Structural Decomposition Methods for Sparse Large-Scale Optimization

by

Navid Matin Moghaddam

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2020 by the
Graduate Supervisory Committee:

Jorge A. Sefair, Chair
Pitu Mirchandani
Adolfo R. Escobedo
Anthony Grubesic

ARIZONA STATE UNIVERSITY

August 2020

ABSTRACT

This dissertation focuses on three large-scale optimization problems and devising algorithms to solve them. In addition to the societal impact of each problem's solution, this dissertation contributes to the optimization literature a set of decomposition algorithms for problems whose optimal solution is sparse. These algorithms exploit problem-specific properties and use tailored strategies based on iterative refinement (outer-approximations). The proposed algorithms are not rooted in duality theory, providing an alternative to existing methods based on linear programming relaxations. However, it is possible to embed existing decomposition methods into the proposed framework. These general decomposition principles extend to other combinatorial optimization problems.

The first problem is a route assignment and scheduling problem in which a set of vehicles need to traverse a directed network while maintaining a minimum inter-vehicle distance at any time. This problem is inspired by applications in hazmat logistics and the coordination of autonomous agents. The proposed approach includes realistic features such as continuous-time vehicle scheduling, heterogeneous speeds, minimum and maximum waiting times at any node, among others.

The second problem is a fixed-charge network design, which aims to find a minimum-cost plan to transport a target amount of a commodity between known origins and destinations. In addition to the typical flow decisions, the model chooses the capacity of each arc and selects sources and sinks. The proposed algorithms admit any nondecreasing piecewise linear cost structure. This model is applied to the Carbon Capture and Storage (CCS) problem, which is to design a minimum-cost pipeline network to transport CO2 between industrial sources and geologic reservoirs for long-term storage.

The third problem extends the proposed decomposition framework to a special

case of joint chance constraint programming with independent random variables. This model is applied to the probabilistic transportation problem, where demands are assumed stochastic and independent. Using an empirical probability distribution, this problem is formulated as an integer program with the goal of finding a minimum-cost distribution plan that satisfies all the demands with a minimum given probability. The proposed scalable algorithm is based on a concave envelop approximation of the empirical probability function, which is iteratively refined as needed.

DEDICATION

To my parents for their endless support.

ACKNOWLEDGMENTS

My sincere thanks go first to my committee chair, Dr. Jorge Sefair, for providing precious guidance and teaching me methodology to conduct this research. He passed on an atmosphere of adventure in regard to research and scientific discovery. His motivation, enthusiasm and support encouraged me to carry out this work. In addition, I am extremely thankful for the suggestions and guidance provided by my committee members.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Mathematical programming has been widely used to model societal problems in transportation, logistics, health-care, and other areas. Despite their ample applicability, some problems are still very difficult to solve for realistic instance sizes. In this dissertation we consider different large-scale optimization problems and investigate several approaches to solve each of them. We develop models and algorithms for efficient solving these large-scale optimization problems. The first two problems have a subjacent network structure. The third problem is about developing decomposition methods to solve a problem that involves uncertainty.

The second chapter, describes a route assignment and scheduling problem in which vehicles need to maintain a minimum distance between them at any time. This problem, which we refer to as RASTC (Route Assignment and Scheduling with Trajectory Coordination), is inspired by modern applications in transportation and logistics, and particularly by the emerging challenge of coordinating driverless vehicles. Formally, a set V of vehicles needs to travel between known origins and destinations in a directed network $G = (N, A)$. Due to safety reasons (e.g. to avoid collisions or to reduce the vulnerability to adversary attacks), the distance between any two vehicles cannot be less than d units at any time. To avoid such *geographic conflict*, RASTC seeks a route and a schedule for each vehicle, specifying the departure time from each node along a route that also minimizes a function of the vehicles' travel times. In addition, RASTC includes other realistic features such as heterogeneous vehicles in terms of speed and minimum and maximum waiting times at any node.

In Chapter 3, we study a fixed-charge pipeline network design problem with appli-

cation to carbon capture and storage (CCS). The CCS problem is to design a network able to transport a target amount of CO2 -denoted by τ- from sources to reservoirs. The network consists of a set of source nodes (S), a set of reservoir nodes (R), and arcs representing pipes (A). Sources and reservoirs are subject to maximum capture and storage capacities, denoted by q_i^s and q_j^r for source i and reservoir j, respectively. Similarly, f_i^s and v_i^s denote the fixed cost (i.e., land purchase, construction, and technology installation) and variable operational cost (i.e., pumping and maintenance) for source i, whereas parameters f_j^r and v_j^r represent the fixed and variable costs for reservoir j, respectively. The minimum and maximum capacity of arc $(i, j) \in A$ —- denoted by l_{ijd} and u_{ijd}—- depend on the chosen pipe diameter $d \in D$, where D is the set of commercially available diameters. Using a pipe incurs a variable operational cost (v_{ijd}) per ton of CO2 transported, and a fixed-charge construction cost (f_{ijd}). This problem is challenging because it requires new methods to preserve scalability of the solution approach.

In Chapter 4, we extend the discussions from previous chapters to tackle a special case of chance constraint programming which involves independent random variables. We focus on solving probabilistic transportation problem with independent random demands. In order to model this problem, we first study different methods to estimate empirical probability distribution of random variables. Based on the probability distribution estimation, we formulate the problem as a chance constraint program in which the goal is to find a minimum cost solution that satisfies the risk constraint. To overcome the challenge of solving the mathematical model, we propose an alternative method to efficiently approximate the empirical probability function. Our method relies on a variant of the decomposition method developed in Chapter 2 and 3.

In addition to the societal impact of the studied problems, this dissertation contributed to the optimization literature a set of efficient decomposition algorithms for

solving structured problems whose optimal solution is sparse. To devise such algorithms, we investigate structure-based problem decomposition strategies that have a core iterative refinement nature. In these strategies, we sequentially solve two type of subproblems, each of smaller size compared to the original problem. The first type of problems are the restricted problems which are same as solving the original problem but on a smaller instance. Solving these problems provide feasible solutions for the original problem hence a bound to the optimal value of the complete problem. The second type of problems are relaxations of the original problem which are iteratively refined to get closer to the feasible region of the complete problem. Solving these problems provide bounds for the complete problem and also provides baseline for creating a problem of the first type in the next iteration. Since this iterative refinement idea is structure dependant, we provide problem specific methods and formulations for each of the studied optimization problems.

Chapter 2

ROUTE ASSIGNMENT AND SCHEDULING WITH TRAJECTORY COORDINATION

This chapter is organized as follows. In Section 2.1 we provide a literature review for this problem. In Section 2.2, we study the geographic conflict and derive disjunctive linear constraints on the vehicles' departure times to represent the geographic conflict. In Section 2.3, we embed the developments from Section 2.2 into an MIP to solve RASTC. Although exact, the MIP's number of variables and constraints makes it computationally challenging for solving moderate- and large-scale instances. To overcome this problem, in Section 2.4 we devise an exact solution approach based on a network decomposition. In Section 2.5, we demonstrate the performance of our approach and explore its limits by solving real instances out of Berlin's road network. Section 2.6 presents our final remarks.

2.1 Introduction and Literature Review

In this chapter we study a route assignment and scheduling problem in which vehicles need to keep a minimum distance from each other at any time. This problem, which we refer to as RASTC (Route Assignment and Scheduling with Trajectory Coordination) is inspired by modern applications in transportation and logistics, and particularly by the emerging challenge of coordinating driverless vehicles. In RASTC, a set of vehicles travel between known origins and destinations in a directed network. Due to safety reasons (e.g. to avoid collisions or to reduce the vulnerability to adversary attacks), the euclidean distance between any two vehicles cannot be less than a given parameter at any time. To avoid such *geographic conflict*, RASTC seeks a

route and a schedule for each vehicle, specifying the departure time from each node along the route that also minimizes a function of the vehicles' travel times. RASTC includes other realistic features such as heterogeneous vehicles in terms of speed and minimum and maximum waiting times at any node.

Route assignment and scheduling problems are common in transportation and logistics applications. Related problems such as vehicle routing with time windows (Bräysy and Gendreau, 2005a,b), time dependent vehicle routing (Malandraki and Daskin, 1992), and dial-a-ride (Cordeau and Laporte, 2007), are close to RASTC as they decide on the vehicles' departure times and seek for optimal routes and schedules. However, they focus on time-dependent demands or dynamic travel times rather than enforcing a minimum distance between vehicles (see, e.g., Pillac *et al.* (2013) and Dixit *et al.* (2019) for comprehensive reviews on dynamic routing problems). Multiple decentralized routing models have focused on obstacle and collision avoidance. In a single-vehicle application, Hu *et al.* (2018) propose a model that selects the best path from a set of candidates to avoid static and moving obstacles while choosing the vehicle's speed and acceleration. For multiple vehicles, Jin *et al.* (2012), Kamal *et al.* (2015), and Zhu and Ukkusuri (2015) propose models and algorithms for intersection control that determine routes and departure schedules to minimize the total travel time of all vehicles across the intersection. Rios-Torres and Malikopoulos (2017) present a comprehensive survey on vehicle coordination approaches for intersections and highway on-ramps. Under the assumption of discrete time, Yu and LaValle (2012) and Ferrati and Pallottino (2013) use time-expanded networks for centralized vehicle coordination in order to construct collision free trajectories. Yamashita *et al.* (2005) propose coordination policies to dynamically adjust vehicle routes while en route, aiming to reduce congestion in connected environments where vehicles share location and destination data.

Collision avoidance is relevant in other applications beyond road traffic. In flexible manufacturing systems, automated guided vehicles (AGVs) are used for material handling and need to be safely routed across the production facility. In this area, Nishi *et al.* (2011) propose a decomposition algorithm to optimally route and schedule AGVs in discrete time that synchronizes vehicles and production schedules. The manufacturing layout is modeled as a network, thus route conflicts prevent two or more AGVs from using a node or edge at the same time. This definition of conflict is also used by Krishnamurthy *et al.* (1993) to optimize AGV routes for known demands and by Adamo *et al.* (2018) to optimize the speed of AGVs for pickup and delivery operations with time windows. Corréa *et al.* (2004) combine constraint programming and mixed-integer programming over a space-time network for dispatching and conflict-free routing of AGVs. Fazlollahtabar and Saidi-Mehrabad (2015) present a survey on existing methodologies for AGV scheduling and routing. Collision avoidance is also relevant in air traffic control of airplanes or unmanned aerial vehicles (UAVs). Using a heuristic approach, Phung *et al.* (2017) solve a path planning problem for a single UAV in the context of infrastructure inspection with static obstacles. In a real-time setting, Frazzoli *et al.* (2001) propose a protocol for airplane conflict resolution in which a centralized traffic controller adjusts aircraft trajectories to minimize the deviation from ideal routes sent by each pilot. Richards and How (2002) investigate the problem of finding optimal trajectories for multiple aircrafts to avoid collisions. The proposed discrete-time problem includes aircraft turning rates and speed decisions as well as collision avoidance constraints, which are embedded into a mixed-integer program. Otto *et al.* (2018) provide a survey on optimization approaches for UAV routing. Trajectory coordination is also relevant in the area of multi-robot path planning (Hoy *et al.*, 2015). Given known origins, destinations, and fixed paths for a set of robots, Abichandani *et al.* (2013) propose a nonlinear optimization problem to de-

termine velocity profiles under collision, kinematics, and communication constraints. Ferrera *et al.* (2013, 2014) provide decentralized approaches for collision avoidance via robot coordination.

Routing and scheduling with a minimum distance requirement is also in the transportation of hazardous materials, where spills or explosions are uncommon but have serious consequences to the environment and humans (Erkut and Verter, 1998). A common approach to mitigate the impact of hazmat accidents is to design routes that satisfy safety, equity, and operational criteria (List *et al.*, 1991; Current and Ratick, 1995). Gopalan *et al.* (1990b) and Gopalan *et al.* (1990a) design vehicle routes under equity considerations that balance the population exposure to hazmat shipments along the path, as it is undesirable to expose the same population to the risk of multiple hazmat shipments at the same time. Toumazis and Kwon (2016) extend the conditional value at risk (CVaR) ideas to develop a risk metric for the design of robust routes that minimize the worst-case consequences of a potential accident. Esfandeh *et al.* (2018) study a network design problem that includes time-dependent road closures that indirectly influence the routes chosen by hazmat vehicles, which helps reducing the population exposure to the risky shipments in space and time. In a closely related study to RASTC, Carotenuto *et al.* (2007) argue that if two hazmat vehicles travel too close to each other and one is involved in an accident, there is a high probability that the other will also be affected. As a result, they enforce a minimum distance between vehicles with a two-stage heuristic approach that first identifies a candidate set of low-risk routes and then determines vehicle departure times while considering that vehicles cannot wait at intermediate nodes. To mitigate the additional risk posed by vehicles traveling too close to each other, each route is discretized and no two vehicles are allowed inside the safety (circular) area around any of the discrete points at the same time. In case of accident or malicious attack affect-

ing a vehicle, this separation gives nearby vehicles enough time to react, limiting the negative consequences on the population. In the military context, the convoy movement problem seeks conflict-free trajectories to move military assets from sources to destinations, while satisfying spatiotemporal constraints. In this area, Thomas *et al.* (2015) propose an algorithm to route convoys across a network, assuming that convoys move as a whole and occupy an edge for some time given their length. For security reasons, a minimum inter-convoy distance is maintained for convoys traveling in the same direction. For the same problem, Chardaire *et al.* (2005) propose a discrete-time integer programming formulation that selects the best route out of a candidate set and a starting time for each convoy, assuming that once a convoy starts traversing the network it continuously moves until reaching its destination. Using an integer programming approach, Kumar and Narendran (2008) determine each convoy's route and departure time while enforcing a minimum inter-convoy headway on shared network components.

The concept of geographic conflict also arises in other *static* problems that focus on finding disjoint paths but ignore the flow scheduling aspect. A classic problem in this area is the design of survivable networks, which aims to find a minimum cost network such that each pair (or subset) of nodes is connected by a given number of arc- or node-disjoint paths (Suurballe, 1974; Kerivin and Mahjoub, 2005; Omran *et al.*, 2013; Diarrassouba *et al.*, 2018; Grötschel *et al.*, 1995). Margolis *et al.* (2018) use this concept to design a resilient supply network by selecting multiple node-disjoint distribution channels, which mitigates the risk of unsatisfied demand due to disruptions at intermediate stages. Other variants in communication problems aim to find disjoint paths that guarantee some degree of information transmission. They include finding arc-disjoint paths in the presence of resilient arcs (i.e., not subject to failure) (Żotkiewicz *et al.*, 2010) and bifurcated routing problems between an origin and a

destination with node and arc use costs (De Jongh *et al.*, 1999). Other approaches focus on the design of disjoint paths under spatial failures. These failures are modeled as disks of known diameter that can be located anywhere in the continuous space. It is assumed that components overlapping with a disk completely fail (or are destroyed) (Neumayer *et al.*, 2009, 2015) or lose some functionality (Sullivan and Smith, 2014). Extending the max-flow min-cut theorem, Neumayer *et al.* (2009, 2015); Kobayashi and Otsuki (2014), and Otsuki *et al.* (2016) study the problems of finding the maximum number of geographically disjoint paths between two nodes and the minimum number of disk failures to disconnect two nodes. In these problems, two paths are geographically disjoint if the minimum distance between them is at least a given value (except in areas close to the origin and destination). Although close to RASTC, these *static* approaches ignore the vehicle route assignment and scheduling aspects, and can only be used when all vehicles travel between the same origin and destination. We refer to these approaches as static as they enforce a minimum distance between paths rather than vehicles.

Applications of RASTC include air and maritime routing, where vehicles need to maintain a steady cruising speed as well as a safety distance to avoid collision. Additional related problems arise in the transportation of hazardous materials, where vehicles cannot be too close to each other given the risk posed to the population in case of accident or malicious attack. Moreover, RASTC is related to the convoy movement problem in the absence of congestion when the convoy can split at any point and vehicles can travel at the speed limit (or any other constant speed) along different routes. In this case, enforcing a minimum separation distance decreases the risk of losing multiple vehicles at once given an airstrike or a roadside bomb. RASTC is also related to AGV routing when the interest is to find routes and schedules for known demands and when operations require each AGV to travel between two

locations only.

We focus on problems with a moderate number of vehicles and assume that cycles are undesirable, as they may unnecessarily expose population to dangerous materials, require additional fuel consumption, and may not be even possible given the network used. We allow heterogeneous vehicles with different speeds, which remain constant once in motion. Each vehicle has a designated speed for each arc, which is not necessarily the same for all arcs. This is plausible for air and maritime routing applications, where vehicles travel at cruising speeds. For more ambitious related problems such as routing of autonomous vehicles, the solution from RASTC provides a benchmark to compare the performance of other models or solution approaches. For instance, the solution quality of a heuristic that allows variable speeds must be no worse than that provided by RASTC. Moreover, RASTC provides a benchmark for any decentralized coordination mechanism, as it assumes a centralized (and optimal) coordination between the vehicles. Although RASTC is not specifically designed to handle variable speeds while in motion, these can be approximated using our approach. Because the goal is to avoid the geographic conflict while still moving towards the destination, it is possible to add extra nodes along an arc where vehicles can wait. In this way, the combination of motion (at constant speed) and waiting can approximate a deceleration profile.

RASTC has received very little attention in the literature. There are no exact models or algorithms available to tackle emerging RASTC problems in continuous time. The following are our main contributions: (1) We introduce an NP-hard routing and scheduling problem that is relevant for current and emerging related applications and that encompasses several realistic features; (2) We develop an alternative linear formulation to the geographic conflict that avoids the euclidean norm when calculating the distance between vehicles and propose a polynomial-time pre-processing

approach to characterize the conflict; (3) We embed such conflict constraints into a mixed-integer programming (MIP) formulation, which we solve using a tailored decomposition technique.

This chapter is organized as follows. In Section 2.2, we study the geographic conflict and derive disjunctive linear constraints on the vehicles' departure times to represent the geographic conflict. In Section 2.3, we embed the developments from Section 2.2 into an MIP to solve RASTC. Although exact, the MIP's number of variables and constraints makes it computationally challenging for solving moderate- and large-scale instances. To overcome this problem, in Section 2.4 we devise an exact solution approach based on a network decomposition. In Section 2.5, we demonstrate the performance of our approach and explore its limits by solving real instances out of Berlin's road network, as well as other randomly generated instances. Section 2.6 presents our final remarks. Appendices A and B contain all the proofs of our propositions.

2.2 Vehicle Coordination Modeling

We enforce the coordination among a set of vehicles, V, by imposing constraints on their departure times from the nodes they traverse in a directed network, $G = (N, A)$. In Section 2.2.1, we characterize the distance between network elements, and in Section 2.2.4 we describe the conditions that vehicles' departure times must satisfy to avoid geographic conflict. We define $\check{\delta}((i,j),k)$ and $\hat{\delta}((i,j),(k,l))$ as functions returning the shortest euclidean distance between arc (i,j) and node k, and between arcs (i,j) and (k,l), respectively. We assume that arcs are straight lines. Non-straight trajectories (e.g., along a winding road) can be approximated by adding intermediate nodes connected by straight arcs.

2.2.1 Distance Between Network Elements

2.2.2 Arc-node Distance

The arc-node distance is the minimum euclidean distance between arc (i, j) and node k, which we denote by $\check{\delta}((i, j), k)$. To calculate $\check{\delta}((i, j), k)$, we use the coordinates of node i, $\mathbf{x}_i$, and the unit vector in the direction of arc (i, j), $\mathbf{u}$. We then obtain the orthogonal projection of $\mathbf{x}_k$ on the line $\mathbf{x}_i + \alpha\mathbf{u}$, which is given by $\mathbf{x}_i + \alpha^*\mathbf{u}$, where $\alpha^* =_{\alpha\in\mathbb{R}} \|\mathbf{x}_i + \alpha\mathbf{u} - \mathbf{x}_k\|$. The value of α^* is unique and can be calculated in closed form given the convexity of the euclidean norm function. Using this projection, we have that

$$\check{\delta}((i, j), k) = \begin{cases} \|\mathbf{x}_i - \mathbf{x}_k\| & \text{If } \alpha^* \leq 0 \\ \|\mathbf{x}_j - \mathbf{x}_k\| & \text{If } \alpha^* \geq \|\mathbf{x}_j - \mathbf{x}_i\| \\ \|\mathbf{x}_i + \alpha^*\mathbf{u} - \mathbf{x}_k\| & \text{If } 0 < \alpha^* < \|\mathbf{x}_j - \mathbf{x}_i\|. \end{cases}$$

2.2.3 Arc-arc Distance

We define the arc-arc distance as the minimum euclidean distance between arcs (i, j) and (k, l), which we denote by $\hat{\delta}((i, j), (k, l))$. We also define $\mathbf{u}_{ij}$ and $\mathbf{u}_{kl}$ as the unit vectors in the direction of arcs (i, j) and (k, l), respectively. Following the same intuition as in the arc-node distance, we find two points along the lines $\mathbf{x}_i + \alpha_{ij}\mathbf{u}_{ij}$ and $\mathbf{x}_k + \alpha_{kl}\mathbf{u}_{kl}$ whose distance is minimum by finding $\boldsymbol{\alpha}^* =_{(\alpha_{ij},\alpha_{kl})\in\mathbb{R}^2} \|\mathbf{x}_i + \alpha_{ij}\mathbf{u}_{ij} - \mathbf{x}_k - \alpha_{kl}\mathbf{u}_{kl}\|$. Using $\boldsymbol{\alpha}^* = (\alpha^*_{ij}, \alpha^*_{kl})$, we have that

$$\hat{\delta}((i, j), (k, l)) = \begin{cases} \left\|\mathbf{x}_i + \alpha^*_{ij}\mathbf{u}_{ij} - \mathbf{x}_k - \alpha^*_{kl}\mathbf{u}_{kl}\right\|, \text{ If } 0 \leq \alpha^*_{ij} \leq \|\mathbf{x}_j - \mathbf{x}_i\| \text{ and } 0 \leq \alpha^*_{kl} \leq \|\mathbf{x}_l - \mathbf{x}_k\| \\ \min\{\check{\delta}((i, j), k), \check{\delta}((i, j), l), \check{\delta}((k, l), i), \check{\delta}((k, l), j)\}, \text{ Otherwise.} \end{cases}$$

Although in this case vector $\boldsymbol{\alpha}^*$ may not be unique (e.g., when arcs are parallel), an optimal solution $\boldsymbol{\alpha}^*$ can still be obtained in closed form given convexity of the euclidean norm function.

2.2.4 Geographic Conflict Modeling

This section describes our approach to prevent geographic conflict by requiring vehicles to maintain a distance of at least d units at any time. We introduce two types of conflicts: *arc-arc*, used to avoid conflict when vehicles are moving, and *arc-node*, used when one of the vehicles is waiting at a node and the other is moving. To impose such conflict constraints only on relevant network components, we define $\Omega = \{((i,j),(k,l)) \subset A \times A : \hat{\delta}((i,j),(k,l)) < d\}$ and $\Psi = \{((i,j),k) \subset A \times N : \check{\delta}((i,j),k) < d\}$. Because arc-arc conflict is symmetric, we add $((i,j),(k,l))$ or $((k,l),(i,j))$ to Ω, but not both. We assume that there is no arc-node conflict involving any vehicle's origin or destination nodes.

2.2.5 Arc-arc Conflict

Suppose that while traversing the network, vehicles g and h use arcs (i,j) and (k,l), respectively, with corresponding velocity vectors $\mathbf{v}_{ij}^g$ and $\mathbf{v}_{kl}^h$, where $((i,j),(k,l)) \in \Omega$. Under these conditions, the distance between g and h at time t is given by $\left\|\mathbf{x_i} + \mathbf{v}_{ij}^g t - (\mathbf{x_k} + \mathbf{v}_{kl}^h(t-\delta))\right\|$, where δ represents the difference in the departure time of h with respect to the departure time of g, which is the reference time. For instance, $\delta < 0$ indicates that h departs δ units of time before g, whereas $\delta > 0$ indicates that h departs δ units of time after g. If $\delta = 0$, then g and h start moving at the same time. We assume that each vehicle's speed is constant while traversing an arc, thus g spends $c_{ij}^g = \frac{\|\mathbf{x}_j - \mathbf{x}_i\|}{\|\mathbf{v}_{ij}^g\|}$ time moving from i to j, and h spends $c_{kl}^h = \frac{\|\mathbf{x}_l - \mathbf{x}_k\|}{\|\mathbf{v}_{kl}^h\|}$ time moving from k to l. This means that any value of δ and t satisfying

the following two conditions leads to a geographic conflict between g and h while they are in motion along arcs (i,j) and (k,l), respectively.

$$\left\|\mathbf{x}_i + \mathbf{v}_{ij}^g t - (\mathbf{x}_k + \mathbf{v}_{kl}^h(t-\delta))\right\| < d \tag{2.1}$$

$$t \in [\max\{0,\delta\}, \min\{c_{ij}^g, c_{kl}^h + \delta\}] \tag{2.2}$$

Condition (2.1) reflects the occurrence of geographic conflict, while (2.2) narrows the time window for conflict analysis down to values of t when vehicles are moving. The term $\max\{0,\delta\}$ captures the earliest time at which both vehicles are in motion and the term $\min\{c_{ij}^g, c_{kl}^h + \delta\}$ captures the time at which the first vehicle arrives at its destination. The following proposition establishes an important property of all the values (δ, t) satisfying (2.1) and (2.2), which we later use to derive some constraints for our mathematical program.

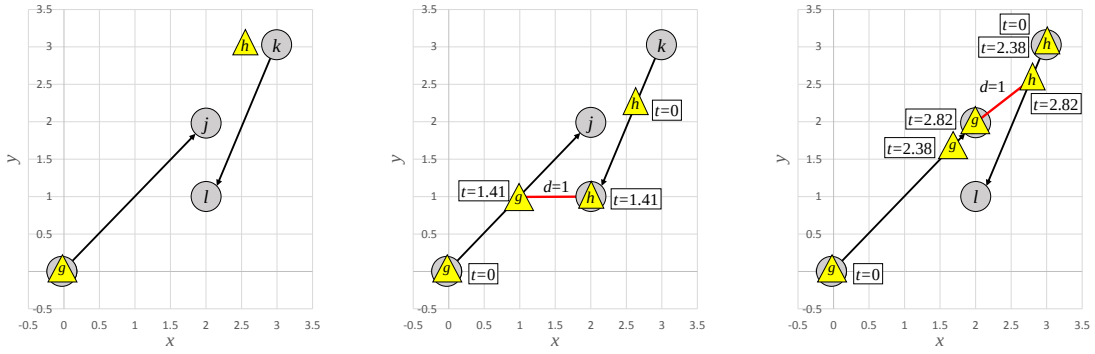
Proposition 1 *If the differences in departure times δ_1 and δ_2, where $\delta_1 \leq \delta_2$, lead to a geographic conflict, then any $\delta \in [\delta_1, \delta_2]$ also leads to a conflict.*

Proposition 1 proves the convexity of the set of values (δ, t) satisfying (2.1) and (2.2), and suggests that the geographic conflict interval can be fully characterized by the minimum and maximum possible difference in the departure times. Based on this observation, we define $\hat{\ell}_{ijkl}^{gh} = \min\{\delta \mid (\delta,t) \text{ satisfies } \left\|\mathbf{x}_i + \mathbf{v}_{ij}^g t - (\mathbf{x}_k + \mathbf{v}_{kl}^h(t-\delta))\right\| \leq d \text{ and (2.2)}\}$ and $\hat{u}_{ijkl}^{gh} = \max\{\delta \mid (\delta,t) \text{ satisfies } \left\|\mathbf{x}_i + \mathbf{v}_{ij}^g t - (\mathbf{x}_k + \mathbf{v}_{kl}^h(t-\delta))\right\| \leq d \text{ and (2.2)}\}$, thus conflict arises if the difference between the departure times falls in the interval $(\hat{\ell}_{ijkl}^{gh}, \hat{u}_{ijkl}^{gh})$. That is, if we let τ_i^g and τ_k^h be the departure times of vehicles g and h from nodes i and k along arcs (i,j) and (k,l), respectively, such that $((i,j),(k,l)) \in \Omega$, then the disjunction $(\tau_k^h - \tau_i^g \leq \hat{\ell}_{ijkl}^{gh}) \vee (\tau_k^h - \tau_i^g \geq \hat{u}_{ijkl}^{gh})$ avoids any conflict when the vehicles are in

motion along these arcs. The optimization problems producing $\hat{\ell}$- and $\hat{u}$-parameters are convex due to Proposition 1 and that Slater's constraint qualification holds given that $((i,j),(k,l)) \in \Omega$. Hence, we can obtain the unique optimal values for $\hat{\ell}$ and $\hat{u}$-parameters in closed-form by solving the corresponding system of first-order Karush-Kuhn-Tucker optimality conditions.

To illustrate the operation of our arc-arc conflict conditions, consider the situation depicted in Figure 2.1a, where vehicle g travels between nodes i and j using arc (i,j), and vehicle h travels from k to l using arc (k,l). The xy-coordinates of nodes i, j, k, and l are $(0,0)$, $(2,2)$, $(3,3)$, and $(2,1)$, respectively. In this case, we assume $d = 1$ for which $((i,j),(k,l))$ clearly belongs to Ω. Assuming speeds equal to one, we obtain $\hat{\ell}^{gh}_{ijkl} = -0.82$, which means that h must depart k at least 0.82 units of time before g to avoid conflict. Figure 2.1b illustrates the situation when $\tau^h_k - \tau^g_i = -0.82$. At the departure time of g (which we denote by $t = 0$ for convenience), vehicle h is almost halfway of its trip to l. At $t = 1.41$, g is halfway to j and h arrives at l, where the distance between vehicles is exactly d. In this case, there is no conflict between g and h, as the distance between them is no less than d at any time while in motion. Note that any departure times such that $\tau^h_k - \tau^g_i < -0.82$ also prevent conflict. The arc-arc conflict analysis also produces $\hat{u}^{gh}_{ijkl} = 2.38$, meaning that h must depart k at least 2.38 units of time after g to avoid conflict. Figure 2.1c illustrates the situation when $\tau^h_k - \tau^g_i = 2.38$. At time $t = 0$, g departs node i while h waits at k until time $t = 2.38$. At $t = 2.82$, g arrives at j and h is at coordinate $(2.8, 2.6)$, whose distance to j is exactly d. As in the previous case, this situation leads to no conflict as the vehicles are never at a distance of less than d while in motion. Indeed, any configuration of departure times such that $\tau^h_k - \tau^g_i \notin (-0.82, 2.38)$ will prevent geographic conflict.

Our arc-arc conflict conditions are general to any network topology as long as the participant arcs belong to Ω. Figure 2.2 depicts three special cases of arc-arc

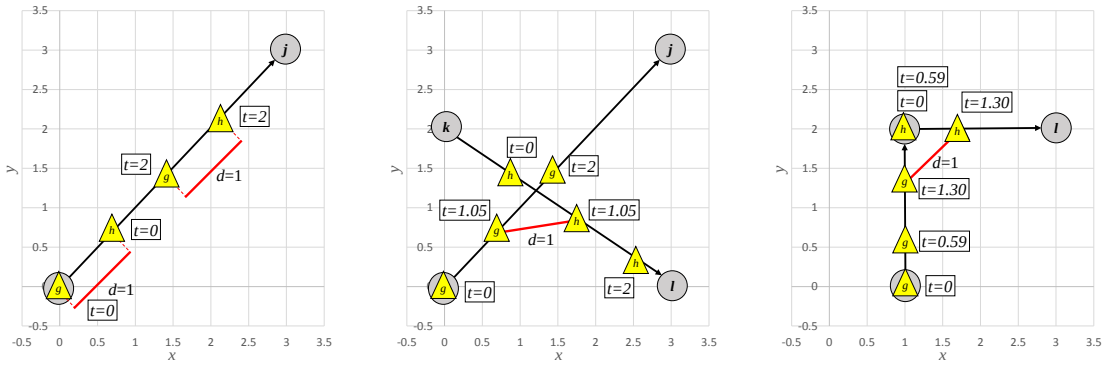


(a) (b) $\tau_k^h - \tau_i^g = -0.82$ (c) $\tau_k^h - \tau_i^g = 2.38$

Figure 2.1: Arc-arc Conflict Analysis

conflicts, in which we assume for simplicity $d = 1$ and unit speeds. In the first case, two vehicles using the same arc (i.e., $(i, j) = (k, l)$) synchronize their departure times to avoid conflict while in motion. In Figure 2.2a, our conflict analysis requires the vehicles to leave node i with a time difference of at least 1 unit, i.e., $\tau_i^h - \tau_i^g \notin (-1, 1)$. Figure 2.2a illustrates that the minimum distance d is preserved when $\tau_i^h - \tau_i^g = -1$. In Figure 2.2b, vehicles transit two arcs that intersect (e.g., a road intersection with no right or left turns), requiring the vehicles to leave the departure nodes with a time difference $\tau_k^h - \tau_i^g \notin (-1.04, 1.55)$ to avoid conflict. Figure 2.2b illustrates the case where $\tau_k^h - \tau_i^g = -1.04$. At time $t = 0$, g departs i while h is almost at the intersection, and at time $t = 1.05$ vehicles are at their minimum distance (d). Figure 2.2c illustrates the case where $j = k$, meaning that g travels towards the location of h such that the departure times from i and j must satisfy $\tau_j^h - \tau_i^g \notin (0.59, 2)$. Figure 2.2c illustrates the case where $\tau_j^h - \tau_i^g = 0.59$, in which g departs i at $t = 0$ and h waits at node j until $t = 0.59$, achieving a minimum distance of d between vehicles at time $t = 1.30$.

Figures 2.1 and 2.2 illustrate how to prevent the geographic conflict while vehicles are moving. However, these constraints allow the distance between vehicles to be less



(a) $\tau_i^h - \tau_i^g = -1$ (b) $\tau_k^h - \tau_i^g = -1.04$ (c) $\tau_j^h - \tau_i^g = 0.59$

Figure 2.2: Special Cases in the Arc-arc Conflict Analysis

than d when one of them is waiting at a node. For instance, the distance between g and h will be less than d if h waits at l after $t = 1.41$ in Figure 2.1b or when g waits at j after $t = 2.82$ in Figure 2.1c. This situation may not be desirable in some related problems (e.g., military convoy planning), requiring additional *arc-node* constraints to prevent such conflict.

2.2.6 Arc-node Conflict

Suppose that vehicle g uses arc (i, j) and vehicle h visits (and possibly waits at) node l such that $((i, j), l) \in \Psi$. In this case, the distance between the two vehicles at time t is given by $\left\|\mathbf{x}_i + t\mathbf{v}_{ij}^g - \mathbf{x}_l\right\|$, where we assume that the departure time of g is the reference time. Proposition 2 establishes an analogous result to Proposition 1 for the arc-node conflict. The proof of Proposition 2 is similar to that of Proposition 1.

Proposition 2 *If the differences in departure times t_1 and t_2, where $t_1 \leq t_2$, lead to a geographic conflict, then any $t \in [t_1, t_2]$ also leads to a conflict.*

Because $((i, j), l) \in \Psi$, we know that geographic conflict exists at some time t such that $0 \leq t \leq c_{ij}^g$. Using Proposition 2, we can calculate a conflict interval $(\check{\ell}_{ijl}^{gh}, \check{u}_{ijl}^{gh})$

for the vehicles' departure times difference by performing the following two steps.

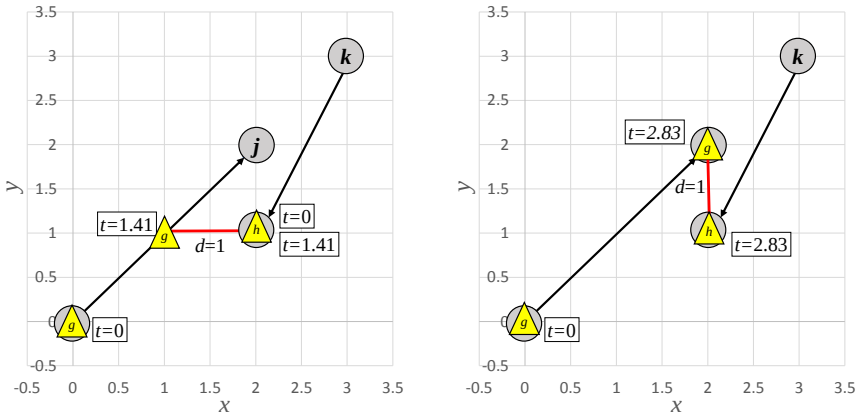
- Step 1: Solve $\left\|\mathbf{x}_i + t\mathbf{v}_{ij}^g - \mathbf{x}_l\right\| = d$ for t and obtain the roots t_1 and t_2, where $t_1 \leq t_2$.
- Step 2: Return $\check{\ell}_{ijl}^{gh} = \max\{0, t_1\}$ and $\check{u}_{ijl}^{gh} = \min\{c_{ij}^g, t_2\}$.

The roots in Step 1 always exist and are real because $((i,j),l) \in \Psi$ and there are no constraints on t. Using this two-step procedure, we establish that no conflict arises if h leaves node l at most $\check{\ell}_{ijl}^{gh}$ units of time after g departs i, or if h arrives at l at least $\check{u}_{ijl}^{gh}$ units of time after g departs i. If we let τ_i^g be the departure time of vehicle g from node i along arc (i,j), τ_l^h be the departure time of vehicle h from node l, and τ_k^h be the departure time of vehicle h from node k along arc (k,l), such that $((i,j),l) \in \Psi$, then the disjunction $(\tau_l^h - \tau_i^g \leq \check{\ell}_{ijl}^{gh}) \vee (\tau_k^h + c_{kl}^h - \tau_i^g \geq \check{u}_{ijl}^{gh})$ avoids any conflict while vehicle g is moving and vehicle h is waiting after arriving into node l using arc (k,l).

Figure 2.3 illustrates the arc-node conflict conditions when g is traveling from i to j, h is waiting at l, $d = 1$, and unit speeds. Clearly, $((i,j),l) \in \Psi$, indicating that g and h will have conflicting locations at some time $t \in [0, \|\mathbf{x}_j - \mathbf{x}_i\|]$. We obtain that $\check{\ell}_{ijl}^{gh} = 1.41$ and $\check{u}_{ijl}^{gh} = 2.83$. Figure 2.3a illustrates the situation when $\check{\ell}_{ijl}^{gh} = 1.41$, which forces h to leave l (to any other node) no later than $t = 1.41$ to avoid conflict. Figure 2.3b describes the situation when $\check{u}_{ijl}^{gh} = 2.83$, which implies that h must arrive at l (from k or any other node) not earlier than $t = 2.83$.

2.3 Mathematical Programming Formulation

We propose an MIP formulation for RASTC that uses a directed network $G = (N, A)$, where N is the set of nodes and A is the set of arcs. A set of vehicles V travels between known origin and destination nodes denoted by s_g and p_g, respectively, for each vehicle $g \in V$. We partition V into sets V_{st} to identify those vehicles traveling



(a) $\tau_l^h - \tau_i^g = 1.41$ (b) (Arrival time of h at l)$-\tau_i^g = 2.83$

Figure 2.3: Arc-node Conflict Analysis

from s to t. Parameters a_j^g and b_j^g denote the minimum and maximum amount of time that vehicle g is allowed to wait at node $j \in N$, if visited. There is no maximum waiting time constraint at nodes s_g and p_g, for any $g \in V$. Departure times within the interval $(\hat{\ell}_{ijkl}^{gh}, \hat{u}_{ijkl}^{gh})$ result in a conflict when vehicles g and h travel on arcs (i,j) and (k,l), respectively, such that $((i,j),(k,l)) \in \Omega$. Similarly, the interval $(\check{\ell}_{ijk}^{gh}, \check{u}_{ijk}^{gh})$ captures the arc-node conflict when vehicle g travels along arc (i,j) and vehicle h uses node k such that $((i,j),k) \in \Psi$. For each vehicle g, we precalculate the shortest-path time from s_g to p_g, denoted by z_{SP}^g, using arc costs given by $c_{ij}^g + a_j^g$, for all $(i,j) \in A$. We use big-M parameters M_g and M_{gh} with positive value for $g, h \in V$.

The binary variable x_{ij}^g is equal to one if and only if $g \in V$ uses arc $(i,j) \in A$, and the continuous variable τ_i^g captures the departure time of g from node $i \in N$. We reformulate the arc-arc and arc-node conflict disjunctive conditions using a big-M approach. We use the binary variable f_{ijkl}^{gh}, which is equal to one if the conflict condition involving $\hat{u}_{ijkl}^{gh}$ is satisfied by vehicles g and h, $g \neq h$, traveling on arcs (i,j) and (k,l), respectively, such that $((i,j),(k,l)) \in \Omega$. This variable is equal to zero if

the departure times of g and h satisfy the disjunctive condition that uses $\hat{\ell}^{gh}_{ijkl}$. The binary variable e^{gh}_{ijk} models the arc-node conflict between vehicles g and h, $g \neq h$, when g is traveling arc (i,j) and h is waiting at node k such that $((i,j),k) \in \Psi$. If the departure times satisfy the arc-node condition that uses $\check{u}^{gh}_{ijk}$, then e^{gh}_{ijk} is equal to one. Otherwise, if they satisfy the condition using $\check{\ell}^{gh}_{ijk}$, then e^{gh}_{ijk} is equal to zero. The decision variable z captures the value of the objective function, which corresponds to the maximum relative deviation from each vehicle's shortest-path time. We define the sets of indices $\Lambda = \{(g,h,i,j,k,l) \mid g,h \in V, g \neq h, ((i,j),(k,l)) \in \Omega\}$, $\Gamma = \{(g,h,i,j,l) \mid g,h \in V, g \neq h, ((i,j),l) \in \Psi\}$, and $\Upsilon = \{(g,h,i,j,k,l) \mid g,h \in V, g \neq h, (i,j),(k,l) \in A, ((i,j),l) \in \Psi\}$ to capture the possible combinations of vehicles and network elements where conflict may occur. Constraints (2.4)–(2.20) describe the feasible region of RASTC.

$$\min \quad z \tag{2.3}$$

$$\text{s.t.} \qquad \sum_{j:(i,j)\in A} x_{ij}^g - \sum_{j:(j,i)\in A} x_{ji}^g = \begin{cases} 1, & \text{If } i = s_g \\ 0, & \text{If } i \in N \setminus \{s_g, p_g\}, \ \forall g \in V, i \in N \\ -1, & \text{If } i = p_g \end{cases} \tag{2.4}$$

$$\tau_i^g + c_{ij}^g + a_j^g \leq \tau_j^g + M_g(1 - x_{ij}^g), \quad \forall g \in V, \ \forall (i,j) \in A \tag{2.5}$$

$$\tau_i^g + c_{ij}^g + b_j^g \geq \tau_j^g - M_g(1 - x_{ij}^g), \quad \forall g \in V, \ \forall (i,j) \in A : i,j \notin \{s_g, p_g\} \tag{2.6}$$

$$\tau_k^h - \tau_i^g \leq \hat{\ell}_{ijkl}^{gh} + M_{gh}[f_{ijkl}^{gh} + (1 - x_{ij}^g) + (1 - x_{kl}^h)], \quad \forall (g,h,i,j,k,l) \in \Lambda \tag{2.7}$$

$$\tau_k^h - \tau_i^g \geq \hat{u}_{ijkl}^{gh} - M_{gh}[(1 - f_{ijkl}^{gh}) + (1 - x_{ij}^g) + (1 - x_{kl}^h)], \quad \forall (g,h,i,j,k,l) \in \Lambda \tag{2.8}$$

$$\tau_l^h - \tau_i^g \leq \check{\ell}_{ijl}^{gh} + M_{gh}[e_{ijl}^{gh} + (1 - x_{ij}^g)], \quad \forall (g,h,i,j,l) \in \Gamma \tag{2.9}$$

$$\tau_k^h + c_{kl}^h - \tau_i^g \geq \check{u}_{ijk}^{gh} - M_{gh}[(1 - e_{ijl}^{gh}) + (1 - x_{ij}^g) + (1 - x_{kl}^h)], \quad \forall (g,h,i,j,k,l) \in \Upsilon \tag{2.10}$$

$$f_{ijkl}^{gh} \leq x_{ij}^g, \quad \forall (g,h,i,j,k,l) \in \Lambda \tag{2.11}$$

$$f_{ijkl}^{gh} \leq x_{kl}^h, \quad \forall (g,h,i,j,k,l) \in \Lambda \tag{2.12}$$

$$e_{ijl}^{gh} \leq x_{ij}^g, \quad \forall (g,h,i,j,l) \in \Gamma \tag{2.13}$$

$$e_{ijl}^{gh} \leq \sum_{k:(k,l)\in A} x_{kl}^h, \quad \forall (g,h,i,j,l) \in \Gamma \tag{2.14}$$

$$\tau_s^{g_1} \leq \cdots \leq \tau_s^{g_{|V_{st}|}}, \quad \forall s,t \in N : |V_{st}| > 1 \tag{2.15}$$

$$\frac{\tau_{p_g}^g}{z_{SP}^g} \leq z, \quad \forall g \in V \tag{2.16}$$

$$\tau_i^g \geq 0, \quad \forall g \in V, i \in N \tag{2.17}$$

$$x_{ij}^g \in \{0,1\}, \quad \forall g \in V, (i,j) \in A \tag{2.18}$$

$$f_{ijkl}^{gh} \in \{0,1\}, \quad \forall (g,h,i,j,k,l) \in \Lambda \tag{2.19}$$

$$e_{ijl}^{gh} \in \{0,1\}, \quad \forall (g,h,i,j,l) \in \Gamma \tag{2.20}$$

Constraints (2.4) impose flow-balance conditions for each node and each vehicle, similar to the classic multicommodity flow problem (Ahuja *et al.*, 1993). Constraints (2.5)–(2.6) help enforcing the waiting times at each node and keep track of each vehicle's travel time. These constraints also help eliminating cycles, as they are assumed to be infeasible for the problems of our interest. Constraints (2.5) guarantee that if vehicle $g \in V$ uses arc (i,j) (i.e., $x_{ij}^g = 1$), then its departure time from node j is at least a_j^g units of time after its arrival time (i.e., $\tau_i^g + c_{ij}^g$), which enforces the minimum waiting time. Similarly, Constraints (2.6) prevent vehicle g from waiting

longer than b_j^g at node j, enforcing the maximum waiting time conditions. Because of the big-M parameters, these constraints are only active when g uses arc (i, j), and are redundant otherwise. We strengthen the formulation by modifying the big-M values as our solution algorithm progresses and more information on the problem's optimal solution becomes available. This is discussed in Section 2.4.2.

Constraints (2.7) and (2.8) impose the arc-arc conflict constraints. If vehicles g and h use arcs (i, j) and (k, l), respectively (e.g., $x_{ij}^g = x_{kl}^h = 1$), and there is an arc-arc conflict between such arcs, then the binary variable f_{ijkl}^{gh} forces the departure times to satisfy either (2.7) or (2.8). If $x_{ij}^g = 0$ or $x_{kl}^h = 0$, or both, then the corresponding arc-arc constraints are relaxed. Constraints (2.9) and (2.10) impose the arc-node conflict constraints. In this case, the binary variable e_{ijl}^{gh} forces the departure times to satisfy exactly one constraint between (2.9) and (2.10), when g is moving along arc (i, j) and h is waiting at l, and $((i, j), l) \in \Psi$. Constraints (2.11)–(2.14) strengthen the formulation by forcing the e- and f-variables to be zero when the network elements involved in the conflict are not used by vehicles g and h because in this case the corresponding Constraints (2.7)–(2.10) will be already relaxed. When multiple identical vehicles travel between the same origin and destination, any permutation of a feasible routing and scheduling plan among vehicles is also feasible. Constraints (2.15) break such symmetry and reduce the feasible space by imposing a nondecreasing order on the vehicles' departure times, which will not affect the optimal solution. In this case, imposing no more than $|V| - 1$ constraints suffices to eliminate the symmetries.

In the absence of conflict, the optimal path for any vehicle is the shortest path to the destination with arc costs given by the sum of travel and minimum waiting times. However, conflict may force a vehicle to deviate from its shortest path or to wait longer than required at one or more nodes, increasing the total travel time. For this reason,

our objective function seeks a *fair* route planning and scheduling that minimizes the maximum deviation from each vehicle's shortest-path time. Constraints (2.16) and the objective function (2.3) model this situation. Constraints (2.17)–(2.19) enforce the nature of the decision variables.

Regarding RASTC's computational complexity, related static problems such as finding the maximum number of geographically disjoint paths can be solved in polynomial time (Neumayer *et al.*, 2009, 2015; Kobayashi and Otsuki, 2014; Otsuki *et al.*, 2016). However, RASTC is NP-hard due to its dynamic nature (see proof in Appendix A).

2.4 Network Decomposition Approach for RASTC

In this section, we develop a network decomposition scheme to expand the limits of our exact model. This is motivated by the prohibitively large number of decision variables and constraints in formulation (2.3)–(**??**), which makes RASTC unsolvable in reasonable time for medium- and large-scale instances using commercial solvers. Our approach is based on two observations: (1) Not all vehicles use all the network components, which means that solving RASTC on a sub-network may produce an optimal solution for the problem on the complete network, and (2) Conflict may not occur on every pair of network components in Ω or Ψ, which means that enforcing a subset of conflict constraints may be enough to produce a conflict-free optimal solution for the complete problem.

2.4.1 Notation and Additional Definitions

We use the notation RASTC$(G, \Lambda, \Gamma, \Upsilon)$ to describe the problem in (2.3)–(**??**) with parameters given by G, Λ, Γ, and Υ, where we assume that other parameters (e.g., V) will not change. We denote the global optimal value of RASTC$(G, \Lambda, \Gamma, \Upsilon)$

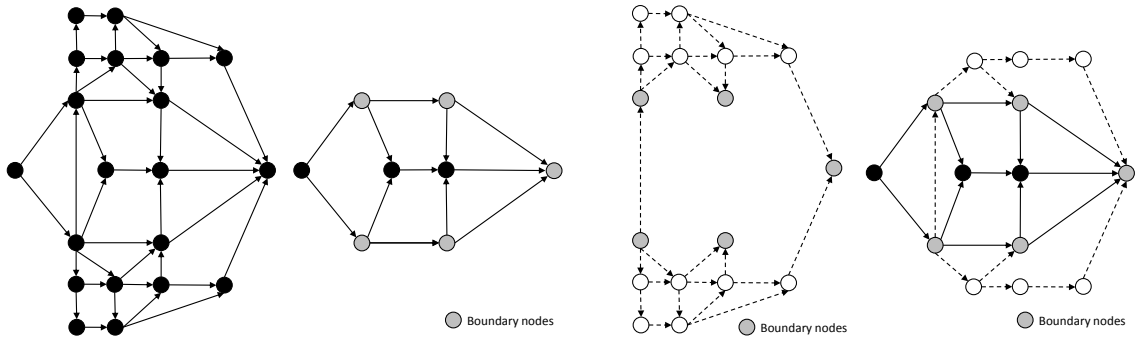
by z^*, and use $z(\cdot)$ to denote the optimal value of RASTC with parameters given by $(\cdot)$. For instance $z(\tilde{G}, \tilde{\Lambda}, \tilde{\Gamma}, \tilde{\Upsilon})$ denotes the optimal solution to RASTC$(\tilde{G}, \tilde{\Lambda}, \tilde{\Gamma}, \tilde{\Upsilon})$. We use the notation $\hat{z}^i(\cdot)$ to represent the optimal value of RASTC if the parameters in $(\cdot)$ produce an upper bound on z^*, and $\check{z}^i(\cdot)$ if they produce a lower bound at iteration i.

We introduce the following network structures that are useful in our analysis. A *reduced network*, $G^r = (N^r, A^r)$, is a subgraph of G (i.e., $N^r \subseteq N$ and $A^r \subseteq A$) that contains at least one path from s_g to p_g, for all $g \in V$. The set of *boundary nodes* B of G^r is the set of nodes in N^r with an incoming or outgoing arc in $A \setminus A^r$. That is, $B = \{i \in N^r \mid (i,j) \in A \setminus A^r \vee (j,i) \in A \setminus A^r\} \subseteq N^r$. The network G^r induces a *complement network* $G^c = ((N \setminus N^r) \cup B, A \setminus A^r)$, which contains all elements in G but not in G^r, and also includes the boundary nodes. We use G^c to search for paths between each pair of boundary nodes $i, j \in B$, $i \neq j$ that can be used to augment G^r. We denote the elements (nodes and arcs) in the k-th shortest-path of vehicle g from i to j in G^c as $\mathcal{P}^k_{ijg}(G^c)$, which allows us define an *augmented network*

$$G^a = \left(\bigcup_{\substack{i,j \in B, i \neq j \\ g \in V,\ k=1,\ldots,K}} \mathcal{P}^k_{ijg}(G^c) \right) \cup G^r,$$

where $G^r \subseteq G^a \subseteq G$, and K is the maximum number of paths allowed. Figure 2.4a shows an initial network $G = (N, A)$ while Figure 2.4b shows a possible reduced network G^r with boundary nodes in gray. Figure 2.4c shows the corresponding complement network and Figure 2.4d shows an augmented network, which in this case contains only one path (if any exist) for each pair of boundary nodes in G^c.

We enforce the following rules when constructing G^a. For each pair of nodes $i, j \in B$, $i \neq j$ and each vehicle g, we augment G^r with $\mathcal{P}^1_{ijg}(G^c)$, which contains the elements of a shortest-path between i and j in G^c with arc costs given by the sum of



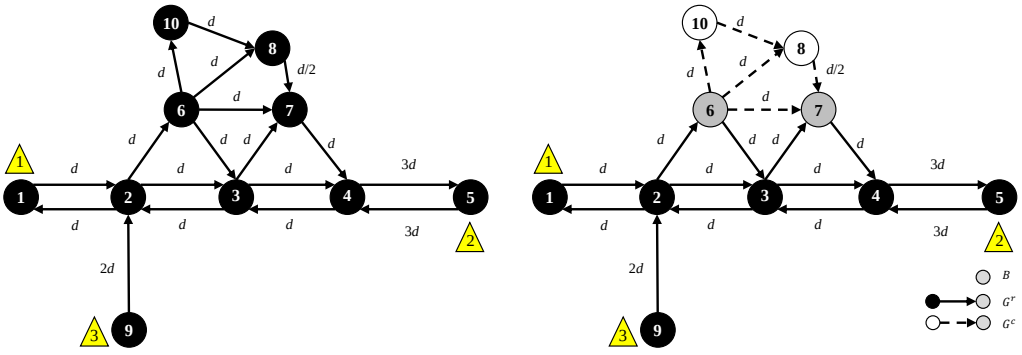
(a) $G = (N, A)$ (b) $G^r = (N^r, A^r)$ (c) $G^c = (N^c, A^c)$ (d) $G^a = (N^a, A^a)$

Figure 2.4: Network Structures Used to Solve RASTC

travel and minimum waiting times for each vehicle. If such path consists of arc (i, j) only, then we also add $\mathcal{P}^2_{ijg}(G^c)$, the *second* shortest-path between i and j in G^c, to G^r in order to allow vehicles to wait at a non-boundary node in G^c, which may be optimal. Note that if $\mathcal{P}^2_{ijg}(G^c) \neq \emptyset$, then it must contain a non-boundary node in G^c that can be used for waiting, which is not the case if the path has only one arc. We illustrate the importance of this construction in the following example, which is also relevant when discussing the correctness of our decomposition algorithm in Section 2.4.2.

This example illustrates an instance of RASTC in which waiting is optimal and where the travel time for some vehicles increases with respect to their shortest path due to the geographic conflict constraint. Consider the network in Figure 2.5a, with $V = \{1, 2, 3\}$, and origin-destination pairs $(s_1, t_1) = (1, 5)$, $(s_2, t_2) = (5, 1)$, and $(s_3, t_3) = (9, 6)$. We assume that vehicles are not allowed to wait at nodes 2 and 6, and that the distance to enforce arc-arc and arc-node conflicts is d. Moreover, we assume unit speeds such that the travel times displayed on each arc are equal to the distance between nodes. The optimal solution to RASTC on G is that all vehicles depart their origins at time 0, using paths $1 \to 2 \to 6 \to 8 \to 7 \to 4 \to 5$,

$5 \to 4 \to 3 \to 2 \to 1$, and $9 \to 2 \to 6$, for Vehicles 1, 2, and 3, respectively. At time $3d$, Vehicle 2 is at node 4 and Vehicle 1 is at node 8, which means that Vehicle 1 must wait $\frac{d}{2}$ units of time until Vehicle 2 arrives at node 3. In this case, the optimal value of RASTC is $z^* = \frac{8d}{6d} = \frac{4}{3}$ (given by Vehicle 1's deviation from its shortest path). Figure 2.5b shows a possible reduced network, G^r. In an optimal solution to RASTC on G^r, Vehicle 1 has to wait $6d$ units of time at node 1 until Vehicle 2 finishes its route, before traveling to node 5 using path $1 \to 2 \to 3 \to 4 \to 5$. As a result, $z(G^r, \Lambda, \Gamma, \Upsilon) = \frac{12d}{6d} = 2$. Observe that Vehicle 1 cannot use nodes 2, 3, 4, 6 or 7 while Vehicle 2 is moving because this will create a conflict. To create G^a, the shortest path between boundary nodes for any vehicle is arc $(6, 7)$. If only this arc is added to G^r and RASTC is solved again on G^a, then the optimal solution will not change because arc $(6, 7)$ alone does not help Vehicle 1 in avoiding Vehicle 2. However, if we add the second shortest path to G^r, which is given by $6 \to 8 \to 7$, then the optimal solution on G^a is optimal to RASTC on G.



(a) $G = (N, A)$ (b) $G^a = (N^a, A^a)$

Figure 2.5: Construction of G^a

By construction, G^r is augmented with elements from G^c, thus the new elements added to G^a contain at least one non-boundary node, either from $\mathcal{P}^1_{ijg}(G^c)$ or $\mathcal{P}^2_{ijg}(G^c)$, for each vehicle g. The distance between these new nodes and some elements already

in G^r may be less than d, creating a conflict. However, if both the maximum waiting time and the geographic conflict constraints are relaxed for these new nodes only, then they can be used for waiting. Using these elements, we define the lower bound problem RASTC-R as a relaxation of RASTC in which Constraints (2.6)–(2.14) are not enforced for elements in $G^a \setminus G^r$ (e.g., white nodes and dotted arcs in Figure 2.4d). That is, there is no geographical conflict or maximum waiting time constraints when vehicles use those elements in $G^a \setminus G^r$. We use $z_R(G, \Lambda, \Gamma, \Upsilon)$ to denote the optimal solution to RASTC-R$(G, \Lambda, \Gamma, \Upsilon)$ and vector $(\mathbf{x}, \boldsymbol{\tau})$ to describe a feasible solution to RASTC or RASTC-R, where $\mathbf{x}$ and $\boldsymbol{\tau}$ contain the values of the x- and τ-variables, respectively. We use $(\hat{\mathbf{x}}, \hat{\boldsymbol{\tau}})$ and $(\check{\mathbf{x}}, \check{\boldsymbol{\tau}})$ to denote optimal solutions to an upper and a lower bound problem, respectively.

The following propositions state some useful bounds for our decomposition algorithm, where we note that $z^* = z(G, \Lambda, \Gamma, \Upsilon)$.

Proposition 3 *The following conditions are satisfied for any network G, any reduced network G^r, and any conflict sets $\bar{\Lambda} \subseteq \Lambda$, $\bar{\Gamma} \subseteq \Gamma$, and $\bar{\Upsilon} \subseteq \Upsilon$.*

1. $z(G^r, \Lambda, \Gamma, \Upsilon) \geq z(G, \Lambda, \Gamma, \Upsilon)$
2. $z(G, \Lambda, \Gamma, \Upsilon) \geq z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$

Proposition 4 *For a given network G and a reduced network G^r such that the optimal solution to RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) is feasible to RASTC($G^r, \Lambda, \Gamma, \Upsilon$), where $\bar{\Lambda} \subseteq \Lambda$, $\bar{\Gamma} \subseteq \Gamma$, and $\bar{\Upsilon} \subseteq \Upsilon$, then $z(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) \geq z(G, \Lambda, \Gamma, \Upsilon)$.*

Proposition 5 *For a given network G, a reduced network $G^r \subset G$, and conflict sets $\bar{\Lambda} \subseteq \Lambda$, $\bar{\Gamma} \subseteq \Gamma$, and $\bar{\Upsilon} \subseteq \Upsilon$, $z(G, \Lambda, \Gamma, \Upsilon) \geq z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$.*

2.4.2 Decomposition Algorithm

Algorithm 1 describes our network decomposition approach. Line 4 initializes conflict sets $\bar{\Lambda}$, $\bar{\Gamma}$, and $\bar{\Upsilon}$ to empty, as they will be dynamically populated when encountering conflicts. Line 4 also initializes the upper and lower bound values, UB_0 and LB_0, and a counter i to track the number of iterations. Line 2 constructs a *feasible* reduced network that guarantees that at least T paths exists from s_g to p_g for every vehicle $g \in V$. This step is performed using Dijkstra's algorithm for $T = 1$ (Dijkstra, 1959) or Yen's k shortest-path algorithm for $T > 1$ (Yen, 1971). The loop in lines 5–14 is executed until convergence and consists of *upper bound* (Lines 5 and 11) and *lower bound* (Lines 7 and 8) routines. Using Proposition 4, Line 5 obtains an upper bound on z^* by solving RASTC on the reduced network using Algorithm 2 (see Section 2.4.3), where conflict constraints are added dynamically in a cutting-plane fashion. This strategy drastically reduces the number of x-, e-, and f-variables, as well as the number of conflict constraints. Line 5 also produces a feasible solution for RASTC$(G, \Lambda, \Gamma, \Upsilon)$, whose objective value and solution are stored in an incumbent in Line 11. In Line 7, our algorithm constructs the augmented network G^a induced by G^r. Line 8 produces a lower bound on z^* by following the rules from Section 2.4.1 to construct G^a. Section 2.4.4 provides a polynomial time algorithm to construct G^a. Using Proposition 5, Line 8 constructs a lower bound on z^* by solving problem RASTC-R$(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, whose optimal value is saved in Line 9.

Lines 10 and 11 verify a first stopping condition for our algorithm. If $(\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$ is feasible to RASTC (i.e., no conflict or maximum waiting time violations in $G^a \setminus G^r$), then such solution is optimal. Lines 13 and 14 verify an additional stopping condition that occurs when the optimal solution to RASTC-R$(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ only uses elements in G^r or when this solution uses elements in $G^a \setminus G^r$ but has the same objective

function value equal to the best know upper bound. In such cases, upper and lower bound values are the same, and the incumbent $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}})$ is optimal. If none of these conditions is satisfied, then G^r is augmented in Line 8, and the algorithm goes to Line 5. Although in Line 8 we augment G^r, we only allow new elements to be used by vehicles needing them, according to $(\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$. This reduces the number of binary variables in the problems solved in Lines 5 and 8. Algorithm 1 can stop at an iteration i in which $UB_i > LB_i$ as a result of Lines 10 and 11.

Algorithm 1 : Network Decomposition Algorithm for RASTC

1: Initialize $\bar{\Lambda} = \emptyset$, $\bar{\Gamma} = \emptyset$, $\bar{\Upsilon} = \emptyset$, $UB_0 = \infty$, $LB_0 = 0$, and set counter $i = 0$
2: Initialize G^r with a set of T paths from s_g to p_g in G for each $g \in V$
3: **while** $UB_i > LB_i$ **do**
4: Set $i = i + 1$
5: Solve RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) to obtain an optimal solution $(\hat{\mathbf{x}}^i, \hat{\boldsymbol{\tau}}^i)$, optimal value $\hat{z}^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, and updated sets $\bar{\Lambda}$, $\bar{\Gamma}$, and $\bar{\Upsilon}$ (see Section 2.4.3)
6: Set $UB_i = \hat{z}^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ and update the incumbent solution $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}}) \leftarrow (\hat{\mathbf{x}}^i, \hat{\boldsymbol{\tau}}^i)$ and objective $\bar{z} = UB_i$
7: Calculate B and construct G^a using G^r (see Section 2.4.4)
8: Solve RASTC-R($G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) to obtain an optimal solution $(\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$ and optimal value $\check{z}^i(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$
9: Set $LB_i = \check{z}^i(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$
10: **if** $(\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$ is feasible for RASTC **then**
11: Incumbent $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}}) \leftarrow (\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$ is optimal with objective $\bar{z} = LB_i$. Go to Step 15
12: **if** $UB_i = LB_i$ **then**
13: Incumbent $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}})$ is optimal with objective $\bar{z} = UB_i$. Go to Step 15
14: **else** Augment G^r with the elements used in $(\check{\mathbf{x}}^i, \check{\boldsymbol{\tau}}^i)$ that are not in G^r
15: Return $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}})$ and $\bar{z}$

At each iteration $i > 1$ of Algorithm 1, we tighten the MIP used in Line 5 by updating the value of the big-M parameters. We use $M_g = UB_{i-1} z^g_{SP}$ and

$M_{gh} = UB_{i-1} \max\{z^g_{SP}, z^h_{SP}\}$ for vehicles $g, h \in V$. We initialize the big-M values using the same expressions, but having a specific *UB*-parameter for vehicle $g \in V$ given by $\sum_{h \in V} z^h_{SP}/z^g_{SP}$, and for each pair of vehicles $g, h \in V$ given by $\max\{\sum_{\ell \in V} z^\ell_{SP}/z^g_{SP}, \sum_{\ell \in V} z^\ell_{SP}/z^h_{SP}\}$. These values capture the worst-case situation in which vehicles move one at a time. Using this strategy, the MIP becomes stronger as Algorithm 1 progresses because the *UB*-values are nonincreasing. Before proving the finite termination and correctness of Algorithm 1, we provide more details on the *upper bound* (Lines 5 and 11) and *lower bound* (Lines 7 and 8) routines.

2.4.3 Upper Bound

We use Algorithm 2 to solve RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) in Line 5 of Algorithm 1. After initializing i, Line 2 solves RASTC over G^r using a subset of conflicts. Line 3 stores an optimal solution and its objective value in an incumbent. The loop in Lines 4–8 iterates until the incumbent solution has no conflict violations. Line 6 identifies such violations using $\hat{\ell}$-, $\hat{u}$-, $\check{\ell}$-, and $\check{u}$-parameters. This can be done in $O(|N|^2|V|^2)$ steps by comparing all arcs in the paths of every pair of vehicles, where $|N|$ bounds the number of arcs in any path. Line 7 solves RASTC using the updated conflicts sets and Line 8 updates the incumbent with the resulting optimal objective value and solution. Line 9 returns an optimal solution to RASTC($G^r, \Lambda, \Gamma, \Upsilon$), its objective value, and the updated conflict sets. Algorithm 2 finishes in a finite number of iterations because of the finite size of the conflict sets. The solution obtained upon termination is feasible to RASTC($G^r, \Lambda, \Gamma, \Upsilon$) and also optimal given Proposition 4. This means that $z(G^r, \Lambda, \Gamma, \Upsilon) = \hat{z}(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$.

Algorithm 2 : Upper Bound Algorithm for RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$)

1: Set counter $i = 0$

2: Solve RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) to obtain an optimal solution $(\mathbf{x}^i, \boldsymbol{\tau}^i)$ and optimal value $z^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$

3: Set $\hat{z}(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) = z^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ and update the incumbent solution $(\hat{\mathbf{x}}, \hat{\boldsymbol{\tau}}) \leftarrow (\mathbf{x}^i, \boldsymbol{\tau}^i)$

4: **while** $(\mathbf{x}^i, \boldsymbol{\tau}^i)$ induces geographic conflict **do**

5: Set $i = i + 1$

6: Identify all the violated arc-arc and node-arc conflicts in $(\mathbf{x}^i, \boldsymbol{\tau}^i)$ and update $\bar{\Lambda}$, $\bar{\Gamma}$, and $\bar{\Upsilon}$

7: Solve RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) to obtain an optimal solution $(\mathbf{x}^i, \boldsymbol{\tau}^i)$ and optimal value $z^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$

8: Set $\hat{z}(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) = z^i(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ and update the incumbent solution $(\hat{\mathbf{x}}, \hat{\boldsymbol{\tau}}) \leftarrow (\mathbf{x}^i, \boldsymbol{\tau}^i)$

9: Return $(\hat{\mathbf{x}}, \hat{\boldsymbol{\tau}})$, $\hat{z}(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, $\bar{\Lambda}$, $\bar{\Gamma}$, and $\bar{\Upsilon}$

2.4.4 Lower Bound

We revisit the importance of the rules to construct G^a. In Figure 2.5b, the shortest path between boundary nodes is arc $(6, 7)$. If only this arc is added to G^r, then Algorithm 1 erroneously stops with a suboptimal solution because the solution to RASTC-R in Line 8 would be the same as the solution to RASTC($G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$). In this case, $z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ is not a lower bound on z^*. If G^a follows the rules from Section 2.4.1, then the optimal solution to RASTC-R($G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}$) is optimal to RASTC-R($G, \Lambda, \Gamma, \Upsilon$).

Using G^r as input, in Line 7 of Algorithm 1 we construct the set of boundary nodes in $O(|A^c|)$ steps and construct G^a by calculating the K (≥ 2) shortest-path between every pair of boundary nodes for each vehicle. This can be done in $O\left(|V||N^r|^2|N^c|(|N^c|\log|N^c| + |A^c|)\right)$ steps using Yen's algorithm (Yen, 1971) and Fibonacci heaps (Fredman and Tarjan, 1987). In practice, we calculate paths between boundary nodes only for those vehicles that have used such nodes. Proposition 6 describes a *filtering* process that avoids adding unnecessary paths to G^a, reducing

the number of decision variables in the problem solved in Line 8 of Algorithm 1. We use the function $c(\mathcal{P})$ to denote the total travel and minimum wait time along path $\mathcal{P}$. Proposition 7 proves the finite termination and correctness of Algorithm 1.

Proposition 6 *Consider that at any given iteration, Algorithm 1 has constructed a reduced network G^r and its corresponding complement network G^c, and that the current upper bound is UB. Then, a shortest-path $\mathcal{P}_{ijg}(G^c)$, with $i, j \in B$, $i \neq j$, and $g \in V$, will not improve UB if for every $g \in V$, $c(\mathcal{P}_{s_g,i,g}(G)) + c(\mathcal{P}_{ijg}(G^c)) + c(\mathcal{P}_{j,p_g,g}(G)) \geq z^g_{SP} UB$.*

Proposition 7 *Algorithm 1 terminates in a finite number of iterations with an optimal solution to RASTC($G, \Lambda, \Gamma, \Upsilon$).*

2.5 Computational Results

We illustrate the features of RASTC and examine the performance of our network decomposition approach on real and randomly generated networks. To perform our computations, we use C++ with CPLEX 12.7 on a desktop computer with an Intel Core i7 2.40 GHz processor and 8.0 GB RAM. We set a solution time limit of 2 hours in all the experiments. With no enhancements, CPLEX cannot solve to optimality any of the proposed instances and sometimes cannot even find an integer feasible solution, while the optimality gaps are very large for those instances where an incumbent is available within the time limit.

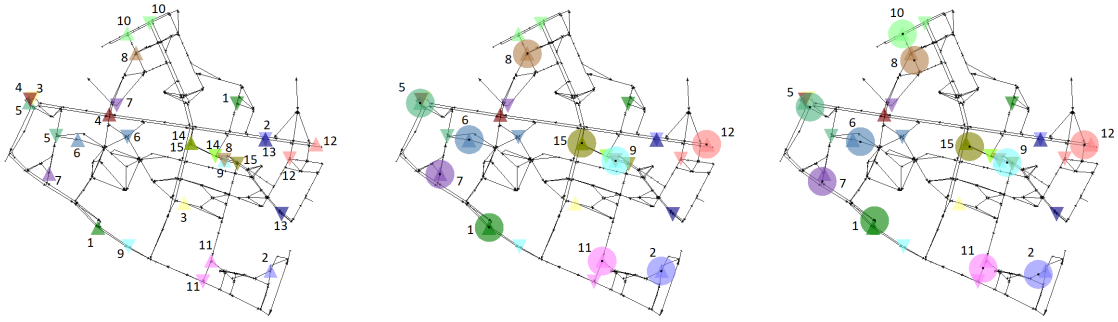
2.5.1 Berlin's Road Network Instances

We use a directed road network from the *Friedrichshain* district in east Berlin, Germany (Transportation Networks for Research Core Team, 2018), with 224 nodes and 523 arcs. The maximum euclidean distance between any two nodes is 2.25 miles,

which we denote by d_{max}. We allow vehicles to wait at nodes an unlimited amount of time and assume a speed of 35 mph for every vehicle. Moreover, we define parameter $\bar{d}$ to limit the maximum euclidean distance that a vehicle can travel between origin and destination. Using this network, Section 2.5.2 illustrates an optimal routing and scheduling plan for $|V| = 15$. Section 2.5.3 describes a procedure to create random instances out of this network and Section 2.5.4 summarizes the performance of our approach on such instances.

2.5.2 Illustrative Example

In this section we describe in detail the features of a RASTC's optimal solution with $|V| = 15$, $\bar{d} = 2d_{max}/3$, and $d = 1050$ ft. Figure 2.6a shows the network and the randomly generated origins (labeled as ▲) and destinations (labeled as ▼). The label next to the triangles is the vehicle index.



(a) (b) $t = 0$ (c) $t = 6$

Figure 2.6: Origin-Destination Pairs and Waiting at Origin Nodes

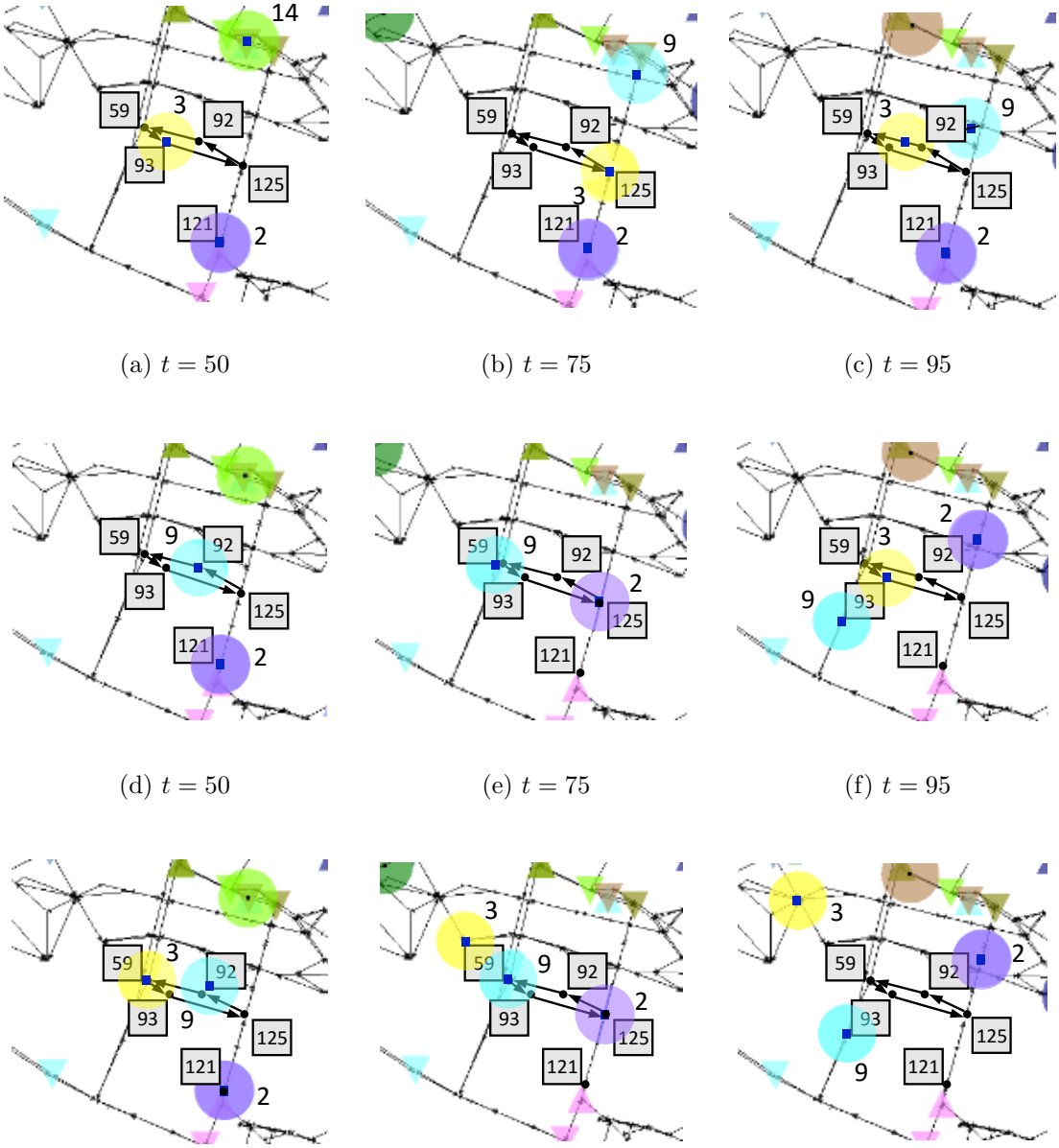
We depict the vehicles' position at various times t, where $t = 0$ is the time at which the first vehicle starts moving. We illustrate the conflict by drawing a circle of diameter equal to d around each vehicle such that any conflict results in overlapping circles. Figure 2.6b shows that 10 vehicles start traveling at $t = 0$, indicating that

some have to wait to avoid conflict. Figure 2.6c shows that Vehicle #10 only starts traveling at $t = 6$, when Vehicle #8 is far enough to avoid conflict. As expected, the minimum distance requirement leads to a deterioration in the travel time for some vehicles because some have to wait at intermediate nodes or need to deviate from a shortest path to avoid conflict. Algorithm 1 solves this instance in 55 seconds, while the MIP formulation could not solve it within 2 hours.

Figure 2.7 illustrates other features of RASTC. The label next to a node is the node index. Figures 4.5a–4.5c show that waiting at a node is optimal for some vehicles. Vehicle #3 departs from Node 93 in Figure 4.5a and because of the road (directed) network structure, it has to make a U-turn visiting Nodes 125, 92, and 59 on the way to its destination. From $t = 50$ to $t = 95$, Vehicle #2 waits at Node 121 to avoid conflict with Vehicle #3 and the approaching Vehicle #9. Figures 4.4d–2.7f illustrate the case where Vehicle #2 is not allowed to wait at Node 121 (i.e., $b^2_{121} = 0$). Vehicle #2 visits Node 121 at $t = 50$ and continues to its destination while Vehicle #3 waits at Node 93. Vehicle #3 starts moving at $t = 95$ when Vehicle #9 is far enough. Figures 2.7g–2.7i illustrate the case of heterogeneous vehicle speeds. We increase Vehicle #3's speed on arc $(93, 125)$ to twice the speed in other arcs. As a result, Vehicle #3 departs Node 93 at some time $t < 50$ and by $t = 50$ is already at Node 59, which allows Vehicles #9 and #2 to freely travel without waiting.

2.5.3 Random Instance Generation

We create instances with 5, 10, 15 and 20 vehicles and with randomly generated origin and destination nodes. To induce various trip lengths, we use $\bar{d} \in \{d_{max}/3, 2d_{max}/3, d_{max}\}$ to represent short (S), medium (M), and long (L) trips, respectively. For each combination of $|V|$ and $\bar{d}$, we generate five random instances (replications). Moreover, for each trip length we solve problems with



(a) $t = 50$ (b) $t = 75$ (c) $t = 95$

(d) $t = 50$ (e) $t = 75$ (f) $t = 95$

(g) $t = 50$ (h) $t = 75$ (i) $t = 95$

Figure 2.7: Waiting at Intermediate Nodes and Heterogeneous Vehicle Speeds

$d \in \{105, 210, 420\}$ (in feet), representing short (S), medium (M), and large (L) distance requirements for the geographic conflict.

2.5.4 Results

In this section, we compare the performance of our decomposition algorithm and the MIP in (2.3)–(**??**). The first three columns of Table 2.1 describe the instance solved, including the number of vehicles ($|V|$) and the different levels of $\bar{d}$ and d. We vary $\bar{d}$ to control the instance difficulty. Increasing $\bar{d}$ leads to longer trips, making it more likely for vehicles to encounter conflict. Longer trips also result in larger G^r and G^a networks, increasing the difficulty of the subproblems. Likewise, increasing d induce more conflicts, which will likely increase the number of iterations needed by our approach.

In Table 2.1, r is the percentage of instances (out of five) solved to optimality within a 2 hour time limit using the proposed decomposition approach. Additionally, t_{min}, t_{avg}, and t_{max} are the minimum, average, and maximum running times (in seconds) across solved instances for each combination of $|V|$, $\bar{d}$ and d. For those instances that timed out, g_{min}, g_{avg}, and g_{max} report the minimum, average, and maximum optimality gap calculated as $100(UB_k - LB_k)/UB_k$, where k is the last iteration before timing out. The optimality gap when $r = 100$ is zero, which we report as "-". We do not report the solution time when $r = 0$. As expected, the solution time increases as $|V|$, $\bar{d}$, and d increase.

Our approach can solve all instances with $\bar{d}$ of type S to optimality, regardless of the number of vehicles and value of d. Table 2.1 also reports the time to obtain and the quality of the first optimality gap. These values are given by t^1_{avg} and $g^1_{avg} = 100(UB_1 - 1)/UB_1$, where t^1_{avg} is the average time required to solve the first upper bound problem. Note that the objective function value in RASTC is always at least equal to one. We also report the average number of iterations to solve an instance to optimality or before time-out, which is given by i_{avg}. On average, our decomposition

algorithm requires few iterations and provides an initial feasible solution (UB) and initial gap in relatively short time.

We calculate the proportion of x-variables used in the decomposition strategy with respect to the MIP. We report the average value of this metric across vehicles and replications (times 100) as a proxy of the average number of network components used by each vehicle. We report these values for the upper and lower bound problems in the last iteration, which we denote by $|\mathbf{x}|^u_{avg}$ and $|\mathbf{x}|^\ell_{avg}$, respectively. We also report the proportion of τ-variables, which we denote by $|\tau|^u_{avg}$ and $|\tau|^\ell_{avg}$. The values c_{avg} and v_{avg} are the proportion of constraints and variables in the last lower bound subproblem (the largest problem solved), relative to to the MIP. These metrics show that our approach significantly reduces the number of variables and constraints used. For example, instances 20-S-L require on average only 0.2% of the constraints, 0.3% of the variables, no more than 3.4% of the arc variables, and no more than 6.21 % of the continuous variables required in the MIP.

2.5.5 Random Network Instances

Section 2.5.1 focus on analyzing the performance of our approach for several instances out of the same road network. In this section, we study the performance of our approach on randomly generated networks. Section 2.5.6 describes a procedure to such instances and Section 2.5.7 summarizes the performance metrics.

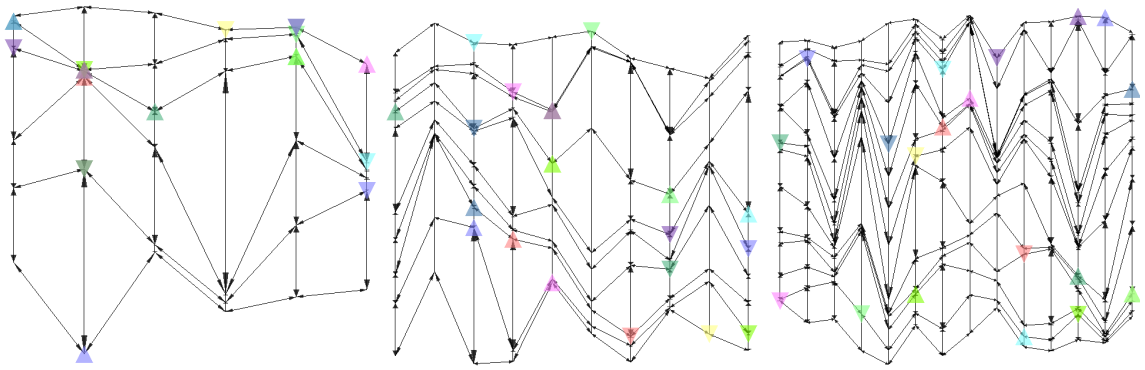
2.5.6 Random Instance Generation

We generate layered networks in order to control the distance between nodes and arcs, which directly affects the existence of conflict. This layered networks also provide many alternative paths for each vehicle, testing the limits of our decomposition

Table 2.1: Computational Performance of the Proposed Decomposition Approach on Berlin's Road Network Instances ($T = 1$, $K = 2$)

$\|V\|$	$\bar{d}$	d	r	t_{min}	t_{avg}	t_{max}	g_{min}	g_{avg}	g_{max}	t^1_{avg}	g^1_{avg}	i_{avg}	$\|\mathbf{x}\|^u_{avg}$	$\|\mathbf{x}\|^\ell_{avg}$	$\|\tau\|^u_{avg}$	$\|\tau\|^\ell_{avg}$	c_{avg}	v_{avg}
5	S	S	100	0.45	1.07	1.84	-	-	-	0.95	0.00	1.00	1.12	1.12	3.07	3.07	0.40	0.31
		M	100	0.81	1.31	1.80	-	-	-	0.76	3.32	1.00	1.12	1.35	3.07	3.34	0.37	0.32
		L	100	0.04	0.54	1.16	-	-	-	0.70	5.01	1.00	1.12	1.39	3.07	3.39	0.27	0.24
	M	S	100	0.68	4.72	18.25	-	-	-	1.45	4.48	1.20	2.36	3.59	5.86	7.46	0.64	0.80
		M	100	0.20	7.91	34.59	-	-	-	1.15	5.34	1.00	2.19	3.66	5.57	7.50	0.57	0.71
		L	100	0.07	9.28	42.55	-	-	-	1.54	7.99	1.60	2.42	3.08	5.91	6.73	0.40	0.48
	L	S	100	0.38	15.67	73.00	-	-	-	1.47	6.95	1.40	2.36	4.52	5.91	9.16	0.75	1.02
		M	100	0.28	11.36	51.09	-	-	-	0.99	7.32	1.20	2.39	4.59	5.96	9.19	0.68	0.92
		L	100	0.41	125.25	610.51	-	-	-	1.25	10.27	2.00	2.94	5.18	6.91	10.03	0.53	0.75
10	S	S	100	2.55	18.40	36.52	-	-	-	2.49	20.81	2.20	1.54	2.60	3.88	5.43	0.40	0.57
		M	100	12.31	29.35	70.90	-	-	-	2.78	23.96	2.40	1.57	2.99	3.89	5.99	0.38	0.54
		L	100	24.99	42.71	85.71	-	-	-	4.04	28.16	3.20	1.87	3.29	4.41	6.38	0.28	0.41
	M	S	100	4.98	52.17	153.41	-	-	-	5.46	16.73	2.20	2.53	4.14	6.09	10.83	0.49	0.73
		M	100	6.46	99.06	333.53	-	-	-	6.90	23.32	2.00	2.95	5.89	6.86	10.83	0.57	0.87
		L	80	10.30	52.36	116.99	43.17	43.17	43.17	8.40	28.48	1.60	2.65	7.31	6.36	13.14	0.47	0.74
	L	S	100	22.37	90.43	198.98	-	-	-	7.29	24.85	2.20	3.10	6.21	7.27	11.33	0.59	0.91
		M	100	48.05	997.42	4294.11	-	-	-	8.82	26.62	3.00	3.43	6.51	7.78	11.66	0.56	0.85
		L	80	61.32	789.72	2846.64	31.91	31.91	31.91	10.60	28.92	3.40	3.51	7.86	7.95	13.78	0.47	0.73
15	S	S	100	6.87	59.40	157.52	-	-	-	9.16	14.36	2.00	1.53	2.77	3.89	5.56	0.32	0.48
		M	100	6.55	97.15	264.72	-	-	-	11.39	18.94	2.40	1.73	3.19	4.25	6.17	0.31	0.47
		L	100	7.74	190.50	337.20	-	-	-	15.30	27.52	3.20	1.93	3.71	4.53	6.75	0.25	0.38
	M	S	100	11.52	324.80	800.78	-	-	-	19.20	24.43	2.40	2.62	5.01	6.12	9.34	0.46	0.73
		M	80	15.96	209.61	369.95	32.91	32.91	32.91	24.18	27.16	2.20	2.74	6.33	6.35	11.19	0.46	0.73
		L	60	24.27	1224.10	3430.73	13.96	27.04	40.12	34.82	35.17	2.00	2.67	6.86	6.27	12.03	0.34	0.55
	L	S	60	199.89	796.85	1933.25	4.92	23.77	42.61	40.19	30.16	2.40	3.18	7.06	7.33	12.82	0.54	0.85
		M	40	269.35	1327.51	2385.67	1.51	26.58	45.82	65.45	35.53	2.60	3.27	8.31	7.55	14.35	0.52	0.80
		L	20	323.40	323.40	323.40	2.63	18.75	46.68	80.13	37.58	2.60	3.56	9.76	8.12	16.61	0.42	0.63
20	S	S	100	10.41	99.50	154.74	-	-	-	16.82	18.75	1.80	1.44	2.68	3.66	5.29	0.27	0.42
		M	100	73.91	144.85	191.03	-	-	-	23.13	28.15	2.00	1.56	2.85	3.87	5.54	0.25	0.39
		L	100	167.70	551.37	1631.14	-	-	-	27.98	33.62	3.60	2.08	3.41	4.65	6.21	0.20	0.31
	M	S	20	587.78	587.78	587.78	20.19	32.33	36.85	87.07	30.26	1.40	2.22	8.01	5.54	13.84	0.46	0.74
		M	40	398.44	484.54	570.64	0.99	25.27	37.88	98.89	31.61	1.80	2.55	8.15	6.17	14.00	0.41	0.66
		L	20	1349.86	1349.86	1349.86	0.31	23.43	48.11	130.25	36.35	2.00	2.65	7.52	6.33	13.08	0.29	0.45
	L	S	40	1151.22	1379.54	1607.86	2.62	14.78	31.71	75.13	30.61	2.40	3.21	9.44	7.48	16.14	0.47	0.75
		M	40	859.81	1319.64	1779.46	1.41	28.77	45.52	224.56	41.23	2.20	3.31	9.91	7.71	16.87	0.46	0.70
		L	0	-	-	-	7.25	31.17	45.69	333.58	47.28	1.50	3.23	10.99	7.64	18.34	0.36	0.54

approach as many iterations may be needed to find useful network components. We create layered networks having n layers and n nodes per layer, resulting in $|N| = n \times n$ nodes. Nodes are arranged in a square of dimension 200×200 such that that layers are evenly separated. The position of each node within a layer is chosen randomly. The i-th node (from top to bottom) of the q-th layer (from left to right) is connected to the i-th node of layers $q+1$ and $q-1$ and to nodes $i-1$ and $i+1$ in the same layer, whenever these nodes exist. The resulting arrangement of nodes and arcs creates multiple conflicts when vehicles move. Figure 3.6 shows random layered networks with $|N| = 6 \times 6$, $|N| = 10 \times 10$, and $|N| = 14 \times 14$ nodes. All vehicles are assumed to have a unit speed.



(a) $(|N|, |A|) = (6 \times 6, 120)$ (b) $(|N|, |A|) = (10 \times 10, 360)$ (c) $(|N|, |A|) = (14 \times 14, 728)$

Figure 2.8: Random Layered Networks of Different Sizes

We create networks with sizes ranging from 4×4 to 14×14 nodes. For each size, we create instances with 10, 15, 20, and 25 vehicles with randomly generated origin and destination nodes for each vehicle and with $d \in \{2, 8\}$, representing short (S) and large (L) distance requirements. We impose no restriction on the trip's length of any vehicle (i.e., $\bar{d} = \infty$) and assume that there are no minimum or maximum waiting times on any node. For each combination of $|N|$, $|V|$, and d, we generate 3 random replications. Because of the number of alternative paths and the number of

arcs, we expect these layered networks to be more difficult than those from Berlin's road network.

2.5.7 Results

Table 2.2 shows the performance of our method on random layered networks. The first three columns characterize the instance, including $|N|$, $|A|$, $|V|$, and d. We compare our decomposition approach (DA) with the MIP solved with CPLEX on the complete network but adding the conflict constraints as encountered via cutting planes. That is, using Algorithm 2 with call RASTC($G, \bar{\Lambda} = \emptyset, \bar{\Gamma} = \emptyset, \bar{\Upsilon} = \emptyset$). The columns t_{DA} and t_{MIP} are the average solution times (in seconds) for our approach and the MIP, respectively, for instances solved within the time limit, whose number is shown in parenthesis. The column g_{DA} is the average optimality gap for the decomposition approach calculated as in Table 2.1. Our decomposition approach solves instances of up to 10 vehicles on 14×14 networks, whereas the largest instance solved with the MIP via cutting planes has 10 vehicles on the easiest setup of a 6×6 network. These results show that the proposed decomposition approach outperforms the MIP. The value of z^* is the average optimal objective function value for instances solved to optimality. If $z^* > 1$, then the geographic conflict requires some vehicles to wait at intermediate nodes or to deviate from their shortest path.

As expected, the performance of our decomposition algorithm depends on multiple factors, including $|N|$, $|V|$, $\bar{d}$, and d. For instance, in Berlin's Friedrichshain network (224 nodes and 523 arcs), we optimally solve some of the 20-vehicle instances with large $\bar{d}$ and medium d, while in the layered networks, we solve instances of 25 vehicles, 100 nodes, and 360 arcs but a small d. Our approach can handle more vehicles for

Table 2.2: Computational Performance of the Proposed Decomposition Approach Versus MIP on Random Layered Network Instances ($T = 1$, $K = 2$)

$(\|N\|, \|A\|)$	$\|V\|$	d	t_{MIP}	t_{DA}	g_{DA}	z^*	$(\|N\|, \|A\|)$	$\|V\|$	d	t_{MIP}	t_{DA}	g_{DA}	z^*
$(4 \times 4, 48)$	10	S	22 (3)	16 (3)	-	1.40	$(6 \times 6, 120)$	10	S	1951 (1)	21 (3)	-	1.07
		L	21 (3)	11 (3)	-	1.42			L	-	19 (3)	-	1.15
	15	S	26 (3)	11 (3)	-	1.44		15	S	-	31 (3)	-	1.15
		L	216 (3)	21 (3)	-	1.45			L	-	62 (3)	-	1.19
	20	S	431 (3)	23 (3)	-	1.42		20	S	-	59 (3)	-	1.23
		L	54 (1)	76 (3)	-	1.42			L	-	233 (3)	-	1.27
	25	S	1788 (1)	108 (3)	-	1.46		25	S	-	977 (3)	-	1.35
		L	-	169 (3)	-	1.49			L	-	1878 (2)	20	1.37
$(8 \times 8, 224)$	10	S	-	49 (3)	-	1.08	$(10 \times 10, 360)$	10	S	-	61 (3)	-	1.03
		L	-	53 (3)	-	1.19			L	-	68 (3)	-	1.08
	15	S	-	63 (3)	-	1.10		15	S	-	174 (2)	35	1.09
		L	-	250 (3)	-	1.20			L	-	96 (1)	40	1.16
	20	S	-	104 (2)	20	1.17		20	S	-	1632 (2)	36	1.08
		L	-	252 (2)	26	1.26			L	-	1550 (1)	38	1.16
	25	S	-	321 (1)	22	1.12		25	S	-	636 (1)	39	1.14
		L	-	1396 (2)	45	1.26			L	-	-	47	-
$(12 \times 12, 528)$	10	S	-	105 (3)	-	1.23	$(14 \times 14, 728)$	10	S	-	888 (2)	28	1.02
		L	-	227 (3)	-	1.17			L	-	509 (1)	39	1.07
	15	S	-	202 (2)	25	1.20		15	S	-	-	42	-
		L	-	324 (3)	-	1.22			L	-	-	43	-
	20	S	-	236 (1)	35	1.08		20	S	-	-	47	-
		L	-	-	34	-			L	-	-	43	-
	25	S	-	-	37	-		25	S	-	-	43	-
		L	-	-	45	-			L	-	-	51	-

some parameter combinations (e.g., small d or $\bar{d}$) or under other problem setups such as imposing conflict constraints only in some parts of the network, which is allowed in our modeling.

2.6 Concluding Remarks

We present an approach to impose geographical conflict conditions in a route assignment and scheduling problem. Using a polynomial-time pre-processing step, we identify regions in the network where geographic conflict may occur and provide conditions on the departure times from each node that avoid conflict. By using a big-M approach to model disjunctive constraints, we reformulate this problem into a mixed-integer program that is very challenging to solve. To improve the solution time, we introduce a decomposition algorithm that takes advantage of the problem's network structure. Instead of solving the problem on the initial (complete) network, we limit our search to the most important sub-networks for each vehicle, which we dynamically construct as the optimization problem is solved. Solving the problem on a reduced network provides an upper bound on the optimal objective function value, which is helpful to eliminate network components that are not used in any optimal solution. We obtain a lower bound by allowing vehicles to use elements outside the reduced network, ignoring conflict or maximum waiting times. This strategy, combined with an iterative procedure to prevent conflicts as they are encountered, helps us maintain a small-sized problem which translates into favorable solution times. Our algorithm is able to solve instances that the MIP formulation cannot solve.

Our approach takes advantage of the sparsity of an optimal solution to construct a reduced network that is sufficient to identify an optimal solution for the complete network. Our approach can be generalized to other problems, where the solution is sparse and possibly without a network structure. The proposed decomposition approach follows the same principle of other classic methods: generate useful problem elements as needed while keeping a "master problem" small. In RASTC, we generate path segments aiming to improve the incumbent solution in the reduced network,

guaranteeing the existence of a feasible solution at any iteration. Our method can be initialized using any set of candidate paths, for instance using the low-risk routes from Carotenuto *et al.* (2007). Moreover, our approach preserves the structure of RASTC at every iteration, which is advantageous because there is always a connection between integer and continuous variables, a known problem in other approaches such as Benders decomposition. Our proposed decomposition approach does not rely on duality theory, avoiding the challenges of potentially weak linear relaxations. However, our approach can be coupled with other methods to solve large problems (e.g., column generation, Benders decomposition, Lagrangian relaxation), which can be used to accelerate the solution of the problems on the reduced or augmented networks.

Although in this chapter we focus on two-dimensional problems where conflict is prevented everywhere in the network, the vehicle coordination analysis also applies to three-dimensional problems (e.g., aerial or underwater vehicles) and other problems where conflict needs to be prevented only in some areas. A future research path is to include a variable speed or a discrete speed profile that vehicles can choose when traversing an arc. Currently, it is possible to approximate such variable speed profile by adding nodes along an arc where vehicles can wait. These mechanisms help approximate any acceleration-deceleration decisions along the route at the expense of additional decision variables and constraints. This is motivated by the observation that geographical conflicts may be avoided not only through route selection and scheduling but also by choosing an appropriate speed at specific times. Additionally, our centralized approach can be used as a benchmark to determine the quality of decentralized approaches that locally resolve the geographical conflicts.

Chapter 3

FIXED-CHARGE NETWORK DESIGN WITH PIECEWISE LINEAR COST FUNCTION

This chapter is organized as follows. In Section 3.1 we provide a literature review of fixed-charge network design problems. In Section 3.2, we study the current formulations for this problem, also introducing a generalized Multiple Choice (MC) formulation. In Section 3.2.3, we propose an alternative formulation that uses a reduced number of variables. Although this alternative formulation is computationally efficient, it reaches its limits when solving large-scale instances. To overcome this problem, we introduce a procedure to iteratively construct simpler cost functions that uses less decision variables. In Section 3.3, we introduce the proposed general method to underestimate the cost function thus providing a relaxation to the problem. We provide a formulation for the problem that utilizes partial convex cost underestimations in Section 3.4. In Section 3.5, we develop an exact algorithm based on a structural decomposition of the network problem. In Section 3.6, we investigate the performance of our approach and explore its limits by solving a set of random instances. Section 3.7 presents our final remarks.

3.1 Introduction and Literature Review

We study a fixed-charge network design problem. Fixed-charge network design problems (FC-NDPs) are network optimization problems in which integer variables represent decisions of whether or not to build —or to operate— an arc or a node. Because they represent general network design and operation models, FC-NDPs are useful in many application domains. These application domains include personnel

scheduling (Balakrishnan and Wong, 1990; Bartholdi III *et al.*, 1980), service network design (Crainic and Rousseau, 1986; Andersen *et al.*, 2009; Crainic, 2000), and logistic network design (Cordeau *et al.*, 2006; Geoffrion and Graves, 1974; Santoso *et al.*, 2005). Solution techniques for network design problems include polyhedral analysis and valid inequalities (Atamtürk, 2002; Atamtürk and Rajan, 2002; Bienstock and Günlük, 1996; Günlük, 1999; Raack *et al.*, 2011), decomposition approaches (Crainic *et al.*, 2001; Cruz *et al.*, 1998; Randazzo and Luna, 2001; Frangioni and Gendron, 2009), and heuristic methods (Ghamlouche *et al.*, 2003; Yaghini *et al.*, 2015; Katayama *et al.*, 2009; Balakrishnan *et al.*, 1989; Kim and Pardalos, 1999).

In this chapter we focus on solving the Carbon Capture and Storage (CCS) problem, which is to design a minimum-cost pipeline network to capture CO2 from sources (e.g., power plants, oil refineries) and to transport it to geologic reservoirs (e.g., depleted oil and gas fields) for its long-term storage. Despite its benefits (Eto *et al.*, 2013; Sathre *et al.*, 2017; Craig *et al.*, 2017), CCS requires large infrastructure investments, which could deter governments and national agencies from its adoption. For this reason, developing new tools to solve large-scale instances to optimality is critical to continue contributing to climate stabilization.

Currently, there are more than 38 large-scale CCS approved projects around the world (Global, 2016). In CCS, we need to decide the location of reservoirs and sources and also the necessary pipe diameters to ensure enough flow to capture a target amount of carbon. Because of the nature of the problem, solving CCS is challenging (Middleton and Bielicki, 2009), demonstrating the need for models and algorithms to solve real-world instances.

Rothfarb *et al.* (1970) introduces an early work on gas pipeline network design in which pipes need to be selected. This work does not consider selection of sources or reservoirs. The selection of pipes and junction nodes for connected tree pipeline

networks is studied in Bhaskaran and Franz (1979). Researchers at Battelle's Joint Global Change Research Institute (JGCRI) present a survey of procedures that make CCS technologies viable climate stabilization candidates to be widely deployed (Dooley *et al.*, 2006). Another CCS framework is discussed in Kobos *et al.* (2007), which considers that sources are directly connected to reservoirs, reducing the CCS formulation to a matching problem. A more realistic and general CCS problem is investigated through the scalable infrastructure model CCS (SimCCS) (Middleton and Bielicki, 2009). This model considers a general network structure in which the set of sources and reservoirs is a subset of the network nodes. A mixed-integer formulation is presented for the case that the cost function is piecewise linear with constant variable cost for each pipe diameter. Other objective functions including elements such as tax and temporal requirements are investigated in Kuby *et al.* (2011) and Middleton *et al.* (2012). To overcome complexity of solving CCS, Middleton (2013) reformulate the discrete capacity as a continuous decision, providing lower bounds for the total network design cost in CCS.

Formally, the CCS problem is to design a network able to transport a target amount of CO2 —denoted by τ— from capturing to storing sites. The network consists of a set of source nodes (S), a set of reservoir nodes (R), a set of intermediate nodes (I), and arcs representing pipes (A). Sources and reservoirs are subject to maximum capture and storage capacities, denoted by q_i^s and q_j^r for source $i \in S$ and reservoir $j \in R$, respectively. Similarly, f_i^s and v_i^s denote the fixed cost (i.e., land purchase, construction, and technology installation) and variable operational cost (i.e., pumping and maintenance) for source $i \in S$, whereas parameters f_j^r and v_j^r represent the fixed and variable costs for reservoir $j \in R$, respectively. The minimum and maximum capacity of arc $(i, j) \in A$ —denoted by l_{ijd} and u_{ijd}— depend on the chosen pipe diameter $d \in D$, where D is the set of commercially available diameters.

Using a pipe of diameter d to transport flow from node i to node j incurs in a variable operational cost v_{ijd} per ton of CO2 transported, and a fixed-charge construction cost f_{ijd}. Solving this problem with traditional mathematical programming approaches is challenging for realistic instances because it requires a large number of discrete variables. Indeed, Guisewite and Pardalos (1990) prove that a generic FC-NDP is NP-hard. As a result, we propose a new method to guarantee the scalability of the solution approach.

In this chapter, we propose a general formulation that admits general piecewise linear cost functions and propose methods for the efficient computation of the optimal network design. This chapter is organized as follows. In Section 3.2, we study the current formulations for this problem, also introducing a generalized Multiple Choice (MC) formulation. We focus on an alternative formulation to the problem in Section 3.2.3. Although this alternative formulation is computationally efficient, it reaches its limits when solving large-scale instances. In Section 3.3, we introduce a general method to provide relaxations of the problem. In Section 3.5 we develop an exact algorithm based on a structural decomposition of the problem network. In Section 3.6, we investigate the performance of our approach and explore its limits by solving random instances. Section 3.7 presents our final remarks.

3.2 Mathematical Programming Formulation

In this section, we study different mathematical formulations to solve the network design problem. In Section 3.2.1, we explore the formulation for the CCS network design problem from Middleton and Bielicki (2009). In Section 3.2.2, we formulate the generalized problem by extending the standard formulation from Section 3.2.1. In Section 3.2.3, we explore an alternative formulation for the problem which requires less binary variables.

3.2.1 Constant Slope Cost Function

In this section, we introduce the formulation for the CCS pipeline network design problem from Middleton and Bielicki (2009). Consider a network with node set N in which $S \subseteq N$ denotes the set of sources and $R \subseteq N$ denotes the set of reservoirs. Each source $i \in S$ has a capacity given by q_i^s. The cost of capturing flow from source $i \in S$ consists of a fixed cost f_i^s and a variable cost v_i^s. Similarly, each reservoir $j \in R$ has a capacity given by q_j^r. In this case, the cost of storing flow in such reservoir consists of a fixed cost f_j^r and a variable cost v_j^r. For arc $(i, j) \in A$, we can build a pipe of diameter $d \in D_{ij}$, where D_{ij} is the set of commercially available pipes that are suitable to connect nodes i and j. Each pipe $d \in D_{ij}$ has a minimum and maximum flow capacity denoted by l_{ijd} and u_{ijd}, respectively. The maximum capacity denotes the maximum flow per time that can traverse the pipe, whereas the minimum capacity denotes the required flow that needs to traverse the pipe to satisfy physical design constraints (i.e., flow-velocity). The cost of transporting flow through arc (i, j) using pipe d consists of a fixed cost f_{ijd} and a variable cost v_{ij}. We use decision variables x_{ij} to represent the flow on arc $(i, j) \in A$, a_i to denote the flow captured at source $i \in S$, and b_j to denote the flow stored in reservoir $j \in R$. Also, we use binary variables y_{ijd}, s_i, and r_j to denote whether or not we build pipe d on arc (i, j), capture flow at source i, and store flow in reservoir j, respectively. The mixed-integer programming

model describing CCS is shown in (3.1)–(3.14).

$$\min \quad \sum_{i\in S}(f_i^s s_i + v_i^s a_i) + \sum_{j\in R}(f_j^r r_j + v_j^r b_j) + \sum_{(i,j)\in A}\sum_{d\in D_{ij}} f_{ijd}y_{ijd} + \sum_{(i,j)\in A} v_{ij}x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} - a_i + b_i = 0, \quad \forall i \in N \tag{3.2}$$

$$x_{ij} - \sum_{d\in D_{ij}} l_{ijd}y_{ijd} \geq 0, \quad \forall (i,j) \in A \tag{3.3}$$

$$x_{ij} - \sum_{d\in D_{ij}} u_{ijd}y_{ijd} \leq 0, \quad \forall (i,j) \in A \tag{3.4}$$

$$a_i - q_i^s s_i \leq 0, \quad \forall i \in S \tag{3.5}$$

$$b_j - q_j^r r_j \leq 0, \quad \forall j \in R \tag{3.6}$$

$$\sum_{i\in S} a_i \geq \tau, \tag{3.7}$$

$$\sum_{d\in D} y_{ijd} \leq 1, \quad \forall (i,j) \in A \tag{3.8}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A \tag{3.9}$$

$$a_i \geq 0, \quad \forall i \in S \tag{3.10}$$

$$b_j \geq 0, \quad \forall j \in R \tag{3.11}$$

$$y_{ijd} \in \{0,1\}, \quad \forall (i,j) \in A, d \in D_{ij} \tag{3.12}$$

$$s_i \in \{0,1\}, \quad \forall i \in S \tag{3.13}$$

$$r_j \in \{0,1\}, \quad \forall j \in R \tag{3.14}$$

Constraints (3.2) impose flow-balance conditions for each node, such that the inflow and the amount captured by the sources (if any) is consistent with the outflow and the amount stored by the reservoirs (if any). Because the binary variable y_{ijd} equals one when pipe d is selected for arc (i, j), Constraints (3.3) and (3.4) limit the minimum and maximum possible flow on arc (i, j) based on the selected minimum and maximum capacity of pipe d. Similarly, Constraints (3.5) and (3.6) limit the maximum flow that can be captured at source $i \in S$ and the maximum flow that can be stored at reservoir $j \in R$ based on their capacity. Constraint (3.7) ensures that we capture at least the target amount (τ) of CO2 across all sources in the network. Constraints (3.8) limit

the number of pipes to be utilized for transportation on arc (i, j) to be at most one. Constraints (3.9)–(3.14) enforce the nature of the decision variables. The objective function in (3.1) seeks to minimize the total cost of network construction and flow transportation through the network. The costs of capturing and storing flows at sources and reservoirs consist of a fixed cost plus a variable cost based on flow. Each pipe utilization also incurs in a fixed construction cost and a variable operation cost based on the pipe utilization (flow). This formulation assumes that the variable cost of operating each pipe depends only on the amount of flow but not on the selected pipe. In other words, the cost of pushing an extra unit of flow through an arc is the same regardless of the pipe diameter used.

3.2.2 Multiple Choice Formulation

We first investigate a more general problem than the formulation in (3.1)–(3.14). In this problem, which we call Generalized Carbon Capture and Storage problem (GCCS), we assume a nondecreasing cost structure as a function of the pipe selection and utilization. This generalization enables us to solve instances of the problem that involve pipe cost functions with diameter-specific variable cost (i.e., variable slope), which implies that the variable cost of transporting CO2 varies depending on the chosen pipe and its level of utilization. Moreover, a general nondecreasing function admits multiple realistic cost structures, including piecewise linear discontinuous forms or any piecewise linear approximation of a continuous cost function. Figure 3.1 shows three examples of possible nondecreasing cost functions that our models can handle. In this figure, each piecewise linear segment corresponds to a pipe with its corresponding fixed and variable costs. Next, we formulate the GCCS problem by extending the standard formulation (3.1)–(3.14).

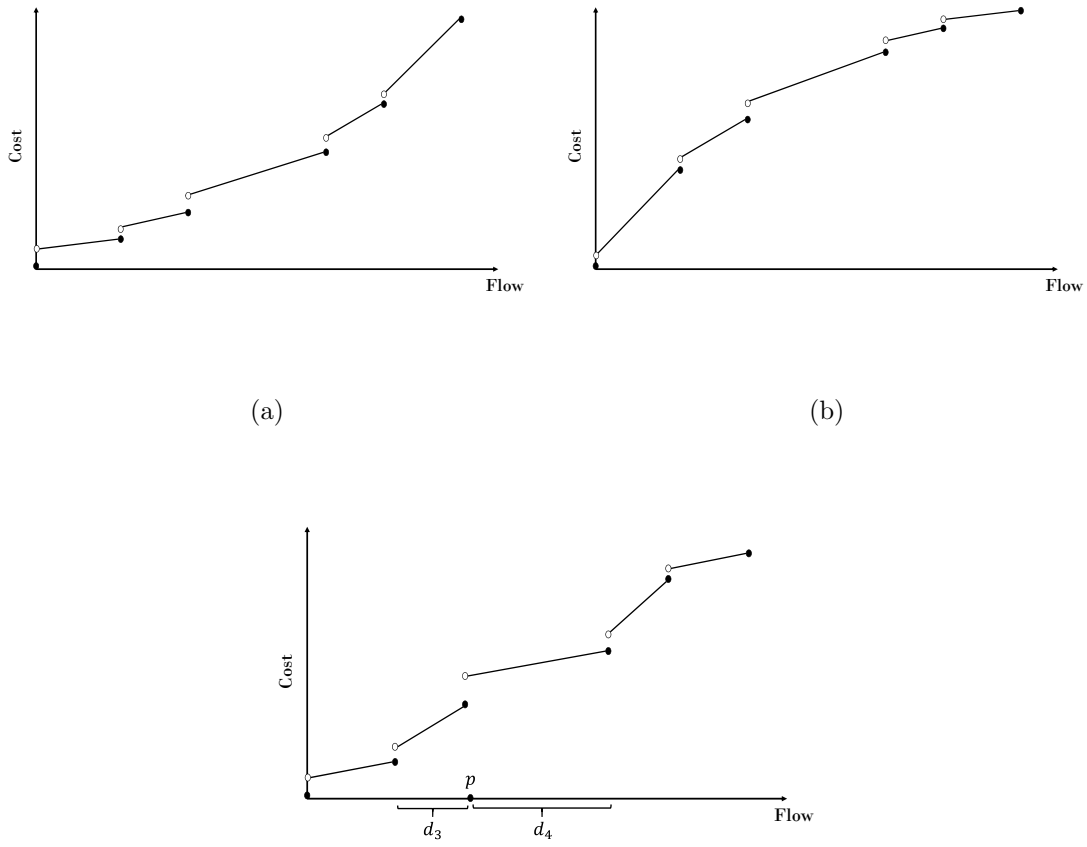


Figure 3.1: Example Nondecreasing Cost Functions

In this formulation, the flow transported through pipe d on arc (i, j) is indicated by variable x_{ijd}, where the additional index 'd' allows distinct variable cost for different pipes in D_{ij}, denoted by v_{ijd}. Constraints (3.5)–(3.8), (3.10)–(3.14), and (3.16)–(3.19), and the objective function in (3.15) describe the GCCS problem.

$$\min \quad \sum_{i \in S}(f_i^s s_i + v_i^s a_i) + \sum_{j \in R}(f_j^r r_j + v_j^r b_j) + \sum_{(i,j) \in A} \sum_{d \in D_{ij}} (f_{ijd} y_{ijd} + v_{ijd} x_{ijd}) \tag{3.15}$$

$$\sum_{j:(i,j) \in A} \sum_{d \in D_{ij}} x_{ijd} - \sum_{j:(j,i) \in A} \sum_{d \in D_{ij}} x_{jid} - a_i + b_i = 0, \quad \forall i \in N \tag{3.16}$$

$$x_{ijd} - l_{ijd} y_{ijd} \geq 0, \quad \forall (i,j) \in A, d \in D_{ij} \tag{3.17}$$

$$x_{ijd} - u_{ijd} y_{ijd} \leq 0, \quad \forall (i,j) \in A, d \in D_{ij} \tag{3.18}$$

$$x_{ijd} \geq 0, \quad \forall (i,j) \in A, d \in D_{ij} \tag{3.19}$$

Constraints (3.16) enforce the balance of flow at each node utilizing the new x-variables. Constraints (3.17) and (3.18) impose a maximum and minimum flow allowed on each pipe. In these constraints, we formulate pipe-specific conditions since we have a pipe-based flow. Constraints (3.19) enforce the non-negativity of the new flow variable. The objective function in (3.15) is to minimize the total network construction and flow transportation costs considering different variable costs for each pipe.

3.2.3 Logarithmic Formulation

We explore an alternative formulation to GCCS, that requires less binary variables and potentially reduces solution times. This is based on the observation that some minimization problems with piecewise linear convex objective function can be modeled as linear programs. However, when such cost structure is combined with a fixed-cost feature, like in classic network design problems, we need to use a set of binary variables to determine whether to utilize network elements (e.g., pipes). This is because the total cost of not using the network element must be equal to zero, whereas any positive level of utilization incurs in a positive cost. For a non-convex piecewise linear cost function, for instance that in Figure 3.1c, Formulation (3.1)–(3.14) uses a binary variable for each segment of the cost function to help accounting for the total cost of operating the chosen pipe.

There are different formulation strategies for non-convex piecewise linear cost functions. Vielma *et al.* (2010) investigate numerous MIP models for these functions. In particular, they propose a Disaggregated Convex Combination (DCC) model and a Convex Combination (CC) model in addition to a Multiple Choice (MC) formulation. The first two methods utilize the convex combination of break points of each segment in the cost function. Formally, let D be the set of line segments in a non-convex piecewise linear function. For a given segment $d \in D$, its endpoints in the function domain are defined by the set $P(d)$ and $P(D)$ denotes the set of all break points. Assuming that we are dealing with a single arc, the flow variable in the DCC model is defined as in (3.20). To illustrate this structure, suppose that a break point p is shared between line segments d_3 and d_4, as shown in Figure 3.1c. In this case, two continuous variables, $\lambda_{d_3,p}$ and $\lambda_{d_4,p}$, are related to break point p. These continuous variables are defined between zero and one, such that the summation of these variables for consecutive endpoints equals one for exactly one segment and the rest take the value of zero. All of such conditions can be achieved through linear constraints.

Equation (3.21) provides the flow formulation for the CC model. In this equation, λ_p is a continuous variable between zero and one related to break point p. In contrast to the DCC model, the CC model has only one continuous variable for each break point which leads to a formulation with less continuous variables. Similar to DCC, only two variables representing consecutive breakpoints can be positive and their sum must be equal to one.

$$x = \sum_{d \in D} \sum_{p \in P(d)} \lambda_{d,p} p \tag{3.20}$$

$$x = \sum_{p \in P(D)} \lambda_p p \tag{3.21}$$

The problem of minimizing a piecewise linear function has strong connections to SOS2 variables. Extending the ideas in Vielma *et al.* (2010), it is possible to use a

logarithmic number of binary variables to model a set of disjoint conditions, when exactly one needs to be satisfied in any problem. Now, we introduce the logarithmic DCC model and describe the logic behind this technique. Suppose that we define the flow variable x using (3.20). To ensure that only continuous variables related to endpoints of exactly one line segment can take non-zero values, we define a set of binary variables in the following fashion. Each line segment is labeled with an integer number between 0 and $|D| - 1$. Each of these numbers can be written using their binary representation, with each digit corresponding to a binary variable y_k ($k \in 1, \ldots, \log_2 |D|$). Using this representation, a combination of zero and one values of the y-variables will uniquely produce the integer number identifying a line segment. We define D_k^1 as the set of line segments with a value of one in the k-th digit of their binary representation. Similarly, we define D_k^0 as the set of line segments with a value zero in the k-th digit of their binary representation. Constraint (3.22) defines the flow through an arc with multiple pipe choices available, where the flow is based on the convex combination of endpoints of the line segments (i.e., minimum and maximum capacity of the pipe). Constraints (3.23) and (3.24) are the non-negativity and the convexity constraints in the flow representation, respectively. The binary nature of the y-variables is enforced by Constraints (3.27). Constraints (3.25) and (3.26) relate the binary reformulation with the λ-values used for each pipe. Note that (3.25) and (3.26) guarantee that the selected endpoints are selected because the solution of the

y-variables will uniquely identify a segment.

$$\sum_{d \in D} \sum_{p \in P(d)} \lambda_{d,p} p = x \tag{3.22}$$

$$\lambda_{d,p} \geq 0, \quad \forall d \in D, p \in P(d) \tag{3.23}$$

$$\sum_{d \in D} \sum_{p \in P(d)} \lambda_{d,p} p = 1 \tag{3.24}$$

$$\sum_{d \in D_k^1} \sum_{p \in P(d)} \lambda_{d,p} \leq y_k, \quad \forall k \in \{1, \ldots, log|D|\} \tag{3.25}$$

$$\sum_{d \in D_k^0} \sum_{p \in P(d)} \lambda_{d,p} \leq 1 - y_k, \quad \forall k \in \{1, \ldots, log|D|\} \tag{3.26}$$

$$y_k \in \{0, 1\}, \quad \forall k \in \{1, \ldots, log|D|\} \tag{3.27}$$

We illustrate the operation of Constraints (3.25) and (3.26) using a piecewise linear cost function with four line segments. In this case, we need two digits to represent all the function's segments because $\log_2 |D| = 2$. In this example, Constraints (3.25) and (3.26) are given by

$$(\lambda_{31} + \lambda_{32}) + (\lambda_{41} + \lambda_{42}) \leq y_1$$

$$(\lambda_{21} + \lambda_{22}) + (\lambda_{41} + \lambda_{42}) \leq y_2$$

$$(\lambda_{11} + \lambda_{12}) + (\lambda_{21} + \lambda_{22}) \leq 1 - y_1$$

$$(\lambda_{11} + \lambda_{12}) + (\lambda_{31} + \lambda_{32}) \leq 1 - y_2,$$

which lead to the following cases based on the different combinations of y-values.

$$y_1 = 0,\ y_2 = 0 \ \rightarrow \begin{cases} (\lambda_{31} + \lambda_{32}) + (\lambda_{41} + \lambda_{42}) \leq 0 \\ (\lambda_{21} + \lambda_{22}) + (\lambda_{41} + \lambda_{42}) \leq 0 \end{cases}$$

$$y_1 = 0,\ y_2 = 1 \ \rightarrow \begin{cases} (\lambda_{31} + \lambda_{32}) + (\lambda_{41} + \lambda_{42}) \leq 0 \\ (\lambda_{11} + \lambda_{12}) + (\lambda_{31} + \lambda_{32}) \leq 0 \end{cases}$$

$$y_1 = 1,\ y_2 = 0 \ \rightarrow \begin{cases} (\lambda_{11} + \lambda_{12}) + (\lambda_{21} + \lambda_{22}) \leq 0 \\ (\lambda_{21} + \lambda_{22}) + (\lambda_{41} + \lambda_{42}) \leq 0 \end{cases}$$

$$y_1 = 1,\ y_2 = 1 \ \rightarrow \begin{cases} (\lambda_{11} + \lambda_{12}) + (\lambda_{21} + \lambda_{22}) \leq 0 \\ (\lambda_{11} + \lambda_{12}) + (\lambda_{31} + \lambda_{32}) \leq 0. \end{cases}$$

These constraints ensure that exactly one of the line segments has non-zero λ-coefficients. Constraint (3.24) guarantees that the corresponding continuous λ-variables sum to one with all other continuous variables equal to zero. The following are all possible cases for the λ-values when combining (3.24) with (3.25) and (3.26) for this example.

$$y_1 = 0,\ y_2 = 0 \ \rightarrow \ \lambda_{11} + \lambda_{12} = 1, \lambda_{21} = \lambda_{22} = \lambda_{31} = \lambda_{32} = \lambda_{41} = \lambda_{42} = 0$$

$$y_1 = 0,\ y_2 = 1 \ \rightarrow \ \lambda_{21} + \lambda_{22} = 1, \lambda_{11} = \lambda_{12} = \lambda_{31} = \lambda_{32} = \lambda_{41} = \lambda_{42} = 0$$

$$y_1 = 1,\ y_2 = 0 \ \rightarrow \ \lambda_{31} + \lambda_{32} = 1, \lambda_{11} = \lambda_{12} = \lambda_{21} = \lambda_{22} = \lambda_{41} = \lambda_{42} = 0$$

$$y_1 = 1,\ y_2 = 1 \ \rightarrow \ \lambda_{41} + \lambda_{42} = 1, \lambda_{11} = \lambda_{12} = \lambda_{21} = \lambda_{22} = \lambda_{31} = \lambda_{32} = 0.$$

Computational results in Vielma *et al.* (2010) suggest that for minimizing a piecewise linear function, the size of the problem (number of line segments) highly affects the performance of each modeling technique. According to their results, the MC model works slightly better than the logarithmic DCC for problems with a small number of segments, whereas the logarithmic DCC shows the best performance for larger instances of the problem. Based on these experiments, we choose the logarithmic DCC modeling technique for our problem formulation. However, we will compare its

performance with MC. To this end, we first adapt the logarithmic DCC formulation to the CCS problem by adding the new set of y-variables and by including an indicator variable to represent whether an arc is used in the optimal design.

In the logarithmic DCC formulation, the summation of continuous λ-variables is equal to one, meaning that exactly one line segment is selected. To reformulate the GCCS using the logarithmic DCC formulation, we first need to model the possibility of not choosing any pipe for an arc. In this case, we add a line segment at the origin of the cost function, for each $(i,j) \in A$. This is illustrated in the piecewise linear cost functions for arc (i,j) in Figure 3.1. This extra line segment plays the role of an indicator variable because choosing non-zero values for the continuous λ-variables in this segment is similar to choosing a zero flow with zero cost. Using this method, GCCS needs less binary variables since their number depends on the logarithm of the number of line segments (i.e., the number of pipes).

To formulate GCCS, we define variable λ_{dp}^{ij} as the continuous weight of endpoint p for line segment d on arc (i,j). We also introduce binary variables y_{ijk} to represent each digit of the binary representation of the cost function for arc (i,j). The formulation for GCCS using the logarithmic reformulation consists of minimizing the objective function in (3.28) subject to Constraints (3.5)–(3.7), (3.10), (3.11), (3.13), (3.14), and (3.29)–(3.34).

$$\min \quad \sum_{i \in S}(f_i^s s_i + v_i^s a_i) + \sum_{j \in R}(f_j^r r_j + v_j^r b_j) + \sum_{(i,j) \in A}\sum_{d \in D}\sum_{p \in P(d)} \lambda_{dp}^{i,j}(v_{ijd}p_{ij} + f_{ijd}) \tag{3.28}$$

$$\text{s.t.} \quad \sum_{j:(i,j) \in A}\sum_{d \in D}\sum_{p \in P(d)} \lambda_{dp}^{ij} p_{ij} - \sum_{j:(j,i) \in A}\sum_{d \in D}\sum_{p \in P(d)} \lambda_{dp}^{ij} p_{ij} - a_i + b_i = 0, \quad \forall i \in N \tag{3.29}$$

$$\sum_{d \in D}\sum_{p \in P(d)} \lambda_{dp}^{ij} = 1, \quad \forall (i,j) \in A \tag{3.30}$$

$$\sum_{d \in D_k^1}\sum_{p \in P(d)} \lambda_{dp}^{ij} \leq y_{ijk}, \quad \forall k \in \{1, \ldots, log|D|\}, (i,j) \in A \tag{3.31}$$

$$\sum_{d \in D_k^0}\sum_{p \in P(d)} \lambda_{dp}^{ij} \leq 1 - y_{ijk}, \quad \forall k \in \{1, \ldots, log|D|\}, (i,j) \in A \tag{3.32}$$

$$\lambda_{dp}^{ij} \geq 0, \quad \forall d \in D, v \in V(d) \tag{3.33}$$

$$y_{ijk} \in \{0,1\}, \quad \forall (i,j) \in A, k \in \{1, \ldots, log|D|\} \tag{3.34}$$

Constraints (3.29) are the flow balance constraints considering that the flow is now the convex combination of endpoints within each line segments. Constraints (3.30)-(3.32) are the reformulation of Constraints (3.24)-(3.26) using the logarithmic DCC formulation. The nature of continuous and binary variables is enforced in Constraints (3.33) and (3.34).

3.3 Cost Function Underestimation

In this section, we propose an approach to construct an underestimator for a piecewise linear cost function that further reduces the number of binary variables needed. This underestimator serves as a lower bound on the total cost and will be embedded into our proposed solution algorithm. We assume that the cost function is nondecreasing, meaning that the total cost (fixed and variable) always increases if more flow is pushed through the arc. We assume no other particular property of the cost function, which implies that convex, concave, or neither functions are allowed. Solving a fixed-charge network design problem with such general cost function (nonconvex in general) is difficult because it requires several binary variables to describe the cost (e.g., using (3.1)–(3.14)). Figure 3.2a illustrates a special type of nondecreasing cost function. The convexity of this function allows the total cost to be modeled using only one binary variable, which also plays the role of indicating whether the arc is chosen to have positive flow. However, this is not the case for general functions, which means that producing underestimators requires some extra mathematical development. In particular, we will discuss the formulation of convex underestimator functions for the MC and logarithmic formulations. The goal of such analysis is to find cost approximations requiring less binary variables than the formulations describing the original cost structure.

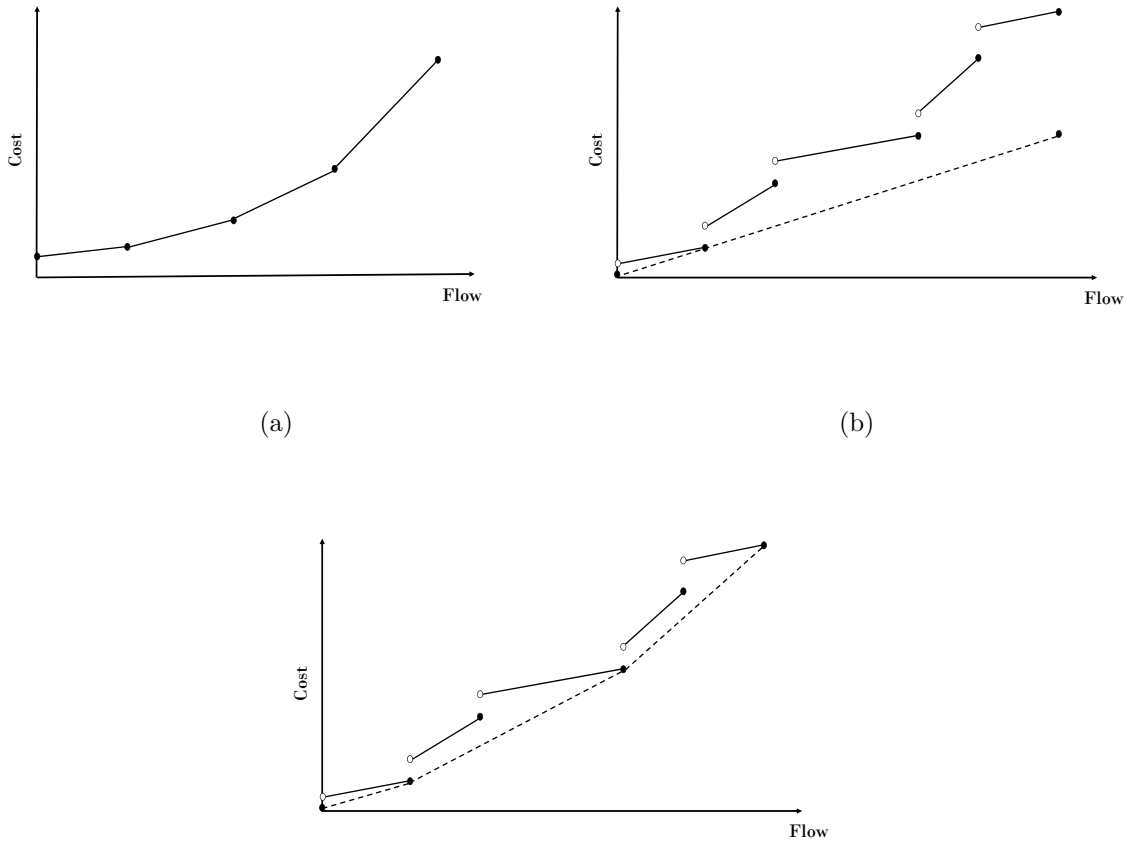


Figure 3.2: Convex Underestimations to the Cost Function

Our approach uses piecewise linear continuous functions to create convex underestimators to any nondecreasing piecewise linear function. Such underestimators are useful because they are tractable and provide a lower bound on the total cost. For example, the function shown as a dashed line in Figure 3.2b is a convex underestimator to the piecewise linear function shown in solid lines. In this case, this underestimator consists of a single line segment, which allows us to approximate the cost function using only one binary variable. However, this cost underestimation considerably un-

derestimates the total cost in some parts of the flow domain due to the few number of line segments. Now, consider the underestimation shown in Figure 3.2c, which consists of three line segments. This cost function provides a tighter and more accurate underestimation to the function in solid lines. Similarly, this convex underestimator only requires one binary variable to approximate the cost.

3.4 Generalized Formulation

Using the results from Section 3.3, we introduce a strategy to approximate non-decreasing piecewise linear cost functions. To motivate this strategy, consider the function shown in Figure 3.3, which consists of five line segments. In our construction, we use the term region to denote a set of consecutive intervals in the function's domain. Let R_{ij} be the set of all regions and C_{ij} be the index set of regions in which the cost function for arc $(i, j) \in A$ is approximated by convex underestimator functions. Similarly, let $\bar{C}_{ij}$ be the set of indices of regions containing only one segment (and with an exact cost representation). The domain of the function in Figure 3.3 contains three regions (R_1, R_2, and R_3) out of which two, $R_1 = \{\tilde{d}_1, \tilde{d}_2\}$ and $R_3 = \{\tilde{d}_4\}$, are the support of a piecewise linear convex underestimator. The elements of R_{ij} whose indices are not in C_{ij} contain single line segments that represent the cost function exactly (e.g., $\tilde{d}_3$). In this example, $C_{ij} = \{1, 3\}$ and $\bar{C}_{ij} = \{2\}$.

3.4.1 Multiple Choice Formulation

We modify the MC formulation to include the case in which the piecewise linear cost functions describing the arc costs include convex underestimator regions. If a line segment does not belong to any region in C (i.e., it is not approximated by a convex underestimator), then the formulation includes a binary variable for this segment and the cost in this segment is modeled as in the third term of (3.15).

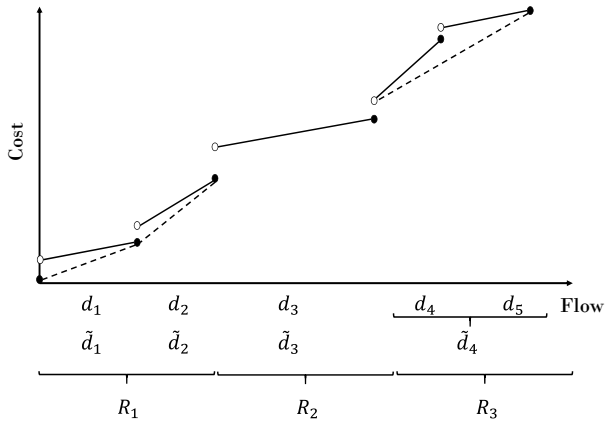


Figure 3.3: Line Segments and Convex Underestimators in the Cost Function

Otherwise, we use the endpoints in the convex region containing such segment in the formulation. To model the objective function, we introduce the continuous variable e_{ij}^k that represents the total cost of any flow in region $k \in C_{ij}$ of arc (i,j). This continuous variable is equal to the highest cost among the line segments in the convex region for any given flow value. Because of convexity and the minimization nature of the objective function, this value is the same as the cost of the given flow using the convex underestimator function. Using these elements, the GCCS problem under the MC model consists of Constraints (3.5)–(3.8), (3.10)–(3.14), (3.16)–(3.19), (3.36), and (3.37), and objective function (3.35).

$$\min \quad \sum_{i\in S}(f_i^s s_i + v_i^s a_i) + \sum_{j\in R}(f_j^r r_j + v_j^r b_j) + \sum_{(i,j)\in A}\sum_{k\in\bar{C}_{ij}}\sum_{\tilde{d}\in R_{ijk}}(f_{ij\tilde{d}}y_{ijk} + v_{ij\tilde{d}}x_{ijk}) + \sum_{(i,j)\in A}\sum_{k\in C_{ij}} e_{ij}^k \tag{3.35}$$

$$\text{s.t.} \qquad e_{ij}^k \geq f_{ij\tilde{d}}y_{ijk} + v_{ij\tilde{d}}x_{ijk}, \quad \forall (i,j)\in A, k\in C_{ij}, \tilde{d}\in R_{ijk} \tag{3.36}$$

$$e_{ij}^k \geq 0, \quad \forall (i,j)\in A, k\in C_{ij} \tag{3.37}$$

The term in the objective function (3.35) related to the construction and operation of pipes is now decomposed into two terms. The first term is the same as in (3.15) for each pipe not in C. The second term contains the summation of the e_{ij}^k variables to de-

fine the cost value of the regions in C across all arcs. Constraints (3.36) together with the objective function linearize the expressions $e^k_{ij} = \max_{\tilde{d} \in R_{ijk}} \{f_{ij\tilde{d}} y_{ijk} + v_{ij\tilde{d}} x_{ijk}\}$, for each $(i,j) \in A$ and $k \in C_{ij}$. In other words, they define the cost function of the regions in C given by the maximum cost among line segments in region R_{ijk} for any given flow level. Note that the x-variables are still defined for each region, but the corresponding cost is calculated by the appropriate line segment within the region using (3.36). Constraints (3.37) are the non-negativity constraints for the e-variables.

3.4.2 Logarithmic Formulation

We reformulate the logarithmic formulation from Section 3.2.3 using the same ideas from Section 3.4.1 and utilizing a continuous variable e^k_{ij} to define the cost in the region R_{ijk} of arc (i,j). The GCCS for logarithmic DCC consists of Constraints (3.5)–(3.7), (3.10), (3.11), (3.13), (3.14), (3.29)–(3.34), (3.39), (3.40), and objective function (3.38).

$$\min \quad \sum_{i \in S} (f^s_i s_i + v^s_i a_i) + \sum_{j \in R} (f^r_j r_j + v^r_j b_j) + \sum_{(i,j) \in A} \sum_{k \in \bar{C}_{ij}} \sum_{\tilde{d} \in R_{ijk}} \sum_{p \in P(k)} \lambda^{i,j}_{k,p} (v_{ij\tilde{d}} p_{ij} + f_{ij\tilde{d}}) + \sum_{(i,j) \in A} \sum_{k \in C_{ij}} e^k_{ij} \tag{3.38}$$

$$\text{s.t.} \quad e^k_{ij} \geq \sum_{p \in P(k)} \lambda^{i,j}_{k,p} (v_{ij\tilde{d}} p + f_{ij\tilde{d}}), \quad \forall (i,j) \in A, k \in C_{ij}, \tilde{d} \in R_{ijk} \tag{3.39}$$

$$e^k_{ij} \geq 0, \quad \forall (i,j) \in A, k \in C_{ij} \tag{3.40}$$

Constraints (3.39) enforce the definition of the cost function value of the regions in C. The non-negativity nature of e-variables is enforced by (3.40). In this case, both λ- and y-variables are defined for each region in C and the cost of the appropriate line segment is calculated using (3.39).

3.5 Progressive Cost Approximation

In this section, we propose an algorithm to solve the GCCS problem, by decomposing it into smaller subproblems that are iteratively solved until an optimal solution for the whole problem is achieved. For this purpose, we define two types of subproblems. The first subproblem is a relaxation of the original problem, hence providing lower bound values to the optimal objective function. By tightening these relaxations, we ensure that the lower bound objective value is nondecreasing through the algorithm's iterations. Similarly, we define a restricted version of the original problem that provides upper bound values to the optimal value of the original problem.

To describe our algorithm, consider an instance of the GCCS problem with arc set given by A. Set D is defined as the union of sets D_{ij}, $\forall (i,j) \in A$, so it contains all the line segments in any arc cost function. Similarly, sets R, C, and $\bar{C}$ are the union of sets R_{ij}, C_{ij}, and $\bar{C}_{ij}$ respectively. We refer to instance of GCCS as $GCCS(R, C, \bar{C})$. The optimal value to such instance is denoted by $z(R, C, \bar{C})$.

3.5.1 Lower Bound Problem

We define the problem $GCCS(\check{R}, \check{C}, \check{\bar{C}})$ as a lower bound subproblem with a cost function that consists of $\check{R}_{ij}$ regions for any given arc (i,j). Set $\check{C}_{ij}$ includes indices of regions in which the cost function for arc $(i,j) \in A$ is approximated by convex underestimator functions. Similarly, set $\check{\bar{C}}_{ij}$ includes indices of regions containing only one segment (and with an exact cost representation).

Proposition 8 *Consider $GCCS(R, C, \bar{C})$ as a GCCS instance with piecewise linear cost functions (original problem) such that $R_{ij} = \{d_1, ..., d_{|D_{ij}|}\}$, $C_{ij} = \emptyset$, and $\bar{C}_{ij} = \{1, ..., |D_{ij}|\}$ for arc $(i,j) \in A$. Let $GCCS(\check{R}, \check{C}, \check{\bar{C}})$ be an underestimated problem, then $z(R, C, \bar{C}) \geq z(\check{R}, \check{C}, \check{\bar{C}})$.*

Proof. First, note that for any given arc (i, j) flow domain in the line segment set $\check{R}_{ij}$ is the same as in the line segment set R_{ij}. Consequently, any feasible flow for $GCCS(\check{R}, \check{C}, \check{\bar{C}})$ is also feasible for $GCCS(R, C, \bar{C})$ and vice versa. Second, $\check{R}_{ij}$ consists of line segments that are exact (regions with indices in $\check{\bar{C}}_{ij}$) and underestimators of R_{ij} (regions with indices in $\check{C}_{ij}$). Hence, for any flow in the underestimated problem the cost of construction and transportation is less than or equal to the cost of that flow in the original problem. We conclude that $z(R, C, \bar{C}) \geq z(\check{R}, \check{C}, \check{\bar{C}})$. ■

3.5.2 Upper Bound Problem

Similar to Section 3.5.1, we define the problem $GCCS(\hat{R}, \hat{C}, \hat{\bar{C}})$ as an upper bound subproblem. That is, $\hat{R}_{ij} \subseteq \{d_1, ..., d_{|D_{ij}|}\}$, $\hat{C}_{ij} = \emptyset$, and $\hat{\bar{C}}_{ij} \subseteq \{1, ..., |D_{ij}|\}$ for arc (i, j).

Proposition 9 *Consider $GCCS(R, C, \bar{C})$ as a GCCS instance with piecewise linear cost functions (original problem) such that $R_{ij} = \{d_1, ..., d_{|D_{ij}|}\}$, $C_{ij} = \emptyset$, and $\bar{C}_{ij} = \{1, ..., |D_{ij}|\}$ for arc $(i, j) \in A$. Let $GCCS(\hat{R}, \hat{C}, \hat{\bar{C}})$ be a restricted problem with $\hat{R}_{ij} \subseteq \{d_1, ..., d_{|D_{ij}|}\}$, $\hat{C}_{ij} = \emptyset$, and $\hat{\bar{C}}_{ij} \subseteq \{1, ..., |D_{ij}|\}$ for arc (i, j), then $z(\hat{R}, \hat{C}, \hat{\bar{C}}) \geq z(R, C, \bar{C})$.*

Proof. First, note that for any arc (i, j), the domain of flow in the line segment set $\hat{R}_{ij}$ is a subset of the domain in the exact line segment set R_{ij}. Consequently, any feasible flow for $GCCS(\hat{R}, \hat{C}, \hat{\bar{C}})$ is also feasible for $GCCS(R, C, \bar{C})$. Second, for any arc (i, j), the line segment set $\hat{R}_{ij}$ is a subset of exact line segments R_{ij}. As a result, for any feasible flow for in the restricted problem the cost of construction and transportation is equal to the cost of that flow in the original problem. We conclude that $z(\hat{R}, \hat{C}, \hat{\bar{C}}) \geq z(R, C, \bar{C})$. ■

3.5.3 Solution Algorithm

The size of the introduced formulations to solve the GCCS problem makes them not competitive for large-scale instances using commercial solvers. The MC formulation is not computationally efficient for larger class of problems because the size of the problem quickly grows with the number of arcs $|A|$ and the number of pipes $|D|$. On the contrary, using the logarithmic formulation for GCCS, reduces the number of binary variables to model the pipe selection decision from $|A||D|$ to $|A|log|D|$. Which still may be quite substantial if the number of pipes and arcs is large.

In this Section, we provide a progressive cost approximation algorithm that can further enhance the computational advantages provided by the logarithmic formulation, allowing us to solve large-scale instances of the problem by using only the relevant line segments in the cost function. We introduce two subroutines that help initialize the cost underestimators and their update along the iterations of the progressive cost approximation algorithm. For a given arc (i, j), Algorithm 3 generates a convex cost underestimation for a subset of the function domain. The input of Algorithm 3 is an instance of $GCCS(R, C, \bar{C})$ along with a set of regions in the function domain R_{ijk} for a given arc (i, j) (Line 1). In Algorithm 3, we iteratively generate the tightest possible convex underestimator for the given region. In Line 3, we start with the leftmost point of the function domain and generate a first underestimator consisting of a line with the highest possible slope. In Lines 4 and 5, Algorithm 3 adds such line segment to the convex region and repeats the process with the rightmost point in the support of the current line. We continue this process until the whole domain is covered by a convex underestimator.

Algorithm 4 updates the cost underestimations in $GCCS(\check{R}, \check{C}, \check{\bar{C}})$ given a solution $\bar{\mathbf{x}} = (x_{ij})_{(i,j)\in A}$. In Line 3, we find the exact cost segment d_b that contains the flow in

Algorithm 3 : Cost Underestimation

1: Consider $R_{ijk} = \{ d_m, ..., d_n\}$ given for arc (i, j)

2: Set $\bar{R} = \emptyset$

3: Start with leftmost point in the flow domain of line segments in region R_{ijk} and set this point as $x_{current}$

4: At point $x_{current}$, find the underestimator line segment with highest slope and denote the intersection of this line d_{new} with line segments in region R_{ijk} as updated $x_{current}$

5: Add d_{new} to $\bar{R}$

6: **if** $x_{current} \neq$ rightmost point in the flow domain of line segments in region R_{ijk} **then**

7: Go to Step 4

8: Return $R_{ijk} \leftarrow \bar{R}$

its domain for all the arcs with positive flow. In Line 4, we redefine R_{ijk} and in Lines 5 and 6 we run the underestimation algorithm for line segments in R_{ijk} on the left and right side of line segment d_b, when applicable this process results in an enhanced underestimation of the exact cost function.

Algorithm 4 : Cost Underestimation Update

1: Consider $\check{R}, \check{C}, \check{\bar{C}}$ and given solution $\bar{\mathbf{x}}$. For each arc (i, j), $\bar{x}_{ij}$ is the flow on that arc which belongs to convex region $R_{ijk} \in R_{ij}$

2: **for** $(i, j) \in A$ *s.t.* $\bar{x}_{ij} \neq 0$ **do**

3: Find exact line segments $\{d_a, ..., d_b, ..., d_c\} \subseteq D_{ij}$ that have the same flow domain as line segments in R_{ijk}. Let d_b be the line segment that contains flow $\bar{x}_{ij}$ in its domain

4: Redefine $R_{ijk} \leftarrow \{d_b\}$

5: For nonempty $\{d_a, ..., d_{b-1}\}$ perform underestimation through Algorithm 3 to obtain R_l

6: For nonempty $\{d_{b+1}, ..., d_c\}$ perform underestimation through Algorithm 3 to obtain R_r

7: Add R_l and R_r to R_{ij} and update indices sets C_{ij} and $\bar{C}_{ij}$ accordingly

Algorithm 5 describes the proposed progressive cost approximation algorithm to solve GCCS. In Lines 1 and 2 of Algorithm 5, we initialize $\check{R}$ by using the convex underestimators of the exact cost function in each arc. In Line 7, we solve the lower

bound problem based on the underestimated cost function with regions set $\check{R}$. In Line 8, we update the cost function underestimators for the next lower bound subproblem. The optimal solution makes it possible to construct an upper bound subproblem's cost structure in Line 10.

Algorithm 5 : Progressive Cost Approximation Algorithm for GCCS

1: For any given arc (i,j), let $\check{R}_{ij} = \{\check{R}_{ij1}, \check{R}_{ij2}\}$, $\check{C} = \{2\}$, $\check{\check{C}} = \{1\}$, $\check{R}_{ij1} = \{d_1\}$, and $\check{R}_{ij2} = \{d_2, ..., d_{|D_{ij}|}\}$ is constructed in Step 2
2: Perform underestimation on $\check{R}_{ij2}$ for each arc (i,j) through Algorithm 3
3: Let $\hat{R} = \hat{C} = \hat{\check{C}} = \emptyset$
4: Set $UB_0 = \infty$, $LB_0 = 0$, and set counter $i = 0$
5: **while** $UB_i > LB_i$ **do**
6: Set $i = i + 1$
7: Solve $GCCS(\check{R}, \check{C}, \check{\check{C}})$ to obtain an optimal solution $\check{\mathbf{x}}^i$, optimal value $\check{z}^i$. Set $LB_i = \check{z}^i$
8: Update $\check{R}, \check{C}, \check{\check{C}}$ with solution $\check{\mathbf{x}}^i$ through Algorithm 4
9: For each arc (i,j), find the line segment d_k in D_{ij} that contains the flow given by $\check{\mathbf{x}}^i$
10: Add $\{d_k\}$ to $\hat{R}_{ij}$ and update $\hat{\check{C}} \leftarrow \hat{\check{C}} \cup \{|\hat{R}_{ij}|\}$
11: Solve $GCCS(\hat{R}, \hat{C}, \hat{\check{C}})$ to obtain an optimal solution $\hat{\mathbf{x}}^i$ and optimal value $\hat{z}^i$
12: Set $UB_i = \hat{z}^i$ and update the incumbent solution $\bar{\mathbf{x}} \leftarrow \hat{\mathbf{x}}^i$ and objective $\bar{z} = UB_i$
13: **if** $UB_i = LB_i$ **then**
14: Incumbent $\bar{\mathbf{x}}$ is optimal with objective $\bar{z}$. Go to Step 15
15: Return $\bar{\mathbf{x}}$ and $\bar{z}$

Note that the update in the lower bound problem in Line 8 results in tighter cost underestimators, thus $LB_{i+1} \geq LB_i$. Also, since the sets of each line segment for upper bound problem iteration i are subsets of those sets in the next iteration, $i+1$, we conclude that $UB_i \geq UB_{i+1}$. Figure 3.4 illustrates the operation of Algorithm 5, emphasizing the progress of the underestimations across iterations. The blue lines are the line segments in D_{ij}, while the red lines are the current underestimator and the green lines are the updated underestimator at each iteration.

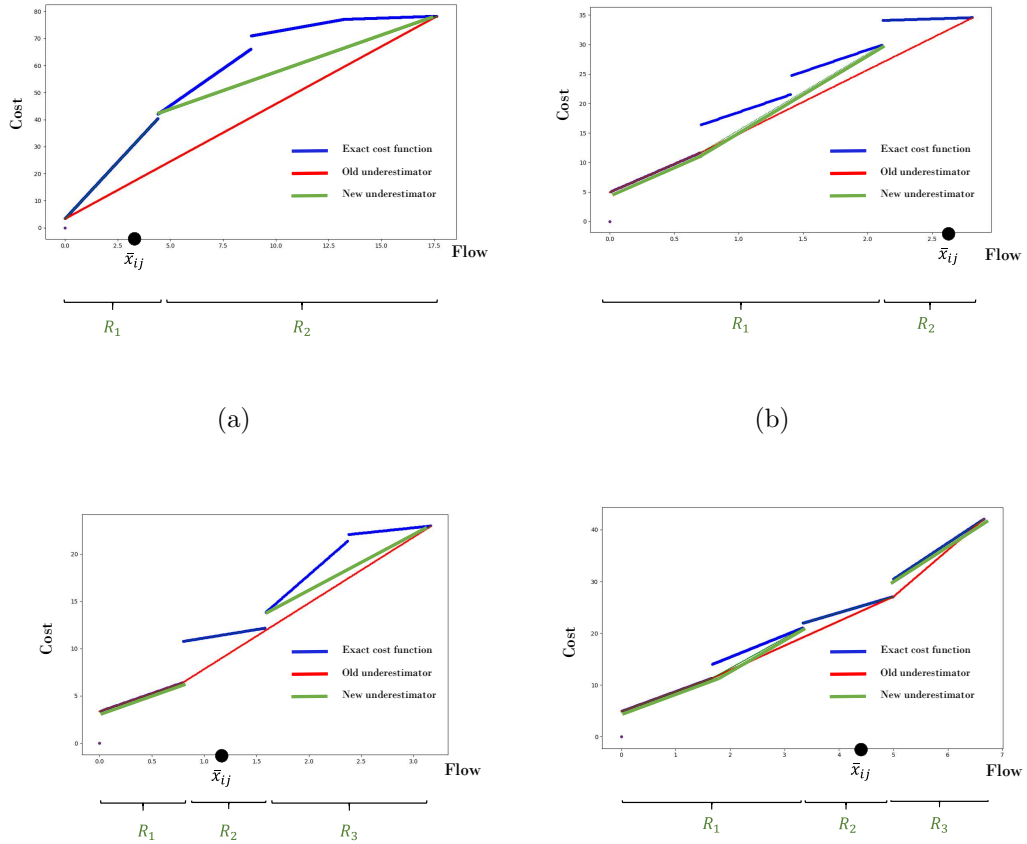


Figure 3.4: Updated Underestimations of Cost Functions

3.6 Computational Results

In this section, we illustrate and examine the performance of our proposed approaches to solve GCCS on real and randomly generated networks. To perform our computations, we use Python with CPLEX 12.7 as optimization solver on a computer with an Intel Core i7 2.40 GHz processor and 8.0 GB RAM. We set a solution time limit of 1 hour in all the experiments.

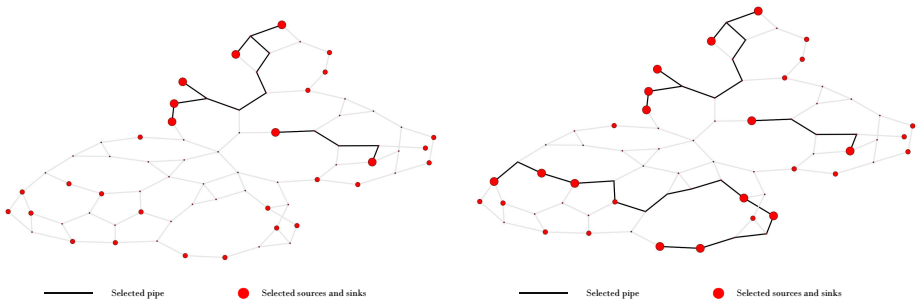
3.6.1 Illustrative Example

We use a pipeline candidate network in the region of Alberta (Canada), consisting of 72 nodes, 200 arcs, 22 sources, and 16 reservoir candidates (Middleton and Brandt, 2013). To generate random cost structures for each arc, we discretize the domain (flow range) of the cost function for each arc into 10 disjoint segments that are evenly distributed. We set the fixed cost of the first segment in the range $[1, 10]$. The variable cost of each cost segment is chosen using a uniform distribution in the range $[0, 1]$ and the step size between line segments is chosen in range $[0, 0.5]$. These parameters fully determine the structure of the cost function. Also, the length of each arc scales the fixed cost and variable cost parameters.

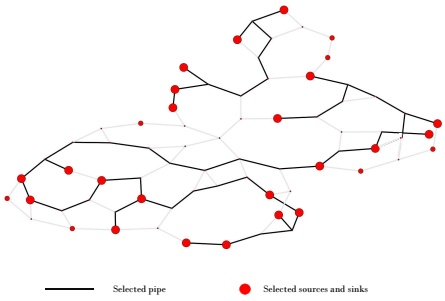
Figure 3.5 illustrates the optimal solution of the GCCS problem for Alberta's pipeline network with randomly generated pipe cost functions. In this figure, sources and sinks that are not utilized in the optimal solution are shown with small circles. Large circles represent sources and sinks that are chosen in the optimal solution. We define $\bar{\tau}$ as the optimal value of the maximum flow problem solved for a network of given capacities for each source, sink, and arc. As expected, increasing amount of target flow makes the optimal solution less sparse.

3.6.2 Random Network Instances

Section 3.6.1 focuses on illustrating solutions of GCCS for some instances out of the same candidate network. In this section, we study the performance of our approach on randomly generated candidate networks. Section 3.6.3 describes a process to generate such instances and Section 3.6.4 reports the results and performance analysis.



(a) $\tau = 0.3 \times \bar{\tau}$ (b) $\tau = 0.6 \times \bar{\tau}$



(c) $\tau = 0.9 \times \bar{\tau}$

Figure 3.5: Effect of Target Flow on the Solution in Alberta Pipeline Network

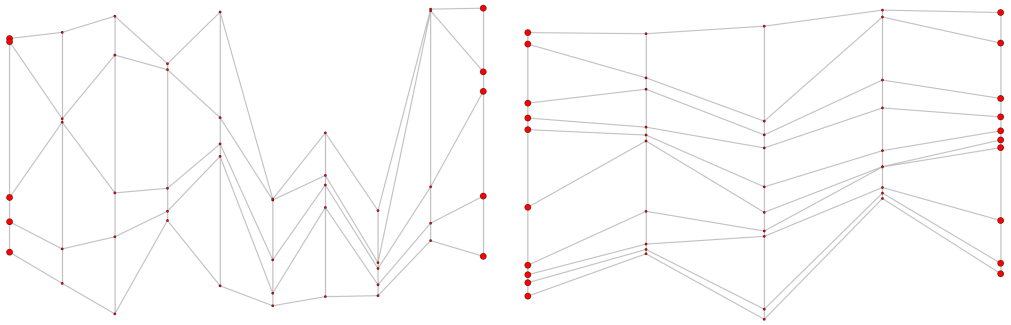
3.6.3 Random Instance Generation

We create layered networks to control the number of sources, sinks, intermediate nodes and arcs. These layered networks also provide many alternative flow routes, testing the limits of our progressive cost approximation approach as many iterations may be needed to enhance underestimators of the cost functions. We generate layered networks with n layers and m nodes per layer, resulting in $|N| = m \times n$ nodes. All m nodes in the first layer are source nodes and all m nodes in the last layer are sink nodes, remaining nodes of the network are considered as intermediate (or transshipment) nodes. The capacities, fixed costs, and variable costs of each source and sink node

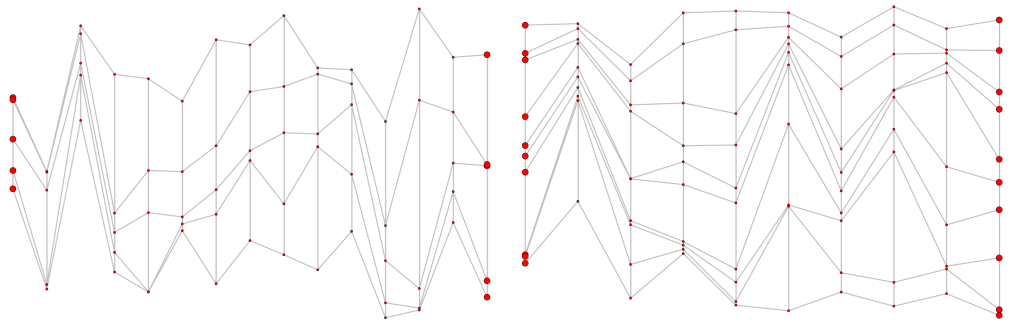
are generated using a uniform distribution in the range $[0, 10]$. We generate random peicewise linear cost functions for each arc by discretizing the domain of the flow for each arc into 10 disjoint segments that are evenly distributed. We pick the fixed cost of the first segment in the range $[1, 10]$. The variable cost of each line segment is chosen in the range $[0, 1]$ and the step size between line segments is chosen in the range $[0, 0.5]$. These two random parameters determine the fixed costs of other segments in the cost function. Also, length of each arc scales the fixed cost and variable cost parameters.

Nodes are located in a square of dimension 100×100 in such a way that layers are evenly distributed. The position of each node within a layer is chosen randomly. The i-th node (from top to bottom) of the j-th layer (from left to right) is connected to the i-th node of layer $j + 1$ and to nodes $i - 1$ and $i + 1$ in the same layer, in case that such nodes exist. We generate networks with 5×10, 10×5, 5×15, 10×10, and 10×15 nodes. Figure 3.6 illustrates five random layered networks of different sizes. The value of $|D|$ represents the number of segments in the cost function of each arc of the network. For the experiments, we set $|D| \in \{30, 60, 120\}$.

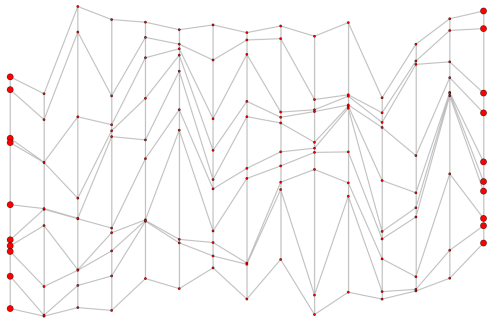
To generate the target flow to capture, we define $\bar{\tau}$ as the optimal value of the maximum flow problem solved for a network of given capacities for each source, sink, and arc. Using $\bar{\tau}$, we set the value of the target flow $\tau \in \{0.3 \times \bar{\tau}, 0.6 \times \bar{\tau}, 0.9 \times \bar{\tau}\}$. Figure 3.7 illustrates optimal configurations of GCCS problems for an instance of size $(|N|, |A|) = (10 \times 10, 270)$ and various target flow values. As expected, increasing the amount of target flow necessitates the construction of more arcs and utilizing more sources and sinks in the network. Because the sparsity of the optimal solution decreases with higher τ values, we expect such instances to be more challenging for the Progressive Cost Approximation Algorithm (Algorithm 5). For each combination of $|N|$, $|D|$, and τ, we create 5 random replications. These replications help averaging



(a) $(|N|, |A|) = (5 \times 10, 125)$ (b) $(|N|, |A|) = (10 \times 5, 130)$



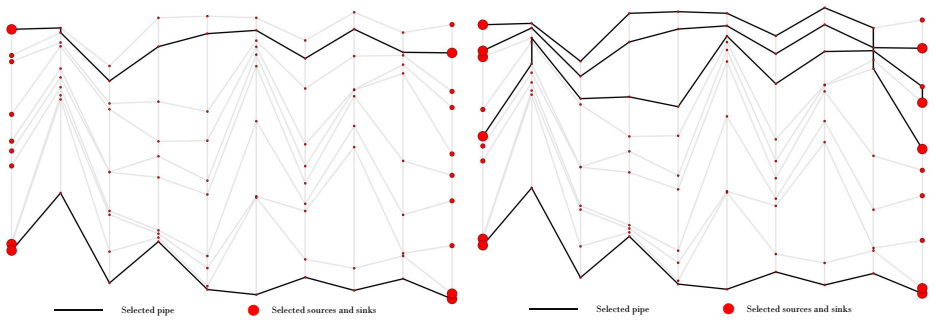
(c) $(|N|, |A|) = (5 \times 15, 190)$ (d) $(|N|, |A|) = (10 \times 10, 270)$



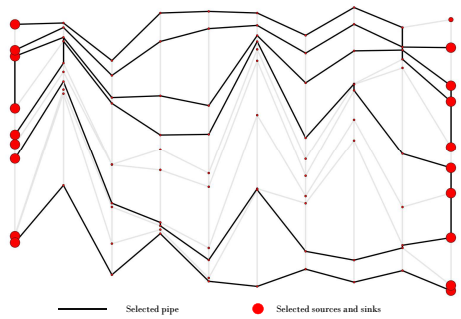
(e) $(|N|, |A|) = (10 \times 15, 410)$

Figure 3.6: Random Networks with Different Layer Size and Number of Layers

the performance metrics and provide more confidence for the computational analysis.



(a) $\tau = 0.3 \times \bar{\tau}$ (b) $\tau = 0.6 \times \bar{\tau}$



(c) $\tau = 0.9 \times \bar{\tau}$

Figure 3.7: Effect of Target Flow on the Solution in Random Layered Networks

3.6.4 Results

We provide computational results and performance metric values in three tables. Table 3.1 shows results for random networks with $|N|$ equal to 5×10 and 10×5. Table 3.2 shows results of experiments on networks with sizes 5×15 and 10×10. Similarly, Table 3.3 provides performance metric values for experiments on networks with 10×15 nodes. The first three columns in each table characterize the instance, including $|N|$, $|A|$, $|D|$, and τ.

We test the performance of four methods on each random instance. We use MC to denote the Multiple Choice MIP formulation and LOG refer to the Logarithmic MIP

model. We have two choices to solve each lower bound and upper bound subproblem using the Progressive Cost Approximation Algorithm. The first choice is to utilize MC method to solve each subproblem, which we refer to as the iterative MC method or MC-P. The second choice is to solve subproblems using the LOG method, which we refer to as LOG-P. The fourth column on each table indicates the method that we have used to solve each replication. The column '*time*' refers to the average solution time (in seconds) for those instances solved within the time limit, whose number is shown in parentheses. The column '*gap*' represents the average optimality gap for instances that time out. In methods MC-P and LOG-P, the gap is calculated as $100(UB_i - LB_i)/UB_i$, where i is the last iteration before timing out. Note that our algorithmic approach is able to solve larger instances to optimality and, in case of timing out, it provides smaller gaps compared with MC and LOG methods without the progressive approximation. Column $|y|$ reports number of binary variables in MC and LOG model and in the last lower bound problem in MC-P and LOG-P methods. When comparing LOG and MC, we observe that LOG results in problems with smaller number of variables, leading to faster solution times and smaller optimality gaps across instances. The number of iterations for each method is reported in column 'i'.

Table 3.1: Computational Performance of the Proposed Approaches on 5×10 and 10×5 Networks

$(\|N\|, \|A\|)$	$\|D\|$	τ		$time$	gap	$\|y\|$	i	$(\|N\|, \|A\|)$	$\|D\|$	τ		$time$	gap	$\|y\|$	i
$(5 \times 10, 125)$	30	0.3	LOG-P	3 (5)	-	149	3	$(10 \times 5, 130)$	30	0.3	LOG-P	59 (5)	-	180	8
			MC-P	2 (5)	-	164	3				MC-P	13 (5)	-	228	8
			LOG	37 (5)	-	635	-				LOG	680 (5)	-	670	-
			MC	21 (5)	-	3760	-				MC	40 (5)	-	3920	-
		0.6	LOG-P	2 (5)	-	149	3			0.6	LOG-P	245 (5)	-	190	7
			MC-P	2 (5)	-	153	3				MC-P	32 (5)	-	244	7
			LOG	11 (5)	-	635	-				LOG	1317 (5)	-	670	-
			MC	6 (5)	-	3760	-				MC	605 (5)	-	3920	-
		0.9	LOG-P	61 (5)	-	178	5			0.9	LOG-P	994 (4)	0.10	201	7
			MC-P	34 (5)	-	215	5				MC-P	109 (5)	-	266	7
			LOG	2031 (3)	1.59	635	-				LOG	1523 (3)	1.00	670	-
			MC	1781 (5)	-	3760	-				MC	1857 (3)	0.60	3920	-
	60	0.3	LOG-P	3 (5)	-	151	4		60	0.3	LOG-P	181 (5)	-	193	10
			MC-P	3 (5)	-	167	4				MC-P	100 (5)	-	272	10
			LOG	56 (5)	-	760	-				LOG	1129 (5)	-	800	-
			MC	75 (5)	-	7510	-				MC	378 (5)	-	7820	-
		0.6	LOG-P	2 (5)	-	150	3			0.6	LOG-P	229 (5)	-	194	8
			MC-P	1 (5)	-	164	3				MC-P	151 (5)	-	258	8
			LOG	13 (5)	-	760	-				LOG	504 (5)	-	800	-
			MC	13 (5)	-	7510	-				MC	420 (2)	1.45	7820	-
		0.9	LOG-P	71 (5)	-	185	5			0.9	LOG-P	1614 (3)	0.13	218	8
			MC-P	61 (5)	-	223	5				MC-P	248 (3)	0.15	306	7
			LOG	1485 (4)	1.78	760	-				LOG	1922 (2)	0.47	800	-
			MC	297 (2)	1.36	7510	-				MC	-	5.85	7820	-
	120	0.3	LOG-P	4 (5)	-	151	4		120	0.3	LOG-P	17 (5)	-	175	7
			MC-P	4 (5)	-	168	4				MC-P	7 (5)	-	222	7
			LOG	60 (5)	-	885	-				LOG	625 (5)	-	930	-
			MC	778 (5)	-	15010	-				MC	794 (3)	3.66	15620	-
		0.6	LOG-P	2 (5)	-	149	3			0.6	LOG-P	897 (5)	-	210	17
			MC-P	1 (5)	-	163	3				MC-P	732 (5)	-	363	17
			LOG	44 (5)	-	885	-				LOG	999 (3)	0.88	930	-
			MC	142 (4)	0.12	15010	-				MC	-	7.59	15620	-
		0.9	LOG-P	204 (5)	-	191	8			0.9	LOG-P	454 (4)	0.00	212	7
			MC-P	192 (5)	-	254	8				MC-P	255 (3)	0.04	297	7
			LOG	2350 (1)	0.35	885	-				LOG	569 (1)	0.36	930	-
			MC	-	3.14	15010	-				MC	-	4.26	15620	-

Table 3.2: Computational Performance of the Proposed Approaches on 5×15 and 10×10 Networks

$(\|N\|, \|A\|)$	$\|D\|$	τ		$time$	gap	$\|y\|$	i	$(\|N\|, \|A\|)$	$\|D\|$	τ		$time$	gap	$\|y\|$	i
$(5 \times 15, 190)$	30	0.3	LOG-P	22 (5)	-	229	4	$(10 \times 10, 270)$	30	0.3	LOG-P	111 (5)	-	328	4
			MC-P	11 (5)	-	259	4				MC-P	29 (5)	-	364	4
			LOG	439 (5)	-	960	-				LOG	70 (5)	-	1370	-
			MC	92 (5)	-	5710	-				MC	254 (3)	1.39	8120	-
		0.6	LOG-P	1407 (1)	0.15	266	4			0.6	LOG-P	3002 (1)	0.32	354	4
			MC-P	592 (4)	-	386	6				MC-P	1145 (4)	0.10	445	6
			LOG	-	1.65	960	-				LOG	-	1.55	1370	-
			MC	-	1.56	5710	-				MC	-	2.38	8120	-
		0.9	LOG-P	1949 (2)	-	295	6			0.9	LOG-P	-	0.57	356	3
			MC-P	379 (5)	-	379	7				MC-P	1863 (2)	0.16	485	6
			LOG	2331 (1)	1.59	960	-				LOG	-	2.75	1370	-
			MC	3403 (1)	-	5710	-				MC	-	3.27	8120	-
	60	0.3	LOG-P	40 (5)	-	241	5		60	0.3	LOG-P	312 (5)	-	340	7
			MC-P	45 (5)	-	282	4				MC-P	482 (5)	-	411	7
			LOG	474 (5)	-	1150	-				LOG	1309 (5)	-	1640	-
			MC	719 (5)	-	11410	-				MC	-	1.51	16220	-
		0.6	LOG-P	1089 (1)	0.66	274	5			0.6	LOG-P	-	0.25	372	5
			MC-P	121 (1)	0.38	424	7				MC-P	389 (1)	0.28	475	6
			LOG	-	10.03	1150	-				LOG	-	1.09	1640	-
			MC	-	6.94	11410	-				MC	-	5.37	16220	-
		0.9	LOG-P	3253 (5)	-	297	7			0.9	LOG-P	-	0.46	379	3
			MC-P	-	0.06	448	6				MC-P	-	0.20	488	4
			LOG	-	0.25	1150	-				LOG	-	2.83	1640	-
			MC	-	1.98	11410	-				MC	-	4.03	16220	-
	120	0.3	LOG-P	21 (5)	-	233	5		120	0.3	LOG-P	71 (5)	-	334	6
			MC-P	23 (5)	-	266	4				MC-P	80 (5)	-	390	6
			LOG	477 (5)	-	1340	-				LOG	960 (2)	0.22	1910	-
			MC	2319 (3)	18.62	22810	-				MC	-	1.65	32420	-
		0.6	LOG-P	-	0.15	286	6			0.6	LOG-P	632 (2)	0.05	373	7
			MC-P	137 (1)	0.08	462	8				MC-P	539 (2)	0.06	407	7
			LOG	-	6.91	1340	-				LOG	-	0.82	1910	-
			MC	-	6.50	22810	-				MC	-	2.35	32420	-
		0.9	LOG-P	346 (5)	-	270	7			0.9	LOG-P	-	0.35	397	4
			MC-P	403 (5)	-	342	7				MC-P	-	0.20	541	4
			LOG	-	0.80	1340	-				LOG	-	3.90	1910	-
			MC	-	6.18	22810	-				MC	-	-	32420	-

Table 3.3: Computational Performance of the Proposed Approaches on 10×15 Networks

$(\|N\|, \|A\|)$	$\|D\|$	τ		$time$	gap	$\|y\|$	i
$(10 \times 15, 410)$	30	0.3	LOG-P	-	1.81	497	3
			MC-P	1483 (1)	0.81	616	4
			LOG	-	11.23	2070	-
			MC	-	25.13	12320	-
		0.6	LOG-P	-	2.27	499	2
			MC-P	-	1.29	602	3
			LOG	-	6.78	2070	-
			MC	-	6.58	12320	-
		0.9	LOG-P	-	1.34	515	3
			MC-P	-	0.85	642	3
			LOG	-	5.71	2070	-
			MC	-	40.89	12320	-
	60	0.3	LOG-P	-	0.96	500	3
			MC-P	-	0.61	615	4
			LOG	-	7.38	2480	-
			MC	-	26.16	24620	-
		0.6	LOG-P	-	1.87	518	2
			MC-P	-	1.33	613	2
			LOG	-	6.22	2480	-
			MC	-	16.13	24620	-
		0.9	LOG-P	-	1.05	525	3
			MC-P	-	0.92	626	3
			LOG	-	11.47	2480	-
			MC	-	12.49	24620	-
	120	0.3	LOG-P	1861 (1)	0.98	520	5
			MC-P	3044 (2)	0.74	639	5
			LOG	-	5.58	2890	-
			MC	-	-	49220	-
		0.6	LOG-P	-	1.22	534	3
			MC-P	-	0.92	695	4
			LOG	-	9.79	2890	-
			MC	-	-	49220	-
		0.9	LOG-P	-	0.73	578	4
			MC-P	-	0.70	718	4
			LOG	-	4.00	2890	-
			MC	-	-	49220	-

3.7 Concluding Remarks

In this chapter we investigated methodologies to design a fixed-charge network with generalized piecewise linear cost function. We formulate this problem using a mixed-integer program that is very challenging to solve. To improve running time, we proposed an alternative formulation that needs less binary variables (logarithmic in terms of number of cost function segments) leading to computational improvements. The logarithmic formulation is still computationally challenging for larger candidate networks. Hence, we introduced a decomposition algorithm that is promising and takes advantage of sparsity of the optimal solution and underestimation of the cost function. Computational experiments suggest that this technique scales well for larger instances based on both number of line segments in cost function and number of arcs in the network.

Chapter 4

PROBABILISTIC TRANSPORTATION PROBLEM WITH INDEPENDENT STOCHASTIC DEMANDS

In this chapter, we leverage on the discoveries from previous chapters to solve a class of Chance Constrained Stochastic Program (CCSP) with independent random variables. This chapter is organized as follows. In Section 4.1, we provide a literature review and introduce the problem to solve. Section 4.2 describes a method to find the empirical probability distribution of the stochastic parameters. In Section 4.3, we discuss the mathematical programming formulation for the probabilistic transportation problem with independent random demands. Because of the computational challenge of solving the resulting MIP when the number of samples is large, Section 4.6 presents an iterative refinement algorithm that is based on similar decomposition principles outlined in previous chapters. We present our computational study in Section 4.7 and some final remarks in Section 4.8.

4.1 Introduction and Literature Review

In general, CCSPs are challenging to solve because of the functional form behind the probabilistic constraints. Although convex deterministic equivalent reformulations exist for some cases (e.g., normal distributions), CCSPs generally result in nonconvex feasible regions. When the first two moments of the random variables are available, approximation methods provide a mechanism to bypass the problem complexity. However, there is no known bound about the quality of the solution in such cases (Ben-Tal *et al.*, 2009).

Sampling the random variables (and consequently the problem parameters) and

then embedding those samples into a deterministic problem is a viable alternative to approximate the subjacent probability distributions. In this case, it is possible to obtain statistical bounds based on a finite scenario sampling approach (Calafiore and Campi, 2005; Campi and Garatti, 2011). A widespread approach based on the finite scenario sampling strategy is the sample average approximation (Luedtke and Ahmed, 2008a; Pagnoncelli *et al.*, 2009). The advantage of this method is that the solution converges to the optimal value of the probabilistic problem as the number of samples increase. Moreover, it is possible to find a confidence interval of the problem's optimal value using this method (Nemirovski and Shapiro, 2006). Despite its advantages, stochastically constrained optimization problems are very challenging to solve using the sample average approximation method. Some approaches focus on probabilistic constraints where the randomness occurs in the constraints' right hand sides (Dentcheva *et al.*, 2000; Beraldi and Ruszczyński, 2002; Saxena *et al.*, 2010; Dentcheva and Martinez, 2013; Lejeune, 2012), while others focus on the case in which the constraint matrix is random (Ruszczyński, 2002; Beraldi and Bruni, 2010; Tanner and Ntaimo, 2010; Luedtke, 2010; Beraldi *et al.*, 2012).

We are interested in solving a special class of CCSP having a joint probability constraint with independent random variables in the right-hand side. This is shown in Constraint (4.1), where the random variables correspond to the $\tilde{b}$-variables and $\mathbf{a}, \mathbf{x} \in \mathbb{R}^q$ are deterministic. The value of $1-\alpha \in (0,1)$ represents the degree of conservatism in the solution. A large value of $1-\alpha$ represents a risk-averse decision maker that wants to satisfy the constraint with a high probability.

$$p\left(\bigwedge_{i=1}^{m} \mathbf{a}_i^T \mathbf{x} \leq \tilde{b}_i\right) = p(\mathbf{a}_1^T \mathbf{x} \leq \tilde{b}_1, \ldots, \mathbf{a}_m^T \mathbf{x} \leq \tilde{b}_m) \geq 1-\alpha. \tag{4.1}$$

Assuming that the $\tilde{b}$-variables are independent, Constraint (4.1) can be reformulated as Constraint (4.2). After applying the natural logarithm in both sides of the

inequality, Constraint (4.2) is equivalent to the linear constraint (4.3). The logarithm transformation preserves the inequality because $\ln(x)$ is a nondecreasing function.

$$p(\mathbf{a}_1^T\mathbf{x} \leq b_1)p(\mathbf{a}_2^T\mathbf{x} \leq b_2)\dots p(\mathbf{a}_m^T\mathbf{x} \leq b_m) \geq 1-\alpha \tag{4.2}$$

$$\ln(p(\mathbf{a}_1^T\mathbf{x} \leq b_1)) + \ln(p(\mathbf{a}_2^T\mathbf{x} \leq b_2)) + \dots + \ln(p(\mathbf{a}_m^T\mathbf{x} \leq b_m)) \geq \ln(1-\alpha) \tag{4.3}$$

Note that Constraint (4.2) implies Constraint (4.4), and therefore it provides a more conservative solution. Moreover, any vector $\mathbf{x} \in \mathbb{R}^q$ satisfying Constraint (4.2) also satisfies Constraints (4.4). If $m = 1$, then Constraint (4.1) reduces to a single probabilistic constraint of the same type of (4.4).

$$p(\mathbf{a}_i^T\mathbf{x} \leq b_i) \geq 1-\alpha, \quad \forall i \in \{1, ..., m\} \tag{4.4}$$

In order to provide a mathematical programming formulation to approximate Constraint (4.3), we first need to characterize the cumulative distribution function (CDF) of each random variable. Section 4.2 presents an overview of the existing estimation approaches of our interest.

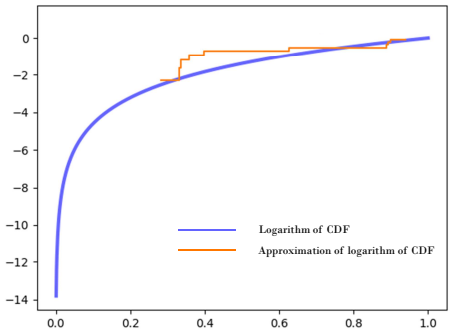
4.2 Cumulative Distribution Function Estimation

There are several approaches to estimate the CDF of a random variable. We follow a sampling based approach to construct the empirical distribution function as an estimation of the true CDF. To do so, we use a step piecewise linear function that facilitates the transformation of Constraint (4.3) into a set of linear constraints. Assuming that $\hat{b}_1, \dots, \hat{b}_n$ are n samples of the random variable of interest, then $P(\tilde{b} \leq y)$ can be approximated by

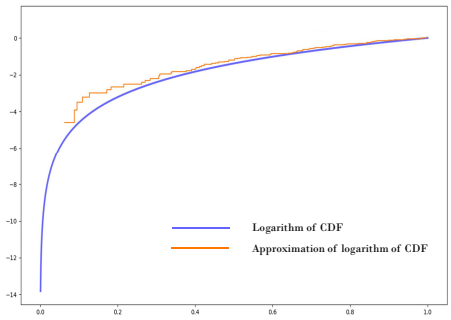
$$\hat{F}_{\tilde{b}}(y) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\hat{b}_i \leq y), \tag{4.5}$$

where $\mathbb{I}(\cdot)$ is an indicator function that takes the value of 1 if the condition $(\cdot)$ is satisfied, and is equal to 0 otherwise. For illustration, Figure 4.1 shows the logarithm

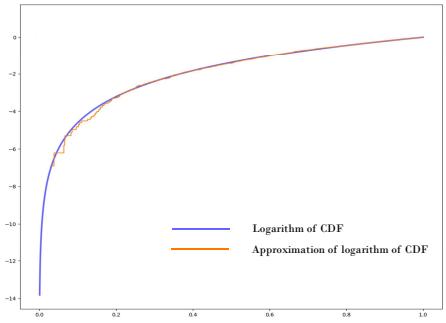
of the theoretical and approximated CDF functions, $\ln(F_{\tilde{b}}(y))$ and $\ln(\hat{F}_{\tilde{b}}(y))$, for a beta distribution with parameters $a = 5$ and $b = 1$ and sample sizes $n = 10$, $n = 100$, and $n = 1000$. As expected, the approximation better describes the theoretical CDF function as the number of samples increase.



(a) $n = 10$



(b) $n = 100$



(c) $n = 1000$

Figure 4.1: Logarithm of the CDF $(\ln(F_{\tilde{b}}(y)))$ and Sample-based Approximation $(\ln(\hat{F}_{\tilde{b}}(y)))$ of a Beta Distribution with $a = 5$ and $b = 1$

4.3 Mathematical Programming Formulation

In this section, we present a mathematical formulation for the probabilistic Transportation problem with Independent Random Demands (TIRD) from Luedtke and Ahmed (2008b). Consider a set of suppliers S and a set of customers C. In this problem, each supplier $i \in S$ has a shipping capacity Q_i and the cost of sending a unit of commodity from supplier $i \in S$ to customer $j \in C$ is c_{ij}. The random demand of customer $j \in C$ is represented by $\bar{d}_j$. The formulation of TIRD is shown in (4.6)–(4.9), where decision variables x_{ij} represent the amount of commodity transported from supplier $i \in S$ to customer $j \in C$.

$$\min \quad \sum_{i \in S} \sum_{j \in C} c_{ij} x_{ij} \tag{4.6}$$

$$\sum_{j \in C} x_{ij} \leq Q_i, \quad \forall i \in S \tag{4.7}$$

$$P\left\{ \bigwedge_{j \in C} \left(\sum_{i \in S} x_{ij} \geq \bar{d}_j \right) \right\} \geq 1 - \epsilon \tag{4.8}$$

$$x_{ij} \geq 0, \quad \forall i \in S, j \in C \tag{4.9}$$

The objective function (4.6) is to minimize the total shipping cost. Constraints (4.7) limit the amount of commodity sent from each supplier to their capacity and Constraint (4.7) enforces a joint probabilistic constraint in which the demand of all customers must the satisfied with a probability of at least $1 - \epsilon$. Note that (4.8) implies the simpler constraints $P\left(\sum_{i \in S} x_{ij} \geq \bar{d}_j\right) \geq 1 - \epsilon, \quad \forall j \in C$. Constraints (4.9) enforce the non-negative nature of the x-variables.

Assuming that the demands are independent random variables, we can rewrite Constraint (4.8) as $\prod_{j \in C} P\left\{\sum_{i \in S} x_{ij} \geq \bar{d}_j\right\} \geq 1 - \epsilon$, which can be reformulated as $\sum_{j \in C} \ln P\left\{\sum_{i \in S} x_{ij} \geq \bar{d}_j\right\} \geq \ln(1 - \epsilon)$ because the logarithm is a nondecreasing function, $0 < P\left\{\sum_{i \in S} x_{ij} \geq \bar{d}_j\right\} \leq 1, \forall j \in C$, and $0 \leq 1 - \epsilon \leq 1$. Using the results

from Section 4.2, we approximate the CDF of random variable $\bar{d}_j$ for each customer j with its empirical distribution, which we obtain from a set of n random samples $D_j = \{\hat{d}_{j1}, \ldots, \hat{d}_{jn}\}$. For the k-th sample point $\hat{d}_{jk} \in D_j$, we define the parameter $p_{jk} = \ln \hat{F}_{\bar{d}_j}(\hat{d}_{jk})$. That is, p_{jk} is the logarithm value of the empirical CDF given by $\hat{F}_{\bar{d}_j}(\hat{d}_{jk}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(\hat{d}_{ji} \leq \hat{d}_{jk})$, as shown in Figure 4.2.

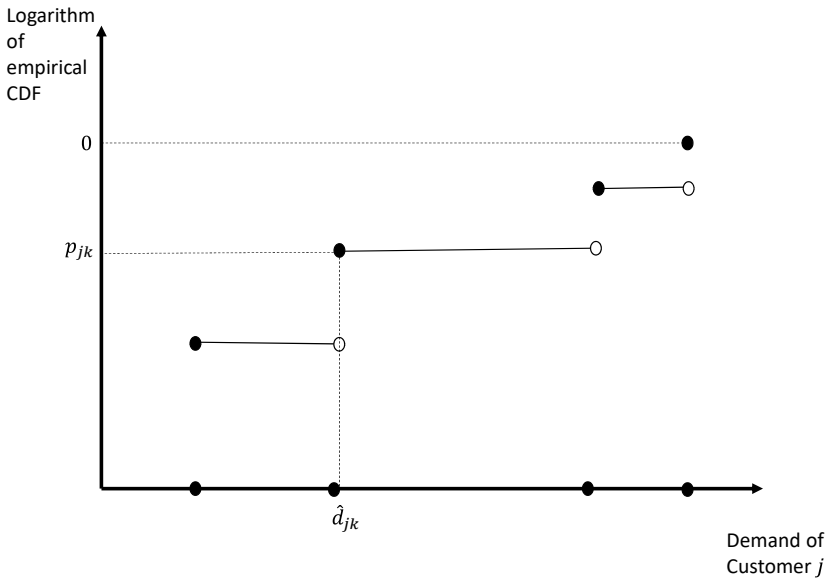


Figure 4.2: Example of $\ln \hat{F}_{\bar{d}_j}(\hat{d}_{jk})$

To formulate this problem as an MIP, we introduce the binary variables z_{jk}, which are related to each sample point $k = 1, \ldots, |D_j|$ for each customer j. Intuitively, z_{jk} determines whether sample point k is used as a reference (cutoff) value to calculate the cumulative probability of random demand $\bar{d}_j$. The objective function in (4.6) and Constraints (4.7), (4.9), and (4.10)–(4.13), describe a reformulation of TIRD when

the CDFs of the random demands are estimated through empirical distributions.

$$\sum_{j \in C} \sum_{k=1}^{n} z_{jk} p_{jk} \geq \ln(1-\epsilon) \tag{4.10}$$

$$\sum_{i \in S} x_{ij} \geq z_{jk} \hat{d}_{jk}, \quad \forall j \in C, k = 1, \ldots, n \tag{4.11}$$

$$\sum_{k=1}^{n} z_{jk} = 1, \quad \forall j \in C \tag{4.12}$$

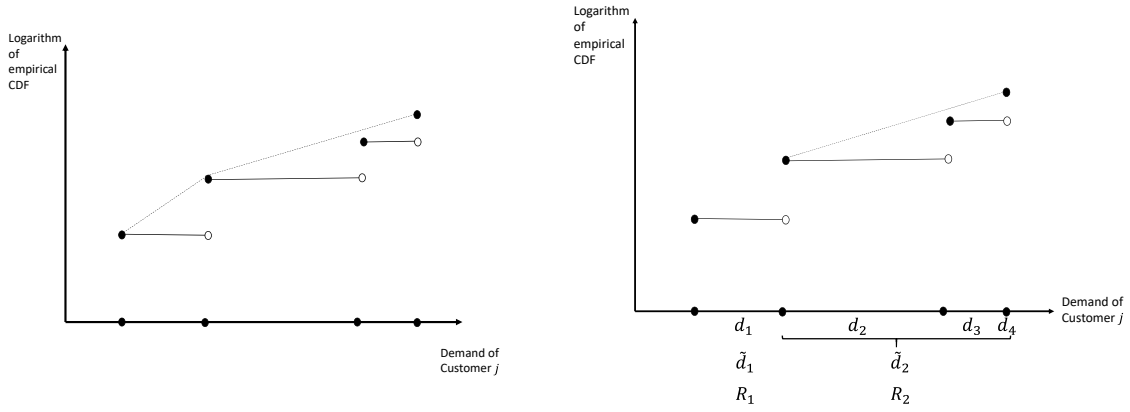
$$z_{jk} \in \{0, 1\}, \quad \forall j \in C, k = 1, \ldots, n \tag{4.13}$$

Constraints (4.10) ensure that the probability of satisfying all demands is at least $1-\epsilon$. Constraints (4.11) enforce both the consistency between the sample value used as a reference to satisfy the probabilistic constraint and the demand satisfaction for each customer. Constraints (4.12) establish that only one sample point can be used as a reference for each customer. Constraints (4.13) enforce the binary nature of the z-variables.

4.4 Overestimation to the Logarithm of Empirical CDF

The sample size required to accurately approximate the CDF may be very large, complicating the solution of the reformulation of TIRD as the number of binary variables depends on the number of samples and customers. To overcome this challenge, we introduce a method to overestimate the logarithm of the empirical CDF function using a piecewise linear concave envelop. Our construction requires considerably less binary variables, which are added as needed to iteratively tighten the overestimation of the function. The overestimator function produces a lower bound on the cost of transporting commodities from suppliers to customers given the (overestimated) probability of satisfying all the demands. Figure 4.3a illustrates an initial concave overestimator of the logarithm of the empirical CDF using a piecewise linear function (dashed lines). This concavity profile allows us to initially approximate the empiri-

cal CDF with no binary variables. However, this overestimator is not exact in some regions of the demand support, requiring a procedure to tighten the estimation wherever is needed. This process demands the use of binary variables, but likely fewer than the TIRD reformulation.



(a) (b)

Figure 4.3: Concave Overestimations to the Logarithm of Empirical CDF Function

4.5 Generalized Formulation

Using the results from Section 4.4, we revise the MIP formulation of TIRD by replacing the sample-point based probabilistic constraint with the piecewise linear overestimator of the logarithm of the CDF. We use the term region to describe a set of consecutive segments in the piecewise linear function's domain. Let R_j be the set of all regions and I_j be the index set of regions where the logarithm of the CDF is approximated by a concave overestimator piecewise linear function for customer $j \in C$. Similarly, let $\bar{I}_j$ be the set of indices of regions containing only one segment (and with exact value of logarithm of the empirical CDF function). For illustration, Figure 4.3b shows a piecewise linear function that contains two regions, R_1 and R_2, (i.e., $R_j =$

$\{R_1, R_2\}$), where $R_2 = \{\tilde{d}_2\}$ is the support of a piecewise linear concave overestimator covering three samples ($\hat{d}_2$, $\hat{d}_3$, and $\hat{d}_4$). The elements of R_j whose indices are not in I_j are regions with a single line segment that represents the logarithm of empirical CDF function exactly (e.g., $\tilde{d}_1$). In this example, $I_j = \{2\}$ and $\bar{I}_j = \{1\}$.

We present a generalization of the MIP formulation for a problem that includes demand regions where the logarithm of the CDF is approximated using piecewise linear concave overestimator function. We define the binary variable z_{jk} for each region $R_{jk} \in R_j$. For each region R_{jk} such that $k \in I_j$, l_{jk} and u_{jk} are the leftmost and rightmost points of the region in the demand domain. Also, p_{jl} and v_{jl} are the intercept and slope of the line segment l that belongs to the function's overestimator function at region k. Note that regions whose indices are in $\bar{I}$ (e.g., R_1 in Figure 4.3a), can be uniquely characterized by the leftmost demand point in the region. This is because larger demand values within the region (i.e., any demand in R_1 in Figure 4.3a) are unattractive as they increase the cost but not the probability in the demand constraint. In contrast, for each region $R_{jk} \in R_j$ whose index is in I_j, we define a continuous variable w_{jk} to represent the demand of customer j within the region.

To account for the logarithm of the empirical CDF value of any region in the objective function, we define the continuous variable e_{jk} for region $R_{jk} \in R_j$ and $k \in I_j$. The e_{jk} variable is the logarithm of the empirical CDF evaluated at any demand point covered by a concave overestimator region R_{jk} for customer j. These continuous variables take the smallest value among the line segments in the region at any given demand value. Because of concavity and the sign of the probability constraint, for any demand in $R_{jk} \in R_j$ and $j \in C$, this value corresponds to the overestimated function. Constraints (4.7), (4.9), (4.12), (4.13), and (4.14)–(4.21) describe the feasible region of TIRD for a general empirical CDF.

$$w_{jk} \geq l_{jk} z_{jk}, \quad \forall j \in C, k \in I_j \tag{4.14}$$

$$w_{jk} \leq u_{jk} z_{jk}, \quad \forall j \in C, k \in I_j \tag{4.15}$$

$$e_{jk} \leq p_{j\tilde{d}} z_{jk} + v_{j\tilde{d}} w_{jk}, \quad \forall j \in C, k \in I_j, \tilde{d} \in R_{jk} \tag{4.16}$$

$$\sum_{j \in C} \sum_{k \in I_j} e_{jk} + \sum_{j \in C} \sum_{k \in \bar{I}_j} p_{jk} z_{jk} \geq \ln(1 - \epsilon) \tag{4.17}$$

$$\sum_{i \in S} x_{ij} \geq l_{jk} z_{jk}, \quad \forall j \in C, k \in \bar{I}_j \tag{4.18}$$

$$\sum_{i \in S} x_{ij} \geq w_{jk}, \quad \forall j \in C, k \in I_j \tag{4.19}$$

$$w_{jk} \geq 0, \quad \forall j \in C, k \in I_j \tag{4.20}$$

$$e_{jk} \geq 0, \quad \forall j \in C, k \in I_j \tag{4.21}$$

Constraints (4.14) and (4.15) ensure that if region k is selected, then the demand must be within the leftmost and rightmost points in the region's domain. Constraints (4.16) linearize the expressions $e_{jk} = \min_{\tilde{d} \in R_{jk}} \{p_{j\tilde{d}} z_{jk} + v_{j\tilde{d}} w_{jk}\}$. In other words, they describe the overestimation of the logarithm of the empirical CDF function for each concave region as a function of the line segments $\tilde{d}$ in the region. The probability calculation in Constraints (4.17) is decomposed into two terms. The first term contains the summation of the e-variables to define the probability contribution from the concave regions. The second term is the same as in (4.10) for each segment that does not belong to a concave region for which the probability contribution is the same as in the logarithm of the empirical CDF. The demand satisfaction requirement is enforced by Constraints (4.18) and (4.19). Finally, Constraints (4.20) and (4.21) define the nature of the w- and e-variables.

4.6 Iterative Refinement Algorithm

We propose an Iterative Refinement Algorithm (IRA) to solve the TIRD problem, which aims at decomposing the problem into simpler subproblems that are iteratively solved until an optimal solution is found. To accomplish this, we define two subproblems. We solve a relaxation of the TIRD problem, producing lower bound values to the optimal objective function value. Refining these subproblems by strengthening the relaxation leads to a sequence of nondecreasing lower bound values along the algorithm iterations. We also solve restricted versions of the TIRD problem, which provide upper bound values to the optimal value of the TIRD problem.

Consider an instance of the TIRD problem aiming to transport commodities from a set of suppliers S to a set of customer C at the minimum cost. We define set D as the union of sets $D_j,\ \forall j \in C$, each of them containing the line segments in the logarithm of the empirical CDF function for the demand of customer j. Similarly, sets R, I, and $\bar{I}$ are the unions of sets R_j, I_j, and $\bar{I}_j$ respectively. We refer to an instance of TIRD as $TIRD(R, I, \bar{I})$ and to its optimal value as $z(R, I, \bar{I})$.

4.6.1 Lower Bound Problem

Consider the lower bound problem $TIRD(\check{R}, \check{I}, \check{\bar{I}})$ with region set $\check{R}_j$ for each customer $j \in C$. Set I_j includes indices of regions in which the piecewise linear function for customer $j \in C$ is approximated by a piecewise linear concave overestimator function. Set $\bar{I}_j$ includes the indices of those regions containing only one segment, which provide the exact value of the logarithm of the empirical CDF function.

Proposition 10 *Consider $TIRD(R, I, \bar{I})$ as a TIRD instance with logarithm of empirical CDF functions (original problem) such that $R_j = \{d_1, ..., d_{|D_j|}\}$, $I_j = \emptyset$, $\bar{I}_j = \{1, ..., |D_j|\}$ for customer $j \in C$. Let $TIRD(\check{R}, \check{I}, \check{\bar{I}})$ be an overestimated prob-*

lem, then $z(R, I, \bar{I}) \geq z(\check{R}, \check{I}, \check{\bar{I}})$.

Proof. First, note that for each customer $j \in C$, the demand domain in any line segment in $\check{R}_j$ is the same as in the exact line segment set R_j. Second, $\check{R}_j$ consists of line segments that are exact (regions with indices in $\check{\bar{I}}$) and overestimators of R_j (regions with indices in $\check{I}_j$). Hence, for for any demand in the overestimated problem the probability value is greater than or equal to the probability value of that demand in the original problem. Consequently, any demand in the exact problem that satisfies the probability constraint, is also feasible for the overestimated problem. Because the feasible set of solutions in the exact problem is a subset of overestimated problem and the objective function value is the same for both of them, then $z(R, I, \bar{I}) \geq z(\check{R}, \check{I}, \check{\bar{I}})$.■

4.6.2 Upper Bound Problem

We define the problem $TIRD(\hat{R}, \hat{I}, \hat{\bar{I}})$ as an upper bound subproblem. That is, $\hat{R}_j \subseteq \{d_1, ..., d_{|D_j|}\}$, $\hat{I}_j = \emptyset$, $\hat{\bar{I}}_j \subseteq \{1, ..., |D_j|\}$ for customer $j \in C$.

Proposition 11 *Consider* $TIRD(R, I, \bar{I})$ *as a TIRD instance with the logarithm of the empirical CDF functions (original problem) defined such that* $R_j = \{d_1, ..., d_{|D_j|}\}$, $I_j = \emptyset$, $\bar{I}_j = \{1, ..., |D_j|\}$ *for each customer* $j \in C$. *Let* $TIRD(\hat{R}, \hat{I}, \hat{\bar{I}})$ *be a restricted problem, then* $z(\hat{R}, \hat{I}, \hat{\bar{I}}) \geq z(R, I, \bar{I})$.

Proof. By definition, the demand domain in set $\hat{R}_j$ is a subset of the domain in set R_j for each customer j. Also, the line segment set $\hat{R}_j$ is a subset of exact line segments R_j. As a result, for any demand in the restricted problem the probability value is the same as the probability value of that demand in the original problem. Hence, any feasible demand for $TIRD(\hat{R}, \hat{I}, \hat{\bar{I}})$ is also feasible for $TIRD(R, I, \bar{I})$.

Considering that objective function values are the same for both problems for any feasible demand, we conclude that $z(\hat{R}, \hat{I}, \hat{\bar{I}}) \geq z(R, I, \bar{I})$.■

4.6.3 Solution Algorithm

The mixed integer programming formulation can be used to solve TIRD using commercial solvers. For an instance with demand sample size $|D|$ and number of customers $|C|$, this formulation requires $|C||D|$ binary variables, which makes it computationally inefficient for large-scale problems. In this section, we introduce an Iterative Refinement Algorithm (IRA) to enhance the performance of the MIP formulation when solving large-scale instances of the TIRD problem.

We introduce two algorithms that are utilized to construct the overestimation functions for the lower bound problem and then update the overestimations in a progressive fashion. Algorithm 6 produces a concave overestimation to the logarithm of the empirical CDF for a subset of the function domain of any given customer. In this algorithm, line segments are iteratively generated to form a concave region that is an overestimator of the logarithm of the empirical CDF.

In Line 3, we consider the leftmost point in the domain of the given region and find the line with lowest slope intersecting with the given exact line segments. In Line 5 the generated line will be added to the concave region. We continue this process until the whole domain of the given region is covered by the concave overestimator.

Algorithm 7 enhances the concave overestimations. The input of the algorithm is an instance of the problem, $TIRD(\check{R}, \check{I}, \check{\bar{I}})$, along with a solution $\bar{\mathbf{w}}$. For each customer $j \in C$, Line 3 finds the exact line segment d_b that contains the solution in its domain. We redefine R_{jk} as $\{d_b\}$ in Line 4. Next, we run Algorithm 6 for the left and right intervals in the domain of the region R_{jk} (Lines 5 and 6), if exist.

Algorithm 8 defines the Iterative Refinement Algorithm to solve TIRD. In Line

Algorithm 6 : Function Overestimation

1: Consider $R_{jk} = \{d_m, ..., d_n\}$ given for customer j

2: Let $\bar{R} = \emptyset$

3: Start with leftmost point in the domain of line segments in R_{jk} and let this point be $x_{current}$

4: At point $x_{current}$, find the overestimator line segment with lowest slope and denote the intersection of this line d_{new} with line segments in R_{jk} as updated $x_{current}$

5: Add d_{new} to $\bar{R}$

6: **if** $x_{current} \neq$ rightmost point in the domain of line segments in R_{jk} **then**

7: Go to Step 4

8: Redefine $R_{jk} \leftarrow \bar{R}$

Algorithm 7 : Function Overestimation Update

1: Consider $\check{R}, \check{I}, \check{\bar{I}}$ and given solution $\bar{\mathbf{w}}$. For each customer j, $\bar{w}_j$ is the demand of that customer which belongs to concave region $R_{jk} \in R_j$

2: **for** $j \in C$ **do**

3: Find exact line segments $\{d_a, ..., d_b, ..., d_c\} \subseteq D_j$ that have the same demand domain as line segments in R_{jk}. Let d_b be the line segment that contains demand $\bar{w}_j$ in its domain

4: Redefine $R_{jk} \leftarrow \{d_b\}$

5: For nonempty $\{d_a, ..., d_{b-1}\}$ perform overestimation through Algorithm 6 to obtain R_l

6: For nonempty $\{d_{b+1}, ..., d_c\}$ perform underestimation through Algorithm 6 to obtain R_r

7: Add R_l and R_r to R_j and update indices sets I_j and $\bar{I}_j$ accordingly

2, we generate an overestimation of the exact function to produce the initial lower bound problem. While the optimal solution is not found, we iteratively solve the lower bound and upper bound subproblems. In Line 7, we solve the lower bound problem and in Line 8, we strengthen the function overestimations to produce lower bound problem that is solved in the next iteration. Utilizing the optimal solution of each lower bound problem, we produce the upper bound subproblem's function in Line 10.

Since the function overestimation is improved to make the lower bound of next

Algorithm 8 : Iterative Refinement Algorithm for TIRD

1: For any given customer $j \in C$, let $\check{R}_j = \{\check{R}_{j1}\}$, $\check{I} = \{1\}$, $\check{\check{I}} = \emptyset$, and $\check{R}_{j1} = \{d_1, ..., d_{|D_j|}\}$ is constructed in Step 2

2: Perform overestimation on $\check{R}_{j1}$ for each customer j through Algorithm 6

3: Let $\hat{R} = \hat{I} = \hat{\hat{I}} = \emptyset$

4: Set $UB_0 = \infty$, $LB_0 = 0$, and set counter $i = 0$

5: **while** $UB_i > LB_i$ **do**

6: Set $i = i + 1$

7: Solve $TIRD(\check{R}, \check{I}, \check{\check{I}})$ to obtain an optimal solution $\check{\mathbf{w}}^i$, optimal value $\check{z}^i$. Set $LB_i = \check{z}^i$

8: Update $\check{R}, \check{I}, \check{\check{I}}$ with solution $\check{\mathbf{w}}^i$ through Algorithm 7

9: For each customer j, find the line segment d_k in D_j that contains the flow given by $\check{\mathbf{w}}^i$

10: Add $\{d_k\}$ to $\hat{R}_j$ and update $\hat{\hat{I}} \leftarrow \hat{\hat{I}} \cup \{|\hat{R}_j|\}$

11: Solve $TIRD(\hat{R}, \hat{I}, \hat{\hat{I}})$ to obtain an optimal solution $\hat{\mathbf{w}}^i$ and optimal value $\hat{z}^i$

12: Set $UB_i = \hat{z}^i$ and update the incumbent solution $\bar{\mathbf{w}} \leftarrow \hat{\mathbf{w}}^i$ and objective $\bar{z} = UB_i$

13: **if** $UB_i = LB_i$ **then**

14: Incumbent $\bar{\mathbf{w}}$ is optimal with objective $\bar{z}$. Go to Step 15

15: Return $\bar{\mathbf{w}}$ and $\bar{z}$

iteration, we observe that $LB_{i+1} \geq LB_i$. In addition, the sets of line segments for the upper bound problem in iteration i are subsets of sets of line segments in iteration $i+1$, resulting in $UB_i \geq UB_{i+1}$. Figure 4.4 illustrates examples of overestimated functions that are enhanced to make the next lower bound problem. The blue lines indicate the line segments in the logarithm of the empirical CDF D_j, whereas the red lines indicate the current overestimators. The green lines are the improved overestimators.

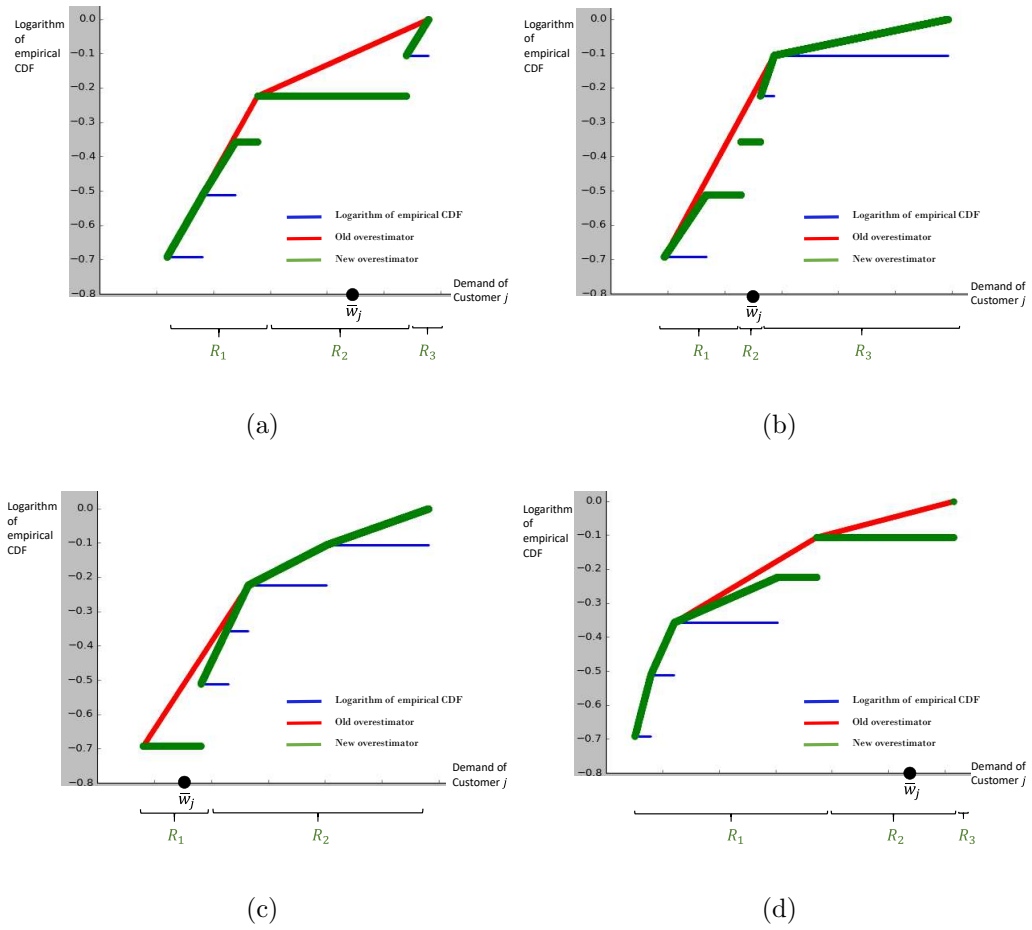


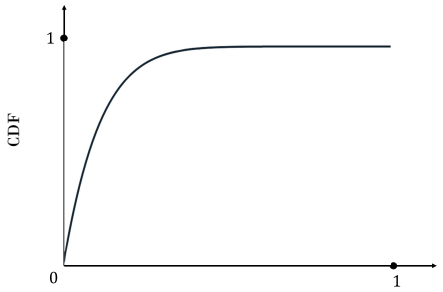
Figure 4.4: Improved Overestimations of Piecewise Linear Functions

4.7 Computational Results

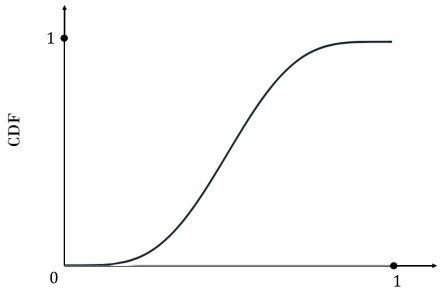
To evaluate the performance of our proposed approach, we solve the TIRD problem on randomly generated instances. We execute computations utilizing Python with CPLEX 12.7 as optimization solver on a desktop computer with an Intel Core i7 2.40 GHz processor and 8.0 GB RAM. In all computations, a time limit of one hour is enforced. The procedure to generate random instances of the TIRD problem is presented in Section 4.7.1. In Section 4.7.2, we present the computational results and analyze the performance of proposed solution approach.

4.7.1 Random Instance Generation

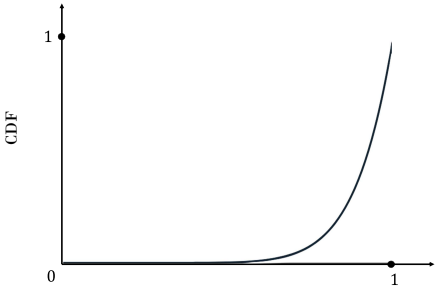
We create random instances varying the number of suppliers $|S|$, number of customers $|C|$, and number of sample points $|D|$. We expect the performance of the proposed approaches to be majorly affected by $|C|$ and $|D|$ since these values determine the number of binary variables in the MIP model. The cost of transporting a unit of commodity from supplier i to customer j, denoted by c_{ij}, is generated using a uniform distribution in the range $[1, 100]$. For an instance with $|S|$ suppliers, the capacity of supplier $i \in S$, denoted by Q_i, is generated using a uniform distribution in the range $[1000/|S|, 1000]$. In addition, demand sample points for customer j are generated from beta distribution with parameters $a = j$ and $b = |C| + 1 - j$ and are multiplied by 1000. This procedure ensures that the demand distributions come from distributions of various shapes. As seen in Figure 4.5, such shape parameters lead to a variety of CDF functions. For the experiments, we select values of $(|S|, |C|)$ from the set $\{(3, 4), (6, 8), (9, 12), (12, 16), (15, 20)\}$ and $|D|$ from the set $\{50, 100, 200, 300, 400, 500, 1000, 2000\}$.



(a) $a = 1$, $b = 9$



(b) $a = 5$, $b = 5$



(c) $a = 9$, $b = 1$

Figure 4.5: CDF of Beta Distribution with Various Shape Parameters

4.7.2 Results

In this section, we compare the computational efficiency of the Iterative Refinement Algorithm (IRA) with the MIP formulation. The first two columns of Table 4.1 define the instance solved for specific $|S|$, $|C|$ and $|D|$ values. In general, we expect instances to be more difficult as $|C|$ and $|D|$ increase. In Table 4.1, t_{IRA} is the running time (in seconds) to solve an instance for each combination of $(|S|, |C|)$ and $|D|$ utilizing the IRA approach. Similarly, t_{MIP} is the running time (in seconds) to solve an instance utilizing the MIP formulation. When an instance times out, we report

the running time as "-". For such instances, g_{IRA} and g_{MIP} are the optimality gaps obtained using IRA and MIP approaches, respectively. The optimality gap of the IRA method is determined as $100(UB_k - LB_k)/UB_k$, where k is the last iteration before timing out. Note that for all values of $(|S|, |C|)$, the IRA method is able to solve instances to optimality for sample sizes 50 and 100. When timing out, the optimality gap obtaned from the IRA method is always below 1% compared to large optimality gaps obtained with the MIP formulation.

We use $|z|_{MIP}$, v_{MIP}, and c_{MIP} to report the number of binary variables, total number of variables, and number of constraints in the MIP model. Similarly, $|z|_{IRA}$, v_{IRA}, and c_{IRA} report the number of binary variables, total number of variables and number of constraints in the last lower bound problem in the IRA method. These metrics indicate that the IRA method significantly reduces the number of variables and constraints used. Also, $|\check{D}|_{avg}$ and $|\check{D}|_{max}$ report the average and maximum number of segments/regions used in last iteration of the lower bound problem. When compared to the number of samples, these values show that the IRA maintains tractable approximations in most of the function domain, requiring only some binaries to retrieve the true value (non-approximated) of the empirical CDF. Parameter i reports number of iterations in the IRA method.

Table 4.1: Computational Performance of the Proposed Approaches on Random TIRD Problem Instances

$(\|S\|,\|C\|)$	$\|D\|$	t_{IRA}	t_{MIP}	g_{IRA}	g_{MIP}	$\|z\|_{IRA}$	$\|z\|_{MIP}$	v_{IRA}	v_{MIP}	c_{IRA}	c_{MIP}	$\|\check{D}\|_{avg}$	$\|\check{D}\|_{max}$	i
(3,4)	50	11	1	-	-	18	104	42	116	71	112	4.50	6	10
	100	9	3	-	-	15	204	45	216	86	212	3.75	7	7
	200	11	115	-	-	17	404	47	416	108	412	4.25	5	8
	300	14	1133	-	-	23	604	53	616	115	612	5.75	7	12
	400	15	955	-	-	23	804	59	816	141	812	5.75	8	11
	500	32	2572	-	-	43	1004	91	1016	211	1012	10.75	17	23
	1000	15	-	-	66	22	2004	60	2016	171	2012	5.50	7	10
	2000	12	-	-	38	19	4004	47	4016	172	4012	4.75	6	9
(6,8)	50	15	198	-	-	26	208	104	256	132	223	3.25	5	10
	100	17	-	-	12	28	408	106	456	158	423	3.50	4	13
	200	166	-	-	47	57	808	153	856	253	823	7.13	14	27
	300	33	-	-	53	36	1208	116	1256	216	1223	4.50	5	19
	400	38	-	-	63	38	1608	120	1656	234	1623	4.75	7	17
	500	33	-	-	59	37	2008	119	2056	241	2023	4.63	7	16
	1000	56	-	-	88	37	4008	117	4056	299	4023	4.63	6	16
(9,12)	50	68	-	-	5	51	312	199	420	204	334	4.25	5	24
	100	95	-	-	28	51	612	203	720	242	634	4.25	7	22
	200	710	-	-	54	59	1212	227	1320	335	1234	4.92	9	26
	300	-	-	0.0014	67	59	1812	227	1920	362	1834	4.92	9	26
	400	-	-	0.0077	75	59	2412	231	2520	396	2434	4.92	9	26
(12,16)	50	89	-	-	14	56	416	298	608	262	445	3.50	5	23
	100	525	-	-	35	60	816	310	1008	327	845	3.75	7	25
	200	-	-	0.0003	58	60	1616	314	1808	389	1645	3.75	7	26
	300	-	-	0.0035	72	63	2416	323	2608	458	2445	3.94	5	26
	400	2793	-	-	79	57	3216	309	3408	455	3245	3.56	5	23
(15,20)	50	889	-	-	19	67	520	429	820	319	556	3.35	5	27
	100	398	-	-	41	65	1020	431	1320	389	1056	3.25	5	25
	200	-	-	0.0763	64	58	2020	432	2320	471	2056	2.90	5	20
	300	-	-	0.0137	76	63	3020	441	3320	560	3056	3.15	7	23
	400	-	-	0.0033	82	54	4020	426	4320	544	4056	2.70	5	18

4.8 Concluding Remarks

Utilizing the empirical CDF function, we transformed a chance constraint program into a mixed inreger program with binary variables related to each segment of the

function domain. This binary variable activates the corresponding segment in the CDF and recovers the value of the empirical cumulative distribution function. Due to computational challenges of solving this MIP model, we introduced an iterative refinement technique to approximate the logarithm of empirical CDF function by concave overestimations. These overestimations are tractable in the sense that they can be represented using a reduced number of binary variables compared to the MIP formulation. Indeed, the computational results show that we only need a small subset of exact line segments to find the optimal solution. The proposed algorithm consists of lower bound and upper bound subproblems that provide very small optimality gaps in case of a time out compared with the large gaps that we obtain from MIP model. This technique makes it possible to solve large instances with reasonable sample size (50 and 100) to optimality within one hour time limit.

REFERENCES

Abichandani, P., K. Mallory and M.-Y. Hsieh, "Experimental multi-vehicle path coordination under communication connectivity constraints", in "Experimental Robotics", pp. 183–197 (Springer, 2013).

Adamo, T., T. Bektaş, G. Ghiani, E. Guerriero and E. Manni, "Path and speed optimization for conflict-free pickup and delivery under time windows", Transportation Science **52**, 4, 739–755 (2018).

Ahuja, R., T. Magnanti and J. Orlin, *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Upper Saddle River, NJ, 1993).

Andersen, J., T. G. Crainic and M. Christiansen, "Service network design with management and coordination of multiple fleets", European Journal of Operational Research **193**, 2, 377–389 (2009).

Atamtürk, A., "On capacitated network design cut–set polyhedra", Mathematical Programming **92**, 3, 425–437 (2002).

Atamtürk, A. and D. Rajan, "On splittable and unsplittable flow capacitated network design arc–set polyhedra", Mathematical Programming **92**, 2, 315–333 (2002).

Balakrishnan, A., T. L. Magnanti and R. T. Wong, "A dual-ascent procedure for large-scale uncapacitated network design", Operations Research **37**, 5, 716–740 (1989).

Balakrishnan, N. and R. T. Wong, "A network model for the rotating workforce scheduling problem", Networks **20**, 1, 25–42 (1990).

Bartholdi III, J. J., J. B. Orlin and H. D. Ratliff, "Cyclic scheduling via integer programs with circular ones", Operations Research **28**, 5, 1074–1085 (1980).

Ben-Tal, A., L. El Ghaoui and A. Nemirovski, *Robust optimization*, vol. 28 (Princeton University Press, 2009).

Beraldi, P. and M. E. Bruni, "An exact approach for solving integer problems under probabilistic constraints with random technology matrix", Annals of operations research **177**, 1, 127–137 (2010).

Beraldi, P., M. E. Bruni and A. Violi, "Capital rationing problems under uncertainty and risk", Computational Optimization and Applications **51**, 3, 1375–1396 (2012).

Beraldi, P. and A. Ruszczyński, "The probabilistic set-covering problem", Operations Research **50**, 6, 956–967 (2002).

Bhaskaran, S. and J. Franz, "Optimal design of gas pipeline networks", journal of the Operational Research Society **30**, 12, 1047–1060 (1979).

Bienstock, D. and O. Günlük, "Capacitated network design—polyhedral structure and computation", Informs journal on Computing **8**, 3, 243–259 (1996).

Bräysy, O. and M. Gendreau, "Vehicle routing problem with time windows, part i: Route construction and local search algorithms", Transportation Science **39**, 1, 104–118 (2005a).

Bräysy, O. and M. Gendreau, "Vehicle routing problem with time windows, part ii: Metaheuristics", Transportation Science **39**, 1, 119–139 (2005b).

Calafiore, G. and M. C. Campi, "Uncertain convex programs: randomized solutions and confidence levels", Mathematical Programming **102**, 1, 25–46 (2005).

Campi, M. C. and S. Garatti, "A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality", Journal of Optimization Theory and Applications **148**, 2, 257–280 (2011).

Carotenuto, P., S. Giordani, S. Ricciardelli and S. Rismondo, "A tabu search approach for scheduling hazmat shipments", Computers and Operations Research **34**, 5, 1328–1350 (2007).

Chardaire, P., G. McKeown, S. Verity-Harrison and S. Richardson, "Solving a time-space network formulation for the convoy movement problem", Operations Research **53**, 2, 219–230 (2005).

Cordeau, J. and G. Laporte, "The dial-a-ride problem: models and algorithms", Annals of Operations Research **153**, 1, 29–46 (2007).

Cordeau, J.-F., F. Pasin and M. M. Solomon, "An integrated model for logistics network design", Annals of operations research **144**, 1, 59–82 (2006).

Corréa, A., A. Langevin and L. Rousseau, "Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming", in "International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming", pp. 370–379 (Springer, 2004).

Craig, M. T., P. Jaramillo, H. Zhai and K. Klima, "The economic merits of flexible carbon capture and sequestration as a compliance strategy with the clean power plan", Environmental science & technology **51**, 3, 1102–1109 (2017).

Crainic, T. G., "Service network design in freight transportation", European journal of operational research **122**, 2, 272–288 (2000).

Crainic, T. G., A. Frangioni and B. Gendron, "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design", Discrete Applied Mathematics **112**, 1-3, 73–99 (2001).

Crainic, T. G. and J.-M. Rousseau, "Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem", Transportation Research Part B: Methodological **20**, 3, 225–242 (1986).

Cruz, F. R., J. M. Smith and G. R. Mateus, "Solving to optimality the uncapacitated fixed-charge network flow problem", Computers & Operations Research **25**, 1, 67–81 (1998).

Current, J. and S. Ratick, "A model to assess risk, equity and efficiency in facility location and transportation of hazardous materials", Location Science **3**, 3, 187–201 (1995).

De Jongh, A., M. Gendreau and M. Labbe, "Finding disjoint routes in telecommunications networks with two technologies", Operations Research **47**, 1, 81–92 (1999).

Dentcheva, D. and G. Martinez, "Regularization methods for optimization problems with probabilistic constraints", Mathematical Programming **138**, 1-2, 223–251 (2013).

Dentcheva, D., A. Prékopa and A. Ruszczynski, "Concavity and efficient points of discrete distributions in probabilistic programming", Mathematical Programming **89**, 1, 55–77 (2000).

Diarrassouba, I., M. Mahjoub, A. Mahjoub and H. Yaman, "Finding failure-disjoint paths for path diversity protection in communication networks", Annals of Telecommunications **73**, 1–2, 5–28 (2018).

Dijkstra, E., "A note on two problems in connexion with graphs", Numerische Mathematik **1**, 1, 269–271 (1959).

Dixit, A., A. Mishra and A. Shukla, "Vehicle routing problem with time windows using meta-heuristic algorithms: a survey", in "Harmony Search and Nature Inspired Optimization Algorithms", edited by N. Yadav, A. Yadav, J. Bansal, K. Deep and J. Kim, vol. 741, pp. 539–546 (Springer, Singapore, 2019).

Dooley, J. J., R. T. Dahowski, C. L. Davidson, M. A. Wise, N. Gupta, S. H. Kim and E. L. Malone, "Carbon dioxide capture and geologic storage: a core element of a global energy technology strategy to address climate change", PNNL, Richland, WA (2006).

Erkut, E. and V. Verter, "Modeling of transport risk for hazardous materials", Operations Research **46**, 5, 625–642 (1998).

Esfandeh, T., R. Batta and C. Kwon, "Time-dependent hazardous-materials network design problem", Transportation Science **52**, 2, 454–473 (2018).

Eto, R., A. Murata, Y. Uchiyama and K. Okajima, "Co-benefits of including ccs projects in the cdm in india's power sector", Energy policy **58**, 260–268 (2013).

Fazlollahtabar, H. and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study", Journal of Intelligent & Robotic Systems **77**, 3-4, 525–545 (2015).

Ferrati, M. and L. Pallottino, "A time expanded network based algorithm for safe and efficient distributed multi-agent coordination", in "52nd IEEE Conference on Decision and Control", pp. 2805–2810 (IEEE, 2013).

Ferrera, E., A. Castaño, J. Capitán, P. Marrón and A. Ollero, "Multi-robot operation system with conflict resolution", in "ROBOT2013: First Iberian Robotics Conference", pp. 407–419 (Springer, 2014).

Ferrera, E., A. Castaño, J. Capitán, A. Ollero and P. Marrón, "Decentralized collision avoidance for large teams of robots", in "16th International Conference on Advanced Robotics (ICAR)", pp. 1–6 (IEEE, 2013).

Frangioni, A. and B. Gendron, "0–1 reformulations of the multicommodity capacitated network design problem", Discrete Applied Mathematics **157**, 6, 1229–1241 (2009).

Frazzoli, E., Z.-H. Mao, J.-H. Oh and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming", Journal of Guidance, Control, and Dynamics **24**, 1, 79–86 (2001).

Fredman, M. and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", Journal of the ACM **34**, 3, 596–615 (1987).

Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness* (W. H. Freeman & Co., Princeton, NJ, 1979).

Geoffrion, A. M. and G. W. Graves, "Multicommodity distribution system design by benders decomposition", Management science **20**, 5, 822–844 (1974).

Ghamlouche, I., T. G. Crainic and M. Gendreau, "Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design", Operations research **51**, 4, 655–667 (2003).

Global, C., "Institute. 2016 the global status of ccs 2016—summary report", Global CCS Institute. See https://www. globalccsinstitute. com (accessed February 2017) (2016).

Gopalan, R., K. Kolluri, R. Batta and M. Karwan, "Modeling equity of risk in the transportation of hazardous materials", Operations Research **38**, 6, 961–973 (1990a).

Gopalan, R., R. R. Batta and M. Karwan, "The equity constrained shortest path problem", Computers and Operations Research **17**, 3, 297–307 (1990b).

Grötschel, M., C. L. Monma and M. Stoer, "Design of survivable networks", in "Handbooks on Operations Research and Management Science", vol. 7, chap. 10, pp. 617–672 (Elsevier Science, North-Holland, Amsterdam, 1995).

Guisewite, G. M. and P. M. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms", Annals of Operations Research **25**, 1, 75–99 (1990).

Günlük, O., "A branch-and-cut algorithm for capacitated network design problems", Mathematical programming **86**, 1, 17–39 (1999).

Hoy, M., A. Matveev and A. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey", Robotica **33**, 3, 463–497 (2015).

Hu, X., L. Chen, B. Tang, D. Cao and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles", Mechanical Systems and Signal Processing **100**, 482–500 (2018).

Jin, Q., G. Wu, K. Boriboonsomsin and M. Barth, "Multi-agent intersection management for connected vehicles using an optimal scheduling approach", in "International Conference on Connected Vehicles and Expo (ICCVE)", pp. 185–190 (IEEE, 2012).

Kamal, M., J. Imura, T. Hayakawa, A. Ohata and K. Aihara, "A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights", IEEE Transactions on Intelligent Transportation Systems **16**, 3, 1136–1147 (2015).

Katayama, N., M. Chen and M. Kubo, "A capacity scaling heuristic for the multicommodity capacitated network design problem", Journal of computational and applied mathematics **232**, 1, 90–101 (2009).

Kerivin, H. and A. Mahjoub, "Design of survivable networks: A survey", Networks **46**, 1, 1–21 (2005).

Kim, D. and P. M. Pardalos, "A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure", Operations Research Letters **24**, 4, 195–203 (1999).

Kobayashi, Y. and K. Otsuki, "Max-flow min-cut theorem and faster algorithms in a circular disk failure model", in "Proceedings of the 2014 IEEE Conference on Computer Communications INFOCOM", pp. 1635–1643 (IEEE, 2014).

Kobos, P., L. Malczynski, D. Borns and B. McPherson, "The 'string of pearls': the integrated assessment cost and source-sink model", in "Conference Proceedings of the 6th Annual Carbon Capture & Sequestration Conference", (2007).

Krishnamurthy, N., R. Batta and M. Karwan, "Developing conflict-free routes for automated guided vehicles", Operations Research **41**, 6, 1077–1090 (1993).

Kuby, M. J., J. M. Bielicki and R. S. Middleton, "Optimal spatial deployment of co2 capture and storage given a price on carbon", International Regional Science Review **34**, 3, 285–305 (2011).

Kumar, P. and T. Narendran, "Integer programming formulation for convoy movement problem", International Journal of Intelligent Defence Support Systems **1**, 3, 177–188 (2008).

Lejeune, M. A., "Pattern-based modeling and solution of probabilistically constrained optimization problems", Operations Research **60**, 6, 1356–1372 (2012).

List, G., P. Mirchandani, M. Turnquist and K. Zografos, "Modeling and analysis for hazardous materials transportation: Risk analysis, routing/scheduling, and facility location", Transportation Science **25**, 2, 100–114 (1991).

Luedtke, J., "An integer programming and decomposition approach to general chance-constrained mathematical programs", in "International Conference on Integer Programming and Combinatorial Optimization", pp. 271–284 (Springer, 2010).

Luedtke, J. and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints", SIAM Journal on Optimization **19**, 2, 674–699 (2008a).

Luedtke, J. and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints", SIAM Journal on Optimization **19**, 2, 674–699 (2008b).

Malandraki, C. and M. Daskin, "Time dependent vehicle routing problems: formulations, properties and heuristic algorithms", Transportation Science **26**, 3, 185–200 (1992).

Margolis, J., K. Sullivan, S. Mason and M. Magagnotti, "A multi-objective optimization model for designing resilient supply chain networks", International Journal of Production Economics **204**, 174–185 (2018).

Middleton, R. S., "A new optimization approach to energy network modeling: anthropogenic co2 capture coupled with enhanced oil recovery", International Journal of Energy Research **37**, 14, 1794–1810 (2013).

Middleton, R. S. and J. M. Bielicki, "A scalable infrastructure model for carbon capture and storage: Simccs", Energy Policy **37**, 3, 1052–1060 (2009).

Middleton, R. S. and A. R. Brandt, "Using infrastructure optimization to reduce greenhouse gas emissions from oil sands extraction and processing", Environmental science & technology **47**, 3, 1735–1744 (2013).

Middleton, R. S., M. J. Kuby, R. Wei, G. N. Keating and R. J. Pawar, "A dynamic model for optimally phasing in co2 capture and storage infrastructure", Environmental Modelling & Software **37**, 193–205 (2012).

Nemirovski, A. and A. Shapiro, "Convex approximations of chance constrained programs", SIAM Journal on Optimization **17**, 4, 969–996 (2006).

Neumayer, S., A. Efrat and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model", Computer Networks **77**, 117–127 (2015).

Neumayer, S., G. Zussman, R. Cohen and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters", in "INFOCOM, 2009 Proceedings IEEE", pp. 1566–1574 (2009).

Nishi, T., Y. Hiranaka and I. Grossmann, "A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles", Computers & Operations Research **38**, 5, 876–888 (2011).

Omran, M. T., J.-R. Sack and H. Zarrabi-Zadeh, "Finding paths with minimum shared edges", Journal of Combinatorial Optimization **26**, 4, 709–722 (2013).

Otsuki, K., Y. Kobayashi and K. Murota, "Improved max-flow min-cut algorithms in a circular disk failure model with application to a road network", European Journal of Operational Research **248**, 2, 396–403 (2016).

Otto, A., N. Agatz, J. Campbell, B. Golden and E. Pesch, "Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey", Networks **72**, 4, 411–458 (2018).

Pagnoncelli, B. K., S. Ahmed and A. Shapiro, "Sample average approximation method for chance constrained programming: theory and applications", Journal of optimization theory and applications **142**, 2, 399–416 (2009).

Phung, M., C. Quach, T. Dinh and Q. Ha, "Enhanced discrete particle swarm optimization path planning for uav vision-based surface inspection", Automation in Construction **81**, 25–33 (2017).

Pillac, V., M. Gendreau, C. Guéret and A. Medaglia, "A review of dynamic vehicle routing problems", European Journal of Operational Research **225**, 1, 1–11 (2013).

Raack, C., A. M. Koster, S. Orlowski and R. Wessäly, "On cut-based inequalities for capacitated network design polyhedra", Networks **57**, 2, 141–156 (2011).

Randazzo, C. and H. P. L. Luna, "A comparison of optimal methods for local access uncapacitated network design", Annals of Operations Research **106**, 1-4, 263–286 (2001).

Richards, A. and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming", in "Proceedings of the 2002 American Control Conference", vol. 3, pp. 1936–1941 (IEEE, 2002).

Rios-Torres, J. and A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps", IEEE Transactions on Intelligent Transportation Systems **18**, 5, 1066–1077 (2017).

Rothfarb, B., H. Frank, D. Rosenbaum, K. Steiglitz and D. J. Kleitman, "Optimal design of offshore natural-gas pipeline systems", Operations research **18**, 6, 992–1020 (1970).

Ruszczyński, A., "Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra", Mathematical Programming **93**, 2, 195–215 (2002).

Santoso, T., S. Ahmed, M. Goetschalckx and A. Shapiro, "A stochastic programming approach for supply chain network design under uncertainty", European Journal of Operational Research **167**, 1, 96–115 (2005).

Sathre, R., L. Gustavsson and N. Le Truong, "Climate effects of electricity production fuelled by coal, forest slash and municipal solid waste with and without carbon capture", Energy **122**, 711–723 (2017).

Saxena, A., V. Goyal and M. A. Lejeune, "Mip reformulations of the probabilistic set covering problem", Mathematical programming **121**, 1, 1–31 (2010).

Sullivan, K. and J. Smith, "Exact algorithms for solving a euclidean maximum flow network interdiction problem", Networks **64**, 2, 109–124 (2014).

Suurballe, J., "Disjoint paths in a network", Networks **4**, 2, 125–145 (1974).

Tanner, M. W. and L. Ntaimo, "Iis branch-and-cut for joint chance-constrained stochastic programs and application to optimal vaccine allocation", European Journal of Operational Research **207**, 1, 290–296 (2010).

Thomas, S., D. Deodhare and M. Murty, "Extended conflict-based search for the convoy movement problem", IEEE Intelligent Systems **30**, 6, 60–70 (2015).

Toumazis, I. and C. Kwon, "Worst-case conditional value-at-risk minimization for hazardous materials transportation", Transportation Science **50**, 4, 1174–1187 (2016).

Transportation Networks for Research Core Team, `https://github.com/bstabler/TransportationNetworks` (2018).

Vielma, J. P., S. Ahmed and G. Nemhauser, "Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions", Operations research **58**, 2, 303–315 (2010).

Yaghini, M., M. Karimi, M. Rahbar and M. H. Sharifitabar, "A cutting-plane neighborhood structure for fixed-charge capacitated multicommodity network design problem", INFORMS Journal on Computing **27**, 1, 48–58 (2015).

Yamashita, T., K. Izumi, K. Kurumatani and H. Nakashima, "Smooth traffic flow with a cooperative car navigation system", in "Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems", pp. 478–485 (ACM, 2005).

Yen, J., "Finding the k shortest loopless paths in a network", Management Science **17**, 11, 712–716 (1971).

Yu, J. and S. LaValle, "Time optimal multi-agent path planning on graphs", in "Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence", (2012).

Zhu, F. and S. Ukkusuri, "A linear programming formulation for autonomous intersection control within a dynamic traffic assignment and connected vehicle environment", Transportation Research Part C: Emerging Technologies **55**, 363–378 (2015).

Żotkiewicz, M., W. Ben-Ameur and M. Pióro, "Finding failure-disjoint paths for path diversity protection in communication networks", IEEE Communications Letters **14**, 8, 776–778 (2010).

APPENDIX A

PROBLEM COMPLEXITY

We define the decision version of RASTC as DRASTC, which seeks to ascertain whether or not there is a feasible path and schedule for every vehicle with maximum deviation from each vehicle's shortest-path time of no more than B. For any given candidate solution, we can verify in $O(|V||N|)$ steps if the path and schedule for each vehicle is feasible in terms of flow balance and travel time consistency, meaning that departure times are consistent with arrival, waiting, and travel times. Moreover, we can verify in $O(|V|^2(|A|^2+|A||N|))$ steps whether the candidate solution is feasible in terms of conflict conditions (arc-arc and arc-node) by examining each pair of vehicles and their departure times from each node. Hence, DRASTC belongs to NP.

We transform any arbitrary instance of the BOOLEAN SATISFIABILITY (SAT) problem, which is known to be NP-complete (Garey and Johnson, 1979), to DRASTC. Suppose that in such SAT instance there are m variables, n clauses, and l_i literals in clause $i \in \{1,\ldots,n\}$. First, we construct a network with a single origin and destination, and $2(n+2)$ node-disjoint layers denoted by $L_0, \bar{L}_0, L_1, \bar{L}_1, \ldots, L_{n+1}, \bar{L}_{n+1}$. We use three types of nodes: *literal*, *consolidation*, and *truth assignment*. Layers L_i and $\bar{L}_i$, for $i = 0,\ldots,n+1$, contain $2m$ literal nodes each, representing each variable and its negation. Layers $\bar{L}_0$, $\bar{L}_{n+1}$, and $L_0,\ldots,L_{n+1}$ have one consolidation node each, and L_0 and $\bar{L}_{n+1}$ contain the origin and destination nodes, respectively. Each intermediate layer L_i and $\bar{L}_i$ contains l_i truth assigner nodes, for $i = 1,\ldots,n$.

For each variable and its negation, we build a *literal path* from origin to destination that uses exactly one literal node in each layer. A literal path consist of an arc from origin to a literal node in L_0, arcs from a literal node in L_i to a literal node in $\bar{L}_i$, for $i = 0,\ldots,n+1$, and arcs from a literal node in $\bar{L}_i$ to a literal node in L_{i+1}, for $i = 0,\ldots,n$. The last arc in any literal path connects a literal node in $\bar{L}_{n+1}$ to the destination node. These $2m$ literal paths are node disjoint (except for the origin and destination nodes) and have identical travel times. We also construct a set of *truth assigner paths* between origin and destination based on the SAT instance clauses. We add an arc between the consolidation node in L_i and all truth assigner nodes in the same layer, and from each truth assigner node in L_i to a unique truth assigner node in $\bar{L}_i$, for $i = 1,\ldots,n$. We also add an arc between the truth assigner nodes in $\bar{L}_i$ and the consolidation node in L_{i+1}, for $i = 1,\ldots,n$. The construction is completed by adding an arc between the origin and the consolidation node in L_0, between consolidation nodes in L_i and $\bar{L}_i$, for $i = 0, n+1$, between consolidation nodes in $\bar{L}_0$ and L_1, and between the consolidation node in $\bar{L}_{n+1}$ and the destination node. Note that under this construction, a truth assigner path visits exactly one truth assigner arc corresponding to a literal in each clause.

To complete the instance construction, we use $|V| = 2m+1$ vehicles with unitary speed and assume $\Gamma = \emptyset$ and $\Upsilon = \emptyset$ (i.e., no arc-node conflicts). We also assume that vehicles are allowed to wait any time at any node. Arc costs are such that the travel time on arcs connecting nodes in L_i and $\bar{L}_i$ is $\delta \in (0,1)$, for $i = 0,\ldots,n+1$, whereas the travel time on literal arcs between layers $\bar{L}_i$ and L_{i+1} is one, for $i = 0,\ldots,n$. The travel time along a truth assigner path between a node in $\bar{L}_i$ and the last visited node in L_{i+1} is also one, for $i = 0,\ldots,n$. Additionally, the travel time from the origin to any node in L_0 and from nodes in $\bar{L}_{n+1}$ to the destination is one, except for the arc from the consolidation node in $\bar{L}_{n+1}$ to the destination, whose travel time is δ. With this construction, literal and truth assigner paths have travel times of $z^\ell = (n+2)(\delta+1)+1$ and $z^t = (n+2)(\delta+1)+\delta$, respectively. Note that in the

absence of conflict, the arrival time at any literal or truth assigner node in any layer is the same regardless of the path taken.

We impose arc-arc conflict constraints on those arcs connecting layers L_q and $\bar{L}_q$ that belong to the literal paths of a variable and its negation, for each $q \in \{0, n+1\}$. For instance, if (i,j) and (k,l) are arcs that connect layers L_q and $\bar{L}_q$ such that (i,j) belongs to the literal path of a variable and (k,l) belongs to the literal path of its negation, then $(g,h,i,j,k,l) \in \Lambda$, for each pair of distinct vehicles g and h. We also add a conflict constraint on those arcs between layers L_q and $\bar{L}_q$ that belong to the literal path of a variable (or its negation) and those in the truth assigner path corresponding to such variable (or its negation) if it belongs to the q-th clause. For instance, suppose that arcs (i,j) and (k,l) connect layers L_q and $\bar{L}_q$, for some $q \in \{1,\ldots,n\}$. If (i,j) belongs to the literal path of a variable that is in the q-th clause and (k,l) belongs to the truth assigner path corresponding to the same variable, then $(g,h,i,j,k,l) \in \Lambda$ for each pair of distinct vehicles g and h. Moreover, there is a conflict constraint on every arc (i,j) connecting the origin and L_0 and L_q and $\bar{L}_q$, for $q = 0,\ldots,n$, such that $(g,h,i,j,i,j) \in \Lambda$, for each pair of distinct vehicles g and h. In general, if there is a conflict constraint involving arcs (i,j) and (k,l), then a vehicle cannot start traveling an arc before another vehicle finishes its trip on the other arc. An example of the DRASTC instance corresponding to the SAT instance $(x) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee \bar{z})$ is shown in Figure A.1.

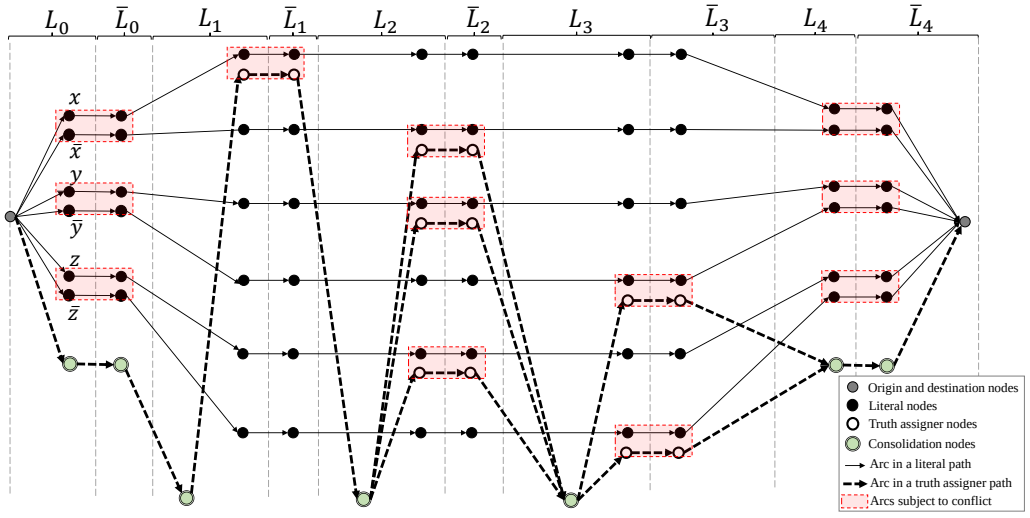


Figure A.1: RASTC Network Construction for the SAT Instance $(x) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{y} \vee \bar{z})$

Before continuing with the proof, we present the following two remarks that help clarify the instance construction. A vehicle traveling a truth assigner path visits exactly one truth assigner arc connecting layers L_q and $\bar{L}_q$, for each $q = 1,\ldots,n$. Because each of the truth assigner arcs between layers L_q and $\bar{L}_q$ is related to a literal in the q-th clause, this is equivalent to selecting the literal to satisfy in such clause. This is the reason behind the name *truth assignment* path.

Sending more than one vehicle on any path results in a travel time deviation of

more than $1 + \delta/z^\ell$. To see this, we examine two cases. If the vehicles are traveling along a truth assigner path, the conflict in the arc connecting the origin and the consolidation node in L_0 forces the second vehicle to wait 1 unit of time at the origin. This means that the earliest arrival time for the second vehicle is z^t+1, which implies a deviation from the shortest-path time of $1 + 1/z^t > 1 + \delta/z^\ell$. If the vehicles are traveling on a literal path, then the same delay makes the earliest arrival time of the second vehicle equal to $z^\ell+1$, which corresponds to a deviation of $1+1/z^\ell > 1+\delta/z^\ell$.

Now, we complete the proof by proving the following propositions, where the target objective for DRASTC is $B = 1 + \delta/z^\ell$.

Proposition 12 *If the answer to* SAT *is "YES", then the answer to* DRASTC *is "YES".*

Proof. Using the truth assignments in the SAT instance solution, construct a feasible solution to DRASTC by assigning $2m$ vehicles to the same number of distinct literal paths and one vehicle to a truth assigner path. For each clause, select only one true literal, which exists since the answer to SAT is "YES". Vehicles traveling literal paths corresponding to the selected true literals wait δ units of time at the origin and arrive at the destination at time z^t with a deviation from their shortest-path times given by $1 + \delta/z^\ell$. To avoid conflict, the remaining vehicles traveling literal paths depart the origin at time 0 and arrive at the destination at time z^ℓ with no delay. As noted in Remark A, the vehicle assigned to a truth assigner path uses the truth assigner arcs connecting layers L_q and $\bar{L}_q$ corresponding to the selected true literal in the q-th clause, for each $q = 1, \ldots, n$. Because vehicles traveling literal paths conflicting with truth assigner arcs are delayed at the origin, the vehicle traveling the truth assigner path arrives at the destination at time z^t with no delay. As a result, this solution is feasible and the maximum deviation from each vehicle's shortest-path time is $B = 1 + \delta/z^\ell$, which means that the answer to DRASTC is "YES". ■

Proposition 13 *If answer to* SAT *is "NO", then the answer to* DMRSS-C *is "NO".*

Proof. Because the answer to SAT is "NO", then any choice of literals to satisfy (one per clause) contains both a variable and its negation. Using Remark A, this implies that any truth assigner path conflicts with literal arcs on both the path of a variable as well as its negation. Select any truth assigner path and suppose without loss of generality that such path conflicts with both literal paths of variable x and $\bar{x}$ (negation of x). Also, suppose that the conflict with the literal path of variable x occurs at Layer L_q, for some $q \in \{1, \ldots, n\}$. Due to Remark A, exactly one vehicle must be assigned to each literal path and one more to a truth assigner path. That is, the vehicle traveling along the truth assigner path conflicts with two different vehicles in two different truth assigner arcs. We examine two cases, based on which vehicle waits at the origin to avoid the conflict in literal arcs connecting layers L_0 and $\bar{L}_0$.

- Case 1: the vehicle traveling the literal path of variable x does not wait at the origin. In this case, this vehicle and that traveling the truth assigner path will arrive at the same time at the literal and truth assigner nodes in L_q. Due to conflict, one vehicle must wait δ units of time while the other reaches layer $\bar{L}_q$. If the vehicle on the truth assigner path waits, the earliest it arrives at the

destination is $z^t + \delta$, which implies a deviation of $1 + \delta/z^t > B$. Thus, the only choice is that the vehicle on the literal path waits δ units of time to avoid the conflict.

- Case 2: the vehicle traveling the literal path of variable x waits δ units of time at the origin. In this case, this vehicle arrives at the literal node in L_q δ units of time after the vehicle on the truth assigner path, avoiding the conflict in this layer.

Combining both cases and using the same analysis for the vehicle traveling the literal path of $\bar{x}$, we obtain that both vehicles traveling on literal paths must wait; one of them to avoid the conflict in layers L_0 and $\bar{L}_0$ and the other to avoid the conflict with the truth assigner path. As a result, both vehicles will arrive at the same time at the literal nodes in layer L_{n+1}. To avoid conflict, one of them must wait additional δ units of time. This means that the earliest arrival time at the destination for one of the vehicles is $z^\ell + 2\delta$, which results in a deviation of $1 + 2\delta/z^\ell > B$. This means that the answer to DMRSS-C is "NO'. ■

Because the instance transformation is polynomial and due to the results in Propositions 12 and 13, we conclude that DMRSS-C is NP-complete.

APPENDIX B

PROOFS

Proof of Proposition 1. Suppose that vehicles g and h use arcs (i,j) and (k,l), respectively, with corresponding velocity vectors $\mathbf{v}_{ij}^g$ and $\mathbf{v}_{kl}^h$, where $((i,j),(k,l)) \in \Omega$. Because δ_1 and δ_2 lead to a geographic conflict, then there exist values t_1 and t_2 such that (δ_1, t_1) and (δ_2, t_2) satisfy (2.1) and (2.2). Define $\lambda \in [0,1]$, so that $\lambda\delta_1 + (1-\lambda)\delta_2 \in [\delta_1, \delta_2]$. We claim that $(\lambda\delta_1 + (1-\lambda)\delta_2, \lambda t_1 + (1-\lambda)t_2)$ also leads to a conflict. To see this, observe that $\left\|\mathbf{x}_i + \mathbf{v}_{ij}^g(\lambda t_1 + (1-\lambda)t_2) - (\mathbf{x}_k + \mathbf{v}_{kl}^h(\lambda t_1 + (1-\lambda)t_2 - \lambda\delta_1 - (1-\lambda)\delta_2))\right\| \leq \lambda \left\|\mathbf{x}_i + \mathbf{v}_{ij}^g t_1 - (\mathbf{x}_k + \mathbf{v}_{kl}^h(t_1 - \delta_1))\right\| + (1-\lambda)\left\|\mathbf{x}_i + \mathbf{v}_{ij}^g t_2 - (\mathbf{x}_k + \mathbf{v}_{kl}^h(t_2 - \delta_2))\right\| < d$, where the first inequality follows from the triangle inequality and the second one from (2.1). Using (2.2), we have that $\max\{0, \lambda\delta_1 + (1-\lambda)\delta_2\} \leq \lambda \max\{0, \delta_1\} + (1-\lambda)\max\{0, \delta_2\} \leq \lambda t_1 + (1-\lambda)t_2$, and that $\min\{c_{ij}^g, c_{kl}^h + \lambda\delta_1 + (1-\lambda)\delta_2\} \geq \lambda \min\{c_{ij}^g, c_{kl}^h + \delta_1\} + (1-\lambda)\min\{c_{ij}^g, c_{kl}^h + \delta_2\} \geq \lambda t_1 + (1-\lambda)t_2$, proving that any difference in the departure times within the interval $[\delta_1, \delta_2]$ leads to a conflict. ■

Proof of Proposition 3. Because G^r contains at least one path from s_g to p_g, for all $g \in V$, any feasible solution to RASTC$(G^r, \Lambda, \Gamma, \Upsilon)$ is also feasible to RASTC$(G, \Lambda, \Gamma, \Upsilon)$. This means that the objective value of any solution to RASTC$(G^r, \Lambda, \Gamma, \Upsilon)$ is an upper bound to the optimal value of RASTC$(G, \Lambda, \Gamma, \Upsilon)$, proving Condition 1. Similarly, any feasible solution to RASTC$(G, \Lambda, \Gamma, \Upsilon)$ is also feasible to RASTC$(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, and therefore provides an upper bound to RASTC$(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, proving Condition 2. ■

Proof of Proposition 4. From Proposition 3 we have that $z(G^r, \Lambda, \Gamma, \Upsilon) \geq z(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, and because the optimal solution to RASTC$(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ is feasible to RASTC$(G^r, \Lambda, \Gamma, \Upsilon)$, we have that $z(G^r, \Lambda, \Gamma, \Upsilon) \leq z(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. This means that $z(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) = z(G^r, \Lambda, \Gamma, \Upsilon) \geq z(G, \Lambda, \Gamma, \Upsilon)$, where the inequality follows from Condition 1 in Proposition 3. ■

Proof of Proposition 5. From Proposition 3 we have that $z(G, \Lambda, \Gamma, \Upsilon) \geq z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. We now prove that $z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) \geq z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. Let $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ be an optimal solution to RASTC$(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. If all paths used in $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ belong to G^r, then $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ is feasible to RASTC-R$(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ because both problems are subject to the same conflict constraints and $G^r \subseteq G^a$. This means that $z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) \geq z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. Now, suppose that some paths in $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ use arcs from G^c (and possibly additional nodes to those in B). This means that at some point, any of such paths must leave and return to G^r at least once using boundary nodes. Without loss of generality, suppose that vehicle g leaves G^r at node i and returns at node j, with $i, j \in B$, $i \neq j$, and let $\mathcal{P}_{ijg}^*(G^c)$ be the path used in $(\mathbf{x}^*, \boldsymbol{\tau}^*)$. Consider the following two cases.

- **Case 1:** $\mathcal{P}_{ijg}^*(G^c)$ is a shortest-path between i and j with no waiting time at any non-boundary node.
- **Case 2:** $\mathcal{P}_{ijg}^*(G^c)$ is not a shortest-path between i and j or it is optimal to wait at any non-boundary node along the path.

Using these cases, we construct a feasible solution to RASTC-R$(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$ based on $(\mathbf{x}^*, \boldsymbol{\tau}^*)$. In this solution, each vehicle uses the same path and schedules in $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ for travel segments in G^r. For each pair of boundary nodes i and j used to leave and return to G^r, vehicle g uses $\mathcal{P}_{ijg}^1(G^a)$ if the condition in Case 1 is satisfied.

This is feasible because G^a either contains $\mathcal{P}^*_{ijg}(G^c)$ or an alternative path with the same travel time. If conditions in Case 2 are satisfied, vehicle g uses $\mathcal{P}^2_{ijg}(G^a)$ with waiting time at a non-boundary node equal to the difference between the travel time in $\mathcal{P}^*_{ijg}(G^c)$ and the travel time in $\mathcal{P}^2_{ijg}(G^a)$. The analysis for every vehicle using arcs in G^c is equivalent and the resulting solution is feasible because the maximum waiting time and conflict constraints are relaxed in RASTC-R for network elements in $G^a \setminus G^r$. This solution has an objective function equal to $z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, which implies that $z(G, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) \geq z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$. ■

Proof of Proposition 6. For each vehicle $g \in V$, $c(\mathcal{P}_{s_g,i,g}(G)) + c(\mathcal{P}_{ijg}(G^c)) + c(\mathcal{P}_{j,p_g,g}(G))$ represents the travel time on a path from s_g to p_g. Using Constraint (2.16) and Proposition 4, we have that in any optimal solution $\tau^g_{p_g} \leq z^g_{SP} z^* \leq z^g_{SP} UB \leq c(\mathcal{P}_{s_g,i,g}(G)) + c(\mathcal{P}_{ijg}(G^c)) + c(\mathcal{P}_{j,p_g,g}(G))$, for all $g \in V$, where $\tau^g_{p_g}$ is the arrival time of g at its destination node p_g. This means that the travel time along path $\mathcal{P}_{s_g,i,g}(G) \cup \mathcal{P}_{ijg}(G^c) \cup \mathcal{P}_{j,p_g,g}(G)$ is not shorter than in the current feasible paths for any vehicle. ■

Proof of Proposition 7. To prove correctness, suppose for a contradiction that an optimal solution $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ exists with objective function value z^*, and that Algorithm 1 stops after k iterations with a solution $(\bar{\mathbf{x}}, \bar{\boldsymbol{\tau}})$ and objective $\bar{z} > z^*$. We examine two cases. If $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ only uses elements in G^r, then it is feasible for RASTC$(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, the last upper bound problem solved in Line 5. This, however, implies that $z^* \geq z(G^r, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) = UB_k \geq \bar{z}$, which contradicts $\bar{z} > z^*$. If $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ uses some elements that are not in G^r, then some vehicle paths leave and return G^r using boundary nodes. By construction of G^a, either $(\mathbf{x}^*, \boldsymbol{\tau}^*)$ is feasible to RASTC-R$(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon})$, the last lower bound problem solved in Line 8, or there is an equivalent solution with the same objective function value. This implies that $z^* \geq z_R(G^a, \bar{\Lambda}, \bar{\Gamma}, \bar{\Upsilon}) = LB_k = \bar{z}$, which also contradicts $\bar{z} > z^*$. To prove finite termination, note that if G^r is not augmented in Line 8 or if G^r is continuously augmented in Line 8 until $G^a = G^r = G$, then the stopping condition in Line 13 is satisfied. ■