \ell_i(\mathbf{x}) + \sum_{(i,j) \in \mathcal{A}} \eta_{ij} h_{ij}(\mathbf{x}) \right), \tag{2.2}$$

$$\mathbf{0} = \nabla_{\mathbf{x}} \left(\boldsymbol{\mu}^T \mathbf{x} + \lambda \mathbf{x}^T \mathbf{V} \mathbf{x} + \sum_{i \in \mathcal{N}} p_i \ell_i(\mathbf{x}) + \sum_{(i,j) \in \mathcal{A}} \eta_{ij} h_{ij}(\mathbf{x}) \right). \tag{2.3}$$

Equations (2.1) & (2.2) and (2.1) & (2.3) are the necessary conditions for optimality for MSDMCF($\bar{\lambda}$) and MVMCF(λ), respectively. Since the objective functions are also convex, these necessary conditions are also sufficient [11]. Our main result now follows.

Proposition 1. *Let* $\mathbf{x}(\lambda)$ *denote an optimal solution vector to the MVMCF(*λ*) problem. This vector is also optimal to MSDMCF(*$\bar{\lambda}$*), if* $\bar{\lambda}$ *satisfies*

$$\bar{\lambda} = 2\lambda\sqrt{\mathbf{x}(\lambda)^T\mathbf{V}\mathbf{x}(\lambda)}. \tag{2.4}$$

Proof. The vector $\mathbf{x}(\lambda)$ must satisfy the KKT necessary conditions for MVMCF(λ) as it is an optimal solution. The constraint system for both problems is the same, and hence it is immediate that $\mathbf{x}(\lambda)$ will satisfy the conditions (2.1) for MSDMCF($\bar{\lambda}$).

As $\mathbf{x}(\lambda)$ satisfies the necessary conditions (2.3) for MVMCF(λ), then we must have

$$-\mu - \sum_{(i,j)\in\mathcal{A}} \eta_{ij}(\lambda)\nabla_{\mathbf{x}}h_{ij}(\mathbf{x}(\lambda)) - \sum_{i\in\mathcal{N}} p_i(\lambda)\nabla_{\mathbf{x}}l_i(\mathbf{x}(\lambda)) = \lambda 2\mathbf{V}\mathbf{x}(\lambda). \tag{2.5}$$

Similarly $\mathbf{x}(\lambda)$ also satisfies (2.2), then

$$-\mu - \sum_{(i,j)\in\mathcal{A}} \eta_{ij}(\lambda)\nabla_{\mathbf{x}}h_{ij}(\mathbf{x}(\lambda)) - \sum_{i\in\mathcal{N}} p_i(\lambda)\nabla_{\mathbf{x}}l_i(\mathbf{x}(\lambda)) = \frac{\bar{\lambda}\mathbf{V}\mathbf{x}(\lambda)}{\sqrt{\mathbf{x}(\lambda)^T\mathbf{V}\mathbf{x}(\lambda)}}, \tag{2.6}$$

Combining the right hand side of both equation (2.5) and (2.6), we get

$$\bar{\lambda} = 2\lambda\sqrt{\mathbf{x}(\lambda)^T\mathbf{V}\mathbf{x}(\lambda)}.$$

Therefore $\mathbf{x}(\lambda)$ satisfies the KKT necessary conditions for MSMCF($\bar{\lambda}$) with λ satisfying (2.2.3). Since the MSMCF($\bar{\lambda}$) objective function is convex, the KKT necessary conditions are also sufficient for optimality.

□

A similar argument gives the reverse direction;

Remark 1. *If a vector* $\mathbf{x}(\bar{\lambda})$ *is optimal for MSDMCF(*$\bar{\lambda}$*), then it is also optimal for MVMCF(*λ*) with*

$$\lambda = \frac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\bar{\lambda})^T\mathbf{V}\mathbf{x}(\bar{\lambda})}}. \tag{2.7}$$

Remark 1 distinguishes this setting from the mean-standard deviation shortest path problem, where the analogous statement fails [49]. Therefore, the MVMCF and the MSDMCF have a closer relationship than the corresponding shortest path problems.

2.3 Algorithm

We use equation (2.2.3) to devise algorithms to solve the MSDMCF($\bar{\lambda}$) problem. If we can identify a weight parameter λ such that

$$f(\lambda) = \lambda - \frac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T\mathbf{V}\mathbf{x}(\lambda)}} = 0, \tag{2.8}$$

solving the MVMCF(λ) will solve the original MSDMCF($\bar{\lambda}$). Therefore, the MSDMCF($\bar{\lambda}$) problem reduces to the problem of finding a root of $f(\lambda)$.

2.3.1 Bisection

A straightforward method to find the root is bisection. In order to use this method, we first need to show that there exists at least one root for $f(\lambda)$ in the domain $\lambda \in [0, \infty)$. To this end, we show that the function takes values of opposite signs when evaluated at the endpoints of the domain, and it is continuous for all $\lambda \in [0, \infty)$. It is trivial to see that it takes a nonpositive value as λ approaches 0, since $\bar{\lambda}$ is nonnegative and all feasible solutions are assumed to have positive variance. Moreover, the variance term in the denominator in $f(\lambda)$ is finite for any value of λ. Let Var_ℓ represent the minimum variance of any feasible flow[1]. Then the variance term is bounded below by Var_ℓ, which is positive by assumption. We can thus conclude that $f(\lambda)$ takes a positive value as λ approaches ∞ since the negative term is finite. A finite upper endpoint of the interval can simply be found by doubling an initial guess λ until $f(\lambda) \geq 0$. Further, $f(\lambda)$ is negative for $\lambda = 0$, so we can set the lower endpoint of the interval to 0.

Finally, since the objective function of MVMCF(λ) is continuous in both $\mathbf{x}$ and λ and strictly convex in $\mathbf{x}$, the minimizer $\mathbf{x}(\lambda)$ is well-defined and continuous in λ by the Maximum Theorem [4]. Therefore $f(\lambda)$ is also continuous, at least one root exists in $[0, \infty)$.

Furthermore, this root is unique, as we show in the next two results. In these results, we use $M(\lambda) \equiv \boldsymbol{\mu}^T \mathbf{x}(\lambda)$ to refer to the mean cost of the optimal

[1]The minimum variance flow's variance cost Var_ℓ can be obtained by solving MCF where the objective only consists of the variance criterion or effectively setting λ to ∞

solution to MVMCF(λ), and $V(\lambda) \equiv \mathbf{x}(\lambda)^T \mathbf{V}\mathbf{x}(\lambda)$.

Lemma 1. *The mean cost $M(\lambda)$ of the optimal solution is nondecreasing in λ, while $V(\lambda)$ is nonincreasing.*

Proof. Let λ_1 and λ_2 be distinct weighting parameters. Without loss of generality we can assume $0 \leq \lambda_1 < \lambda_2$. Since $\mathbf{x}(\lambda_1)$ minimizes $\boldsymbol{\mu}^T\mathbf{x} + \lambda_1 \mathbf{x}^T\mathbf{V}\mathbf{x}$, we have

$$M(\lambda_1) + \lambda_1 V(\lambda_1) \leq M(\lambda_2) + \lambda_1 V(\lambda_2)\,. \tag{2.9}$$

Similarly,

$$M(\lambda_2) + \lambda_2 V(\lambda_2) \leq M(\lambda_1) + \lambda_2 V(\lambda_1)\,. \tag{2.10}$$

Multiplying inequality (2.9) by λ_2, inequality (2.10) by λ_1, and subtracting gives

$$(\lambda_2 - \lambda_1) M(\lambda_1) \leq (\lambda_2 - \lambda_1) M(\lambda_2) \tag{2.11}$$

whence it follows that $M(\lambda_1) \leq M(\lambda_2)$, that is, M is nondecreasing.

Furthermore, since $M(\lambda_1) \leq M(\lambda_2)$, inequality (2.10) can only be satisfied if $V(\lambda_1) \geq V(\lambda_2)$, showing that V is nonincreasing and completing the lemma. □

Proposition 2. *The function defined in equation (2.8) has exactly one root.*

Proof. The above discussion establishes the existence of a root; we now show that this root is unique.

By contradiction, assume that $f(\lambda_1) = f(\lambda_2) = 0$ for some $\lambda_1 \neq \lambda_2$. Proposition 1 ensures that $\mathbf{x_1} \equiv \mathbf{x}(\lambda_1)$ and $\mathbf{x_2} \equiv \mathbf{x}(\lambda_2)$ are both optimal to MSDNFP($\bar{\lambda}$). Since this problem is convex, the set of optimal solutions is convex, and $(1-\alpha)\mathbf{x_1} + \alpha\mathbf{x_2}$ is optimal as well for any $\alpha \in [0, 1]$.

Since all these solutions are optimal, they all have equal objective function values, so

$$\boldsymbol{\mu}^T((1-\alpha)\mathbf{x_1}+\alpha\mathbf{x_2})+\bar{\lambda}\sqrt{((1-\alpha)\mathbf{x_1}+\alpha\mathbf{x_2})^T\mathbf{V}((1-\alpha)\mathbf{x_1}+\alpha\mathbf{x_2})} \quad (2.12)$$

does not depend on α. The first term in (2.12) is linear in α; since the sum is constant, this implies that the second term must also be linear, that is,

$$\sqrt{((1-\alpha)\mathbf{x_1}+\alpha\mathbf{x_2})^T\mathbf{V}((1-\alpha)\mathbf{x_1}+\alpha\mathbf{x_2})}=(1-\alpha)\sqrt{\mathbf{x_1}^T\mathbf{V}\mathbf{x_1}}+\alpha\sqrt{\mathbf{x_2}^T\mathbf{V}\mathbf{x_2}} \quad (2.13)$$

Squaring both sides gives

$$\begin{aligned}&(1-\alpha)^2\mathbf{x_1}^T\mathbf{V}\mathbf{x_1}+\alpha^2\mathbf{x_2}^T\mathbf{V}\mathbf{x_2}+2\alpha(1-\alpha)\mathbf{x_1}^T\mathbf{V}\mathbf{x_2}\\&=(1-\alpha)^2\mathbf{x_1}^T\mathbf{V}\mathbf{x_1}+\alpha^2\mathbf{x_2}^T\mathbf{V}\mathbf{x_2}+2\alpha(1-\alpha)\sqrt{(\mathbf{x_1}^T\mathbf{V}\mathbf{x_1})(\mathbf{x_2}^T\mathbf{V}\mathbf{x_2})}\,, \quad (2.14)\end{aligned}$$

since $\mathbf{V}$ is symmetric or, after simplifying,

$$\mathbf{x_1}^T\mathbf{V}\mathbf{x_2}=\sqrt{(\mathbf{x_1}^T\mathbf{V}\mathbf{x_1})(\mathbf{x_2}^T\mathbf{V}\mathbf{x_2})}\,, \quad (2.15)$$

But $\mathbf{V}$ is also positive definite, so $\mathbf{x}^T\mathbf{V}\mathbf{y}$ forms an inner product space. The Cauchy-Schwarz inequality therefore asserts that (2.15) holds only if $\mathbf{x_1}$ and $\mathbf{x_2}$ are linearly dependent, that is, if $\mathbf{x_1}=\beta\mathbf{x_2}$ for some $\beta\neq 0$. The only choice that satisfies the flow conservation constraints for both $\mathbf{x_1}$ and $\mathbf{x_2}$ is $\beta=1$; therefore $\mathbf{x_1}=\mathbf{x_2}$, and the solutions corresponding to λ_1 and λ_2 are in fact identical.

As a result, $V(\lambda_1)=V(\lambda_2)$. But $f(\lambda)=\lambda-\bar{\lambda}/(2\sqrt{V(\lambda)})$, so $f(\lambda_1)=f(\lambda_2)=0$ would imply $\lambda_1=\lambda_2$, a contradiction. □

2.3.2 Finding an initial λ

It is possible to reduce the search space for the root-finding algorithms by finding a smaller initial interval which includes the root. Let Var_h represent an upper bound on the variance of optimal solutions with any λ to the MVMCF(λ) problem. An efficient way to obtain such a bound is to solve a linear minimum cost flow problem with the mean costs, essentially setting λ to 0, and setting Var_h to be the variance of such a solution. This is an upper bound on the variance of the optimal solution. Therefore, any λ with

$$\lambda \leq \frac{\bar{\lambda}}{2\sqrt{Var_h}},$$

also satisfies

$$\lambda \leq \frac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)}},$$

where $\mathbf{x}$ is obtained from solving the MVMCF(λ) at that λ. Hence, we can set the lower bound for the interval that includes the root to $\lambda_{low} = \bar{\lambda}/2\sqrt{Var_h}$.

It is also possible to find an upper bound on the interval in a similar fashion. Doing so would require solving a quadratic MCF. There is an alternative, simpler procedure which provides a looser upper bound: if we set λ to $\bar{\lambda}/2$, and if $\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)} > 1$ then $f(\lambda) > 0$. By changing units one can always satisfy the condition $\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)} > 1$, and re-solve the problem after scaling. Therefore we can set $\lambda_{high} = \bar{\lambda}/2$.

However, our computational experiments show that the former approach performs better. Specifically, let Var_ℓ represent a lower bound on the variance of optimal solutions with any λ to the MVMCF(λ) problem. In order to obtain such a bound, one needs to solve a quadratic minimum cost flow problem with

	Initialization		Time (s)		Iteration #	
	λ_{low}	λ_{high}	BSC	NR	BSC	NR
Naive	0	$\bar{\lambda}/2$	154.85	19.15	21	2
Custom	$\bar{\lambda}/2\sqrt{Var_h}$	$\bar{\lambda}/2\sqrt{Var_\ell}$	26.77	11.63	1	1

Table 2.1: Initialization procedure benefits - on a network with 4096 nodes and average degree of 64.

the variance term alone. Var_ℓ will lead to the maximum possible value for the negative term in $f(\lambda)$. Then, any λ with

$$\lambda \geq \frac{\bar{\lambda}}{2\sqrt{Var_\ell}},$$

also satisfies

$$\lambda \geq \frac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)}},$$

where the $\mathbf{x}$ is obtained from solving the MVMCF(λ) at that λ. Hence, we can set the upper bound for the interval that includes the root to $\lambda_{high} = \bar{\lambda}/2\sqrt{Var_\ell}$

Table 2.1 illustrates the benefits of using custom bounds found with the procedure described in Subsection 2.3.2. It compares iteration numbers and the running time of the algorithms for both naive and custom bounds, on a dense network with 4096 nodes and degree 64. The custom initialization helps the algorithms to start very close to $\lambda^\star$, and therefore iteration numbers and running times are much lower.

2.3.3 Newton's algorithm

Although the bisection method is guaranteed to converge, it only has a linear convergence rate and may need many iterations to converge, each of which requires solving a mean-variance problem. An alternative is to seek a root for

Figure 2.1: Pseudocode for BSC ($\bar{\lambda}$, *TOL)*

$\tilde{\mathbf{x}}_h \leftarrow MVMCF(\lambda = 0), \tilde{\mathbf{x}}_l \leftarrow MVMCF(\lambda = \infty)$;
$Var_h \leftarrow \tilde{\mathbf{x}}_h \mathbf{V} \tilde{\mathbf{x}}_h, Var_\ell \leftarrow \tilde{\mathbf{x}}_l \mathbf{V} \tilde{\mathbf{x}}_l$;
$\lambda_{low} \leftarrow \bar{\lambda}/2\sqrt{Var_h}, \quad \lambda_{high} \leftarrow \bar{\lambda}/2\sqrt{Var_\ell}$;
$Found \leftarrow False$;
while *not* $Found$ **do**
  $\lambda \leftarrow (\lambda_{high} + \lambda_{low})/2$;
  $\mathbf{x}(\lambda) \leftarrow \arg\min(\text{MVMCF}(\lambda))$;
  $f(\lambda) = \lambda - \dfrac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)}}$;
  if $|f(\lambda)| \leq TOL$ **then**
    $Found \leftarrow True$;
  else
    if $f(\lambda) > 0$ **then**
      $\lambda_{high} \leftarrow \lambda$;
    else
      $\lambda_{low} \leftarrow \lambda$;

$f(\lambda)$ with the Newton-Raphson method. This method is simple to implement, and under certain conditions has quadratic convergence [11]. However, this method requires calculating the derivative of $f(\lambda)$, which involves solving an auxiliary optimization problem. The Newton update for $f(\lambda)$ is given by:

$$\lambda_{n+1} = \left[\lambda_n - \frac{f(\lambda_n)}{f'(\lambda_n)}\right]^+, \tag{2.16}$$

where the $[\cdot]^+$ operator outputs 0 when the input is negative and otherwise does not modify the input. Let $\boldsymbol{\xi}$ represent the vector of derivatives of the optimal solution $\mathbf{x}$ with respect to λ, $\boldsymbol{\xi} = d\mathbf{x}/d\lambda$. We can then write $f'(\lambda)$ as

$$f'(\lambda) = 1 + \frac{\bar{\lambda}\mathbf{x}(\lambda)^T\mathbf{V}\boldsymbol{\xi}}{2(\mathbf{x}(\lambda)^T\mathbf{V}\mathbf{x}(\lambda))^{3/2}}. \tag{2.17}$$

In this section, we identify the derivatives $\boldsymbol{\xi}$ using sensitivity analysis, using similar techniques as in Boyles [16] and Jafari & Boyles [46]. The derivative of the optimal solution vector with respect to the weight parameter λ can be interpreted as the sensitivity of the solution to changes in λ.

Let $C_{ij}(x_{ij}) = E[c_{ij}]x_{ij} + \lambda Var[c_{ij}]x_{ij}^2$ represent the cost of arc ij, and $C'_{ij} = E[c_{ij}]+2\lambda Var[c_{ij}]$ its derivative.. Then, using $\mathbf{p}$ to represent the dual variables for the flow conservation constraints, the Karush-Kuhn-Tucker conditions for

MVMCF(λ) require

$$C'_{ij}(x_{ij}) + p_i - p_j \geq 0 \qquad \forall (i,j) : x_{ij} = 0 \tag{2.18a}$$

$$C'_{ij}(x_{ij}) + p_i - p_j = 0 \qquad \forall (i,j) : 0 < x_{ij} < u_{ij} \tag{2.18b}$$

$$C'_{ij}(x_{ij}) + p_i - p_j \leq 0 \qquad \forall (i,j) : x_{ij} = u_{ij} \tag{2.18c}$$

$$\sum_{(j,k)\in\mathcal{A}} x_{jk} - \sum_{(i,j)\in\mathcal{A}} x_{ij} = d_j \quad \forall j \in \mathcal{N} \tag{2.18d}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{2.18e}$$

that hold at optimality. Let $\mathcal{J}^*$ represent the set of arcs with $C'_{ij}(x_{ij})+p_i-p_j = 0$, and further partition $\mathcal{J}^*$ into sets $\mathcal{J}^*_+$, $\mathcal{J}^*_0$, and $\mathcal{J}^*_-$ according to whether $x_{ij} = 0$, $0 < x_{ij} < u_{ij}$, or $x_{ij} = u_{ij}$ at optimality, respectively. (The sets $\mathcal{J}^*_+$ and $\mathcal{J}^*_-$ are empty unless the optimal solution is degenerate.)

Let $\boldsymbol{\varphi}$ represent the marginal change in $\mathbf{p}$, when the weight parameter λ is perturbed, $\boldsymbol{\varphi} = d\mathbf{p}/d\lambda$. Differentiating the KKT conditions with respect to λ, we have

$$2Var[c_{ij}](x_{ij} + \lambda\xi_{ij}) + \varphi_i - \varphi_j = 0 \quad \forall (i,j) \in \mathcal{J}^* \tag{2.19a}$$

$$\sum_{(j,k)\in\mathcal{J}^*} \xi_{jk} - \sum_{(i,j)\in\mathcal{J}^*} \xi_{ij} = 0 \qquad \forall j \in \mathcal{N} \tag{2.19b}$$

$$\xi_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{J}^*_+ \tag{2.19c}$$

$$\xi_{ij} \quad free \qquad \forall (i,j) \in \mathcal{J}^*_0 \tag{2.19d}$$

$$\xi_{ij} \leq 0 \qquad \forall (i,j) \in \mathcal{J}^*_- \tag{2.19e}$$

$$\xi_{ij} = 0 \qquad \forall (i,j) \in \mathcal{A}\backslash\mathcal{J}^* \tag{2.19f}$$

which show how the optimal $\mathbf{x}$ and $\mathbf{p}$ change with λ.

A solution $\boldsymbol{\xi}$ to this problem could be obtained by solving this set of linear equations and inequalities; indeed, in the typical case where the optimal solution is nondegenerate it is simply a linear system of equations that can be

solved using standard techniques. Regardless of degeneracy, we can recognize this system as the optimality conditions of the following quadratic program:

$$
\begin{aligned}
\min_{\boldsymbol{\xi}} \quad & 2 \sum_{(i,j)\in\mathcal{J}^*} Var[c_{ij}] x_{ij} \xi_{ij} + \lambda \sum_{(i,j)\in\mathcal{J}^*} Var[c_{ij}] \xi_{ij}^2 \\
\text{s.t.} \quad & \sum_{(j,k)\in\mathcal{J}^*} \xi_{jk} - \sum_{(i,j)\in\mathcal{J}^*} \xi_{ij} = 0 \qquad \forall j \in \mathcal{N} \\
& \xi_{ij} \geq 0 && \forall (i,j) \in \mathcal{J}^*_+ \\
& \xi_{ij} \quad free && \forall (i,j) \in \mathcal{J}^*_0 \\
& \xi_{ij} \leq 0 && \forall (i,j) \in \mathcal{J}^*_- \\
& \xi_{ij} = 0 && \forall (i,j) \in \mathcal{A} \backslash \mathcal{J}^*
\end{aligned}
\qquad (\Delta(\lambda, \mathbf{x}))
$$

Note that $\mathbf{x}$ is a parameter in this formulation, the optimal solution of the MVMCF(λ), and that $\boldsymbol{\xi}$ is the only decision variable. Furthermore, this optimization problem is very nearly an instance of MVMCF, restricted to the links in the set $\mathcal{J}^*$, without capacities, with different sign constraints, and with mean link costs replaced with $2Var[c_{ij}]x_{ij}$.

There are advantages to obtaining $\boldsymbol{\xi}$ by solving this MVMCF variant, rather than solving the linear system directly. Using existing algorithms for MVMCF exploits problem structure, and our experiments showed that they were faster and more numerically stable, and furthermore provide a natural framework for identifying solutions at a customizable level of precision (at early iterations, high-precision solutions to these derivatives are likely not necessary).

The pseudocode in Figure 2.2 outlines the Newton-Raphson-based search procedure which uses the flow sensitivity procedure to determine the derivatives. Since evaluating $f(\lambda)$ requires solving an optimization problem, it is unclear what conditions guarantee convergence of the algorithm. In the next subsection, we provide a fail-safe to alleviate the lack of convergence guarantee for

Figure 2.2: Pseudocode for NR ($\bar{\lambda}$, *TOL)*

$\tilde{\mathbf{x}}_h \leftarrow MVMCF(\lambda = 0)$;
$Var_h \leftarrow \tilde{\mathbf{x}}_h \mathbf{V} \tilde{\mathbf{x}}_h$;
$\lambda \leftarrow \bar{\lambda}/2\sqrt{Var_h}$;
while ***not*** $Found$ **do**
  $\mathbf{x}(\lambda) \leftarrow \arg\min(\text{MVMCF}(\lambda))$;
  $f(\lambda) = \lambda - \dfrac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)}}$;
  if $|f(\lambda)| \leq TOL$ **then**
    $Found \leftarrow True$;
  else
    $\boldsymbol{\xi} \leftarrow \arg\min \Delta(\lambda, \mathbf{x})$;
    $f'(\lambda) = 1 + \dfrac{\bar{\lambda} \mathbf{x}(\lambda)^T \mathbf{V} \boldsymbol{\xi}}{2(\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda))^{3/2}}$;
    $\lambda = \left[\lambda - \dfrac{f(\lambda)}{f'(\lambda)} \right]^+$;

the pure Newton algorithm. We note that the method converged for all the test instances in our experiments, despite the lack of convergence proof.

2.3.4 Hybrid algorithm

The third algorithm (NR-BSC) is a hybrid of the first two, primarily using a Newton step size with bisection as a fallback to ensure convergence, as in Press et al. [82]. Specifically, we switch to a bisection step whenever the current Newton-Raphson step suggests a solution that is out of the bracket, or whenever the bracket size is not reducing rapidly enough. The resulting procedure has a best-case quadratic convergence rate and worst-case linear convergence rate.

It is easy to check for the first condition to see if the step would take the

solution out of bounds. However, to check the second condition, a definition for 'rapidly enough' is needed. In our implementation, we use a simple condition, and check if $|f(\lambda)|$ is smaller than the $|f(\lambda)|$ in the previous iteration. This approach prevents possible divergent behaviors in the pure NR algorithm. The pseudocode of the algorithm is provided in Figure 2.3.

2.4 Generalization to non-separable parametric convex cost problems

In this section, we discuss the applicability of the proposed methods to certain generalizations. First, we note that the (BSC) method is applicable more generally, to not only for the MCF problems but for general case of non-separable convex cost problem;

$$\min_{\mathbf{x}\in\mathcal{X}} \quad \mu(\mathbf{x}) + \bar{\lambda} g\left(v(\mathbf{x})\right) \tag{2.20}$$

with bounded linear constraint system $\mathcal{X}$, where μ and v are a separable and differentiable function of $\mathbf{x}$, g is a strictly monotone, increasing and differentiable function, and lastly the composition $g \circ v$ is convex. Such a function is necessarily invertible, and applying g^{-1} as a transformation will yield the convex separable problem:

$$\min_{\mathbf{x}\in\mathcal{X}} \quad \mu(\mathbf{x}) + \bar{\lambda} v(\mathbf{x}) \tag{2.21}$$

By following the same procedure as in the proof of Proposition 1, we can find a relation between the two problems.

Figure 2.3: Pseudocode for NR-BSC ($\bar{\lambda}$, *TOL)*

$\tilde{\mathbf{x}}_h \leftarrow MVMCF(\lambda = 0)$;
$Var_h \leftarrow \tilde{\mathbf{x}}_h \mathbf{V} \tilde{\mathbf{x}}_h$;
$\lambda_{low} \leftarrow \bar{\lambda}/2\sqrt{Var_h}$;
while *not* $Found$ **do**
  $\mathbf{x}(\lambda) \leftarrow \arg\min(\text{MVMCF}(\lambda))$;
  $f(\lambda) = \lambda - \dfrac{\bar{\lambda}}{2\sqrt{\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda)}}$;
  if $|f(\lambda)| \leq TOL$ **then**
    $Found \leftarrow True$
  else
    if $|f(\lambda)| \leq |f(\lambda_{prev})|$ **then**
      $\boldsymbol{\xi} \leftarrow \arg\min \Delta(\lambda, \mathbf{x})$;
      $f'(\lambda) = 1 + \dfrac{\bar{\lambda} \mathbf{x}(\lambda)^T \mathbf{V} \boldsymbol{\xi}}{2(\mathbf{x}(\lambda)^T \mathbf{V} \mathbf{x}(\lambda))^{3/2}}$;
      $f(\lambda_{prev}) \leftarrow f(\lambda)$;
      $\lambda_{prev} \leftarrow \lambda$;
      $\lambda \leftarrow \lambda - \dfrac{f(\lambda)}{f'(\lambda)}$;
      if $\lambda_{low} < \lambda < \lambda_{high}$ **then**
        if $f(\lambda) > 0$ **then**
          $\lambda_{high} \leftarrow \lambda$;
        else
          $\lambda_{low} \leftarrow \lambda$
      else
        Update the bounds using λ_{prev} **and perform Bisection step**
    else
      Update the bounds using λ_{prev} **and perform Bisection step**

$$\lambda = \bar{\lambda}\frac{\partial g\left(v(\mathbf{x})\right)}{\partial v(\mathbf{x})}. \tag{2.22}$$

In the above equations, the vector $\mathbf{x}$ is the solution vector obtained by solving (2.21) at λ — in order to not the clutter the notation, instead of $\mathbf{x}(\lambda)$ we referred to it as $\mathbf{x}$. Note that differentiability of both g and v are necessary for Proposition 1 to follow. Moreover, as $g \circ v$ is convex by assumption, the KKT necessary conditions are also sufficient for optimality of (2.20) and therefore the optimal solution for (2.21) with λ satisfying (2.22) is also optimal for (2.20). By similar arguments made earlier, one can show that the function $f(\lambda)$ is continuous. Moreover, it takes values of opposite signs when evaluated at the endpoints of the domain. By assumption, g is monotonically increasing, and therefore the derivative is positive for any λ in the domain. Then, $f(\lambda)$ takes a nonpositive value as λ approaches 0. The criterion v term in $f(\lambda)$ is finite for any value of λ, so $f(\lambda)$ takes a positive value as λ approaches ∞, as the negative term will be finite with this assumption. If in addition, the function g is twice differentiable and we have an MCF problem, one can carry on the sensitivity analysis and also use the Newton-Raphson.

In practice, optimization problems of this form might arise when capturing the utilities with an exponential function. Other functions such as quadratic, Ackley, Brent, and Brown fall into the class of functions for g that satisfies the required conditions to use in this framework. We refer the reader to an extensive survey of benchmark functions [47] for more applicable functions that fall into this class.

It is also possible to arrive at this form starting from other optimization problems. For instance, the single criterion MCF problem — such as minimizing

only the expected value — with nonlinearly evaluated budget constraints

$$\begin{array}{ll} \min\limits_{\mathbf{x}} & \mu(\mathbf{x}) \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \\ & g\left(v(\mathbf{x})\right) \leq B, \end{array} \tag{2.23}$$

can be cast into the form

$$\begin{array}{lll} \min\limits_{\mathbf{x}} \max\limits_{\bar{\lambda} \geq 0} & \mu(\mathbf{x}) + \bar{\lambda}\left(g\left(v(\mathbf{x})\right)\right) - \bar{\lambda}B \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{array} \tag{2.24}$$

by Langrangianizing the budget constraints. For a given $\bar{\lambda}$, we then have an outer minimization problem which is of the form we consider in this section. Note that Langrangianizing the budget constraints will lead to a max min problem. Strong duality holds if there exists a feasible solution $\mathbf{x}$ for which $g(v(\mathbf{x})) < B$, as such point would be an interior point and so the Slater's condition will then hold. In this case, one can swap the max min problem into min max problem and end up with the optimization problem in (2.24).

2.5 Computational Experiments

In this section, we assess the performance of the proposed algorithms and compare them to the performance of the CPLEX solver. We compare the methods using the same benchmark suite, and thus provide intuition into their performance on networks with different characteristics, including, how dense the network is, how restricting are the capacities on the arcs. All of the computational experiments are performed on a quad-core 2.8 GHz computer

with 16 GB RAM. The code used for the computational experiments and analysis is provided at https://github.com/cangokalp/mean-std.

2.5.1 Benchmark networks

The performance of the methods are evaluated on the networks generated with the well-known random generator NETGEN [51]. We use the benchmark suite created in [50], which was designed to compare linear MCF solution methods.

In the NETGEN problem families, the arc costs and capacities are uniformly drawn from [1, 10,000] and [1, 1,000], respectively. There are approximately $\sqrt{n}$ supply and demand nodes, and the average supply per supply node is set to 1000.

There are four problem families created with above characteristics:

- **NETGEN-8.** Sparse networks, with average node outdegree of 8 ($m = 8n$).
- **NETGEN-SR.** Dense networks, with average node outdegree of $\sqrt{n}$ ($m \approx n\sqrt{n}$).
- **NETGEN-LO-8.** Same as NETGEN-8, except the average supply per supply node is 10.
- **NETGEN-LO-SR.** Same as NETGEN-SR, except the average supply per supply node is 10.

Arc capacities in NETGEN-LO-8 and NETGEN-LO-SR impose only loose bounds for feasible flows, as the average supply per supply node is small.

We use the arc costs in the instances as the mean arc costs $E[c_{ij}]$. We sample a coefficient of variation COV_{ij} for each link, drawn uniformly from $[0.15, 0.3]$,

and thus set the standard deviation as $\sigma_{ij} = COV_{ij}E[c_{ij}]$. This interval for COV_{ij} represents typical variation in transportation networks [7].

2.5.2 Comparison of algorithms

The reported running times for the algorithms NR and BSC include the time elapsed for finding the interval for λ. We do not report the hybrid algorithm in the tables and figures below as its performance is almost identical to the NR method since the "failsafe" bisection steps were rarely used. Both of the line search methods used convergence criterion of $TOL = 10^{-8}$. The MVMCF subproblems are solved using CPLEX solver. All comparisons were done using $\bar{\lambda} = 10$. We also address how the performance changes for different values of $\bar{\lambda}$ later in this section.

For each graph family, each method's performance was measured by seconds needed to achieve 0.01% "optimality gap" – the percentage gap between the method's objective and the best objective found by all three algorithms. The reported running times are averaged over 5 instances for each problem.

Tables 2–5 provide the absolute running times in seconds, and the best running times are bolded. Figures 4–7 provide corresponding plots, using logarithmic scales so the relative difference between methods is clearly apparent across all problem sizes tested.

In the tables, the size of the network is indicated by the number of nodes and the average degree per node in each row. NR method outperforms the other methods on every experiment. While the BSC method outperforms CPLEX on dense networks for smaller problem sizes, it has a worse trend than CPLEX in all cases. All of the methods' solution times increase by about an order of magnitude when the number of nodes is held fixed and the density of the

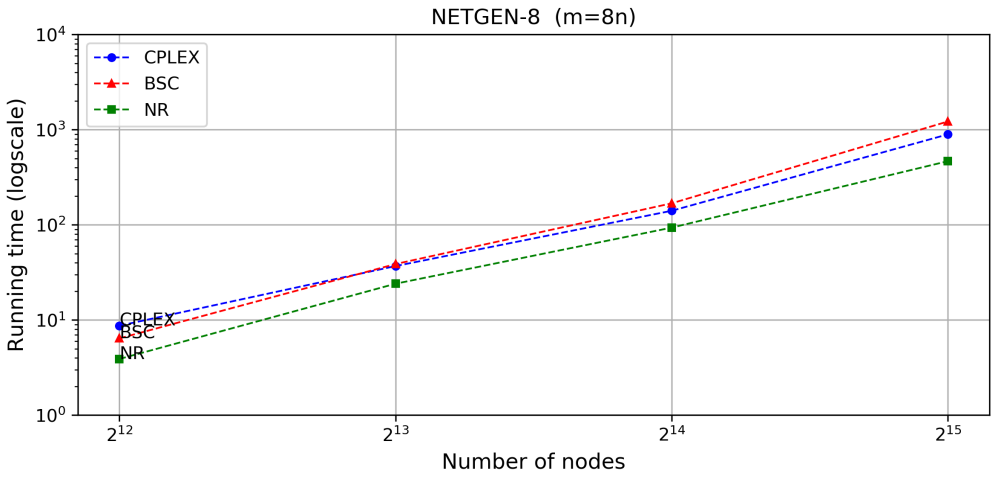

Figure 2.4: Comparison of the algorithms on NETGEN-8 families (logarithmic scale).

network increased.

Additionally, Tables 2–5 also provide the average number of iterations for the proposed algorithms to achieve the gap level. The NR method requires fewer iterations for all families except NETGEN-LO-SR. The solution time of the NR method is better than the BSC method, despite requiring more iterations for this family. This is due to each method requiring a different amount of time to find the initial λ. The time for each method to achieve its first objective includes only the time elapsed for finding initial λ. For the NR method, initial λ is set to λ_{low}, and to find this lower bound, a linear MCF problem needs to be solved. On the other hand, for the BSC method, initial λ is set to $(\lambda_{low} + \lambda_{high})/2$ which requires finding both the lower bound and the upper bound. The latter requires solving a quadratic MCF problem and thus is more costly.

Figure 2.8 presents the convergence behavior of the algorithms on the NETGEN-

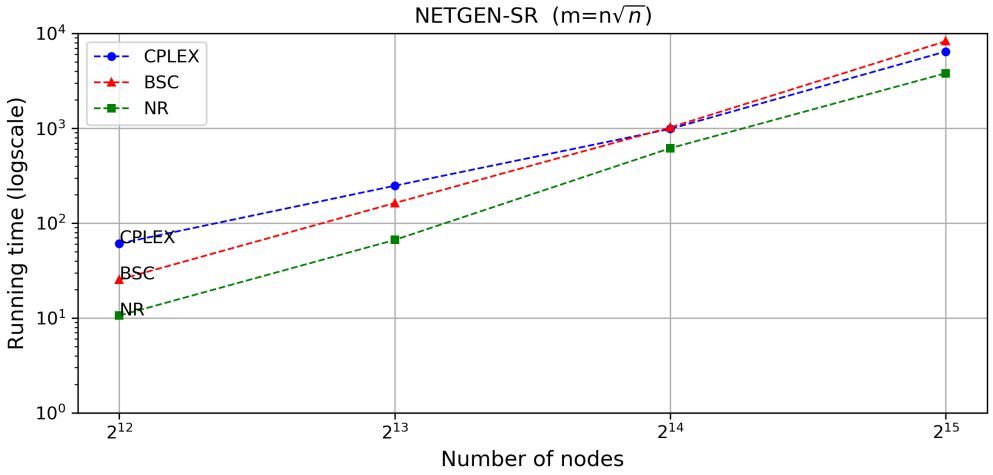

Figure 2.5: Comparison of the algorithms on NETGEN-SR families (logarithmic scale).

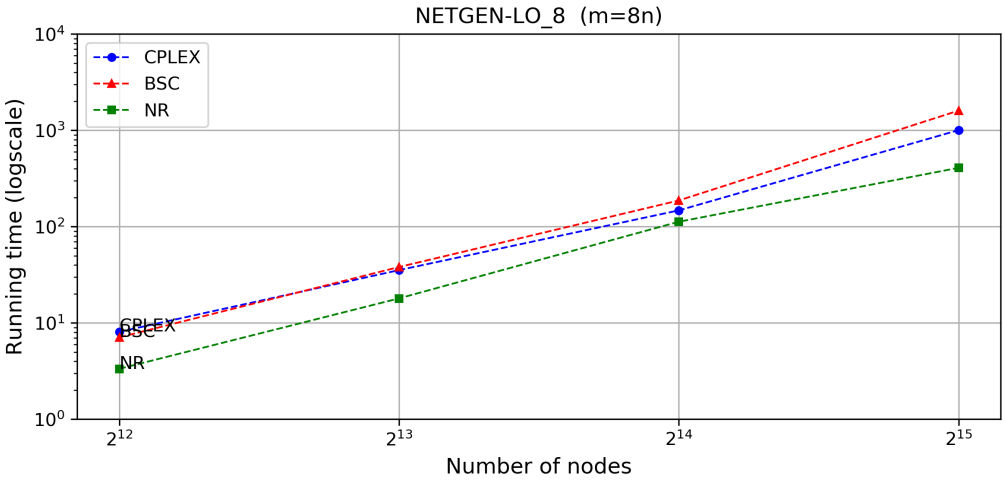

Figure 2.6: Comparison of the algorithms on NETGEN-LO-8 families (logarithmic scale).

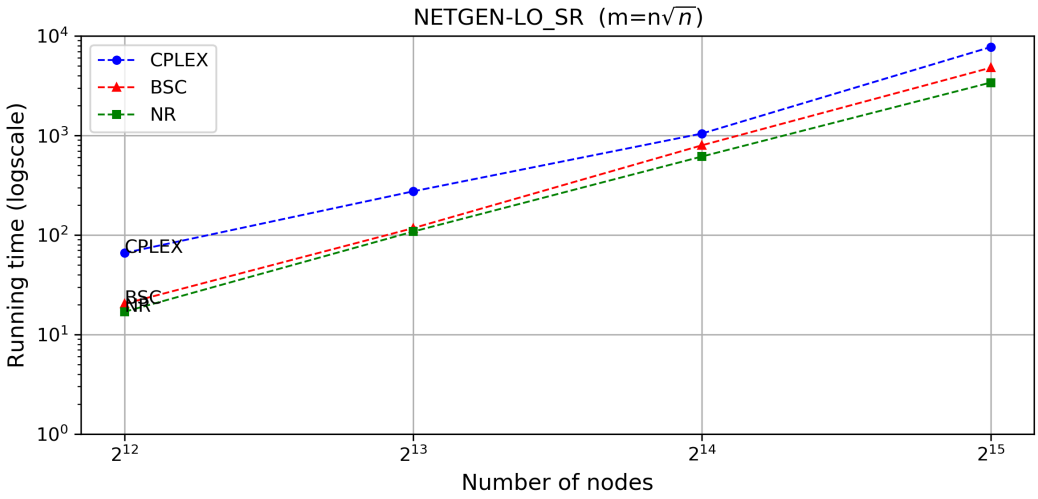

Figure 2.7: Comparison of the algorithms on NETGEN-LO-SR families (logarithmic scale).

Size		Time (s)			Avg. Iteration #	
n	deg	CPLEX	BSC	NR	NR	BSC
2^{12}	8	8.70	6.40	**3.89**	2.0	2.2
2^{13}	8	36.90	38.74	**24.06**	2.0	2.0
2^{14}	8	140.64	168.51	**93.55**	1.2	1.2
2^{15}	8	893.78	1220.60	**467.02**	1.0	1.0

Table 2.2: Comparison on NETGEN-8 instances.

Size		Time (s)			Avg. Iteration #	
n	deg	CPLEX	BSC	NR	BSC	NR
2^{12}	64	60.96	25.40	**10.60**	1.2	1.0
2^{13}	90	247.85	162.85	**66.53**	1.2	1.0
2^{14}	128	984.32	1019.80	**616.12**	1.2	1.2
2^{15}	181	6441.52	8265.37	**3804.74**	1.0	1.2

Table 2.3: Comparison on NETGEN-SR instances.

Size		Time (s)			Avg. Iteration #	
n	deg	CPLEX	BSC	NR	BSC	NR
2^{12}	8	8.08	7.08	**3.34**	2.0	2.0
2^{13}	8	35.38	38.11	**17.99**	2.0	1.8
2^{14}	8	147.63	187.12	**112.51**	2.0	1.6
2^{15}	8	1005.59	1604.47	**408.43**	2.0	1.0

Table 2.4: Comparison on NETGEN-LO-8 instances.

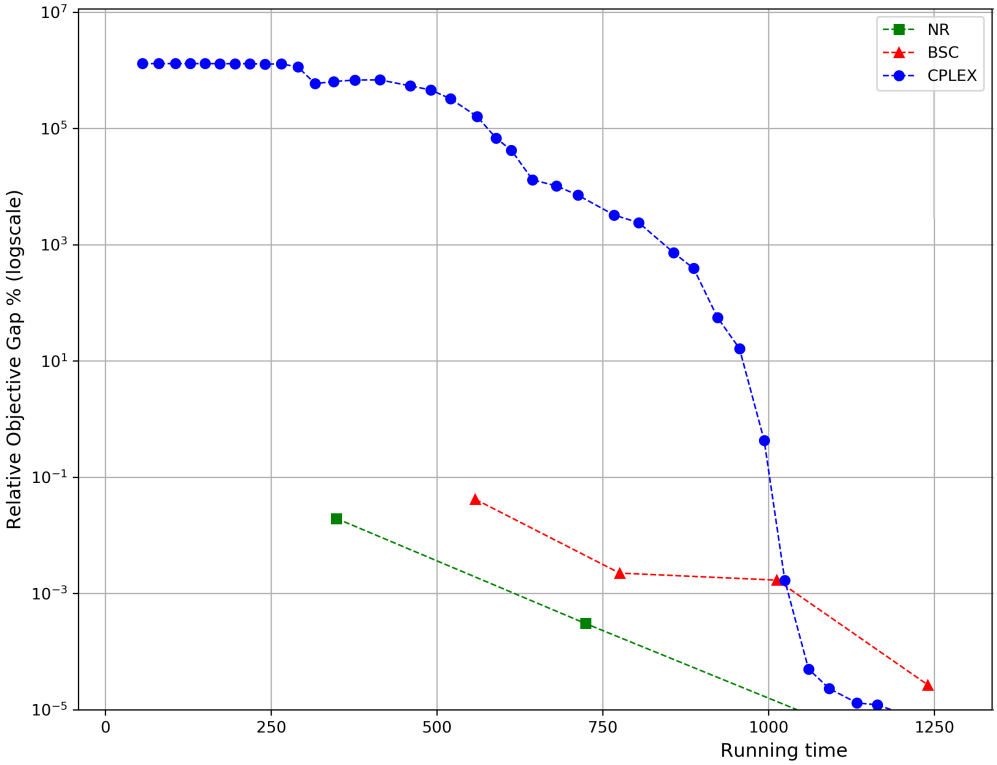

Figure 2.8: Convergence behavior.

Size		Time (s)			Avg. Iteration #	
n	deg	CPLEX	BSC	NR	BSC	NR
2^{12}	64	65.94	20.34	**16.91**	1.2	2.0
2^{13}	90	274.49	117.14	**108.20**	1.2	2.0
2^{14}	128	1044.87	796.01	**614.13**	2.0	1.8
2^{15}	181	7727.51	4789.31	**3397.46**	1.4	1.4

Table 2.5: Comparison on NETGEN-LO-SR instances.

		NETGEN-8				NETGEN-SR				NETGEN-LO-8				NETGEN-LO-SR			
Gap	Method	2^{12}	2^{13}	2^{14}	2^{15}	2^{12}	2^{13}	2^{14}	2^{15}	2^{12}	2^{13}	2^{14}	2^{15}	2^{12}	2^{13}	2^{14}	2^{15}
10^{-1}	CPLEX	8.70	36.90	138.02	872.25	59.47	242.40	961.95	6266.61	7.97	34.54	144.63	981.43	63.01	265.72	1031.35	7528.59
	BSC	6.40	38.74	168.51	1220.60	24.12	153.43	953.44	8265.37	5.92	28.62	139.49	1218.74	19.40	110.43	552.81	3920.18
	NR	3.89	24.06	93.55	467.02	10.60	66.53	513.04	3184.48	1.52	8.94	65.73	408.43	10.42	56.61	340.59	2922.85
10^{-2}	CPLEX	8.70	36.90	140.64	893.78	60.96	247.85	984.32	6441.52	8.08	35.38	147.63	1005.59	65.94	274.49	1044.87	7727.51
	BSC	6.40	38.74	168.51	1220.60	25.40	162.85	1019.80	3804.74	7.08	38.11	187.12	1604.47	20.34	117.14	796.015	4789.31
	NR	3.89	24.06	93.55	467.02	10.60	66.53	616.12	8265.37	3.34	17.99	112.51	408.43	16.91	108.20	614.13	3397.46
10^{-3}	CPLEX	8.70	36.90	143.45	920.82	63.49	261.11	1022.64	6746.57	27.74	109.89	464.80	1043.12	68.58	289.56	1077.93	8015.43
	BSC	6.40	38.74	168.51	1220.60	29.65	162.85	1155.14	8854.55	7.08	39.77	226.26	1988.61	28.07	124.54	936.20	5672.07
	NR	3.89	24.06	93.55	467.02	20.99	121.62	1002.74	5845.18	3.34	20.27	143.81	882.13	18.46	118.63	687.83	5329.62

Table 2.6: Time elapsed to achieve gap levels.

LO-SR family on a representative problem instance with 2^{14} nodes. The BS and NR methods we propose start very close to the optimal solution, thanks to the tight interval found for the parameter using the procedure described in Subsection 2.3.2. Both of the methods achieve a percentage gap of 0.1% in their first iteration. Similar behavior is observed in other graph families and instances. The time needed to achieve various gap levels is shown in Table 2.6. For dense networks, for the early iterations CPLEX has a much higher gap value than the methods we propose. Moreover, the performance from NR and BSC methods can be further optimized by tuning the precision to which the subproblems are solved, since high-precision subproblem solutions are likely more useful in later iterations than in earlier ones (in these experiments, no such optimization was done).

2.5.3 Sensitivity to reliability

In this subsection, we emphasize the need for a reliable model by showing the difference in solutions between a reliable model versus a deterministic model where arc costs are stochastic. Additionally, we also investigate how the performance of the algorithms changes with respect to the changes in the reliability parameter $\bar{\lambda}$. In our experiments, standard deviation was generated uniformly from $[0.15E[c_{ij}], 0.3E[c_{ij}]]$ as that represents typical variation in transportation networks [7], however in other types of networks this problem parameter might be very different. To capture the possible effects of higher or lower variation for the arc costs, in the set of experiments we perform in this subsection we allow $\bar{\lambda}$ to range from 0.1 to 1000 and investigate the sensitivity of some of the problem metrics.

In terms of modeling, Figure 2.9 plots the percentage relative gap between the objective value of a deterministic model, that only considers the mean cost, and the objective value of the mean-standard deviation model versus the reliability parameter $\bar{\lambda}$ on a small network with 1024 nodes and 8192 arcs. As the reliability becomes more and more important to the decision maker, performance of the deterministic model deteriorates. In such situations, where reliability is important, using a mean-standard deviation model may outweigh the additional computation costs over optimizing expected performance only. Moreover, Figure 2.10 demonstrates that a significant decrease in the standard deviation cost can be traded off with a relatively small increase in the mean cost, especially when $\bar{\lambda}$ is small. It is thus possible to substantially improve reliability with a small impact to mean cost.

Figures 11–14 plot performance of the algorithms with respect to different reliability parameters for each of the graph family in the benchmark suite.

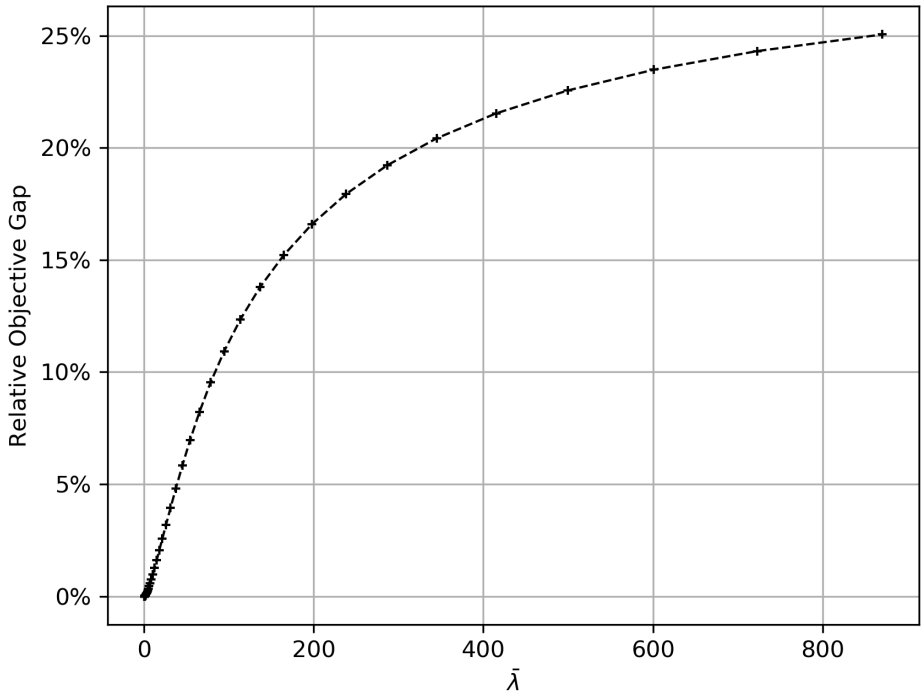

Figure 2.9: NETGEN-8 with $n = 2^{10}$.

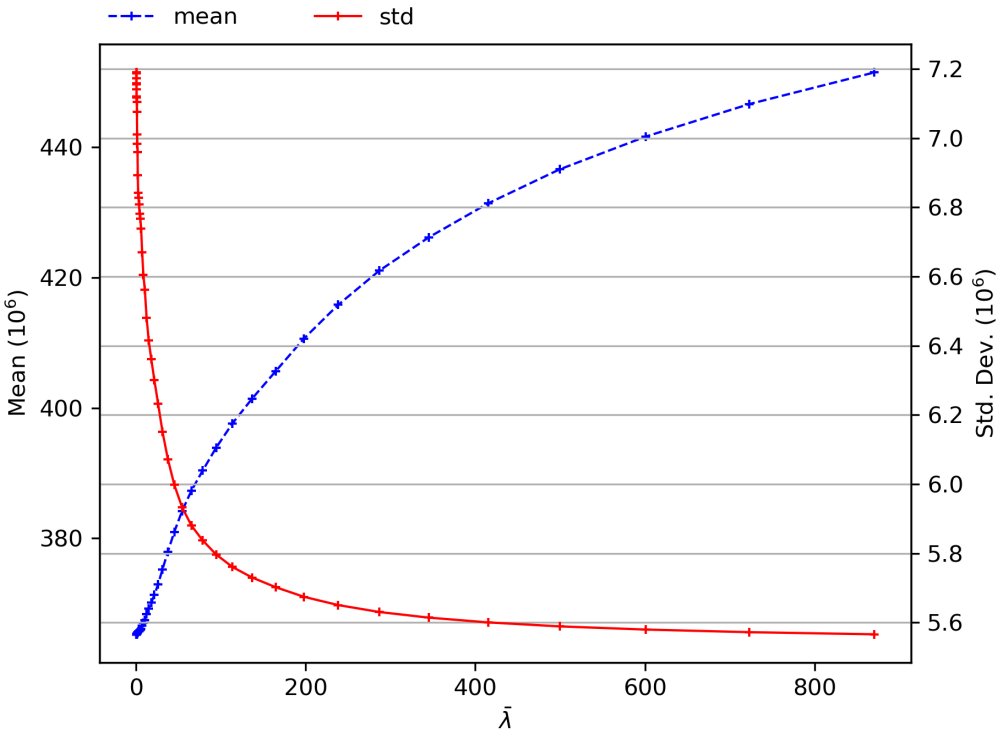

Figure 2.10: Criteria trade-off.

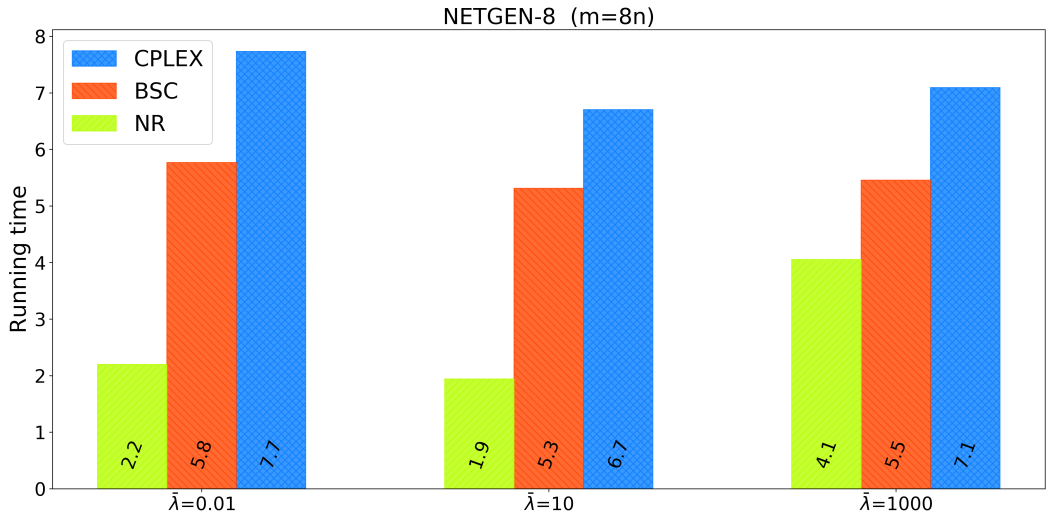

Figure 2.11: Sensivity to λ on NETGEN-8 instances.

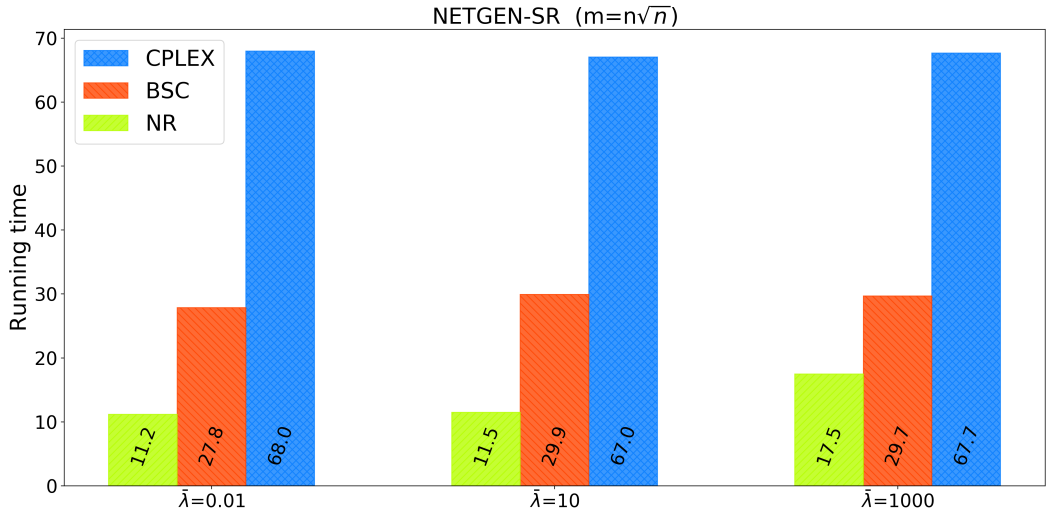

Figure 2.12: Sensivity to λ on NETGEN-SR instances.

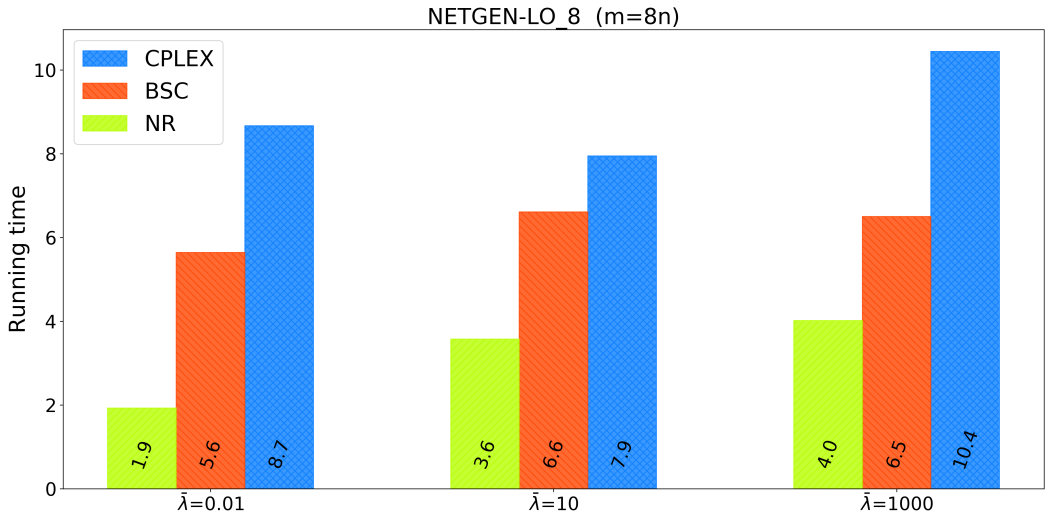

Figure 2.13: Sensivity to λ on NETGEN-LO-8 instances.

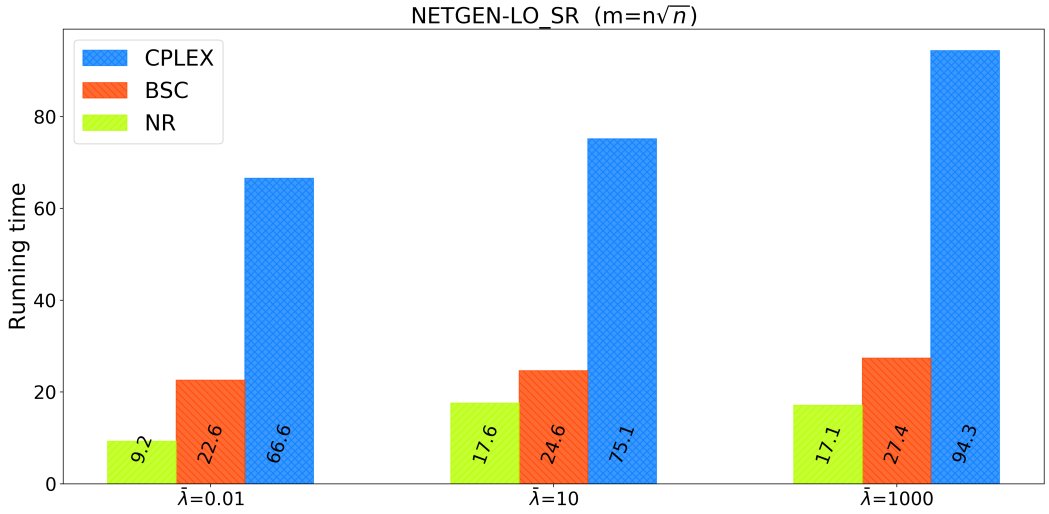

Figure 2.14: Sensivity to λ on NETGEN-LO-SR instances.

Amongst all the methods, performance of the BSC method is the most robust against the variation in the λ parameter. On the other hand, the performance of the NR method and CPLEX seems to be affected when the λ varies. While this is observable in all families, the change in runtime is especially notable on NETGEN-LO-SR family.

2.6 Conclusion

This chapter described three solution algorithms for the MSDMCF. The proposed methods solve the MSDMCF problem by repeatedly solving easier MVMCF problems. The effectiveness of the algorithms relies on the number of easier MVMCF problems required to be solved until the particular weight parameter λ is found. The algorithms differ in the root-finding method that they use. We also provide a procedure to find tighter upper and lower bounds for the root-finding methods, which is shown to improve the performance significantly. Amongst all, the BSC method is the simplest to implement. However, it needs more iterations to converge compared to the NR method. In contrast, the NR method requires solution derivatives, which can be obtained through sensitivity analysis. In each iteration of the NR method, we thus solve two problems, one subproblem and one auxiliary problem for finding the derivatives. The starting λ for the NR method is crucial, as starting far from the root may cause divergent behavior. In order to alleviate this potentially divergent behavior of the pure Newton method, we also provide a "failsafe" Hybrid method. These algorithms can also be applied to more general GNPMCF problems.

In our experiments, we compare running times of the algorithms to achieve a gap level of 0.01%. The NR method outperforms CPLEX and BSC on

every problem instance. In contrast, BSC method outperforms CPLEX for small instances of dense network families, while performing competitively or worse for larger instances. Another advantage of the NR and BSC methods is achieving very good solution, very fast. This can even be improved by changing the strategy to find the initial λ. Spending less effort for finding an initial parameter for the algorithms to start with, will result in time savings while trading off with solution quality.

The runtime of the proposed algorithms provided in this chapter can be further improved in several ways. We used CPLEX solver to solve the MVMCF subproblems. However, faster solution methods [72, 20, 67, 48] specialized for separable convex MCF problems could reduce runtime significantly. Another approach could be finding ways to improve the root-finding procedure, possibly exploring or modifying the methods to descend even faster than the ones provided. One can also do analysis on early stopping for early iterations in the proposed methods. The framework can be used for any problem with linear constraints and continuous variables, where the objective function meets the requirements. Other potential directions for future research is to investigate the case where the second criterion is concave and differentiable, and considering dependent arc costs.

Chapter 3

System Optimal Parking Search

This chapter describes a network flow problem where the nonlinearity lies in the constraints, rather than the objective function. The primary motivation is to study parking behavior in urban environments, representing competition among drivers for limited parking availability, and uncertainty in whether an available space exists at any given location and point in time.

Indeed, in certain neighborhoods drivers searching for parking form a significant proportion of overall traffic volume. Shoup [93] claims that in urban areas approximately 34% of congestion are due to cruising for parking. Similar findings were reported in a study in Franfurt, Germany for trips to the city center during the peak period [5].

Nevertheless, most urban traffic models, including microsimulation and traffic assignment, tend to ignore the need to search for parking at trip destinations. Researchers have addressed this deficiency in several ways. One approach is to transform the network with additional links to represent parking options and the link costs are assigned to represent the delay associated with the parking search [21, 54, 57, 53]. However, these models fail to capture the additional congestion caused by people searching for parking by assuming a deterministic cost for the additional parking links. Another approach is agent-based simulation [97, 10, 36, 26, 56], which has the advantage of incorporating fine-grained

details of driver behavior and parking dynamics. However, these advantages also present challenges in calibration and validation, and generally require estimating a relatively large number of parameters. A third approach, largely favored by economists, develops analytical results in stylized environments (e.g., the network is a homogeneous circle, or has only a single bottleneck) [2, 115, 116, 85, 107, 84]. While elegant results can be derived in such cases, it is unclear whether the topology of actual urban environments maps well to such environments, and to what extent policies derived there have practical insight.

An alternative model, developed by Boyles et al. [19], relies on a network transformation and changes to the flow conservation constraints. The resulting model is explicitly stochastic (and therefore models drivers searching for parking), is parsimonious in its assumptions (the only behavior assumed is that drivers take the action minimizing expected travel cost[1]), and scalable to large network systems. We build on this model and describe it below.

The Boyles et al. [19] model identifies a *user equilibrium*, in which drivers behave independently and aim to minimize their private travel cost. Transportation systems exhibit externalities, where one agent's choices impact the utility of others — for instance, as more drivers search for parking in the same location, they decrease the probability that any one of them will actually find such a space. Such externalities are seen in many systems involving competition for scarce resources (e.g., the celebrated tragedy of the commons).

In these settings it is valuable to identify a *system optimum*, which minimizes

[1]This is a generalized cost which can include travel time, walking time, parking cost, and so forth, with appropriate weights.

total cost across all agents. When externalities exist, the system optimum is generally different from the user equilibrium. System optimum solutions are useful to identify how much room there is to improve a given situation (a performance bound on any policy or intervention), and to suggest policies which can reduce total cost. In terms of parking, one example of a policy is to adjust the price of parking. Intuition suggests that the price of parking should be higher where demand exceeds supply; the value of a system optimum model could be to identify precisely how much higher this price should be.

The contribution of this chapter is to formulate the system optimum problem using the parking model of [19], and present a descent algorithm to find such a local optimum. Because the constraints of the problem are nonlinear, the resulting optimization problem is nonconvex and such methods cannot guarantee a global optimum. At a high level, we derive the marginal impact of additional flow on any link on total cost, and shift flows in a way that decreases total cost. We provide a sensitivity analysis to obtain the solution derivatives which are then used in marginal cost definitions. Despite the nonlinear flow conservation constraints, we can identify the necessary marginal costs by solving a linear system.

The approaches in Gallager [35] and Bar-Gera [6] are similar in spirit to our approach and derivations. However, both researchers considered the problem in an acyclic network (possibly a subset of the full network, as only an acyclic subgraph is used at equilibrium). In our setting we cannot make such an assumption — indeed, the effects of drivers "circling" for parking is a central feature, and requires cycles in the network. We therefore cannot use their results directly, and must consider the problem in networks with arbitrary topology.

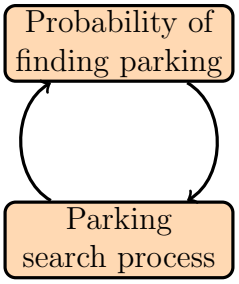

Figure 3.1: Mutual Dependency.

The remainder of this chapter is organized as follows. We first introduce the definitions and notations for the parking search model in [19] in Section 3.1. In Section 3.2 we provide the sensitivity analysis for given flow vector. Lastly, Section 3.4 describes the algorithm we propose.

3.1 Parking Search Model

This section describes the parking model of Boyles et al. [19], which we will build on to develop a system optimum algorithm. The central idea is to capture the mutual dependency between the probability of finding parking at any location, and the strategies drivers use as they search for parking. That is, the probability of finding parking space on a given link depends on how many drivers are searching for parking on that link, but drivers base their searching strategies based on the likelihood of finding parking. (Figure 3.1.)

The basis of this model is a network transformation, shown in Figure 3.2. The sets N and A of nodes and links in the network G are partitioned into several types. The physical transportation network (roads and intersections) are modeled with *regular* nodes and links, respectively denoted by the sets N_R and A_R.

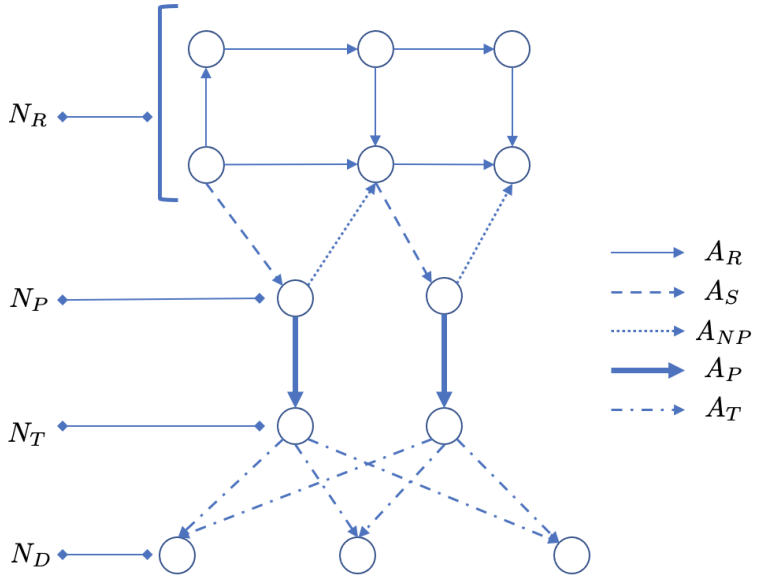

Figure 3.2: Transformed Network - Figure taken from [19].

For every location where parking is available (either on-street or in a lot or garage), a *parking* node is created; N_P is the set of all such nodes. Each parking node is associated with one *searching* link, one *parking* link, and one no-parking link. The searching link enters the parking node, and represents drivers who wish to park there, and will do so if they can find an available space. The parking and no-parking links leave the parking node. The former represents the searching drivers who are successful in finding a space, and the latter those who fail to find a space. The sets of searching, parking, and no-parking links are respectively denoted A_S, A_P, and A_{NP}.

Each parking link terminates at a *transfer* node (forming the set N_T). Finally, the set of *destination* nodes is denoted N_D. Each transfer node is connected to each destination node with a transfer link, from the set A_T. Critically, and

unlike many transportation network models, the destination nodes are distinct from the physical nodes travelers pass through *en route* to their trip ends.

We assume that there is exactly one parking node for each regular link (that is, there is a bijection between A_R and N_P). No generality is lost with this assumption, since we can add artificial links or nodes with suitable costs or parking functions (defined below) to satisfy it. As a result, we can use the convenient notation (i,j) to refer to a regular link, and $(i,j)_S$, $(i,j)_P$, and $(i,j)_{NP}$ to the searching, parking, and no-parking link corresponding to the parking node associated with (i,j).

Every link has a generalized cost t_{ij} representing the total disutility of travel on the link. Depending on the type of link, this may represent in-vehicle travel time, a monetary fee for parking, time spent walking from a parking space to the destination, the mental burden of having to search for a parking space, and so on. For simplicity we assume these are constants and do not depend on flow — the model can be generalized to make these flow-dependent, without major changes to the results that follow. We will use x_{ij} to denote the total flow on a link.

For each parking node the probability of finding a parking space is assumed to be a function of the flow on its corresponding searching link. That is, if (i,j) is the regular link corresponding to this parking node, the probability that any searching driver finds parking is $p_{ij}(x_{ij}^S)$. The number of drivers which can actually park is therefore $f_{ij}(x_{ij}^S) \equiv x_{ij}^S p_{ij}(x_{ij}^S)$, and the number of drivers which fail to park is $x_{ij}^S - f_{ij}(x_{ij}^S)$. It is reasonable to assume f_{ij} is nondecreasing and differentiable with $0 \leq f'_{ij} \leq 1$, and we make this assumption throughout. The parking capacity of link $(i,j)_P$ is defined as $C_{ij,P} = \sup_{x \geq 0}\{f_{ij}(x)\}$, and the parking capacity of the entire network is $C = \sum_{(i,j)_P \in A_P} C_{ij,P}$.

Driver behavior is represented by the fraction of drivers at each regular node which choose to leave on a particular link. As the links leaving a regular node are either regular links or searching links, these fractions describe both routing and parking search behavior. Specifically, let α_{ij}^d be the fraction of drivers passing through node i with a final destination of d that exit this node on link $(i,j) \in A_R \cup A_S$. These must be nonnegative, and all α_{ij} values leaving node i must sum to one. Drivers do not have any choice at other nodes: at a parking node, whether they successfully park or not depends on the physical availability of a space, represented by f_{ij}; at a transfer node they always choose the link connected to their destination node. Finally, the number of drivers starting at node $i \in N_R$ and destined for node $d \in N_D$ is q_{id}, and the total network demand is $D = \sum_{i \in N_R} \sum_{d \in N_D} q_{id}$.

Collecting these definitions, the flow conservation equations are:

$$x_{ij}^d = \alpha_{ij}^d \left(q_{id} + \sum_{(h,i) \in \mathcal{A}} x_{hi}^d \right) \qquad \forall i \in N_R, (i,j) \in A_R \cup A_S, d \in N_D \quad (3.1)$$

$$x_{ij}^P = f_{ij}(x_{ij}^S) = x_{ij}^S p_{ij}(x_{ij}^S) \qquad \forall (i,j)_P \in A_P \quad (3.2)$$

$$x_{ij}^{NP} = x_{ij}^S - x_{ij}^P \qquad \forall (i,j)_{NP} \in A_{NP} \quad (3.3)$$

$$x_{ij,T,d}^d = x_{ij,P}^d \qquad \forall (i,j) \in A_T, d \in N_D \quad (3.4)$$

$$\sum_{(i,j)_T \in A_T} x_{ij,T,d}^d = \sum_{i \in N_R} q_{id} \qquad \forall d \in N_D \quad (3.5)$$

Network loading refers to the process of determining $\mathbf{x}$ values from this system of equations, given values of the splitting fractions $\boldsymbol{\alpha}$. For an example, given the $\boldsymbol{\alpha}$ values and network in Figure 3.4, the flows resulting from network loading are shown in Figure 3.5.

Since the functions f_{ij} are nonlinear, the existence or uniqueness of such a solution are not trivial. A vector of splitting fractions $\boldsymbol{\alpha}$ are *weakly feasible* if they are nonnegative and sum to one for each node; the weakly feasible set is denoted

$$\Omega = \left\{ \boldsymbol{\alpha} \in \mathbb{R}_+^{|N_D|(|A_R|+|A_S|)} : \sum_{ij \in A} \alpha_{ij}^d = 1 \qquad \forall i \in N_R, d \in N_D \right\}. \tag{3.6}$$

Such a vector is *strongly feasible* if it is weakly feasible, and if the flow conservation equations have a finite solution in $\mathbf{x}$. The strongly feasible set is denoted Ω_S.

The following results were shown in Boyles et al. [19]:

- *(Necessary condition for strong feasibility.)* Unless $D \leq C$, no strongly feasible solution exists.
- *(Sufficient condition for strong feasibility.)* If $D < C$, all f_{ij} are strictly increasing, and the network is strongly connected, then at least one strongly feasible solution exists.
- If $D = C$, a strongly feasible solution may or may not exist.
- *(Uniqueness.)* If $\boldsymbol{\alpha}$ is strongly feasible and all f_{ij} are strictly increasing, there is exactly one solution $\mathbf{x}$ to the flow conservation equations.
- *(Computability.)* Given a strongly feasible $\boldsymbol{\alpha}$, the algorithm LoadNetwork described in the paper converges linearly to the unique solution $\mathbf{x}$. For completeness, we reproduce this algorithm in Figure 3.3.
- The strongly feasible set Ω_S is open relative to Ω.
- The strongly feasible set Ω_S may not be convex.

Algorithm LoadNetwork$(G, \boldsymbol{\alpha}, \epsilon)$

{Arguments are a graph G (as defined in Section 3.1 and equipped with $p_{ji}(x)$ functions), an exogenous vector of splitting proportions $\boldsymbol{\alpha}$ and a convergence tolerance $\epsilon > 0$.}
{Initialization}
for all $(i,j) \in \mathcal{A}, d \in N_D$ **do**
  $x_{ij}^d \leftarrow 0$
end for
for all $i \in N_R$ **do**
  $\eta_i^d \leftarrow q_{id}$
end for
while $\max_i \{\eta_i\} > \epsilon$ **do**
  Choose i such that η_i is maximal.
  {Push flow not searching for parking}
  for all $(i,j)_R \in A_R, d \in N_D$ **do**
    $x_{ij,R} \leftarrow x_{ij,R} + \alpha_{ij,R}^d \eta_i^d$
    $\eta_j \leftarrow \eta_j + \alpha_{ij,R}^d \eta_i^d$
  end for
  {Push flow searching for parking}
  for all $(i,j)_S \in A_S$ **do**
    $y_{ij,NP}^d \leftarrow x_{ij,NP}^d$ {Temporary value for updating imbalance at j}
    $x_{ij,S}^d \leftarrow x_{ijS}^d + \alpha_{ij,S}^d \eta_i^d \qquad \forall d \in N_D$
    $x_{ij,P}^d \leftarrow f_{ij}(x_{ij}^S) \left(x_{ij,S}^d / \sum_{d' \in N_D} x_{ij,S}^{d'} \right) \qquad \forall d \in N_D$
    $x_{ij,T,d}^d \leftarrow x_{ij,P}^d$
    $x_{ij,NP}^d \leftarrow x_{ij,S}^d - x_{ij,P}^d \qquad \forall d \in N_D$
    $\eta_j^d \leftarrow \eta_j^d + x_{ij,NP}^d - y_{ij,NP}^d \qquad \forall d \in N_D$
  end for
  $\eta_i \leftarrow 0$
end while
return **x**

Figure 3.3: The LoadNetwork algorithm introduced in Boyles et al. [19].

The latter two properties are unpleasant, since optimizing over open or non-convex sets is difficult. Boyles et al. [19] describe a procedure to amend this difficulty, assuming that trips terminate at each node with a small probability ϵ_S (drivers that "give up" searching). With this modification, all $\boldsymbol{\alpha}$ values are strongly feasible, and furthermore any strongly feasible solution in the original problem can be approached asymptotically as $\epsilon_S \to 0$.

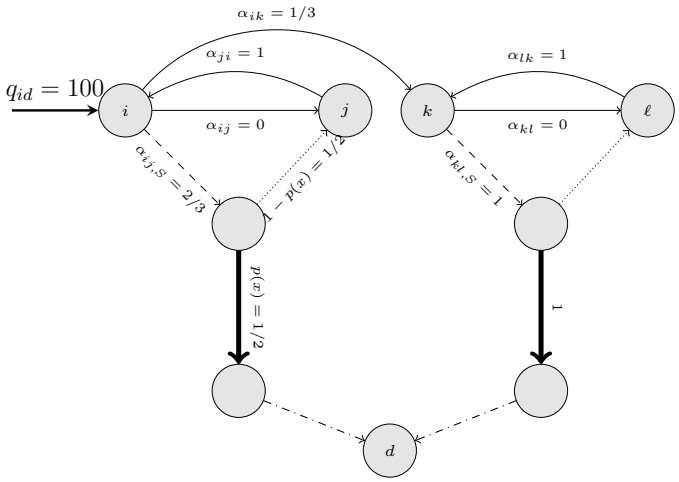

Figure 3.4: Network Splitting Proportions.

3.2 System optimal formulation

This section introduces the system optimal model based on the network loading procedure described in the previous section. Recall that t_{ij} is the generalized cost/disutility on link ij, reflecting travel time, monetary cost, walking time, mental stress, and so on.

For simplicity, and to focus our discussion on parking congestion (the main object of our interest), we make the following simplifying assumptions:

- There is a single destination shared by all drivers. This simplifies the

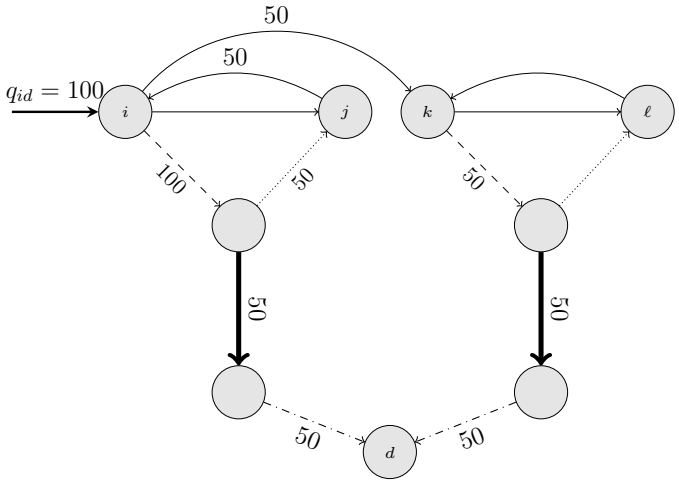

Figure 3.5: Network Flows.

notation (allowing us to omit d indices) and our methods naturally generalize to the case of multiple destinations without difficulty.

- The link costs t_{ij} are constants which do not depend on flow. The source of congestion in our model comes only from competition for scarce parking spaces. The derivations that follow can be extended in a straightforward (but tedious) way to handle traffic congestion on links as well.

- All weakly feasible solutions are strongly feasible ($\Omega_S = \Omega$). As discussed in the previous section, this can be done by introducing a suitably small ϵ_S value and assuming that drivers "give up" searching with this probability at each node; any strongly feasible solution can be approximated to arbitrary precision as $\epsilon_S \to 0$, and link flows increase without bound for any non-strongly feasible solution (and thus are irrelevant to the system optimal problem, which aims to minimize total cost).

  For the examples shown in the chapter, the network is constructed in a

way that this is almost everywhere true (in the measure-theoretic sense), so we do not have to introduce an explicit ϵ_S value.

Algorithm LoadNetwork (Figure 3.3) shows how the link flows $\mathbf{x}$ can be computed in terms of the splitting fractions $\boldsymbol{\alpha}$ which represent driver behavior. The total cost experienced by all drivers in the system can therefore be written as

$$T(\boldsymbol{\alpha}) = \sum_{ij \in A} x_{ij}(\boldsymbol{\alpha}) t_{ij} \,. \tag{3.7}$$

A system optimum solution is a feasible vector of splitting fractions $\boldsymbol{\alpha}$ which minimizes equation (3.7). With our assumptions, the optimization problem can be written as

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}} & T(\boldsymbol{\alpha}) \\ \text{s.t.} & \sum\limits_{ij} \alpha_{ij} = 1 \forall i \in N \\ & \alpha_{ij} \geq 0 \quad \forall ij \in A \end{array} \tag{3.8}$$

If we Langranganize the constraints with the dual variables π_i and θ_{ij}, we obtain the Lagrangian function

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\zeta}) = \sum_{ij} x_{ij}(\boldsymbol{\alpha}) t_{ij} - \sum_{i \in N} \pi_i \sum_{ij} \alpha_{ij} + \sum_{i} \pi_i + \sum_{ij} \theta_{ij} \alpha_{ij} \,, \tag{3.9}$$

The necessary optimality conditions then include

$$\frac{\partial \sum_{kl} x_{kl} t_{kl}}{\partial \alpha_{ij}} \geq \pi_i \quad \textbf{if} \quad \alpha_{ij} = 0 \quad \forall ij \in A \tag{3.10}$$

and

$$\frac{\partial \sum_{kl} x_{kl} t_{kl}}{\partial \alpha_{ij}} = \pi_i \quad \textbf{if} \quad \alpha_{ij} > 0 \quad \forall ij \in A \,. \tag{3.11}$$

These conditions dictate that all links (i, j) emanating from the same node with $\alpha_{ij} > 0$ have equal and minimal marginal cost $\Gamma_{ij} \equiv \dfrac{\partial \sum_{kl} x_{kl} t_{kl}}{\partial \alpha_{ij}}$.

However, the mapping $\mathbf{x}(\boldsymbol{\alpha})$ is nonlinear and nonconvex, so these conditions are not sufficient for global optimality. As a result, the method we develop can only find a local minimum.

3.3 Sensitivity of parking flows

The optimality conditions in the previous section involve the partial derivatives of total system cost $T(\boldsymbol{\alpha})$ with respect to the splitting fractions $\boldsymbol{\alpha}$:

$$\frac{\partial T(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \sum_{ij \in A} \frac{\partial x_{ij}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} t_{ij} \,. \tag{3.12}$$

To develop an algorithm based on these conditions, we therefore need to calculate the sensitivity of the flows to the changes in the proportions $\frac{\partial x_{ij}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$. It is not immediately obvious how to do so, because networks with cycles create dependencies among these sensitivities.

To illustrate this, consider the network in Figure 3.6. There is only one destination, so the d superscripts are suppressed for brevity. Intuitively, drivers enter at node 1 ($q_1 = 50$) and must choose whether to park at a "good" lot which is close to the destination but has limited capacity, or a "bad" lot which is far from the destination but has unlimited capacity. Drivers who cannot find a space in the "good" lot can cycle back around and try again. In keeping with the notational convention that every link is associated with a parking node, and that every node has a no-parking link, you can imagine that the bad lot is associated with the link $(2, 1)$ in the network, with the no-parking link suppressed since all drivers attempting to park in the bad lot will succeed; the figure draws them separately for clarity.

Additional features of the problem and specific parameter values:

- Because parking is guaranteed in the bad lot, the flow on its parking link will be the same as the flow on link 6, so we don't have to consider it in the following analysis.
- The link costs are $c_1 = 1$, $c_5 = 2$, and $c_6 = 4$; all other costs are zero. That is, a small amount of effort must be expended to search for parking in the "good" lot, whether or not you find a space; circling around to try again has a slightly larger cost; and parking in the bad lot has an even higher cost.
- The parking flow on link 3 (the good lot) is given by $x_3 = 10(1-e^{-x_1/10})$.
- The only places where drivers make choices in this network is at nodes 1 and 2; these are reflected by α_1, α_2, α_5, and α_6. When we add the feasibility constraints, there will only be two independent variables, since $\alpha_1+\alpha_2 = 1$ and $\alpha_5+\alpha_6 = 1$. But for the purposes of writing derivatives, we can first consider all four as independent, and then introduce the constraint later.
- Any solution with $\alpha_5 < 1$ is strongly feasible.

The (nonlinear) system of equations defining flow conservation in this network

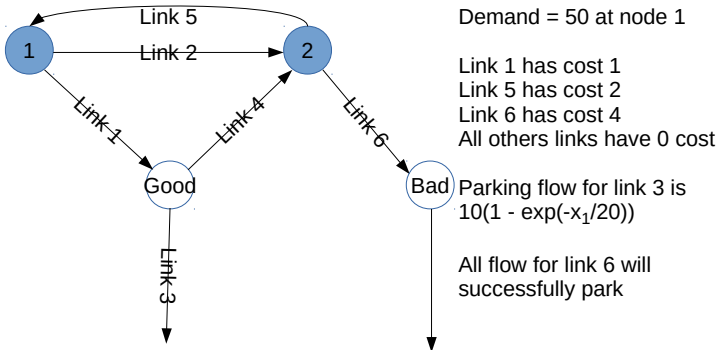

Figure 3.6: Small network for demonstration.

are as follows:

$$x_1 = \alpha_1(q_1 + x_5) \tag{3.13}$$

$$x_2 = \alpha_2(q_1 + x_5) \tag{3.14}$$

$$x_3 = f_3(x_1) \equiv 10(1 - e^{-x_1/10}) \tag{3.15}$$

$$x_4 = x_1 - x_3 \tag{3.16}$$

$$x_5 = \alpha_5(x_2 + x_4) \tag{3.17}$$

$$x_6 = \alpha_6(x_2 + x_4) \tag{3.18}$$

For instance, according to the flow conservation equations for this network, a marginal change in α_1 would change the flows on the x_1 and x_2. Change in x_2 changes the flow on x_5, which in return changes the flow on x_1. Therefore, we need to differentiate each equation and solve the resulting system of equations to find the partial derivatives of link flows with respect to a link proportion.

Differentiating each equation with respect to α_1 gives the *linear* system

$$\frac{\partial x_1}{\partial \alpha_1} = q_1 + x_5 + \alpha_1 \frac{\partial x_5}{\partial \alpha_1} \tag{3.19}$$

$$\frac{\partial x_2}{\partial \alpha_1} = \alpha_2 \frac{\partial x_5}{\partial \alpha_1} \tag{3.20}$$

$$\frac{\partial x_3}{\partial \alpha_1} = f_3'(x_1) \frac{\partial x_1}{\partial \alpha_1} \tag{3.21}$$

$$\frac{\partial x_4}{\partial \alpha_1} = \frac{\partial x_1}{\partial \alpha_1} - \frac{\partial x_3}{\partial \alpha_1} \tag{3.22}$$

$$\frac{\partial x_5}{\partial \alpha_1} = \alpha_5 \left(\frac{\partial x_2}{\partial \alpha_1} + \frac{\partial x_4}{\partial \alpha_1} \right) \tag{3.23}$$

$$\frac{\partial x_6}{\partial \alpha_1} = \alpha_6 \left(\frac{\partial x_2}{\partial \alpha_1} + \frac{\partial x_4}{\partial \alpha_1} \right) \tag{3.24}$$

which can be written in matrix form as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & -\alpha_1 & 0 \\ 0 & 1 & 0 & 0 & -\alpha_2 & 0 \\ -f_3'(x_1) & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & -\alpha_5 & 0 & -\alpha_5 & 1 & 0 \\ 0 & -\alpha_6 & 0 & -\alpha_6 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial \alpha_1} \\ \frac{\partial x_2}{\partial \alpha_1} \\ \frac{\partial x_3}{\partial \alpha_1} \\ \frac{\partial x_4}{\partial \alpha_1} \\ \frac{\partial x_5}{\partial \alpha_1} \\ \frac{\partial x_6}{\partial \alpha_1} \end{bmatrix} = \begin{bmatrix} q_1 + x_5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.25}$$

If you repeat the process with the derivatives with respect to α_2, α_5, or α_6 you will see that the matrix on the left-hand side is identical; only the right-hand side vector changes. This is good news, since we only have to invert the matrix once, and then reuse it to find the derivatives of all x_i with respect to all α_j.

As a specific example, assume that initially $\alpha_1 = \alpha_2 = \alpha_5 = \alpha_6 = 1/2$. (This produces the solution $x_1 = x_2 = 45.1$, $x_3 = 9.9$, $x_4 = 35.2$, $x_5 = x_6 = 40.1$.) Substituting these specific numbers into the systems of equations above, we find that the Jacobian of $\mathbf{x}$ with respect to $\boldsymbol{\alpha}$ is

$$J_{\mathbf{x}}(\boldsymbol{\alpha}) = \left[\frac{\partial x_i}{\partial \alpha_j}\right] = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cccc} \alpha_1 & \alpha_2 & \alpha_5 & \alpha_6 \end{array} \\ \begin{bmatrix} 134.4 & 44.8 & 79.8 & 0 \\ 44.3 & 134.9 & 79.8 & 0 \\ 1.5 & 0.5 & 0.9 & 0 \\ 132.9 & 44.3 & 78.9 & 0 \\ 88.6 & 89.6 & 160 & 0 \\ 88.6 & 89.6 & 79.3 & 80.2 \end{bmatrix} \end{array} \tag{3.26}$$

We have not yet taken into account the constraints $\alpha_1+\alpha_2 = 1$ and $\alpha_5+\alpha_6 = 1$ representing weak feasibility. In fact, a solution can be determined by two independent variables, the "searching fraction" β_1 and "cycling fraction" β_5. Given values of these "nonbasic" variables, we have $\alpha_1 = \beta_1$, $\alpha_2 = 1 - \beta_1$, $\alpha_5 = \beta_5$, and $\alpha_6 = 1 - \beta_6$ as the values of all splitting fractions. The Jacobian of $\mathbf{x}$ with respect to β_1 and β_5 is easily computed from the chain rule:

$$\frac{\partial x_1}{\partial \beta_1} = \frac{\partial x_1}{\partial \alpha_1}\frac{\partial \alpha_1}{\partial \beta_1} + \frac{\partial x_1}{\partial \alpha_2}\frac{\partial \alpha_2}{\partial \beta_1} = \frac{\partial x_1}{\partial \alpha_1} - \frac{\partial x_1}{\partial \alpha_2}\,, \tag{3.27}$$

and so forth. So the change in $\mathbf{x}$ values with respect to β_1 is simply the first column of (3.26) minus the second:

$$\left[\tfrac{\partial x_i}{\partial \beta_1}\right] = \begin{bmatrix} 89.6 & -90.6 & 0.99 & 88.6 & -0.99 & -0.99 \end{bmatrix}^T . \tag{3.28}$$

Likewise, the change in $\mathbf{x}$ values with respect to β_2 is simply the difference between the third and fourth columns of (3.26):

$$\left[\tfrac{\partial x_i}{\partial \beta_2}\right] = \begin{bmatrix} 79.8 & 79.8 & 0.88 & 78.9 & 159 & -0.88 \end{bmatrix}^T . \tag{3.29}$$

We now move beyond this specific example. In general, the Jacobian of $\mathbf{x}$ with respect to $\boldsymbol{\alpha}$ solves the matrix equation

$$\mathbf{A} J_{\mathbf{x}} = \mathbf{B}\,, \tag{3.30}$$

where the $|A| \times |A|$ matrix $\mathbf{A}$ can be written in block form, partitioned based on the type of link. Without loss of generality, we can assume that the links in A_S, A_P, and A_{NP} appear in the same order based on their common parking node. The matrix then takes the form

$$\mathbf{A} = \begin{array}{c} \\ A_R \\ A_S \\ A_P \\ A_{NP} \\ A_T \end{array} \begin{array}{c} \begin{array}{ccccc} A_R & A_S & A_P & A_{NP} & A_T \end{array} \\ \begin{bmatrix} \mathbf{I} + \mathbf{A_1} & \mathbf{0} & \mathbf{0} & \mathbf{A_2} & \mathbf{0} \\ \mathbf{A_3} & \mathbf{I} & \mathbf{0} & \mathbf{A_4} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_5} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix} \end{array}, \tag{3.31}$$

incorporating the following submatrices:

- Row i and column j of matrix $\mathbf{A_1}$ is $-\alpha_i$ if the regular link j is immediately upstream of regular link i, and 0 otherwise.
- Row i and column j of matrix $\mathbf{A_2}$ is $-\alpha_i$ if the no-parking link j is immediately upstream of regular link i, and 0 otherwise.
- Row i and column j of matrix $\mathbf{A_3}$ is $-\alpha_i$ if the regular link j is immediately upstream of searching link i, and 0 otherwise.
- Row i and column j of matrix $\mathbf{A_4}$ is $-\alpha_i$ if the no-parking link j is immediately upstream of searching link i, and 0 otherwise.
- Matrix $\mathbf{A_5}$ is diagonal, with the i-th diagonal element equal to $-f_i'$ evaluated at the current value of its corresponding searching link.

This matrix is irreducibly diagonally dominant for any weakly feasible α (since $f' \leq 1$ by assumption) and therefore the Levy-Desplanques theorem ensures it is nonsingular, so a unique solution exists.

The $|A| \times |A_R \cup A_S|$ matrix $\mathbf{B}$ takes the form

$$\mathbf{B} = \begin{array}{c} \\ A_R \cup A_S \\ A_P \\ A_{NP} \\ A_T \end{array} \begin{array}{c} A_R \cup A_S \\ \left[\begin{array}{c} \mathbf{B_1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{array}\right] \end{array}, \tag{3.32}$$

where $\mathbf{B_1}$ is a diagonal matrix. Its i-th element is given by $q_{h(i)} + \sum_{g \in RS(h(i))} x_g$, where $h(i)$ is the (regular) tail node of link i, and $RS(h(i))$ is its reverse star.

Equation (3.30) gives the derivatives of the link flows, treating each α_i value as independent from the others. To reflect the weak feasibility constraint, for each regular node we can choose all but one of the outgoing links as "nonbasic;" the remaining "basic" link's flow can be determined based on the others. The Jacobian of $\mathbf{x}$ with respect to the nonbasic $\boldsymbol{\alpha}$ values is obtained by subtracting each regular node's basic column from its associated nonbasic columns, then deleting the basic columns. The derivations were given for the case where there is only one destination, in general same procedure can be applied for the multiple destination case and the sensitivities $\frac{\partial \mathbf{x}}{\partial \alpha_{ij}^d}$ can be found.

With the Jacobian of $\mathbf{x}$ with respect to $\boldsymbol{\alpha}$ in hand, it is straightforward to calculate the marginal costs $\Gamma_{ij} = \partial T / \partial \alpha_{ij}$, which form the gradient of T.

3.4 System Optimal Assignment Algorithm

For a node i, consider two outgoing links (i, j) and (i, k), their associated splitting fractions α_{ij} and α_{ik}, and their marginal costs Γ_{ij} and Γ_{ik}. If $\Gamma_{ij} > \Gamma_{ik}$ and $\alpha_{ij} > 0$, the necessary optimality conditions 3.10 & 3.11 imply that this $\boldsymbol{\alpha}$ is not optimal. A step towards equalizing the marginal costs, that is, a small decrease in α_{ij} and increase in α_{ik}, will drive $\boldsymbol{\alpha}$ closer to the necessary

condition and result in a reduction in total system cost $T(\boldsymbol{\alpha})$.

For a particular node i, let the nonbasic link nb_{ij} and basic link b_{ij} be the highest marginal cost link and lowest marginal cost link respectively;

$$nb_{ij} \in \underset{ij \in FS(i)}{\arg\max}\, \Gamma_{ij} \tag{3.33}$$

$$b_{ij} \in \underset{ij \in FS(i)}{\arg\min}\, \Gamma_{ij} \tag{3.34}$$

where $FS(i)$ is link i's forward star. The algorithm we propose iterates through two procedures until a convergence criterion is met. First, given the network flows $\mathbf{x}(\boldsymbol{\alpha})$, we will consider every node i, and shift proportions from nb_{ij} to b_{ij}, reducing $T(\boldsymbol{\alpha})$ at each step, we will call this the update procedure. Once the proportions are shifted and new α obtained after performing the update procedure, the corresponding $\boldsymbol{x}(\boldsymbol{\alpha})$ is then found using the flow push procedure LoadNetwork$(G, \boldsymbol{\alpha}, \epsilon)$. These steps iterate until the necessary conditions 3.10 & 3.11 are approximately satisfied.

In the update procedure, first, we need to determine how much proportion should be shifted from the nonbasic link to the basic one. Let Δ_{ij} denote the difference in marginal cost between the nonbasic and basic link:

$$\Delta_i = \Gamma_{nb_{ij}} - \Gamma_{b_{ij}}\,, \tag{3.35}$$

We choose a shift proportional to Δ, so that larger marginal cost differences result in larger shifts:

$$\delta_i = \eta \Delta_i \tag{3.36}$$

where η is the step size.

Finding an appropriate step size is crucial to allow the objective function to descend at each iteration. For this purpose, an exact line search to minimize T would be too costly at each iteration, while a fixed step size would require too many iterations (since a small η is needed to ensure convergence). As a middle ground, we use an inexact backtracking line search with an Armijo stopping rule.

Once a desired shift amount η is decided, we then project this shift onto the feasible set with the following procedure:

$$\eta \leftarrow \min\{\eta, \alpha_{nb_{ij}}/\Delta_{ij}\} \tag{3.37}$$

$$\alpha_{nb_{ij}} \leftarrow \alpha_{nb_{ij}} - \eta\Delta_{ij} \tag{3.38}$$

$$\alpha_{b_{ij}} \leftarrow \alpha_{b_{ij}} + \eta\Delta_{ij}\,. \tag{3.39}$$

The pseudocode is given in Algorithm 1. We test the algorithm on the toy network represented in Figure 3.6 with only two variables. A user equilibrium[2], shown in Figure 3.7, occurs when $\alpha_1 = 0.776$, $\alpha_2 = 0.224$, $\alpha_5 = 0$, and $\alpha_6 = 1$; this solution has a total cost of 200. A system optimum solution is shown in Figure 3.8. In this solution, $\alpha_1 = 0.277$, $\alpha_2 = 0.723$, $\alpha_5 = 0$, $\alpha_6 = 1$, and the total cost is 184.

3.5 Conclusion

We presented an algorithm to identify a (locally) system optimal assignment for the model described in [19]. To do this, the main contribution was the

[2]Not "the" because the nonlinearity of the problem means there may be multiple equilibria.

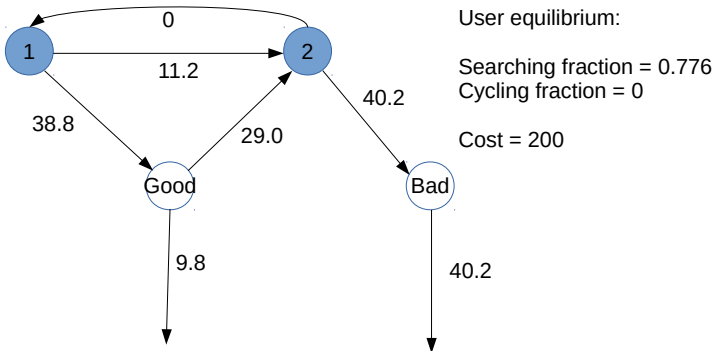

Figure 3.7: User equilibrium link flows.

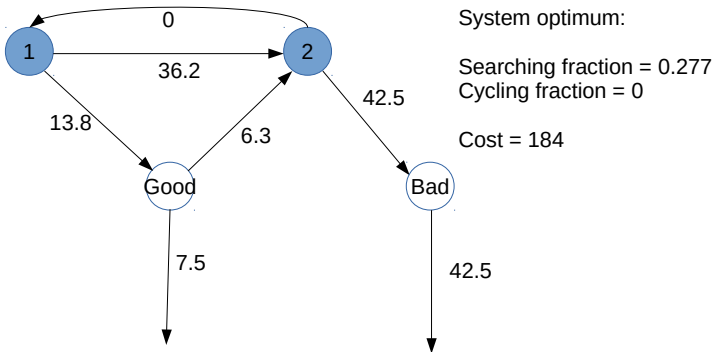

Figure 3.8: System optimum link flows.

Algorithm 1: SO_Algorithm(G,ξ, ϵ)

Initialization (can also initialize with user equilibrium or other feasible solution);
foreach $d \in N_D$ **do**
  foreach $i \in N_R$ **do**
    foreach $ij \in FS(i)$ **do**
      $\alpha_{ij}^d \leftarrow 1/|FS(i)|$;
cost_history $\leftarrow []$;
converged $\leftarrow$ **False**;
while ***not*** *converged* **do**
  $i \leftarrow 1$;
  foreach $d \in N_D$ **do**
    Flow push procedure;
    $\boldsymbol{x} \leftarrow$ `NetworkLoading(`G`,` $\boldsymbol{\alpha}$`,` ϵ`)`;
    Update procedure;
    cost $\leftarrow \sum_{ij} x_{ij} t_{ij}$;
    Push cost onto cost_history;
    $J_{\boldsymbol{x}} \leftarrow \mathbf{A^{-1}B}$;
    foreach $i \in N_R$ **do**
      $\Gamma_{ij} \leftarrow \frac{\partial \sum_{kl} x_{kl} t_{kl}}{\partial \alpha_{ij}^d}$;
      $nb_{ij} \leftarrow \arg\max_{ij \in FS(i)} \Gamma_{ij}$, $b_{ij} = \arg\min_{ij \in FS(i)} \Gamma_{ij}$;
      $\Delta_i \leftarrow \Gamma_{nb_{ij}} - \Gamma_{b_{ij}}$;
      Select desired η using Armijo rule ;
      $\eta \leftarrow \min\{\eta, \alpha_{nb_{ij}}/\Delta_{ij}\}$;
      $\alpha_{nb_{ij}} \leftarrow \alpha_{nb_{ij}} - \eta\Delta_{ij}$;
      $\alpha_{b_{ij}} \leftarrow \alpha_{b_{ij}} + \eta\Delta_{ij}$;
  Check for convergence;
  if $i > 10$ **then**
    if *cost - cost_history[-10]* $< \xi$ **then**
      converged = **True**

derivation of the solution sensitivities in the case of cyclic flows, generalizing similar work in acyclic networks (which relied critically on the existence of a topological order that does not exist with cycles).

Future work should address the computational properties of the algorithm we develop on larger networks, and explore options for escaping local optima. The properties of the system optimal solution should also be studied to identify guidance for urban parking policies. For instance, if the system-optimal state can be related to the user-equilibrium state, it may be possible to determine parking prices which can shift the user equilibrium to a system optimum.

Chapter 4

Post-Disaster Recovery Sequencing Strategy for Road Networks

4.1 Introduction

Natural disasters and extreme hazards such as seismic events, floods, terrorist attacks or hurricanes can substantially damage transportation systems along with other infrastructure systems such as power, water, and gas. The 1989 Loma Prieta earthquake resulted in the collapse of multiple bridges and required retrofitting for all Bay area bridges, 1994 Northridge earthquake had 286 collapsed highway bridges [44], 2008 earthquake in Wenchuan, China affected 1,657 bridges [122], and Hurricane Irene damaged over 300 bridges and 2,000 miles of highways in Vermont [65]. Planning resilient transportation networks is necessary to minimize the effects of such disasters on system performance. Zhang et al. [114] classify resilience in terms of *mitigation strategies* put in place before a disaster to increase robustness of the system; *emergency response* actions taken immediately after the event; and *long-term recovery*, as the system is gradually reinstated through reconstructions of the impaired network.

This chapter focuses on long-term recovery phase for road networks. Specifically, we assume that a subset of links is initially unusable due to damage or destruction, and must be repaired. Given the time needed to repair each link,

we investigate the ordering for rebuilding these links which minimizes the total system travel time (TSTT) experienced over the repair horizon, assuming that these links must be repaired sequentially, and that the traffic state reaches user equilibrium at each stage of repairs. This problem is difficult both due to its combinatorial nature (the number of repair sequences is the factorial of the number of damaged links), and because the user equilibrium assumption means that many instances of the traffic assignment problem (TAP) must be solved as subproblems.

Most past investigations into these type of problems either focus on small instances (where all sequences can be enumerated) or use general-purpose metaheuristics such as genetic algorithms or tabu search. The problem can also be cast in the form of an optimal scheduling problem, and techniques from that domain can be applied. However, as we show below, methods that do not account for the interdependency between damaged facilities can provide poor solutions.

This article describes a specialized heuristic, tailored to the traffic assignment problem, which produces high-quality solutions in a reasonable amount of time, even on large network instances. In particular:

- We show that the complexity of the problem can be reduced from factorial to exponential by deriving an analogue of Bellman's optimality principle from dynamic programming.
- We build on this principle by designing a customized bidirectional search heuristic, using lower and upper bounds based on traffic assignment properties.

- We use these bounds to create specialized branching and pruning rules within the heuristic.
- A beam search is further used to reduce the number of solutions which are explored.
- Our numerical results show that this customized heuristic produces higher-quality solutions than alternative heuristics.

The remainder of the chapter is organized as follows. We review past literature on the problem, then introduce our specific problem formulation and associated notation. We follow this with a high-level overview of the bidirectional search heuristic, and then its specific components: lower and upper bounds used to guide the search direction, and other speed-up techniques. The computational experiments come next, and the chapter is concluded by a summary of the key contributions and discussion of future research directions.

4.1.1 Literature review

This section focuses on reconstruction sequencing problems in transportation networks. For comprehensive reviews on resilience, we refer the reader to studies by Faturechi & Miller-Hooks [31] and Zhou et al. [120] in transportation systems; Xu et al. [106] in electricity systems; Wang et al. [100] in communication systems; and Luna et al. [64] study water distribution network recovery. The rest of this section focuses on studies in the field of transportation networks.

For post-disaster reconstruction in transportation networks, Faturechi & Miller-Hooks [31] categorize performance metrics into three measures: functional, topological, and economical. Merschman et al. [66] seek to find the repair

sequence minimizing a combination of these measures. The case study focuses on a sample of four bridges, and they enumerate every sequence for evaluation. Chen & Miller-Hooks [24] study the recovery of intermodal freight transport network by proposing a stochastic mixed-integer program to identify recovery actions to maximize demand met. Miller-Hooks et al. [68] aim to improve pre-disaster mitigation and post-disaster scheduling in order to maximize expected traffic demand accommodated. Moreover, Zhang et al. [113] use the average number of reliable independent pathways as a performance metric. Direct and indirect costs combined with a topological metric quantify performance in Zhang & Ali [112].

In this chapter, we use TSTT as a functional measure to capture the level of service of a network, not just raw connectivity. Other studies [15, 99, 109] use a similar performance measure, as part of a multi-stage framework in order to obtain the reconstruction sequence.

These studies differ in the assumptions they make, the way they model the repairs, and the cost functions. Bocchini & Frangopol [15] use TSTT and total travel distance as performance measures. They propose a multi-objective bi-level optimization model to find a restoration sequence to minimize weighted combination of cost and time till target functionality level. Capacity restoration is modeled as a continuous process, with no constraints on simultaneous repairs. A genetic algorithm approach is then used to solve a problem on a small instance with 38 links.

Vugrin et al. [99] propose a bi-level model with a multi-criteria objective function, a weighted combination of system impact (measured by TSTT) and the cost of recovery operations. Their model considers repair tasks requiring resources of one or more types. Each repair task return partial capacity on the

link. Their model also allows simultaneous repair actions as long as demand does not exceed total resource availability. They do not mention a solution method, directly presenting results for computational experiments on a small 9 node and 15 link network with four damaged links.

Ye & Ukkusuri[109] consider the sum of system performance during reconstruction as their cost function. They incorporate day-to-day traffic assignment problems to model the evolution of flow during recovery, which is used as the lower level problem. However, this requires usage of a path-based solution method. These methods are very memory intensive and require further assumptions about path flows due to lack of uniqueness at optimality. Tabu search is used to obtain solutions. They allow simultaneous recovery operations while budget constraints are met. Their largest numerical experiment considers Sioux Falls network (24 nodes and 76 links) with 6 damaged links.

Gehlot et al. [38] consider the problem of finding optimal control sequence for infrastructure repair, where components continue to degrade while not being repaired. Such deterioration often not modeled in other studies considering optimal sequencing problem. Their variation of the problem aims to find a sequence that bring maximum number of components to the desired recovery threshold without letting them to degrade to an undesirable threshold level. In an extension [37], they then consider a case where they model precedence constraints among the components into the problem.

Rey and Bar-gera[88] study a similar problem to the one we consider in this chapter, aiming to finding a reconstruction policy minimizing total system travel time. They provide a mixed-integer linear program (MILP) in order to produce exact solutions to the resulting bi-level program. They then use

the exact solutions generated by MILP to compare three greedy heuristic approaches on small instances where there are 9-10 repair projects. Our problem is in fact a special case of theirs, where repairs must be conducted sequentially (and not simultaneously). However, the heuristic we develop for this special case outperforms the more general heuristics they developed, both in terms of solution quality and the size of problem instances it can handle, by exploiting the sequentiality property. Our numerical experiments provide more details on this comparison, using greedy heuristic approaches similar to theirs as a benchmark.

A key difficulty in solving these problems is the large, combinatorial set of feasible solutions. For $|\mathcal{B}|$ damaged links, there are $|\mathcal{B}|!$ possible repair sequences, each requiring $|\mathcal{B}|$ solutions of the traffic assignment problem to evaluate total cost. This is reflected in the computational results of these studies, with relatively small networks being used as case studies. Additionally, these studies do not utilize any properties or underlying structure of TAP, treating it as a black box sub-problem. In this chapter, we seek a more scalable approach for large-scale networks by exploiting such properties.

4.2 Problem Statement

The basis of our model is the static traffic assignment problem (TAP), originally formulated by Beckmann [8] and described at length in the books by Patriksson [80] and Boyles et al. [18]. Consider a network with a set of nodes $\mathcal{N}$, links $\mathcal{A}$, and zones $\mathcal{Z}$. Each link has a travel time T_a which is a nondecreasing function of the flow on that link alone. The travel demand between zones r and s is d_{rs}, which we assume to be fixed, an assumption common to most of the other literature on this problem [15, 99]. Let Π^{rs} denote the set

of simple paths connecting zones r and s, and $\Pi = \cup_{(r,s)\in\mathcal{Z}^2}\Pi^{rs}$ the set of all such paths.

Let $\mathcal{B} \subset \mathcal{A}$ denote the set of damaged links which must be repaired; let $N = |\mathcal{B}|$. The time needed to repair each damaged link $b \in \mathcal{B}$ is D_b. We assume that the network remains strongly connected even without these links (this assumption can be relaxed by introducing artificial links between each origin and destination, with a large fixed travel time expressing the social cost of failing to serve a trip.) We further assume that repairs proceed sequentially, so there are N network states as each damaged link is restored, each lasting for the duration of repair for that link, plus a terminal state after all links are restored. The non-terminal states will be indexed by $t \in \{1, \ldots, N\}$; for the most part we will be unconcerned about the terminal state, since the delay in the final state does not depend on the order of repairs, and the scope of the optimization problem ends after all repairs are done. The sequence of repairs is represented by the binary decision variables y_b^t, which equal 1 iff link b is the one being repaired during stage t, and z_b^t, which equal 1 iff link b was repaired prior to stage t (and is thus usable by travelers in stage t).

Within each stage t, we assume that the network flows are consistent with Wardrop's principle [102] and are at user equilibrium, in that every used path between each origin and destination has equal and minimal travel time. This assumption is also common [15, 99]. Effectively, we assume that the time to rebuild each link is long enough for a new equilibrium to arise; empirically, this has been observed to be several weeks [121]. For the case of major infrastructure likely to be damaged by disaster, such as bridges, we believe this to be reasonable. (See the Conclusions section for some discussion on relaxing this assumption.) We will denote by x_a^t the equilibrium flow on link a during

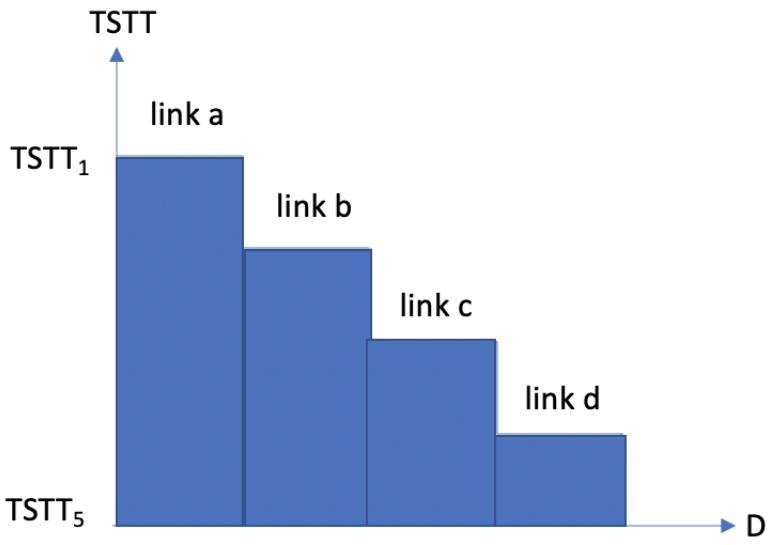

Figure 4.1: The shaded area represents the total delay over the repair horizon.

stage t, and by h_π^t an equilibrium flow on path π in this stage.

The total system travel time (TSTT) during each day in stage t is given by $\sum_a x_a^t T_a(x_a^t)$, and this state persists for D_b days, where b is the link repaired in stage t. In what follows, it will be convenient to use $TSTT_t$ to denote the total system travel time *during* stage t. Since links are repaired at the end of each stage, $TSTT_1$ represents the total system travel time with no links repaired, $TSTT_2$ the total system travel time with only one link repaired, and so on. Define $TSTT_{t+1}$ to be the total system travel time at the equilibrium solution after all links are restored. (We do not actually have to calculate this solution explicitly, since we stop counting delay once the last link is restored.) The total delay is then the area under the graph of TSTT over the repair horizon (Figure 4.1).

This is expressed by the following bilevel optimization problem, in which M

is a sufficiently large constant:

$$\begin{aligned}
\min_{\mathbf{y},\mathbf{z}} \quad & \sum_{t=1}^{N} \sum_{a \in \mathcal{A}} x_a^t T_a(x_a^t) \sum_{b \in \mathcal{B}} y_b^t D_b \\
\text{s.t.} \quad & \sum_{t'=1}^{t-1} y_b^{t'} = z_b^t \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \\
& \sum_{b \in \mathcal{B}} y_b^t = 1 \quad \forall t \in \{1, \ldots, N\} \\
& \sum_{t=1}^{N} y_b^t = 1 \quad \forall b \in \mathcal{B} \\
& y_b^t \in \{0,1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\} \\
& z_b^t \in \{0,1\} \quad \forall b \in \mathcal{B}, t \in \{1, \ldots, N\}
\end{aligned} \tag{4.1}$$

where each $\mathbf{x_t}$ is optimal to:

$$\begin{aligned}
\min_{\mathbf{x_t},\mathbf{h_t}} \quad & \sum_{a \in \mathcal{A}} \int_0^{x_a^t} T_a(x)\, dx \\
\text{s.t.} \quad & \sum_{\pi \ni a} h_\pi^t = x_a^t \quad \forall a \in \mathcal{A} \\
& \sum_{\pi \in \Pi^{rs}} h_\pi^t = d_{rs} \quad \forall (r,s) \in \mathcal{Z}^2 \\
& h_\pi^t \geq 0 \quad \forall \pi \in \Pi \\
& x_b^t \leq M z_b^t \quad \forall b \in \mathcal{B}
\end{aligned} \tag{4.2}$$

The problem is evidently a scheduling problem. A standard technique for such problems would be to calculate the ratio of repair benefits to repair duration for each link, and repair links in descending order of this ratio. However, in our problem the benefit of repairing a link depends on which other links have already been repaired, and which remain unusable — the "actual" repair benefits in the optimal sequence are unknowable *a priori*. One can nevertheless

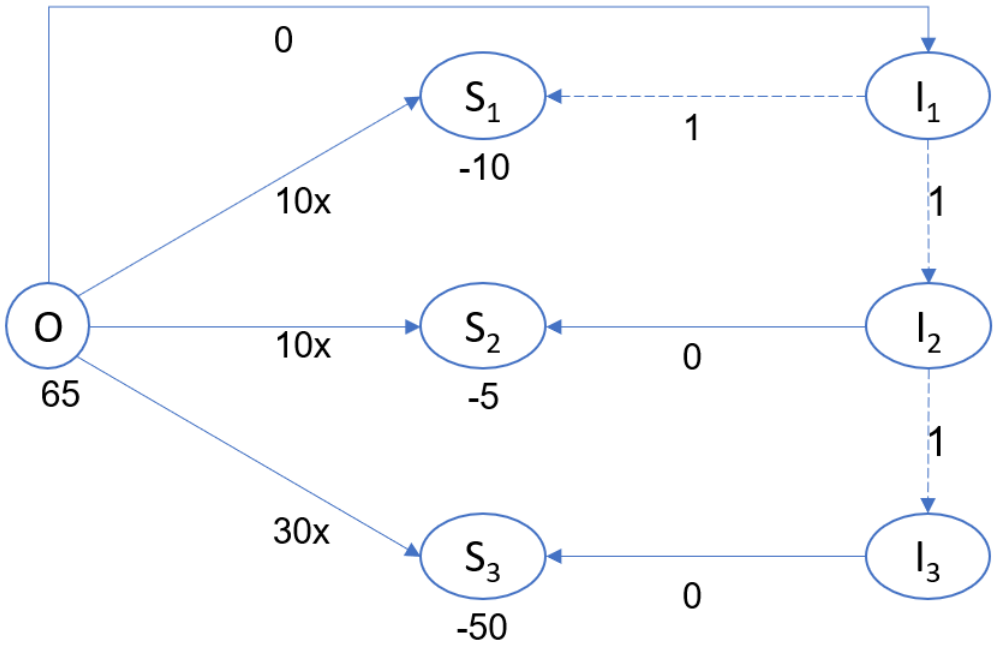

Figure 4.2: Toy Network

estimate these benefits; one example is a "greedy" heuristic based on identifying the unrepaired link maximizing the benefit-duration ratio given the current state of the network; repairing this link; recalculating these ratios for the remaining unrepaired links; and so forth.

This heuristic performs poorly in networks with strong dependencies between damaged links. Consider the network in Figure 4.2. Intact links are shown as solid and disrupted links are shown as dashed. The labels on the links show the cost functions $T_a(x_a)$. The repair durations for the links are $D_{I_1-S_1} = D_{I_1-I_2} = 20$ and $D_{I_2-I_3} = 10$.

The greedy heuristic would reconstruct links in the order $I_1 - S_1$, $I_1 - I_2$, and

$I_2 - I_3$ based on the immediate ratio of benefit to duration. However, repairing link $I_2 - I_3$ would provide a large benefit (66%) if repaired after link $I_1 - I_2$, and before link $I_1 - S_1$. Greedy methods encounter similar problems with links in series.

Therefore, a more intelligent method is needed, that reflects the dependencies between damaged links. In particular, the benefit of repairing a link can depend strongly on where it lies in the repair sequence. (Later in the chapter, we explore whether pathological behavior like the above is common in more realistic networks. In particular, we will compare our solution method to this "greedy" heuristic as a benchmark.)

At first glance, this is not encouraging, and one might think that sequence-dependent repair benefits would require us to enumerate all possible sequences. As formulated, the problem has $|\mathcal{B}|!$ feasible solutions, one for each permutation of damaged links; there is clearly a bijection between each such permutation and feasible values of $\mathbf{y}$ and $\mathbf{z}$.

However, there is a subsequence optimality condition that can be applied to greatly reduce the number of solutions to examine. Any repair sequence can be divided into two subsequences, $[b_1, \ldots, b_i]$ and $[b_{i+1}, \ldots, b_N]$, and we can consider "restricted" versions of the problem where the sets of links in the two subsequences are fixed, but the order within one or both subsequences can vary.

Theorem 1. *Let* $[b_1, b_2, \ldots, b_N]$ *be a repair sequence corresponding to an optimal solution of* (4.1) *and* (4.2)*, and let* $[b_1, \ldots, b_i]$, $[b_{i+1}, \ldots, b_N]$ *be any division into subsequences. For* every *restricted problem formed by fixing the links in the "head" in some permutation of* $\{b_1, \ldots, b_i\}$*, the optimal repair order*

in the "tail" is $[b_{i+1}, \ldots, b_N]$*. Likewise, for every restricted problem fixing the links in the tail in a specific order, the optimal repair order in the "head" is* $[b_1, \ldots, b_i]$*.*

Proof. We prove the first part of the claim; the second follows by the same argument.

The key observation is that $TSTT_t$ only depends on the *set* of links repaired before stage t, and not on the *order* in which they are repaired: the lower-level problem (4.2) only depends on the upper-level through the variables z_b^t, which are defined in the upper-level problem by a sum of y_b variables, which does not depend on order. Likewise, the total duration of any repair sequence (or subsequence) only depends on the D_b values of its links, and not the order in which they are repaired.

It follows that in any restricted problem of the form in the theorem, $TSTT_i$ is the same no matter what the ordering is in the head and tail subsequences. Likewise, the durations of the head and tail are the same for any permutation of the s within their subsequences. Nevertheless, the specific value of the objective function may depend on the order within these sequences. This means that the objective function can be decomposed into the sum of two independent areas, one corresponding to the head subsequence and the other to the tail (See Figure 4.3).

Therefore, the "restricted" problems for every possible permutation of $\{b_1, \ldots, b_i\}$ as head differ only by a constant term added to the objective, representing the different areas under the head subsequence. By assumption, $[b_{i+1}, \ldots, b_N]$ is the optimal tail when the head is $[b_1, \ldots, b_i]$, so it must be optimal for every other ordering too. □

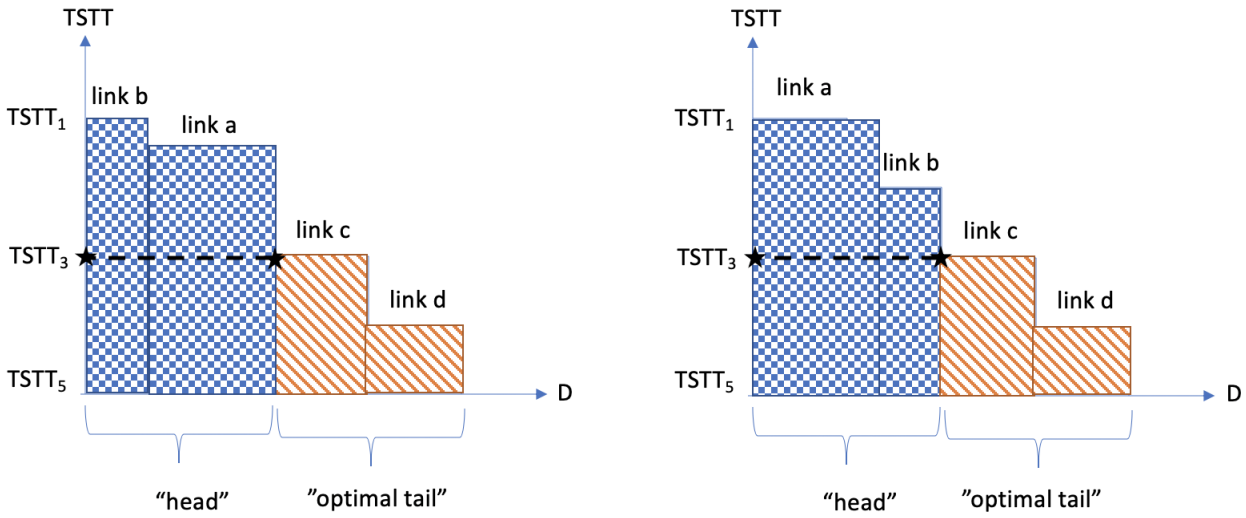

Figure 4.3: Demonstrating separability of the formulation by subsequence; $TSTT_3$ and the optimal ordering of links c and d is independent of the order of a and b.

The theorem extends trivially to division into more than two subsequences; once a partitioning into subsequences is fixed, the optimal ordering within each partition can be determined independently. The upshot is that we can formulate this sequencing problem using dynamic programming, since Theorem 1 shows that Bellman's optimality principle holds. This is the foundation of the heuristic we describe in the following sections.

4.3 Overview of solution method

This section provides a high-level description of our solution approach; more specifics and pseudocode are found in the following sections.

Using Theorem 1, we can reformulate our problem as the following dynamic program: define a state for each subset of $\mathcal{B}$ (representing a condition where a

particular subset of links has been repaired), create transitions between states differing by the addition of only one link (representing repair of that link), and equip each transition with a cost equal to the appropriate TSTT multiplied by the repair duration. Solving a shortest path problem on the resulting graph would produce the optimal solution. In practice, there are still too many states to build the entire graph — even though there are just $2^{|\mathcal{B}|}$ states, far fewer than the $|\mathcal{B}|!$ feasible sequences, this approach is intractable even with only 10–20 damaged links.

We therefore construct this graph incrementally, using heuristic procedures to only generate transitions and states which are likely to lead to good solutions. Our framework is a *bidirectional search* which grows this graph starting both at the initial state (no repairs) and at the terminal state (all links repaired). This graph has two search fronts, a forward front rooted at the initial state, and a backward front rooted at the terminal state.

At each iteration of the search, a state is chosen from either the forward search front or the backward search front. This state is used to construct successor (or predecessor) states. States on the forward front have a cost label representing the portion of the objective function accrued from the initial state up through that state. States on the backward front have a cost label representing the portion of the objective function accrued from that state until the terminal state. These cost labels are calculated exactly.

We also calculate heuristic upper and lower bounds on the cost of the optimal subsequence connecting states on opposite fronts; by Theorem 1 we can restrict our attention to the remaining links that lie between the states on the two fronts. These heuristics are based on the intuitive observation that the marginal benefit of repairing a link diminishes as more links are repaired,

which we can translate into bounds on sequences of repairs. (See the following sections and the numerical results for more investigation of this intuition.) These bounds are also used for early pruning, and to control which branches are created.

We finally adopt a beam search-like strategy for further restricting the number of states which are generated.

4.4 Bi-directional search

We propose a bi-directional, best-first search approach. Following conventions in this literature, we will use the subscripts f and b to indicate the forward and backward search directions, respectively. A collection of states that have been created, but not yet expanded into successor states, is maintained in a set *Open*. After expansion, states are removed from the *Open* set. As in the A^* algorithm, the search progresses by selecting a state from *Open* with minimum estimated cost (actual cost accrued thus far plus a heuristic estimate of the remaining cost), and expanding it by creating successor states. On the forward front, successors represent an additional link being repaired. On the backward front, a successor represents one fewer link being repaired (the search proceeds "backwards" from the state of full repair).

The root states for the two fronts are denoted by α for the forward front (nothing repaired), and ω for the backward front (everything repaired). At a particular state s on the forward front, let $Y_f(s)$ denote the set of repaired links. If s is on the backward front, $Y_b(s)$ denotes the set of links which remain unrepaired. With this convention, $Y_f(\alpha)$ and $Y_b(\omega)$ are both empty, and successors on both fronts are formed by adding one link. We also maintain labels $g_f(s)$ and $g_b(s)$, representing the cost of the best-known partial sequence

from the initial state to s on the forward front, and from s to the final state on the backward front, and predecessors $p_f(s)$ and $p_b(s)$ which will be used to trace the optimal path.

The search is aided by a heuristic function $h(s)$, which estimates the cost of the partial sequence connecting state s to another state. The heuristic uses front-to-front evaluations, i.e., the cost $f(s)$ is estimated from state s on one front to eligible states on the other front. We will use $ub(s, s')$ and $h(s, s')$, respectively, to represent the heuristic upper and lower bounds on the cost between states s and s'. The details for calculating $h(s)$ are provided in the next section.

The bidirectional search heuristic is stated in Procedure 2. Our procedure will require repeated solution of the traffic assignment problem, with the same OD matrix $\mathbf{d}$, but a different link set. We will use the notation $TAP(A)$ as a shorthand for "solve TAP with arc set A, and return the total system travel time at equilibrium," and $TSTT(s)$ to mean $TAP(A)$ with the network repairs corresponding to state s.

4.5 Heuristic Function

This section first derives heuristic lower and upper bounds $h(s, s')$ and $ub(s, s')$ of the cost of connecting states s and s' on opposite search fronts through an optimal repair subsequence, then uses these to construct a heuristic $f(s)$ of the total cost from state s to the target (if s is on the forward front) or initial state (if s is on the backward front). If this heuristic is a lower bound on the actual repair cost, it is called *admissible* and the BidirectionalSearch will be exact, terminating with an optimal repair sequence. [42]. However, the larger the values of $f(s)$, the fewer states will be generated, so there is a

Algorithm 2: BIDIRECTIONALSEARCH

Input: $\mathcal{B}, D, r$ **Output:** $incumbent$

Initialize $Open_f \leftarrow \varnothing, Open_b \leftarrow \varnothing, UB \leftarrow \infty\, incumbent = \varnothing$;
Initialize $g_f(\alpha) \leftarrow 0$, $g_b(\omega) \leftarrow 0$, $p_f(\alpha) \leftarrow \varnothing$, $p_b(\omega) \leftarrow \varnothing$, $Y_f(\alpha) \leftarrow \varnothing$, $Y_b(\omega) \leftarrow \varnothing$;
Set $counter \leftarrow 0$;
Calculate $TSTT(\alpha) \leftarrow TAP(\mathcal{A} \setminus \mathcal{B})$ and $TSTT(\omega) \leftarrow TAP(\mathcal{A})$;
bb, **wb** $\leftarrow$ PREPROCESSING($\mathcal{B}, \mathbf{D}$) (see Procedure 6) ;
while $Open_f \neq \varnothing$ ***and*** $Open_b \neq \varnothing$ ***and*** $\max(\min_{s \in Open_f} f(s), \min_{s \in Open_b} f(s)) < UB$ **do**
  if $|Open_f| < |Open_b|$ **then**
    $Open_f, Open_b, UB, S^{incumbent} \leftarrow$ EXPANDF($Open_f, Open_b, UB, incumbent$) (see Procedure 3)
  else
    $Open_f, Open_b, UB, S^{incumbent} \leftarrow$ EXPANDB($Open_f, Open_b, UB, incumbent$) (see Procedure 4)
  if $counter \equiv 0 \pmod r$ **then**
    foreach $s' \in Open_b$ **do**
      $UB, incumbent \leftarrow$ FINDBOUNDS($s', Open_b, backward, UB, incumbent$) ;
    foreach $s' \in Open_f$ **do**
      $UB, incumbent \leftarrow$ FINDBOUNDS($s', Open_f, forward, UB, incumbent$) ;
    $Open_f, Open_b \leftarrow$ BEAMSEARCH($Open_f, Open_b$) (see Procedure 11);
  $counter \leftarrow counter + 1$;

Algorithm 3: ExpandF

Input: $Open_f, Open_b, UB, incumbent$

Output: $Open_f, Open_b, UB, incumbent$

Choose $s \in \arg\min_{s \in Open_f} f(s)$;

Remove state s from $Open_f$;

foreach $s' \in Open_b$ **do**

  $NV = \mathcal{B} \setminus \{Y_f(s) \cup Y_b(s')\}$;

  $UB, incumbent \leftarrow$ CheckSoln$(s, s', NV, UB, incumbent)$ (see Procedure 5) ;

if $f(s) > UB$ **then**

  Non-optimality proven, no need to expand, return with the current values;

Let $EXT \leftarrow \mathcal{B} \setminus Y_f(s)$ represent potential expansions of s;

Initialize $children \leftarrow \varnothing$;

foreach $b \in EXT$ **do**

  $create \leftarrow$ Branching$(s, \ell, forward)$ (see Procedure 10);

  if *create* **then**

    Let $Y_{temp} \leftarrow Y_f(s) \cup \{b\}$;

    Calculate $TSTT_{temp} \leftarrow TAP((\mathcal{A} \setminus \mathcal{B}) \cup Y_{temp})$;

    Let $g_{temp} \leftarrow g(s) + D_b TSTT_{temp}$;

    Let s'' be the state for which $Y_f(s'') = Y_{temp}$ (if any);

    if s'' does not exist or $g_{temp} < g_f(s'')$ **then**

      Create s'' if it does not exist;

      Update $g_f(s) \leftarrow g_{temp}$;

      Update $p_f(s) \leftarrow s$;

    Add s'' to *children*;

foreach $s'' \in children$ **do**

  $UB, incumbent \leftarrow$ FindBounds$(s'', Open_b, forward, UB, incumbent)$ (see Procedure 7) ;

  $Open_f, Closed_f, incumbent \leftarrow$ Pruning$(Open_f, s'', incumbent, UB)$ (see Procedure 9) ;

Algorithm 4: ExpandB

ExpandB function:
Input: $Open_f, Open_b, UB, incumbent$
Output: $Open_f, Open_b, UB, incumbent$
Choose $s \in \arg\min_{s \in Open_b} f(s)$;
Remove state s from $Open_b$;
foreach $s' \in Open_f$ **do**
    $NV = \mathcal{B} \setminus \{Y_f(s) \cup Y_b(s')\}$;
    $UB, incumbent \leftarrow$ CheckSoln$(s', s, NV, UB, incumbent)$ (see Procedure 5) ;
if $f(s) > UB$ **then**
    Non-optimality proven, no need to expand, return with the current values;
Let $EXT \leftarrow \mathcal{B} \setminus Y_f(s)$ represent potential expansions of s;
Initialize $children \leftarrow \varnothing$;
foreach $b \in EXT$ **do**
    $create \leftarrow$ Branching$(s, \ell, backward)$ (see Procedure 10);
    if $create$ **then**
        Let $Y_{temp} \leftarrow Y_b(s) \cup \{b\}$;
        Calculate $TSTT_{temp} \leftarrow TAP(\mathcal{A} \setminus Y_{temp})$;
        Let $g_{temp} \leftarrow g(s) + D_b TSTT_{temp}$;
        Let s'' be the state for which $Y_f(s'') = Y_{temp}$ (if any);
        if s'' does not exist or $g_{temp} < g_b(s'')$ **then**
            Create s'' if it does not exist;
            Update $g_b(s) \leftarrow g_{temp}$;
            Update $p_b(s) \leftarrow s$;
        Add s'' to $children$;
foreach $s'' \in children$ **do**
    $UB, incumbent \leftarrow$ FindBounds$(s'', Open_f, backward, UB, incumbent)$ (see Procedure 7) ;
    $Open_b, Closed_b, incumbent \leftarrow$ Pruning$(Open_b, s'', incumbent, UB)$ (see Procedure 9) ;

Algorithm 5: CHECKSOLN

Input:s, s', NV, UB, *incumbent* **Output:**UB, *incumbent*

```
if NV = ∅ then
    g ← g(s) + g(s') if g < g(incumbent) then
        g(incumbent) ← g;
        g ← (s, s');
if |NV| = 1 then
    Identify b ∈ NV;
    g ← g(s) + D_b TSTT(s) + g(s');
    if g < g(incumbent) then
        g(incumbent) ← g;
        g ← (s, s');
UB ← min{UB, g(incumbent)} ;
```

tension between guaranteeing optimality (low f values) and tractability.

We propose a custom heuristic, based on the intuition that the marginal impact of repairing a link is likely higher when most other links are unrepaired and there are fewer alternate routes, and lower once the network is nearly restored to full functionality. It is not hard to construct counterexamples where this principle does not hold. Still, in our numerical experiments, this procedure performs well, suggesting that such counterexamples are rare in practice. Our procedure for calculating f also contains a weaker, "backstop" bound which is applied if this principle is clearly violated.

As a preprocessing step, for each damaged link b we calculate heuristic upper and lower bounds on the *benefit* of rebuilding this link, calculated as the reduction in TSTT when link b is reconstructed. Clearly this benefit depends on where in the repair sequence b lies; we will use $rb_b(\mathbf{y})$ to denote this *realized benefit* of repairing b if the full sequence is given by $\mathbf{y}$.

The heuristic upper and lower bounds on the realized benefit are denoted by bb_b and wb_b (the mnemonic is "best benefit" and "worst benefit"). We calculate bb_b as the reduction in TSTT if b is repaired first: $bb_b = TSTT(\mathcal{A} \setminus \mathcal{B}) - TSTT((\mathcal{A} \setminus \mathcal{B}) \cup \{b\})$. Likewise, wb_b is the reduction in TSTT if b is repaired last: $wb_b = TSTT(\mathcal{A} \setminus \{b\}) - TSTT(\mathcal{A})$. Our heuristic is derived under the assumption that $wb_b \leq rb_b(\mathbf{y}) \leq bb_b$ for every damaged link b and every feasible $\mathbf{y}$. This preprocessing step requires solving $2|\mathcal{B}| + 2$ instances of TAP, and the pseudocode is given in Procedure 6. The bb_b and wb_b estimates are then updated, if necessary, every time a TAP instance is solved during the bidirectional search process.

Algorithm 6: PREPROCESSING

Input: $\mathcal{B}, \mathbf{D}, TSTT(\alpha), TSTT(\omega)$ **Output:** $\mathbf{bb}, \mathbf{wb}$

foreach $b \in \mathcal{B}$ **do**

- $TSTT \leftarrow TAP((A \setminus \mathcal{B}) \cup \{b\})$;
- $bb_b \leftarrow TSTT(\alpha) - TSTT$;
- $TSTT \leftarrow TAP(A \setminus \{b\})$;
- $wb_i \leftarrow TSTT - TSTT(\omega)$

4.5.1 Bounds on the cost connecting states

Here we describe how to calculate upper and lower bounds on the optimal cost of connecting two states on opposite search fronts, using the upper and lower bounds on the benefits of repairing individual links. For concreteness, let s and s' be on the forward and backward search fronts, respectively. These two states are *compatible* if $Y_f(s) \cap Y_b(s') = \emptyset$, representing that a feasible solution can be formed by connecting them by a suitable subsequence of intermediate links $NV = \mathcal{B} \setminus (Y_f(s) \cap Y_b(s'))$. Let $n = |NV|$, and assume s and s' are compatible (else we can ignore the possibility of connecting them).

If $n = 0$ or 1, then s and s' together form a feasible solution, whose head is given by the predecessors leading from α to s, and whose tail is given by the predecessors leading from s' to ω. In such a case, we calculate the total cost of this solution, and update our global upper bound UB on the optimal cost if possible. (Procedure 5) The remainder of this subsection assumes $n > 1$, so there are at least two additional links that must be sequenced for repair to connect the fronts.

By Theorem 1, we can decompose the cost of any solution passing through states s and s' into three parts: the cost incurred between α and s (where the optimal order is assumed already known and represented by the labels at s); the cost incurred between s and s' (which is unknown), and the cost incurred between s' and ω (the optimal value of which is again assumed already known). This unknown cost in the connecting segment is represented by the gray shaded area of the middle region in Figure 4.4.

We start by calculating a lower bound on this cost, assuming that $wb_b \leq rb_b$ in any sequence we consider. Without loss of generality we assume that the links in NV are links $\{1, \ldots, n\}$, and that they are indexed in order of decreasing worst benefit-to-repair duration ratio:

$$\frac{wb_1}{D_1} \geq \frac{wb_2}{D_2} \geq \cdots \geq \frac{wb_n}{D_n}.$$

Theorem 2. *If $wb_b \leq rb_b(\mathbf{y})$ in any feasible sequence $\mathbf{y}$, then the cost of any segment connecting s and s' is at least $TSTT(s')\sum_{i=1}^{n} D_i + \sum_{1\leq i\leq j\leq n} D_i wb_j$.*

Proof. The total system travel time at any stage between s and s' is the sum of $TSTT(s')$ and the realized benefit of all links yet to be repaired (for example,

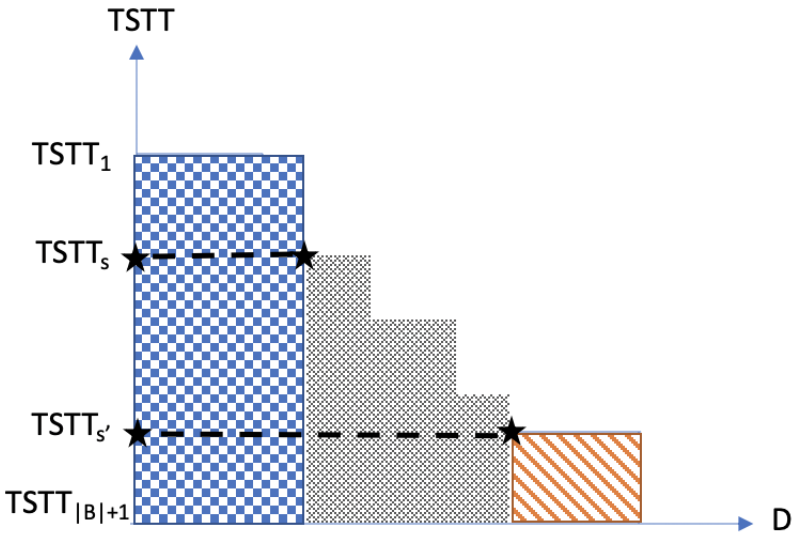

Figure 4.4: Dividing the cost of a solution into a head (left section; represented by state s); a middle section (yet to be determined); and a tail (right section; represented by state s')

in Figure 4.4 , TSTT decreases stepwise from $TSTT(s') + rb_1 + rb_2 + rb_3$ as each link is repaired.), and the total cost accrued during this stage is found by multiplying by its duration. Consider any permutation of the links in NV, and let $b(i)$ denote the link repaired in the i-th position in this permutation. We now prove the following inequality, where the left-hand side represents the cost of the permutation; adding $TSTT(s') \sum_{i=1}^{n} D_i$ to both sides will then establish the theorem.

$$\sum_{1 \le i \le j \le n} D_{b(i)} rb_{b(j)} \ge \sum_{1 \le i \le j \le n} D_i wb_j \, .$$

The sum on the right-hand side contains a term for every possible value of i and j satisfying $1 \le i \le j \le n$. We now show that no matter what permutation is considered, for all $1 \le i \le j \le n$ the left-hand side either contains a term of the form $D_i wb_j$ or $D_j wb_i$, but not both. Indeed, if i is repaired before j in the permutation, then there is no term $D_j wb_i$, and if j is repaired before i there is no term $D_i wb_j$. So, taken as an *unordered* set, the indices $\{b(i), b(j)\}$ in the left-hand sum are all distinct. There are $n(n+1)/2$ sets of the form $\{i, j\}$ with $i, j \in \{1, \ldots, n\}$, exactly the number of terms in the left-hand sum, so this sum must therefore range over all possibilities of $\{i, j\}$, either in the order $D_i wb_j$ or $D_j wb_i$.

Therefore, for each set $\{i, j\}$ with $i, j \in \{1, \ldots, n\}$, we can pair a term in the left-hand side with a corresponding term on the right-hand side, and this will exhaust all terms in both sums. If the inequality holds for each corresponding pair of terms, we are done. These $n(n+1)/2$ inequalities will either take the form $D_i rb_j \ge D_i wb_j$ or $D_j wb_i \ge D_i wb_j$. In the first case the inequality is clear, since $rb_j \ge wb_j$ by assumption. In the second case, the term on the right-hand side must satisfy $i \le j$. Dividing by both durations and applying $rb_j \ge wb_j$, it

is sufficient to show $wb_i/D_i \geq wb_j/D_j$, which is true by the assumed indexing of the links in NV. □

A corresponding upper bound $TSTT(s')\sum_{i=1}^{n} D_i + \sum_{1\leq i\leq j\leq n}\sum_{j=i}^{n} bb_j D_i$ on the connecting cost can be derived by the same argument, *mutatis mutandis*, ordering the links by ascending bb_b/D_b. As an easy corollary of Theorem 2, if the failed links are independent in the sense that rb_b does not depend on $\mathbf{y}$, the optimal strategy is to repair them in decreasing order of rb_b/D_b.

Now, there are cases where the assumption $wb_b \leq rb_b(\mathbf{y})$ is clearly false; in particular, if $\sum_{b\in NV} wb_b > TSTT(s) - TSTT(s')$, then $rb_b < wb_b$ for at least one link. In such a case we use the weaker, trivial bound $(TSTT(s) - TSTT(s'))\min_{b\in NV}\{D_b\}$. Likewise, in calculating the upper bound, if $\sum_{b\in NV} bb_b < TSTT(s) - TSTT(s')$, then the assumption $rb_b(\mathbf{y}) \leq bb_b$ is clearly false, and we use the weaker bound $(TSTT(s) - TSTT(s'))\sum_{b\in NV} D_b$.

In our numerical experiments, we observe that this weaker "backstop" is rarely used (less than 3% of calculations).

4.5.2 Bounds on the total cost from each state

The bidirectional search algorithm requires an estimate of the *total* cost from s (on the forward front) to the target state ω, or of the total cost from α to s' (on the backward front). This section shows how we can translate the lower and upper bounds $h(s, s')$ and $ub(s, s')$ for a specific pair of states s and s' to bounds on the cost of connecting the particular state s to ω via *any* compatible state on the backward front (or connecting α to s' via any compatible state on the forward front). We will use $f(s)$ and $ub(s)$ to denote these calculated bounds. For specificity, we will assume s is on the forward front; the procedure

is essentially the same on the backward front.

To find the lower bound $f(s)$ on the optimal cost, we find the weakest of the lower bounds connecting s to s', since any of these might be used in the optimal solution:

$$f(n) = \min_{s' \in Open_b} \{g_f(s) + h(s, s') + g_b(s')\}.$$

This lower bound is used to prioritize states to explore in the bidirectional search.

For the upper bound $ub(s)$, we use the strongest of the upper bounds connecting s to s'. This is justified since the cost of the optimal solution is no greater than the cost of the upper bound on the (s, s') pair it uses, and this cost is no greater than its upper bound. Therefore

$$ub(n) = \min_{s' \in Open_b} \{g_f(s) + ub(s, s') + g_b(s')\}$$

even though the states s' used in the minimizations defining f and ub may be different. In our procedure, these bounds are updated every r iterations, and progressively get tighter as the accuracy of the wb and bb estimates improves. This upper bound is used for pruning of other states.

The pseudocodes for the heuristic calculation are in Procedures 7 and 8.

4.6 Other Speedup Techniques

We further enhance the bidirectional search in several ways. First, we use the upper bounds to prune search branches by comparing the lower bound on its cost $f(s)$ to UB, the best-known global upper bound on the optimal cost. If $f(s) > UB$, then the state can be pruned, and we do not need to examine

Algorithm 7: FINDBOUNDS

Input: s, $Open$, sd, UB, $incumbent$ **Output:** UB, $incumbent$ $E \leftarrow \varnothing$;
foreach $s' \in Open$ **do**
  if $Y(s) \cap Y(s') = \varnothing$ **then**
    $E \leftarrow E \cup \{s'\}$;
$f \leftarrow \infty, ub \leftarrow \infty$;
foreach $s' \in E$ **do**
  $f(s') \leftarrow g(s) + g(s')$, $ub(s') \leftarrow g(s) + g(s')$;
  $NV \leftarrow \mathcal{B} \setminus \{Y(s) \cup Y(s_n)\}$;
  if $sd = f$ **then**
    $TSTT_{fw} = TSTT(s), TSTT_{bw} = TSTT(s')$;
  else
    $TSTT_{bw} = TSTT(s'), TSTT_{bw} = TSTT(s)$;
  $UB, incumbent, h(s, s'), ub(s, s') \leftarrow$
   BOUNDSFORPAIR$(s, s', sd, TSTT_{fw}, TSTT_{bw}, NV, UB, incumbent)$
   (see Procedure 8) ;
  $f \leftarrow min\{f(s') + h(s, s'), f\}$, $ub \leftarrow \min\{ub(s') + ub(s, s'), ub\}$
$f(s) \leftarrow f$, $ub(s) \leftarrow ub$;

Algorithm 8: BOUNDSFORPAIR

Input:s, s', sd, $TSTT_{fw}$, $TSTT_{bw}$, NV, UB, $incumbent$ **Output:**UB, $incumbent$, $h(s, s')$, $ub(s, s')$

If $sd = b$, swap s, s' ;

UB, $incumbent \leftarrow$ CHECKSOLN(s,s',NV,UB,incumbent);

Sort links in NV in descending order of wb_b/D_b. **if** $\sum_i wb_i < TSTT_{fw} - TSTT_{bw}$ **then**

  $f(s, s') \leftarrow 0$;

  $sum \leftarrow \sum_i wb_i$; **foreach** $b \in NV$ **do**

    $f(s, s') \leftarrow f(s, s') + D_i sum$;

    $sum \leftarrow sum - w_i$;

else

  $f(s, s') \leftarrow (TSTT_{fw} - TSTT_{bw})(\min_b\{D_b\})$;

Sort links in NV in descending order of bb_b/D_b. **if** $\sum_i bb_i > TSTT_{fw} - TSTT_{bw}$ **then**

  $ub(s, s') \leftarrow 0$;

  $sum \leftarrow \sum_i b_i$;

  foreach $b \in NV$ **do**

    $ub(s, s') \leftarrow ub(s, s') + D_i sum$;

    $sum \leftarrow sum - b_i$;

else

  $ub(s, s') \leftarrow (TSTT_{fw} - TSTT_{bw}) \sum_b D_b$;

any states leading from it. Procedure 9 provides the formal statement for the pruning procedure. Furthermore, we obtain a feasible solution every r iterations by expanding the least heuristic cost node to completion using a greedy approach. If necessary, the upper bound is then updated.

Second, we cache TAP solutions, to obviate unnecessary computations of TSTT for the same set of links in different order.

Third, we limit branching by avoiding sequences that are likely to have locally suboptimal subsequences. This is heuristically done by identifying the last

link repaired in s (obtained from the predecessor label), and computing the ratio of its estimated benefit and repair duration to that of the candidate link in the new branch. We expect that links with a higher benefit-duration ratio should be repaired first, and avoid creating branches that violate this ordering. Procedure 10 describes this process formally.

Fourth, our procedure requires repeated solution of TAP as a subproblem. As a convex nonlinear optimization problem, TAP can only be solved approximately, and the stopping criterion should be chosen to balance accuracy without wasting time on unnecessary precision. Based on the guidance from Patil et al. [79], the value of $TSTT$ is almost always within 1% of its equilibrium value when the relative gap (a common convergence criterion) is below 10^{-4}. A tighter gap of 10^{-5} or 10^{-6} is sometimes used to ensure stability of individual link flows. In our experiments we test both 10^{-4} and 10^{-6}.

Finally, to reduce the number of states stored in memory for large instances and to further speed up the search process, we incorporate a beam search strategy. We define the *level* of a state to be the number of links in its set Y. After every r iterations, we prune states with costs over $(100 + k)\%$ of the lowest heuristic cost at that level. Increasing k improves the solution quality at the expense of increasing state space size and overall computation time. The parameters k and r can be tuned as required. We use an adaptive strategy where k is decreased for later iterations as the wb and bb estimates improves every iteration as well as the heuristic cost estimate approaches the unknown optimal cost. Procedure 11 describes this formally. The beam search strategy can be modified to only apply to particular levels or after a certain number of states have been generated.

Algorithm 9: PRUNING

Input:$Open$, s, $incumbent$, UB **Output:**$Open$, $Closed$, UB
if $f(s) \leq UB$ **then**
    $Open \leftarrow Open \cup \{s\}$;

Algorithm 10: BRANCHING

Input:s, b, sd **Output:**$create$
$create \leftarrow False$;
Let $b' \leftarrow Y(s) \setminus Y(p(s))$ be the last link changed in s;
if $sd = f$ *and* $wb_b/D_b < bb_{b'}/D_{b'}$ **then**
    $create \leftarrow True$;
if $sd = b$ *and* $wb_{b'}/D_{b'} < bb_b/D_b$ **then**
    $create \leftarrow True$;

Algorithm 11: BEAMSEARCH

***BeamSearch*:**
Input:$Open$, $Closed$, k **Output:**$Open$, $Closed$
foreach $i \in \{1, \ldots, N\}$ **do**
    Let $Open_f^i \leftarrow \{s \in Open_f : |Y_f(s)| = i\}$;
    Let $Best_f^i$ be a subset of $Open_f^i$ of size k, with the largest $f(s)$ values;
    Let $Open_b^i \leftarrow \{s \in Open_b : |Y_b(s)| = i\}$;
    Let $Best_b^i$ be a subset of $Open_b^i$ of size k, with the largest $f(s)$ values;
$Open_f \leftarrow \cup_{i=1}^{N} Best_f^i$;
$Open_b \leftarrow \cup_{i=1}^{N} Best_b^i$;

4.7 Numerical experiments

This section describes our tests of the procedures described above. All code is available on Github [40]; for solving the traffic assignment subproblems we use our implementation of Algorithm B [17]. For testing, we used the Sioux Falls and Anaheim networks from the Transportation Test Networks repository [96]. Sioux Falls has 24 nodes and 76 links; Anaheim has 416 nodes and 914 links.

We compare our method with three other heuristics, which are simpler in form. The greedy method (GM) involves constructing a sequence myopically, at each stage choosing a link to repair that leads to the greatest immediate benefit. Specifically, assume that we have already repaired a set of links Y. The next bridge chosen by GM maximizes

$$\frac{(TSTT(Y) - TSTT(Y \cup \{i\})}{D_i}$$

over all unrepaired links $i \in \mathcal{B} \setminus Y$. This approach requires solving $O(|\mathcal{B}|^2)$ instances of TAP.

The "lazy greedy method" (LGM) is even simpler, repairing bridges in order of their best benefit-to-repair time ratios bb_i/D_i. This method requires solving $O(|\mathcal{B}|)$ TAP instances, and represents the intuitive idea that links with higher benefits and shorter repair times should be repaired first. (Recall that Theorem 2 implies that this is in fact the optimal repair order if the damaged links are independent and do not affect each other.)

The simplest method, IF (importance factor), is to simply repair links in decreasing order of their pre-disruption flow; this represents the intuitive solution of repairing the most heavily-used links first. This method is naïve, but requires solving only a single TAP instance.

4.7.1 Numerical experiments

Three damage levels ($|\mathcal{B}| = \{8, 16, 32\}$) are considered for each network to generate problem instances. Two different sampling methods are used to generate set $\mathcal{B}$. For the first method, $\mathcal{B}$ is obtained by weighted sampling of links using pre-disaster flows as weights. This is intended to choose impactful and potentially disruptive links. For the second method, $\mathcal{B}$ is obtained by location based sampling of the links. A node is randomly chosen and assigned as the disaster center, followed by a set of affected nodes based on distance from the disaster center. $\mathcal{B}$ is then chosen from links adjacent to this nodeset using pre-disaster flow weighted sampling.

Each damaged link i is assigned a repair duration D_i based on its capacity C_i and length ζ_i, reflecting that higher-capacity and longer links generally have more lanes and require more labor to reconstruct. The repair length for the links with higher $C_i\zeta_i$ are sampled from a gamma distribution with higher mean and variance, whereas the repair length for the links with a lower product are sampled from a gamma distribution with lower mean and variance. Specifically, the mean and variance parameters for the gamma distribution with respect to the percentiles of $C_i\zeta_i$ are given in Table 4.1.

Percentile of $C_i\zeta_i$	Mean (in weeks)	Variance (in days)
$0^{th} - 25^{th}$	6	17
$25^{th} - 50^{th}$	10	22
$50^{th} - 75^{th}$	12	24
$75^{th} - 95^{th}$	14	32
$95^{th} - 100^{th}$	16	39

Table 4.1: Repair duration sampling parameters.

The beam search is used on a search front once the iteration number exceeds

250 (this is another hyperparameter chosen empirically). This avoids early pruning of states and allows complex interactions to be captured.

In the results below, M refers to the bidirectional search heuristic, where the traffic assignment subproblems are solved to a relative gap of 10^{-6}. RM (relaxed method) uses a slightly weaker convergence threshold of 10^{-4} for the TAP subproblems. Both M and RM produced optimal solutions when there are 6 bridges and enumeration is possible.

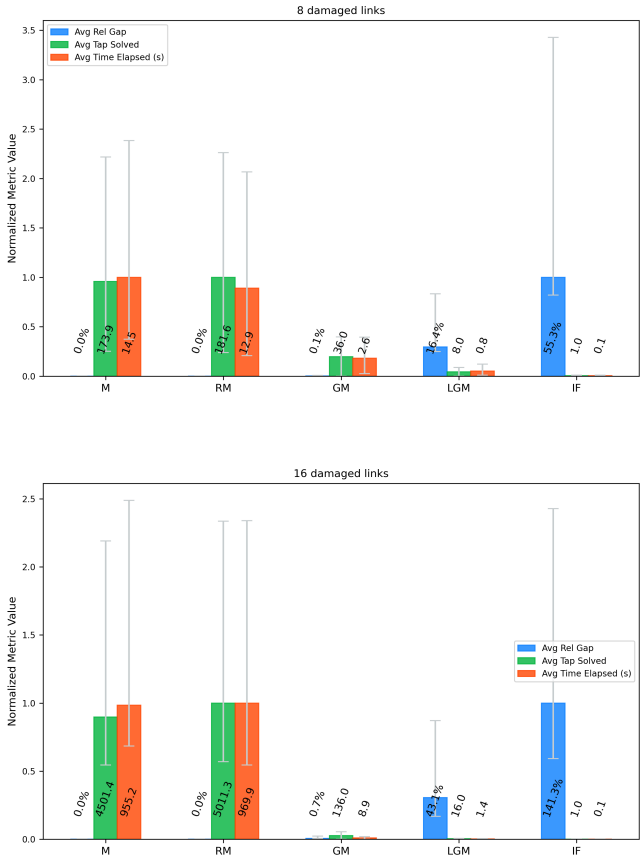

Figure 4.5: Comparison on Sioux Falls, flow weighted sampling.

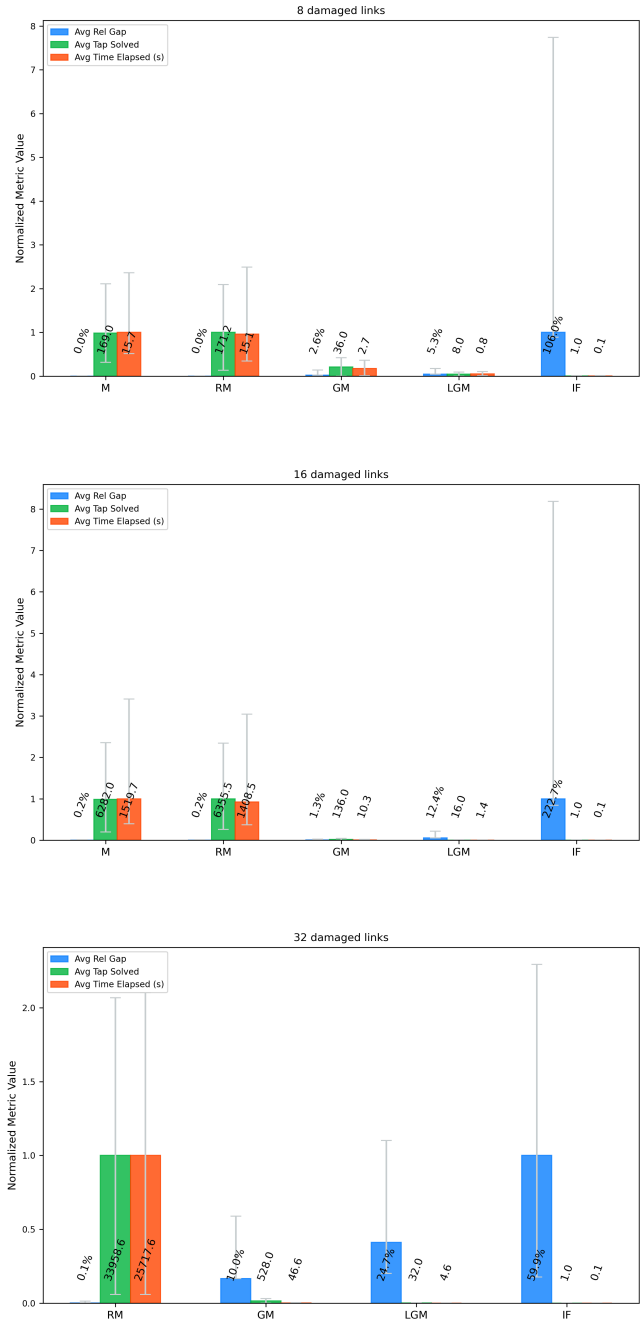

Figure 4.6: Comparison on Anaheim, flow-weighted sampling.

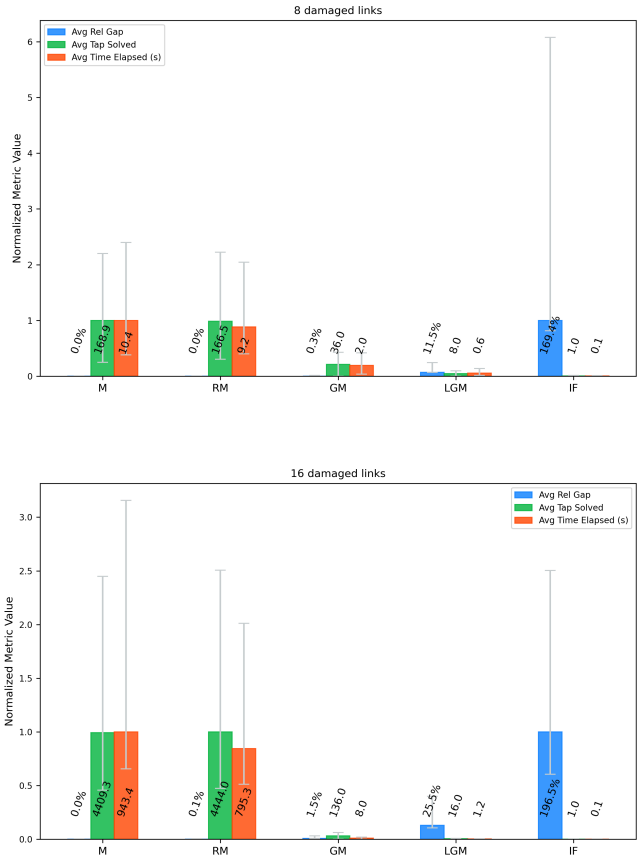

Figure 4.7: Comparison on Sioux Falls, location based sampling.

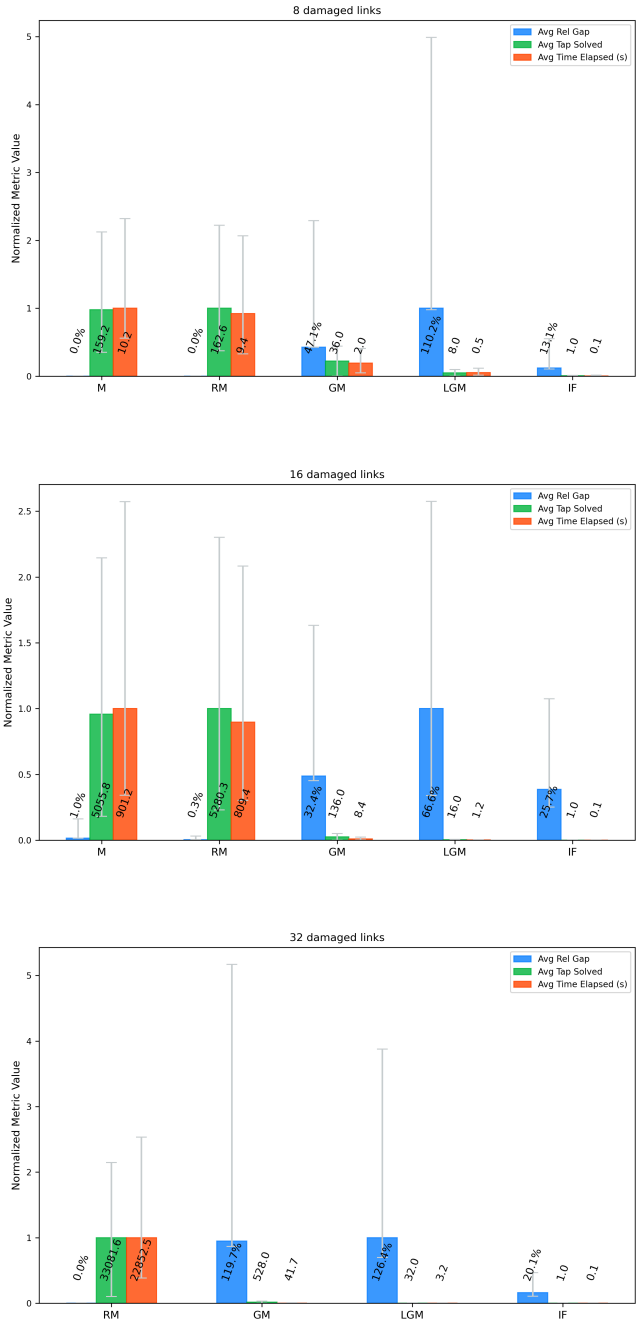

Figure 4.8: Comparison on Anaheim, location based sampling.

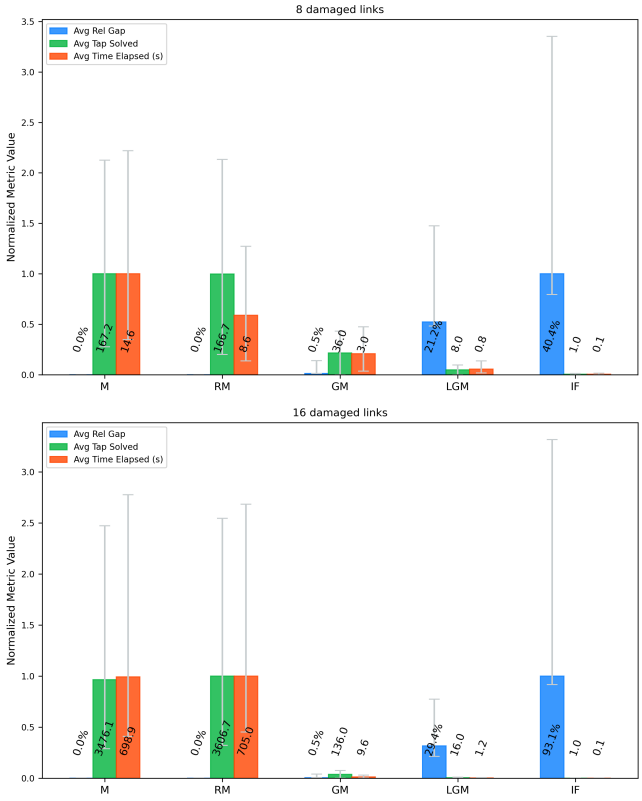

Figure 4.9: Comparison on Sioux Falls with increased demand, flow weighted sampling.

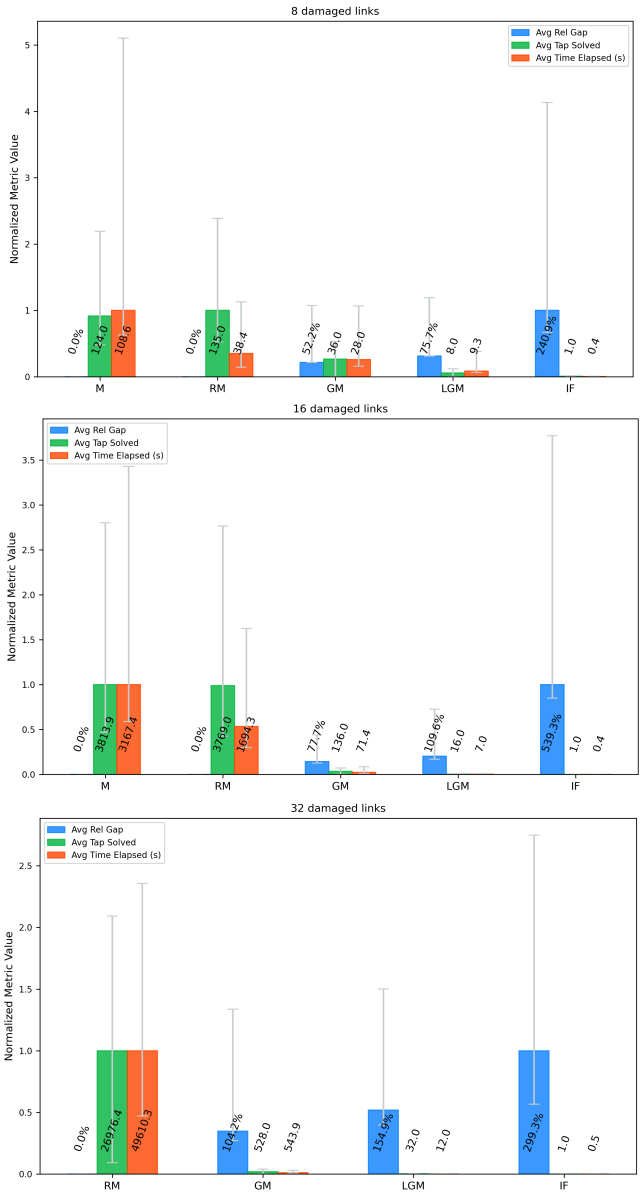

Figure 4.10: Comparison on Anaheim with increased demand, flow-weighted sampling.

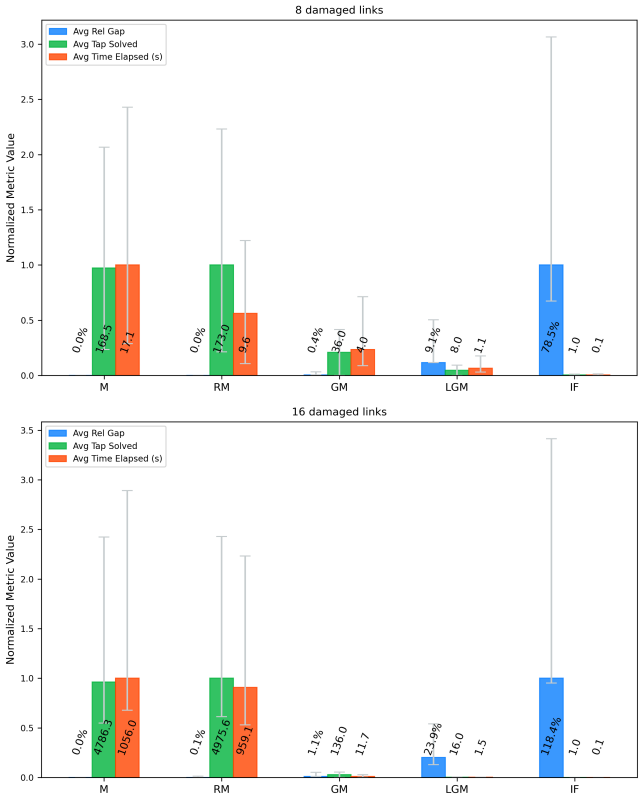

Figure 4.11: Comparison on Sioux Falls with increased demand, location based sampling.

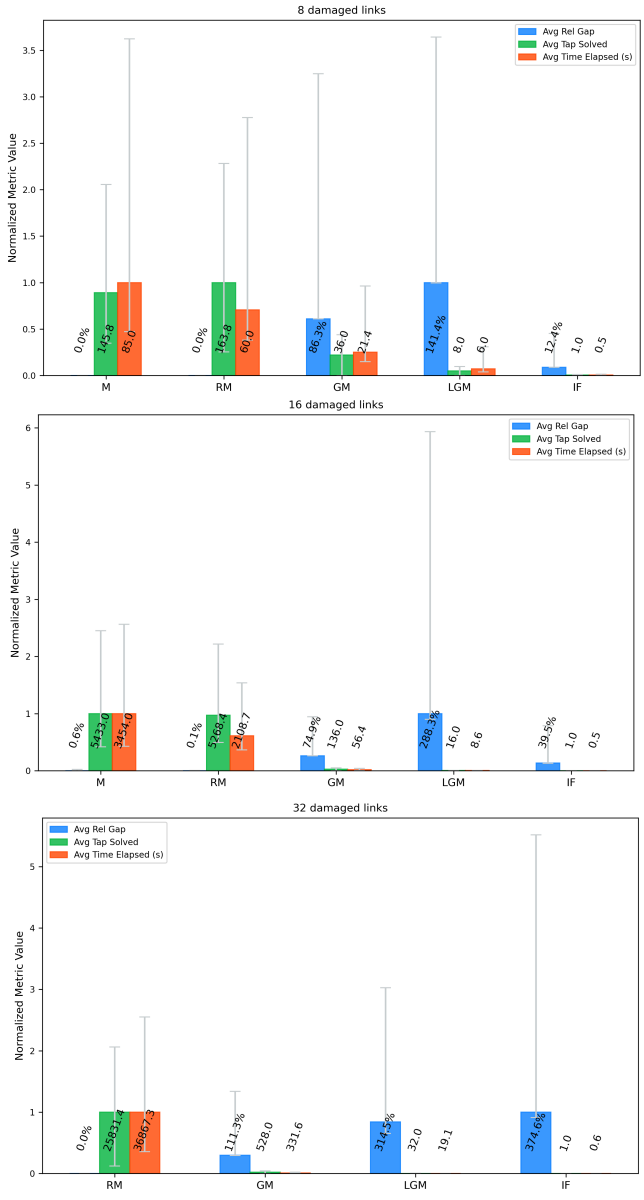

Figure 4.12: Comparison on Anaheim with increased demand, location based sampling.

Figures 4.5–4.12 compare the performance of five methods (M, RM, GM, LGM, and IF). These figures show the solution quality and computational requirements, as expressed by (1) the percentage gap between the objective function value and the best-known objective function value[1]; (2) the number of TAP subproblems which were solved; and (3) the computation time. The results were averaged over ten random instances. Since these three quantities are in different units, the height of the bars in these figures is normalized, with 1 indicating the highest average value across the methods. Lastly, the error bars represent the range of values – as the quantities are normalized, the range represented here is in terms of multiple of the highest average.

Figures 4.5–4.8 consider the networks with standard parameters and compares the performances differences of the methods across different impact levels and sampling methods. Figures 4.9–4.12 report similar comparisons, but on more challenging instances created by multiplying all demand values by 4, representing higher congestion.

The performance differences between the bidirectional search and greedy heuristics is fairly small, but increases with the size of the problem scale — for instance, with flow-weighted sampling, when there are 32 damaged links, a significant difference in solution quality is seen. With location-based sampling, the advantage of bidirectional search over the greedy methods is larger even with fewer damaged links. Location-based sampling apparently generates more challenging problem instances for simpler heuristics. This is likely because the damaged links are closer to each other geographically and thus interact more heavily, recalling the example in Figure 4.2.

[1]For small instances this can be found by enumeration; for large instances we take the minimum value over all the methods.

For the cases with increased demand, the performance differential between our methods (M and RM) and the other heuristics is still relatively small for the Sioux Falls experiment. As a small network, there are few potential paths between most OD pairs. A larger performance gap exists for the larger networks, where there are more link dependencies. The objective gap between the greedy method (GM) and our proposed method (M) for 8 damaged links is about 50% for flow-weighted sampling and 86% for location-based sampling. For the same network, increasing the number of damaged links to 16 provides objective gap of 77% and 74%.

Solving the TAP instances to a relaxed gap of 10^{-4} also did not affect performance in any noticeable way. In the Anaheim network this reduced computation time significantly without compromising performance. There was a slight difference in the number of TAP instances solved; we hypothesize this is due to the slight error in the TAP objective function introduced by looser convergence criteria leading to pruning different branches of the search tree.

Based on this observation, we use only RM for larger problem instances, since it saves computation time without affecting solution quality. The objective gap rises above 110% for 32 damaged link instances[2]. The computation time needed for the bidirectional search on the 32 link instances is significantly higher than the instances with fewer damaged links. However, this concern can be remedied by changing some of the settings in the beam search procedure, trading computation time and solution accuracy. Additionally, for several location-based sampling instances, IF is seen to outperform GM, especially when situations like Figure 4.2 arise.

[2]The results for the experiments with 32 damaged links are averaged over 5 instances as they take significantly longer

Our proposed method produces substantially higher-quality solutions than all three of the simpler heuristics (GM, LGM, and IF). Although they require less time, the computation times we report are very reasonable given the time scales involved in planning for disaster recovery. We also note the large variance in performance of these alternative methods, as shown by the error bars in the plots, suggesting that the simpler heuristics do not provide reliable solution quality.

4.8 Conclusion

In this chapter, we provide a bi-directional search approach for the link reconstruction sequencing problem in a post-disaster transportation network problem. We exploit properties of the traffic assignment problem to develop specialized branching and pruning procedures based on upper and lower estimates of solution cost, and embed the resulting procedure in a beam search. The resulting heuristic is then able to solve larger problem instances than the heuristic approaches studied in the literature for this problem. Our experiments indicates substantial improvements in performance over simpler heuristics.

Future research should explore relaxing assumptions made in this model. Perhaps the two most significant assumptions are (1) that traffic reaches equilibrium at each stage and (2) that repairs can only proceed sequentially and not concurrently.

Regarding the former, adopting a day-to-day flow evolution would invalidate Theorem 1, since the sequence of repairs would become significant in determining the future strategy, and not just the set of repaired bridges. Unless another concise state definition is possible, or unless alternative dominance criteria could be established, it would seem that many more potential sequences

would need to be evaluated.

Regarding the latter, allowing concurrent repairs would also greatly increase the number of feasible solutions, but perhaps insight from job-shop scheduling or other domains would be helpful. Research into either of these topics would be valuable.

Chapter 5

Conclusion

This dissertation considered three network problems which violate assumptions in their classical formulations, making them more difficult to solve, but perhaps more relevant to specific applications:

- A minimum-cost flow problem where the objective is to minimize a weighted sum of the mean flow cost and its standard deviation. As the standard deviation is nonlinear and nonseparable by link, the objective function is more challenging to optimize than in the conventional minimum-cost flow problem. Nevertheless, we derived a relationship between this problem and the (separable) problem of minimizing a weighted sum of cost mean and variance, showing that optimal solutions to the former are also optimal to the latter, for a suitable choice of weighting parameter. We showed that this weighting parameter can be found through a line search, and developed three algorithms: a bisection method, a Newton method, and a hybrid method that offers the superior typical-case convergence rate of the Newton method while retaining the superior worst-case performance of bisection.

- A network flow problem with nonlinear flow conservation constraints, motivated by parking management in urban areas. By formulating the problem in the space of splitting fractions, rather than link flows, we

are able to maintain a mathematically-convenient representation of the feasible space, at the cost of having a nonlinear (indeed, nonconvex) objective function. We obtain a descent algorithm for finding a local minimum of total cost, showing how the gradient of the cost function can be obtained through the Jacobian of link flows with respect to splitting fractions.

- A network repair sequencing problem, where at each stage the flow reaches a user equilibrium. The complication in this problem, relative to classical scheduling problems, is the spatial dependencies between network links: the benefit of repairing a damaged link can depend heavily on which other links have already been repaired. Despite this dependency, we show that the problem can be solved by dynamic programming. The state space is extremely large, so we develop a bidirectional search heuristic that obtains high-quality solutions while exploring a relatively small number of states. This heuristic uses custom branching and pruning rules tailored to this specific problem, using heuristic bounds we derive on the benefit of repairing particular subsets of links. We show that this method produces solutions on networks an orders of magnitude larger than those reported in past literature.

Our hope is that the techniques designed for these specific problems have value in other applications. In the first case, similar techniques may be applicable for other optimization problems that can be related to simpler objective functions; in the second, that our definition of marginal cost can be related to the user equilibrium problem in a way that identifies optimal parking prices to assist in urban planning; and in the third, that the bounds we develop may have value in other bilevel network optimization problems, such as network design or toll

optimization.

Bibliography

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1993.

[2] Richard Arnott and John Rowse. Modeling parking. *Journal of Urban Economics*, 45:97–124, 1999.

[3] Alper Atamtürk and Muhong Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[4] Lawrence M Ausubel and Raymond J Deneckere. A generalized theorem of the maximum. *Economic Theory*, 3(1), 1993.

[5] Kay Axhausen, John Polak, and Manfred Boltze. Effectiveness of parking guidance and information systems: Recent evidence from Nottingham and Frankfurt am Main. 01 1993.

[6] Hillel Bar-Gera. Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36:398–417, 2002.

[7] John Bates. Challenges and accomplishments of modeling impacts of policy initiatives. *In Association for European Transport and Contributors*, 2008.

[8] Martin Beckmann, Charles B McGuire, and Christopher B Winsten.

Studies in the economics of transportation. Technical Report RM-1488-PR, RAND corporation, 1956.

[9] Aharon Ben-Tal, Laurent E Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28. Princeton University Press, 2009.

[10] Itzhak Benenson, Karel Martens, and Slava Birfir. Parkagent: An agent-based model of parking in the city. *Computers, Environment and Urban Systems*, 32(6):431–439, 2008. GeoComputation: Modeling with spatial agents.

[11] Dimitri P Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[12] Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Operations Research*, 61(5):1218–1242, 2013.

[13] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1–3):49–71, 2003.

[14] John R Birge and James K Ho. Optimal flows in stochastic dynamic networks with congestion. *Operations Research*, 41(1):203–216, 1993.

[15] Paolo Bocchini and Dan Frangopol. Restoration of bridge networks after an earthquake: Multicriteria intervention optimization. *Earthquake Spectra*, 28(2):427–455, 2012.

[16] Stephen D Boyles. Bush-based sensitivity analysis for approximating subnetwork diversion. *Transportation Research Part B*, 46:139–155, 2012.

[17] Stephen D Boyles. Tap-b implementation. *https://github.com/spartalab/tap-b*, 2020. Accessed: 2020-05-20.

[18] Stephen D Boyles, Nicholas E Lownes, and Avinash Unnikrishnan. *Transportation Network Analysis*, volume 1. 0.85 edition, 2020.

[19] Stephen D Boyles, Shoupeng Tang, and Avinash Unnikrishnan. Parking search equilibrium on a network. *Transportation Research Part B: Methodological*, 81:390–409, 2015. Optimization of Urban Transportation Service Networks.

[20] Stephen D Boyles and S Travis Waller. A mean-variance model for the minimum cost flow problem with stochastic arc costs. *Networks*, 56(3):215–227, 2010.

[21] Nourht C C, El-Reedy T Y, and Ismail H K. A combined parking and traffic assignment model. *Traffic Engineering and Control*, 22:524–530, 1981.

[22] Anthony Chen and Zhong Zhou. The α-reliable mean-excess traffic equilibrium model with stochastic travel times. *Transportation Research Part B: Methodological*, 44(4):493–513, 2010.

[23] Bi Y Chen, William H K Lam, Agachai Sumalee, Qingquan Li, and Mei L Tam. Reliable shortest path problems in stochastic time-dependent networks. *Journal of Intelligent Transportation Systems*, 18(2):177–189, 2014.

[24] Lichun Chen and Elise Miller-Hooks. Resilience: An indicator of recovery capability in intermodal freight transport. *Transportation Science*, 46(1):109–123, 2012.

[25] Peng Chen and Yu M Nie. Bicriterion shortest path problem with a general nonadditive cost. *Transportation Research Part B: Methodological*, 57:419–435, 2013.

[26] Karel Dieussaert, Koen Aerts, Steenberghen Thérèse, Sven Maerivoet, and Karel Spitaels. Sustapark: An agent-based model for simulating parking search, 01 2009. Presented at the 12th AGILE International Conference on Geographic Information Science, Hannover, Germany.

[27] Augusto Eusébio and José R Figueira. Finding non-dominated solutions in bi-objective integer network flow problems. *Computers & Operations Research*, 36(9):2554–2564, 2009.

[28] Augusto Eusébio, José R Figueira, and Matthias Ehrgott. On finding representative non-dominated points for bi-objective integer network flow problems. *Computers & Operations Research*, 48:1–10, 2014.

[29] Yueyue Fan, Robert E Kalaba, and James E Moore. Arriving on time. *Journal of Optimization Theory and Applications*, 127(3):497–513, 2005.

[30] Yueyue Fan and Yu M Nie. Optimal routing for maximizing the travel time reliability. *Networks and Spatial Economics*, 6(3-4):333–344, 2006.

[31] Reza Faturechi and Elise Miller-Hooks. Measuring the performance of transportation infrastructure systems in disasters: A comprehensive review. *Journal of Infrastructure Systems*, 21(1), 2014.

[32] Mogens Fosgerau and Leonid Engelson. The value of travel time variance. *Transportation Research Part B: Methodological*, 45(1):1–8, 2011.

[33] Mogens Fosgerau and Anders Karlström. The value of reliability. *Transportation Research Part B: Methodological*, 44(1):38–49, 2010.

[34] Steven A Gabriel and David Bernstein. The traffic equilibrium problem with nonadditive path costs. *Transportation Science*, 31(4):337–348, 1997.

[35] Robert Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, 25(1):73–85, January 1977.

[36] Mariano Gallo, Luca D'Acierno, and Bruno Montella. A multilayer model to simulate cruising for parking in urban areas. *Transport Policy*, 18:735–744, 09 2011.

[37] Hemant Gehlot, Shreyas Sundaram, and Satish V Ukkusuri. Approximation algorithms for the recovery of infrastructure after disasters under precedence constraints. *IFAC-PapersOnLine*, 52(20):175–180, 2019.

[38] Hemant Gehlot, Shreyas Sundaram, and Satish V Ukkusuri. Optimal sequencing policies for recovery of physical infrastructure after disasters. In *2019 American Control Conference (ACC)*, pages 3605–3610, 2019.

[39] Gregory D Glockner and George L Nemhauser. A dynamic network flow problem with uncertain arc capacities: formulation and problem structure. *Operations Research*, 48(2):233–242, 2000.

[40] Can Gokalp. Bridge repair repository. *https://github.com/cangokalp*, 2020. Accessed: 2020-05-25.

[41] Horst W Hamacher, Christian Roed Pedersen, and Stefan Ruzika. Multiple objective minimum cost flow problems: A review. *European Journal of Operational Research*, 176(3):1404–1422, 2007.

[42] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[43] Dorit Hochbaum. Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, 153(1):257–296, September 2007.

[44] George W Housner and Jr. Thiel, Charles C. The continuing challenge: Report on the performance of state bridges in the northridge earthquake. *Earthquake Spectra*, 11(4):607–636, 1995.

[45] Kevin R Hutson and Douglas R Shier. Extended dominance and a stochastic shortest path problem. *Computers & Operations Research*, 36(2):584–596, 2009.

[46] Ehsan Jafari and Stephen D Boyles. Improved bush-based methods for network contraction. *Transportation Research Part B*, 83:298–313, 2016.

[47] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 08 2013.

[48] P V Kamesam and R R Meyer. *Multipoint methods for separable nonlinear networks*, pages 185–205. Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.

[49] Alireza Khani and Stephen D Boyles. An exact algorithm for the mean-standard deviation shortest path problem. *Transportation Research Part B: Methodological*, 81:252–266, 2014.

[50] Zoltán Király and Péter Kovács. Efficient implementations of minimum-cost flow algorithms. *Acta Universitatis Sapientiae, Informatica*, 4, 2012.

[51] D Klingman, A Napier, and J Stutz. Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20(5):814–821, 1974.

[52] Fernando A Kuipers, Song Yang, Stojan Trajanovski, and Ariel Orda. Constrained maximum flow in stochastic networks. In *2014 IEEE 22nd International Conference on Network Protocols*, pages 397–408. IEEE, 2014.

[53] William H K Lam, Zhi-Chun Li, Hai-Jun Huang, and Shuai C Wang. Modeling time-dependent travel choice problems in road networks with multiple user classes and multiple parking facilities. *Transportation Research Part B: Methodological*, 40(5):368–395, 06 2006.

[54] William H K Lam, M L Tam, and M G H Bell. Optimal road tolls and parking charges for balancing the demand and supply of road transport facilities. In M. A. P. Taylor, editor, *Proceedings of the 15th International Symposium on Transportation and Traffic Theory*, pages 561–582, 2002.

[55] Haijune Lee and P Simin Pulat. Bicriteria network flow problems: Continuous case. *European Journal of Operational Research*, 51(1):119–126, 1991.

[56] Fabien Leurent and Houda Boujnah. Traffic equilibrium in a network model of parking and route choice, with search circuits and cruising flows. In *Proceedings of EWGT2012*, Paris, 2012. EURO Working Group on Transportation.

[57] Shenzhi Li, Christopher D Janneck, Aditya P Belapurkar, Murat Ganiz, Xiaoning Yang, Mark Dilsizian, Tianhao Wu, John M Bright, and William M Pottenger. Mining higher-order association rules from distributed named entity databases. In *2007 IEEE Intelligence and Security Informatics*, pages 236–243, 2007.

[58] J Shung Lin, Chin C Jane, and John Yuan. On reliability evaluation of a capacitated-flow network in terms of minimal pathsets. *Networks*, 25(3):131–138, 1995.

[59] Jsen S Lin. Reliability evaluation of capacitated-flow networks with budget constraints. *IIE Transactions*, 30(12):1175–1180, 1998.

[60] Yi K Lin. A simple algorithm for reliability evaluation of a stochastic-flow network with node failure. *Computers & Operations Research*, 28(13):1277–1285, 2001.

[61] Yi K Lin. Using minimal cuts to evaluate the system reliability of a stochastic-flow network with failures at nodes and arcs. *Reliability Engineering & System Safety*, 75(1):41–46, 2002.

[62] Yi K Lin. On a multicommodity stochastic-flow network with unreliable nodes subject to budget constraint. *European Journal of Operational Research*, 176(1):347–360, 2007.

[63] Yi K Lin. Reliability evaluation for an information network with node failure under cost constraint. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(2):180–188, 2007.

[64] Ronaldo Luna, Nandini Balakrishnan, and Cihan H Dagli. Postearthquake recovery of a water distribution system: Discrete

event simulation using colored petri nets. *Journal of Infrasturcture Systems*, 17(1):25–34, 2011.

[65] Neale F Lunderville. Irene recovery report: A stronger future. *A representative to the Governor of Vermont, State of Vermont*, 2011.

[66] Eric Merschman, Mehrnaz Doustmohammadi, Abdullahi M Salman, and Michael Anderson. Postdisaster decision framework for bridge repair prioritization to improve road network resilience. *Transportation Research Record*, 2674(3):81–92, 2020.

[67] R R Meyer. Two-segment separable programming. *Management Science*, 25(4):385–395, 1979.

[68] Elise Miller-Hooks, Xiaodong Zhang, and Reza Faturechi. Measuring and maximizing resilience of freight transportation networks. *Computers and Operations Research*, 39(7):1633–1643, 2012.

[69] Siamak Moradi, Andrea Raith, and Matthias Ehrgott. A bi-objective column generation algorithm for the multi-commodity minimum cost flow problem. *European Journal of Operational Research*, 244(2):369–378, 2015.

[70] Ishwar Murthy and Sumit Sarkar. A relaxation-based pruning technique for a class of stochastic shortest path problems. *Transportation Science*, 30(3):220–236, 1996.

[71] Ishwar Murthy and Sumit Sarkar. Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function. *European Journal of Operational Research*, 103(1):209–229, 1997.

[72] V A Nguyen and Y.-P Tan. Minimum convex cost flow problem. In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, volume 2, pages 1248–1252, 2003.

[73] Yu Nie and Yueyue Fan. Arriving-on-time problem: discrete algorithm that ensures convergence. *Transportation Research Record: Journal of the Transportation Research Board*, (1964):193–200, 2006.

[74] Yu M Nie. Multi-class percentile user equilibrium with flow-dependent stochasticity. *Transportation Research Part B: Methodological*, 45(10):1641–1659, 2011.

[75] Yu M Nie and Xing Wu. Shortest path problem considering on-time arrival probability. *Transportation Research Part B: Methodological*, 43(6):597–613, 2009.

[76] Yu M Nie, Xing Wu, and Tito Homem-de Mello. Optimal path problems with second-order stochastic dominance constraints. *Networks and Spatial Economics*, 12(4):561–587, 2012.

[77] Evdokia Nikolova and Nicolás E Stier-Moses. A mean-risk model for the traffic assignment problem with stochastic travel times. *Operations Research*, 62(2):366–382, 2014.

[78] Fernando Ordóñez and Nicolás E Stier-Moses. Wardrop equilibria with risk-averse users. *Transportation Science*, 44(1):63–86, 2010.

[79] Priyadarshan N Patil, Katherine C Ross, and Stephen D Boyles. Convergence behavior for traffic assignment characterization metrics. *Transportmetrica A: Transport Science*, pages 1–35, 2020.

[80] Michael Patriksson. *The traffic assignment problem: models and methods*. VSP, 1994.

[81] A Arun Prakash, Ravi Seshadri, and Karthik K Srinivasan. A consistent reliability-based user-equilibrium problem with risk-averse users and endogenous travel time correlations: formulation and solution algorithm. *Transportation Research Part B: Methodological*, 114:171–198, 2018.

[82] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press., 1992.

[83] P Simin Pulat, Fenghueih Huarng, and Haijune Lee. Efficient solutions for the bicriteria network flow problem. *Computers & Operations Research*, 19(7):649–655, 1992.

[84] Zhen (Sean) Qian and Ram Rajagopal. Optimal occupancy-driven parking pricing under demand uncertainties and traveler heterogeneity: a stochastic control approach. *Transportation Research Part B*, 67:144–165, 2014.

[85] Zhen (Sean) Qian, Feng (Evan) Xiao, and H M Zhang. Managing morning commute traffic with parking. *Transportation Research Part B*, 46(7):894–916, 2012.

[86] Andrea Raith and Matthias Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36(6):1945–1954, 2009.

[87] Andrea Raith and Antonio Sedeño-Noda. Finding extreme supported solutions of biobjective network flow problems: An enhanced parametric

programming approach. *Computers & Operations Research*, 82:153–166, 2017.

[88] David Rey and Hillel Bar-Gera. Long-term scheduling for road network disaster recovery. *International Journal of Disaster Risk Reduction*, 42:101353, 2020.

[89] Antonio Sedeño-Noda and C González-Martın. The biobjective minimum cost flow problem. *European Journal of Operational Research*, 124(3):591–600, 2000.

[90] Suvrajeet Sen, Rekha Pillai, Shirish Joshi, and Ajay K Rathi. A mean-variance model for route guidance in advanced traveler information systems. *Transportation Science*, 35(1):37–49, 2001.

[91] Ravi Seshadri and Karthik K Srinivasan. Robust traffic assignment model: formulation, solution algorithms and empirical application. *Journal of Intelligent Transportation Systems*, 21(6):507–524, 2017.

[92] Mehrdad Shahabi, Avinash Unnikrishnan, and Stephen D Boyles. An outer approximation algorithm for the robust shortest path problem. *Transportation Research Part E: Logistics and Transportation Review*, 58:52–66, 2013.

[93] Donald C Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006. Parking.

[94] Raj A Sivakumar and Rajan Batta. The variance-constrained shortest path problem. *Transportation Science*, 28(4):309–316, 1994.

[95] Karthik K Srinivasan, AA Prakash, and Ravi Seshadri. Finding most reliable paths on networks with correlated and shifted log-normal travel

times. *Transportation Research Part B: Methodological*, 66:110–128, 2014.

[96] Ben Stabler. Transportation networks. *https://github.com/bstabler/TransportationNetworks*, 2020. Accessed: 2020-05-20.

[97] Russell G Thompson and Anthony J Richardson. A parking search model. *Transportation Research Part A: Policy and Practice*, 32(3):159–170, 1998.

[98] László A Végh. A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. *SIAM Journal on Computing*, 45(5):1729–1761, 2016.

[99] Eric D Vugrin, Mark A Turnquist, and Nathanael J K Brown. Optimal recovery sequencing for enhanced resilience and service restoration in transportation networks. *International Journal of Critical Infrastructures*, 10(3/4):218–246, 2014.

[100] Jianping Wang, Chunming Qiao, and Hongfang Yu. On progressive network recovery after a major disruption. *Proceedings of IEEE INFOCOM*, 2011.

[101] Judith YT Wang, Matthias Ehrgott, and Anthony Chen. A bi-objective user equilibrium model of travel time reliability in a road network. *Transportation Research Part B: Methodological*, 66:4–15, 2014.

[102] John G Wardrop and James I Whitehead. Correspondence. some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(5):767–768, 1952.

[103] Xing Wu. Study on mean-standard deviation shortest path problem in stochastic and time-dependent networks: A stochastic dominance based approach. *Transportation Research Part B: Methodological*, 80:275–290, 2015.

[104] Tao Xing and Xuesong Zhou. Finding the most reliable path with and without link travel time correlation: A lagrangian substitution based approach. *Transportation Research Part B: Methodological*, 45(10):1660–1679, 2011.

[105] Tao Xing and Xuesong Zhou. Reformulation and solution algorithms for absolute and percentile robust shortest path problems. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):943–954, 2013.

[106] Ningxiong Xu, Seth D Guikema, Rachel A Davidson, Linda K Nozick, and Zehra Ça gnan. Optimizing scheduling of post-earthquake electric power restoration tasks. *Earthquake Engineering & Structural Dynamics*, 36:265–284, 01 2006.

[107] Hai Yang, Wei Liu, Xiaolei Wang, and Xiaoning Zhang. On the morning commute problem with bottleneck congestion and parking space constraints. *Transportation Research Part B*, 58:106–118, 2013.

[108] Lixing Yang and Xuesong Zhou. Optimizing on-time arrival probability and percentile travel time for elementary path finding in time-dependent transportation networks: Linear mixed integer programming reformulations. *Transportation Research Part B: Methodological*, 96:68–91, 2017.

[109] Qing Ye and Satish V Ukkusuri. Resilience as an objective in the op-

timal reconstruction sequence for transportation networks. *Journal of Transportation Safety & Security*, 7(1):91–105, 2015.

[110] Chao Zhang, Xiaojun Chen, and Agachai Sumalee. Robust wardrop's user equilibrium assignment under stochastic demand and supply: expected residual minimization approach. *Transportation Research Part B: Methodological*, 45(3):534–552, 2011.

[111] Leilei Zhang and Tito Homem-de Mello. An optimal path model for the risk-averse traveler. *Transportation Science*, 51(2):518–535, 2016.

[112] Ning Zhang, Alice Alipour, and Laura Coronel. Application of novel recovery techniques to enhance the resilience of transportation networks. *Transportation Research Record: Journal of the Transportation Research Board*, pages 138–147, 2018.

[113] Weili Zhang, Naiyu Wang, and Charles Nicholson. Resilience-based post-disaster recovery strategies for road-bridge networks. *Structure and Infrastructure Engineering*, 13(11):1404–1413, 2017.

[114] Weili Zhang, Naiyu Wang, Charles Nicholsonc, and Mohammad H Tehrani. A stage-wise decision framework for transportation network resilience planning. *arXiv preprint arXiv:1808.03850*, 2018.

[115] Xiaoning Zhang, Hai-Jun Huang, and H M Zhang. Integrated daily commuting patterns and optimal road tolls and parking fees in a linear city. *Transportation Research Part B*, 42(1):38–56, 2008.

[116] Xiaoning Zhang, Hai Yang, and Hai-Jun Huang. Improving travel efficiency by parking permits distribution and trading. *Transportation Research Part B*, 45(7):1018–1034, 2011.

[117] Yufeng Zhang and Alireza Khani. An algorithm for reliable shortest path problem with travel time correlations. *Transportation Research Part B: Methodological*, 121:92–113, 2019.

[118] Yuli Zhang, Zuo J Max Shen, and Shiji Song. Parametric search for the bi-attribute concave shortest path problem. *Transportation Research Part B: Methodological*, 94:150–168, 2016.

[119] Yuli Zhang, Zuo J Max Shen, and Shiji Song. Lagrangian relaxation for the reliable shortest path problem with correlated link travel times. *Transportation Research Part B: Methodological*, 104:501–521, 2017.

[120] Yaoming Zhou, Junwei Wang, and Hai Yang. Resilience of transportation systems: Concepts and comprehensive review. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4262–4276, 2019.

[121] Shanjiang Zhu, David Levinson, Henry X Liu, and Kathleen Harder. The traffic and behavioral effects of the I-35W Mississippi River Bridge collapse. *Transportation Research Part A*, 44:771–784, 2010.

[122] Weilin Zhuang, Zhenyu Liu, and Jinsong Jiang. Earthquake-induced damage analysis of highway bridges in wenchuan earthquake and countermeasures. *Chinese Journal of Rock Mechanics and Engineering*, 28:1377–1387, 2009.