DECOMPOSITION ALGORITHMS FOR CERTAIN INTEGER PROBLEMS OVER NETWORKS

A Dissertation
Presented to
The Academic Faculty

By

Yijiang Li

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

August 2023

DECOMPOSITION ALGORITHMS FOR CERTAIN INTEGER PROBLEMS OVER NETWORKS

Thesis committee:

Dr. Nikolaos Sahinidis
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. John-Paul Clarke
Cockrell School of Engineering
The University of Texas at Austin

Dr. Santanu Dey
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Alejandro Toriello
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Diego Cifuentes
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date approved: July 20 2023

All theory depends on assumptions which are not quite true. That is what makes it theory.

Robert M. Solow

To my family

ACKNOWLEDGMENTS

There are many people to whom I owe many thanks for their guidance, help, love, and support for the past five years. First and foremost, I would like to express my deepest gratitude to my advisors, Nikolaos Sahinidis and Santanu Dey. I have had the great privilege to spend a lot time working with and learning from them who have always been supportive to me. I especially enjoyed their passion and intellectual depth during our meetings and their lighthearted sense of humors outside of work. Special thanks should go to John-Paul Clarke. Professor Clarke guided me into the realm of academic research and offered his support and guidance that began in my undergraduate study and continued throughout my PhD journey. I would also like to thank Diego Cifuentes and Alejandro Toriello for serving as my committee members and providing valuable feedback and suggestions to this thesis.

I am grateful to the financial support from the U.S. Department of Energy for my PhD study and to my collaborators Naresh Susarla, Markus Drouven, and Miguel Zamarripa for many insightful discussions that have provided me with fresh perspectives and expanded my understanding. I am grateful to my mentors and colleagues during my internships at Argonne National Laboratory and Amazon. In particular, I thank Sven Leyffer, Kibaek Kim, Matt Menickelly, Yuan Li, Xiaoyan Si, Semih Atakan, Louis Faugère, and Ye Chun.

I would like to thank my fellow graduate students who brought fun and inspirations to my life, to name a few, Minas Chatzos, Prakirt Jhunjhunwala, Sajad Kohdadadian, Anatoliy Kuznetsov, Chungjae Lee, Yan Li, Shancong Mou, Jiachen Shi, Yuyang Shi, Hairong Wang, Zilong Wang, Haotian Wu, Enpeng Yuan, Shaowu Yuchi, and Keyu Zhu. I would also like to thank Zaiwei Chen, Chenghao Duan, Scott Guan, Yiguo Liu, Ke Pan, and Ci Song with whom I have had numerous get-togethers and trips with laughter and joy. Last but not least, I would like to thank my friends, Hongzhao Guan, Kim Huang, Chenkai Shao, Pat Wang, Richard Wang, Robert Xu, and Jimmy Zhang for their accompany in the best and worst of times for more than ten years.

I wanted to thank my parents for their endless love and support for every success and setback I have gone through and every decision I made. Thank you for all the encouragements and inspirations in all of my pursuits and dreams. I am deeply indebted to my wife, Starry, for her unconditional love and understanding. Her abundant knowledge in history, art and sociology has sparked between us numerous long yet fruitful conversations. Our shared passions in travel, culinary experiences and museum-going have enriched my life outside academic research with pleasures and cherished memories.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Integer optimization stands as a fundamental and widely embraced tool in addressing many real-world problems. Among the abundant applications of integer optimization, numerous network-related problems arise and exhibit significant influence in many areas such as transportation, energy systems, and supply chain management. The intricate nature of these problems often results in complicated large-scale formulations that are computationally expensive to solve directly. Instead, decomposition is a more computationally tractable means in many cases. In this thesis, we focus on a few integer problems over networks and relevant decomposition algorithms.

First, we investigate the airport flight-to-gate assignment problem, where the goal is to minimize the total delays by optimally assigning each scheduled flight to a compatible gate. We provide a column generation approach for solving this problem. Specifically, we use a set covering formulation for the master problem and decompose the pricing problem such that each gate is the basis for an independent pricing problem to be solved for assignment patterns with negative reduced costs. We use a combination of an approximation algorithm based on the submodularity of the underlying set and dynamic programming algorithm to solve the independent pricing problems. We show that the dynamic programming algorithm is pseudo-polynomial under the special cases of integer inputs. We also design and employ a rolling horizon method and block decomposition algorithm to solve the large-sized instances. Finally, we perform extensive computational experiments using both synthetic and real-world operational data from Atlanta Hartsfield-Jackson International Airport to validate the performance of our approach.

Second, we focus on the gas network design problem. Gas networks are used to transport natural gas, which is an important resource for both residential and industrial customers throughout the world. The gas network design problem is a challenging nonlinear and non-convex optimization problem. Additional challenges arise due to the binary vari-

ables needed to model the pipe sizing and status of certain network components. We propose a decomposition framework that separates the binary variables from the challenging nonlinear and non-convex constraints. In particular, we utilize a two-stage procedure that involves a convex reformulation of the original problem that solves for the binary variables and iteratively checks whether the set of values of binary variables produce a set of feasible flows and potentials. Finally, we conduct experiments on the Gaslib-582 network, a publicly available benchmark instance, to validate and analyze the performance of our framework.

Third, we consider the water network design and operation problems in one single problem. In general, water network problems can be divided into two categories, the design problem and the operation problem. The design problem considers pipe sizing and placements of pump stations, while the operation problem is commonly multiple time period problem that accounts for temporal changes in supply and demand, and considers the scheduling of the installed pump stations. We focus our attention on the so-called produced water that is co-produced from oil and gas. In solving the resulting formulation, we observe that it is more difficult to obtain a primal (feasible) solution while the dual bounds can be improved by the solver relatively easier. As a result, we propose two methods to obtain good primal (feasible) solutions. One method is based on a similar decomposition framework that is used in chapter 3 while the other method is based on time decomposition. We conduct computational experiments on a network derived from The Produced Water Optimization Initiative (PARETO) ([1]) case study.

Fourth, we study the resiliency of infrastructure networks. An infrastructure network generally consists of multiple types of infrastructure facilities that are interdependent. In the event of natural disaster, some of the infrastructure nodes can be damaged and disabled creating failures and such failures can propagate to other facilities that depend on the disabled facilities creating a cascade of failures and eventually a potential system collapse. We propose a bilevel interdiction model to study this problem of cascading failures in an

interdependent infrastructure system with a probabilistic dependency graph. We utilize a Benders type decomposition algorithm to solve the resulting formulation. Computational experiments are performed using synthetic networks with partial real-world data to validate the performance of this algorithm.

Some of the contents of the thesis can be found in [2, 3, 4, 5].

CHAPTER 1
INTRODUCTION

Optimization has been a fundamental tool in study a wide range of real-world problems across various domains, including transportation, energy systems, and supply chain management. Mixed-Integer Linear Programming (MILP) and Mixed-Integer Nonlinear Programming (MINLP) are particularly well-suited because of the discrete nature of some of the decisions involved. MILP and MINLP consider a combination of integer (binary) and continuous decision variables providing a flexible framework to capture the essences of the problems. Many applications can be effectively modeled using linear constraints. Nonetheless, there are cases where nonlinear phenomena that cannot be easily and accurately modeled by linear constraints introduce nonlinearity into the model. For instance, energy system often involves nonlinear constraints based on laws of physics or operational limits of system components. Furthermore, problems in these domains frequently revolve around networks or can be formulated based on classical network problems. For example, problems in energy system could consider the construction or expansion of natural gas or water networks and problems in transportation often relate to optimizing flows on networks.

Extensive research in the theories and development of general-purpose solvers have greatly advanced the field of solving MILP and MINLP. Various solvers demonstrate relatively strong performances in solving small-scale to medium-scale models. However, as real-world problems continue to grow in complexities, the resulting models become increasingly large-scale or even extreme-scale, characterized by large numbers of variables and constraints and the computational resources needed to explore the solution space increase exponentially, leading to excessive computational time. To address the challenges, ongoing research focuses on developing novel techniques and algorithmic improvements.

Among the diverse techniques, decomposition methods play a significant role. Decomposition involves breaking down a problem into smaller and more computationally contractable subproblems.

Formally, a MILP or a MINLP can be expressed as

$$\min f(x, z) \tag{1.0.1}$$

$$\text{s.t. } g(x, z) \leq 0 \tag{1.0.2}$$

$$x \in \mathbb{R}^m \tag{1.0.3}$$

$$z \in \mathbb{Z}^n, \tag{1.0.4}$$

where $f(x, z)$ and $g(x, z)$ are functions of the decision variables x and z. The objective function $f(\cdot, \cdot)$ and constraints $g(\cdot, \cdot)$ can be linear or nonlinear functions. x are the continuous variables while z are the integer variables. In some cases, z are restricted to be binary, i.e., $z \in \{0, 1\}^n$. [6] and [7] offer a wealth of discussions covering both the theoretical foundations and practical applications of MILP and MINLP.

The rest of the thesis is organized as follows. In chapter 2, we study the airport flight-to-gate assignment problem. A column generation scheme is proposed for problem. A combination of approximation algorithm and dynamic programming is used in solving the pricing problems. For the large-sized instances, a rolling horizon framework is implemented on top of the proposed approach. Chapters 3 considers the natural gas network design problem. To separate the nonlinearity and non-convexity from the large number of binary variables, we propose a decomposition framework in which the core component consists of a master problem that solves for the binary variables and a subproblem that checks the feasibility of the values obtained from the master problem. Chapter 4 combines the water network design and operation problems into one single problem. Two decomposition algorithms are proposed to obtain good primal (feasible) solutions. One is based on the framework introduced in chapter 3 and the other is based on time decomposition. In

chapter 5, we identify the most severe contingencies in an infrastructure network. A bilevel formulation and a Benders type decomposition algorithm are proposed. Lastly, in chapter 6, we conclude the thesis.

Note that each chapter in this thesis is self-contained. In other words, there may be overlap of notations across the chapters. We also discuss the relevant literature and contributions in each chapter.

CHAPTER 2
USING SUBMODULARITY WITHIN COLUMN GENERATION TO SOLVE THE FLIGHT-TO-GATE ASSIGNMENT PROBLEM

2.1 Introduction

2.1.1 Motivation and literature survey

Airports throughout the world have seen a rapid increase in the numbers of flights and passengers over the past decade. Not withstanding the current decline due to COVID-19, the international Air Transport Association [8] expects 7.2 billion passengers to travel in 2035, a near doubling of the 3.8 billion air travelers in 2016. The vast majority of current airport facilities, in particular airport terminals, are not sized to handle such traffic. And, although expansions in capacity have been planned for the long term, in the near future the requisite capacity expansions are unlikely to materialize. Airports will therefore see increases in delays and associated increases in the cost of delays that are similar to or in excess of the $\$1.6$ billion or 6 percent increase, from $\$26.6$ to $\$28.2$ billion, that was observed between 2017 and 2018 [9]. These expected increases in the magnitude and cost of delays can only be mitigated through improved airport operations management.

One critical area of airport operations is the optimal and efficient assignment of arriving flights to gates. The airport flight-to-gate assignment problem has been extensively studied by many researchers in both the operations research and aviation communities. Many models and algorithms have been proposed. For a detailed review of past work in this area, we refer the readers to [10] and [11], but give a brief overview of the popular objectives and the common solution methods.

There are three popular objectives to consider in the models. The first one is the maximization of passenger satisfaction levels. In particular, researchers consider walking dis-

tances and waiting or transit time as proxies for the passenger satisfaction level. A more detailed approach involves a differentiation between the transfer passengers and destination passengers. In these models, the costs associated with the assignments of two flights to any two gates reflect the distance between the two gates and the connection time between the two flights. Interested readers are referred to [12, 13, 14, 15]. The second class of objective focuses on airport and airline operations. Some papers, such as [16] and [17], consider the minimization of the number of ungated flights or equivalently the number of flights assigned to the remote gates or introduce costs only incurred when the flights are assigned to remote gates to differentiate the priorities of the flights. Moreover, [18] and [19] point out that airlines likely have preferences over which set of gates to park their flights leading to the consideration of maximizing total gate preference scores. The third popular objective is to improve the robustness of the solution to schedule variations. Due to the uncertain nature of the air transportation system, arrival times and departure times are likely to be stochastic subject to various factors such as weather and maintenance. Early arrival and late departure may cause the arriving flight to wait for the departing flight at the assigned gate to push back. This phenomenon is referred to as a gate conflict. Other studies consider robustness with respect to minimizing the expected gate conflict time. In the event of major disruptions such as severe weather or maintenance delays, airport has to be able to quickly recover from the deviations from the original arrival schedules and adjust the flight-to-gate assignments to reduce any potential delays. Readers are referred to the work of [20] and [21]. Since the above mentioned three objectives are all very important to take into account and neither of the objectives is superior than the other two, some combinations of them are also explored. The weighted sum method is more common in such line of considerations, but a pareto-based approach has been utilized as well. The papers [22] and [23] are examples of studies that use weighted sum approach while [24] approaches this problem with a pareto local search method.

Due to the general $\mathcal{NP}$-hard nature of the problem, it can be extremely difficult to solve

the problems directly and exactly. Many solution algorithms and heuristics have been proposed. In much of the work to date, an integer programming or mixed integer programming formulation is first presented and solver is called to obtain a solution and validate the model. These formulations are large and complex because of the many complicated constraints required to reflect real world phenomena. Consequently, it is very time-consuming if not impossible to obtain solutions within reasonable amount of time and it is challenging to improve these solution methodologies. Readers are referred to [21] and [25] for the details of these models. In addition to integer programming or mixed integer programming approach, [26] uses a stochastic optimization model to capture the uncertain nature of the problem and study the occurrence of gate conflicts. The paper [27] uses a column generation method together with a branch-and-bound algorithm to obtain integral solutions. They give a description of the column generation method and a general strategy to apply the branch-and-bound algorithm. However, they have not given any explicit mathematical formulations for the column generation scheme and they have not discussed any strategies or methods to tackle the pricing problems. Due to the limitations/challenges presented in using exact formulations for this problem, heuristics are extensively used. Popular heuristics such as genetic algorithms and Tabu search algorithms have been explored. Tabu search algorithm performs a local search around the current solution and prohibits searching at previous search points by keeping a list of those points for a few iterations while genetic algorithm is inspired from the evolutionary idea where a chromosome of higher fitness is more likely to survive and it involves operations of selections, crossover, and mutations. For an implementation of these two metaheuristics, readers are referred to [15]. Moreover, [28] implemented a MIP-based heuristic which is adapted from many popular heuristic and they show that the adaptation gives a good performance. Lastly, there are some other heuristics based on neighborhood search, breakout local search, Fuzzy Bee Colony optimization (FBCO), and particle swarm optimization (PSO). Readers are referred to [29, 30, 31] for these heuristics.

2.1.2 Contributions of this chapter

As we discussed previously, there are gaps in both the formulations and solution methodologies of the flight-to-gate assignment problem that need to be addressed. In this chapter, we focus on the airport operations and aim to minimize the arrival delays over all the flights. We propose an exact solution method that is capable of obtaining the flight-to-gate assignments at busy hub airports where a large number of flights are seen on a daily basis. Our main contributions are summarized below.

1. **Column generation formulation.** We present an explicit column generation formulation with a set covering master problem and we decompose the pricing problem into independent problems to which we apply both heuristic and exact algorithms to obtain favorable assignment patterns. Such decomposition allows us to simplify the pricing problem formulations and reduce the size of the individual pricing problem significantly. Experiments show that this formulation is far more efficient than a conventional compact mixed integer programming formulation.

2. **Pricing problem algorithms.** We explore a few approximation methods and exact algorithms to efficiently solve individual pricing problems after the decomposition. In particular,

    - We show and exploit the submodularity of the objective functions of the pricing problems to design an approximation algorithm. To the best of our knowledge, this is the first use of submodularity to efficiently solve the pricing problems in a column generation setting.
    - We derive an exact dynamic programming algorithm based on a recursive formula and show that the direct implementation of the algorithm works reasonably well in practice. We show the algorithm to be pseudo-polynomial in the special case of integer inputs and present a new heuristic based on this observation.

- We propose a block decomposition heuristic and prove it is a 2-approximation algorithm for large-sized instances.

The structure of the rest of this chapter is as follows. In Section 2.2, we provide a description of the flight-to-gate assignment problem. In Section 2.3, we provide the formulation for the column generation algorithm, and the exact algorithms and heuristics used for solving the pricing problems are discussed in Section 2.4. Specifically, a submodular maximization approximation algorithm is described in Subsection 2.4.2 and a dynamic programming algorithm with complexity analyses is presented in Subsection 2.4.3. In Subsection 2.4.5, we present a new block decomposition approximation for large-sized instances. We discuss the branching scheme and our method for retrieving integer feasible solutions in Section 2.5. Results of the computational experiments are presented in Section 2.6. Finally, we make some concluding remarks and discuss possible extensions of this work in Section 2.7 .

2.2 Problem setup

The flight-to-gate assignment problem may be described as follows. Let the set of flights be $\mathcal{F}$ and the set of gates be $\mathcal{G}$. Each flight $i \in \mathcal{F}$ has an arrival time, a_i, an airline, and an aircraft type which determines its minimum turn time, τ_i, the minimum duration for which the flight must remain after parking at its assigned gate before being ready for departure. Each gate $k \in \mathcal{G}$ has a buffer time, b_k, that has to be observed between consecutive flights, and a set of allowable aircraft types. The solution of the problem gives an assignment in which each flight is assigned to one gate. For each flight i, we define its arrival delay as the difference between its arrival time, a_i and the park time at its assigned gate, t_i^g. The decisions are the gate assignment, x_{ik}, the park time, t_i^g, and the push back time, t_i^p, for $i \in \mathcal{F}$ and $k \in \mathcal{G}$. In particular, the decision variables x_{ik} are equal to 1 if flight i is assigned to gate k and otherwise equal to 0. In addition, we also consider the notion of compatibility, α_{ik}, where a flight can only be assigned to a compatible gate. In

particular, α_{ik} equal to 1 if flight i is compatible to gate k and equal to 0 otherwise. The compatibilities are determined by various factors including gate types (heavy or regular), flight types (heavy or regular), and airline preferences. Two additional conditions have to be met for an assignment to be valid:

1. A flight occupies its assigned gate for a duration of at least its minimum turn time, τ_i, for $i \in \mathcal{F}$.

2. A pair of flights that are assigned to the same gate cannot occupy the gate at the same time and the buffer time, b_k, must be observed before the gate becomes ready for the next flight.

We further assume independence of the gates which exclude interference between the gates. This assumptions can be justified from a practical perspective. Note first that we are only considering arrival delays in this chapter. The current practices of many airports is that pushbacks of the departing flights are coordinated by the ramp towers. Whenever a pushback of a flight is in the way of an arriving flight into its assigned gate, the pushback is delayed. Note that such delays are rare in many of the large airport where there are multiple taxi ways in the ramp area to minimize blockage of arriving flights by the departing flights. Consequently, what affects the park time of an arriving flight most is its assigned gate and the departing flight at the assigned gate. Based on the above descriptions, we give a compact mixed integer programming formulation below,

$$\min \sum_{i \in \mathcal{F}} (t_i^g - a_i) \tag{2.2.1}$$

$$\text{subject to} \sum_{k \in \mathcal{G}} x_{ik} = 1, \ \forall i \in \mathcal{F} \tag{2.2.2}$$

$$t_i^g + \tau_i \leq t_i^p, \ \forall i \in \mathcal{F} \tag{2.2.3}$$

$$t_i^g \geq a_i, \ \forall i \in \mathcal{F} \tag{2.2.4}$$

$$t_i^p + b_k - t_j^g \leq M(2 - x_{ik} - x_{jk}), \ i < j, \ \forall i, j \in \mathcal{F}, \ \forall k \in \mathcal{G} \tag{2.2.5}$$

$$x_{ik} \leq \alpha_{ik},\ \forall i \in \mathcal{F}, j \in \mathcal{G} \tag{2.2.6}$$

$$x_{ik} \in \{0,1\},\ \forall i \in \mathcal{F}, k \in \mathcal{G} \tag{2.2.7}$$

$$t_i^g, t_i^p \geq 0,\ \forall i \in \mathcal{F}, \tag{2.2.8}$$

where M is a sufficiently large constant. The expression in (2.2.1) gives the objective function which minimizes the total arrival delays. Constraint (2.2.2) ensures that each flight is assigned to exactly one gate. Constraint (2.2.3) ensures the flight parks at a gate for at least a duration of the minimum turn time (condition 1 above). Constraint (2.2.4) ensures that the park time, t_i^g, is no earlier than the arrival time, a_i. We want to emphasize that we do not impose an upper bound on the park times, t_i^g, since it is possible that an arriving flight has to wait at its assigned gate for a long period of time for the departing flight that is currently occupying the gate to pushback due to various reasons, such as weather, technical difficulties, or security reasons, although the chance of such long wait is slim. In addition, imposing such upper bounds on the park times can lead to infeasibility sometimes. Condition 2 above leads to constraint (2.2.5) which observes the buffer time, b_k. It considers all pairs of flights $i < j$. When they are assigned to the same gate, the difference between t_i^p and t_j^g must be at least the buffer time of the gate, b_k. Constraint (2.2.6) is the compatibility constraint to ensure the flights are only assigned to the compatible gates. Constraints (2.2.7) and (2.2.8) are binary and non-negativity requirements on the decision variables respectively. We will show in computational experiments that this compact formulation is not ideal to obtain the desired flight-to-gate assignment efficiently. We instead propose a column generation approach to solve the problem. Note also that this setup is in fact similar to what airlines do for their flight-to-gate assignments in the real world. Interested readers are referred to [32].

2.3 Column generation formulation

2.3.1 Master problem consideration

Constraint (2.2.2) in the compact mixed integer programming formulation suggests a set partitioning master problem should be used in the column generation formulation. However, as many papers, such as [33] and [34], have pointed out, a set covering master problem formulation is numerically more stable when solving its linear programming relaxation compared to a set partitioning master problem. Therefore, we present a set covering master problem formulation where the constraint in which each flight is assigned to exactly one gate is replaced by a constraint in which each flight is assigned to at least one gate. Since this gives a relaxation and we are minimizing the total arrival delay that is non-increasing after removing a flight from a gate, the set covering master problem either produces a set partitioning assignment or a set covering assignment in which we can fix each flight with multiple assignments to one of its assigned gates and recover a set partitioning assignment that has a total arrival delay at most as large as that of the original set covering assignment.

2.3.2 The set covering master problem

We define the following parameters:

$$
\begin{aligned}
P_k &:= \text{ the set of all feasible assignment patterns for gate } k \\
\delta_{ip}^k &:= \begin{cases} 1 & \text{if flight } i \text{ is assigned on pattern } p \in P_k \text{ of gate } k \\ 0 & \text{otherwise} \end{cases} \\
c_p^k &:= \text{ the arrival delay of pattern } p \in P_k.
\end{aligned}
$$

Note that δ_{ip}^k and c_p^k are constants that can be computed for a given assignment pattern. The decision variables z_p^k $(k \in \mathcal{G}, p \in P_k)$ are equal to 1 if pattern p of gate k is used and 0

otherwise. Then the set covering master problem is given by

$$\min \sum_{k\in\mathcal{G}} \sum_{p\in P_k} c_p^k z_p^k \tag{2.3.1}$$

$$\text{subject to} \sum_{k\in\mathcal{G}} \sum_{p\in P_k} \delta_{ip}^k z_p^k \geq 1,\ \forall i \in \mathcal{F} \quad (\pi_i) \tag{2.3.2}$$

$$\sum_{p\in P_k} z_p^k = 1,\ \forall k \in \mathcal{G} \quad (\mu_k) \tag{2.3.3}$$

$$z_p^k \in \{0,1\},\ \forall k \in \mathcal{G}, p \in P_k. \tag{2.3.4}$$

The objective (2.3.1) is an expression for total arrival delays by summing the arrival delays of all patterns over all gates. Constraint (2.3.2) is the cover constraint that ensures each flight is assigned to at least one gate. Constraint (2.3.3) is the availability constraint that ensures only one pattern of each gate is selected. Last constraint (2.3.4) is the binary requirements on the decision variables z_p^k $(k \in \mathcal{G}, p \in P_k)$ and relaxing these constraints to $0 \leq z_p^k \leq 1$ gives the linear programming relaxation of the set covering master problem. The linear programming relaxation is then solved in each iteration.

2.3.3 The pricing problems

In the pricing problem, we construct feasible assignment patterns for each gate. If we associate dual variables π_i with (2.3.2) and μ_k with (2.3.3), the reduced cost $\bar{c}_p^k$ $(k \in \mathcal{G}, p \in P_k)$ of a pattern p of gate k is given by

$$\bar{c}_p^k = c_p^k - \sum_{i\in\mathcal{F}} \delta_{ip}^k \pi_i - \mu_k. \tag{2.3.5}$$

Given a feasible solution z to the linear programming relaxation of (2.3.2) - (2.3.4), we know that z is optimal if and only if the optimal assignment patterns generated from the

pricing problem have non-negative reduced costs. Then the pricing problem is given by

$$\min\{c_p^k - \sum_{i\in\mathcal{F}} \delta_{ip}^k \pi_i - \mu_k : k \in \mathcal{G}, p \in P_k\}. \tag{2.3.6}$$

Since we assume each gate is independent, we can further decompose the pricing problem into $|\mathcal{G}|$ independent pricing problems, one for each gate. The k^{th} pricing problem is given by

$$\min\{c_p^k - \sum_{i\in\mathcal{F}} \delta_{ip}^k \pi_i - \mu_k : p \in P_k\}. \tag{2.3.7}$$

For the decision variables, similar to the compact formulation (2.2.1) - (2.2.8), we use binary decision variables x_{ik} which are equal to 1 if flight i is assigned at gate k and equal to 0 otherwise, and continuous variables t_i^g and t_i^p for the park times and push back times of flight i respectively. Recall the definitions of the constants δ_{ip}^k and c_p^k which are given by

$$\delta_{ip}^k := \begin{cases} 1 & \text{if flight } i \text{ is assigned on pattern } p \in P_k \text{ of gate } k \\ 0 & \text{otherwise} \end{cases}$$

$$c_p^k := \text{ the arrival delay of pattern } p \in P_k.$$

When the pricing problems are solved to generate a new assignment pattern p, the values of the constants, δ_{ip}^k, are determined using the values of the decision variables x_{ik} and c_p^k is the sum of all individual arrival delay, $t_i^g - a_i$. Let M denote a large constant, then the k^{th} pricing problem is given by

$$\min \sum_{i\in\mathcal{F}} (t_i^g - a_i) - \sum_{i\in\mathcal{F}} x_{ik}\pi_i - \mu_k \tag{2.3.8}$$

$$\text{subject to } t_i^g + \tau_i \le t_i^p, \ \forall i \in \mathcal{F} \tag{2.3.9}$$

$$t_i^g \ge a_i, \ \forall i \in \mathcal{F} \tag{2.3.10}$$

$$t_i^p + b_k - t_j^g \leq M(2 - x_{ik} - x_{jk}),\ \forall i < j, \quad i, j \in \mathcal{F} \tag{2.3.11}$$

$$x_{ik} \leq \alpha_{ik},\ \forall i \in \mathcal{F}, k \in \mathcal{G} \tag{2.3.12}$$

$$x_{ik} \in \{0, 1\},\ \forall i \in \mathcal{F},\ k \in \mathcal{G} \tag{2.3.13}$$

$$t_i^g, t_i^p \geq 0,\ \forall i \in \mathcal{F}. \tag{2.3.14}$$

The objective function (2.3.8) is obtained by expressing c_p^k explicitly in decision variables t_i^g and parameters a_i, and substituting the parameters δ_{ip}^k by the decision variables x_{ik}. Constraints (2.3.9) - (2.3.14) follow directly from the constraints (2.2.3) - (2.2.8) in the compact mixed integer programming formulation. However, we do not need constraint (2.2.2). For flights that are not assigned to this gate, we assume their arrival delays to be zero.

The solutions x_{ik} from a pricing problem form a potential assignment pattern. If the corresponding reduced cost given by the objective value is negative, then this assignment pattern is favorable and added to the master problem.

We want to point out that μ_k is a constant in the k^{th} pricing problem and thus dropping this term we can equivalently solve (2.3.8) - (2.3.14) by solving the following maximization problem:

$$\max\ -\sum_{i \in \mathcal{F}} (t_i^g - a_i) + \sum_{i \in \mathcal{F}} x_{ik}\pi_i \tag{2.3.15}$$

$$\text{subject to}\ \ (2.3.9) - (2.3.14).$$

This equivalent problem is intuitively easier to interpret as we may consider π_i as the benefits of accepting flight i at the cost of an arrival delay $t_i^g - a_i$. Consequently, we are attempting to maximize the total net benefits over all flights. We refer to this maximization problem as the pricing problem for any gate in all following sections and use the terms total net benefits and reduced cost interchangeably.

2.4 Solving the pricing problem

2.4.1 Pre-processing

The decomposition of the pricing problem into smaller pricing problems allows additional pre-processing of the set of flights $\mathcal{F}$ for each gate. Consider a particular gate k, we can construct a subset of flights $\mathcal{F}' \subseteq \mathcal{F}$ to include only compatible flights and flights with positive dual variables π_i. To see this fixing is correct, the incompatible flights are not allowed to be assigned to this gate while accepting the flights with $\pi_i \leq 0$ does not contribute any benefit at the cost of arrival delays of other flights. For the rest of the discussions in this section, we use the subset of flights $\mathcal{F}'$ and assume $|\mathcal{F}'| = n$. Note that for different gates, the set $\mathcal{F}'$ could be different.

2.4.2 Submodular approximation algorithm

Submodularity is a well studied property of set functions and there are many existing algorithms that can approximate the task of maximizing a submodular function very efficiently and effectively. We notice that, in the context of the flight-to-gate assignment problem, the submodularity of the objective function (2.3.15) is very intuitive. Consider any gate and given a set of accepted flights at this gate, the impact of accepting an additional flight on the total arrival delay of this set of flights is at least as large as that of accepting the additional flight with a subset of those flights.

Formally consider the pricing problem for gate k and recall that our objective after pre-processing the set of flights is

$$\max \ -\sum_{i\in\mathcal{F}'}(t_i^g - a_i) + \sum_{i\in\mathcal{F}'} x_{ip}\pi_i.$$

For simplicity, we denote $t_i^g - a_i$ by $\triangle t_i^g$ and clearly $\triangle t_i^g \geq 0$. We define a function

$f : 2^{\mathcal{F}'} \mapsto \mathbb{R}$ as follows,

$$f(A) = -\sum_{i \in A} \triangle t_i^{g,A} + \sum_{i \in A} \pi_i. \tag{2.4.1}$$

We further denote the variables $\triangle t_i^g$ under a set of assigned flights A by an additional superscript A and assume that the values of $\triangle t_i^{g,A}$ for $i \notin A$ to be zero. By the definition of f, we have that $f(\emptyset) = 0$. For a given set of assigned flights A, the constraints (2.3.9) - (2.3.11) can be incorporated when evaluating $f(A)$. Specifically, let the flights in the set A be $1, 2, \ldots |A|$ and the total benefits $\sum_{i \in A} \pi_i$ can be computed. What remains is to compute the arrival delays. Note that the first flight parks at the gate at its arrival time, a_1, and pushes back after a duration of its minimum turn time at $a_1 + \tau_1$. The gate becomes available for the second flight after a duration of the buffer time at $a_1 + \tau_1 + b_k$. Consequently, an arrival delay of $a_1 + \tau_1 + b_k - a_2$ on the second flight is incurred if the second flight arrives earlier than $a_1 + \tau_1 + b_k$ and it parks at $a_1 + \tau_1 + b_k$. Otherwise, there is no arrival delay on the second flight and it parks at its arrival time, a_2. The same procedure can be applied to the rest of flights in the set A to compute $f(A)$.

Note that since the subset of flights $\mathcal{F}'$ are all compatible to this gate and the above procedure incorporates constraints (2.3.9) - (2.3.11), we can pose the pricing problems as the following equivalent unconstrained maximization problem

$$\max\{f(A) : A \subseteq 2^{\mathcal{F}'}\}. \tag{subMax}$$

In order to show the submodularity of the function f, we need the following observation about the variables $\triangle t_i^g$.

Lemma 2.4.1. *If $A \subseteq B \subseteq \mathcal{F}'$, then*

$$\sum_{i \in B} \triangle t_i^{g,B} \geq \sum_{i \in A} \triangle t_i^{g,A}.$$

proof of Lemma 2.4.1. We can rewrite the left hand side expressions into

$$\begin{aligned}\sum_{i\in B} \triangle t_i^{g,B} &= \sum_{i\in A} \triangle t_i^{g,B} + \sum_{i\in B\setminus A} \triangle t_i^{g,B} \\ &\geq \sum_{i\in A} \triangle t_i^{g,A} + \sum_{i\in B\setminus A} \triangle t_i^{g,B}.\end{aligned}$$

The last inequality is valid because $A \subseteq B$, for any $i \in A$, thus the arrival delay of i in the set of assigned flights B of larger cardinality is at least as large as its arrival delay in the set of assigned flights $A \subseteq B$. Additionally, since $\triangle t_i^{g,B} \geq 0$, we have the desired inequality. □

We next show that f is a submodular function.

Lemma 2.4.2. *f is a submodular set function defined on $\mathcal{F}'$.*

proof of Lemma 2.4.2. First note that we use the expression "the park time of a flight is pushed to the right" to mean the park time of this flight is delayed and consequently this flight experiences a larger arrival delay. Formally, given a set of flights A, the park time of flight i is pushed to the right in another set of flight B if $\Delta t_i^{g,A} \leq \Delta t_i^{g,B}$.

Now suppose that $A \subseteq B \subseteq \mathcal{F}'$ and $u \notin \mathcal{F}'\setminus B$, we compare $f(A\cup\{u\}) - f(A)$ against $f(B \cup \{u\}) - f(B)$,

$$\begin{aligned}f(A \cup \{u\}) - f(A) &= -\sum_{i\in A\cup\{u\}} \triangle t_i^{g,A\cup\{u\}} + \sum_{i\in A\cup\{u\}} \pi_i - \left(-\sum_{i\in A} \triangle t_i^{g,A} + \sum_{i\in A} \pi_i\right) \\ &= -\left(\sum_{i\in A} \triangle t_i^{g,A\cup\{u\}} - \sum_{i\in A} \triangle t_i^{g,A}\right) + (\pi_u - \triangle t_u^{g,A\cup\{u\}}). \quad (2.4.2)\end{aligned}$$

Similarly, we have that

$$f(B \cup \{u\}) - f(B) = -\left(\sum_{i\in B} \triangle t_i^{g,B\cup\{u\}} - \sum_{i\in B} \triangle t_i^{g,B}\right) + (\pi_u - \triangle t_u^{g,B\cup\{u\}}). \quad (2.4.3)$$

Since $A \subseteq B$, the arrival delay of u in the set of assigned flights B is at least as large as that of u in the set of assigned flights A and then $\triangle t_u^{g,A\cup\{u\}} \leq \triangle t_u^{g,B\cup\{u\}}$. Thus $\pi_u - \triangle t_u^{g,A\cup\{u\}} \geq \pi_u - \triangle t_u^{g,B\cup\{u\}}$. For the first term in the expressions (2.4.2) and (2.4.3), we consider the following cases when accepting an additional flight u to A and B:

1. The park times remain unchanged for all $i \in B$ and thus $i \in A$. Both terms are zero. We have $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$.

2. The park times are pushed to the right for a set of flights $I \subseteq B$. If $I \cap A \neq \emptyset$, following similar arguments for u, we see that the increase in the arrival delay of a flight $i \in I \cap A$ in the set of assigned flights B is at least as large as that of i in the set of assigned flights A. In addition, there are potentially increases in arrival delays of flights in the set $I \cap A^c$. Therefore, the first term in (2.4.2) is less negative than that in (2.4.3). If $I \cap A = \emptyset$, the first term in (2.4.2) is zero while the first term in (2.4.3) is non-positive. We have again $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$.

The two cases above verify that when we accept an additional flight u, we have $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ and this shows that f is a submodular function. $\square$

We want to point out that f is not monotonic in general. Suppose that $A \subseteq B \subseteq \mathcal{F}'$, then

$$\begin{aligned} f(B) - f(A) &= -\sum_{i\in B} \triangle t_i^{g,B} + \sum_{i\in B} \pi_i - \left(-\sum_{i\in A} \triangle t_i^{g,A} + \sum_{i\in A} \pi_i \right) \\ &= -\left(\sum_{i\in B} \triangle t_i^{g,B} - \sum_{i\in A} \triangle t_i^{g,A} \right) + \sum_{i\in B\setminus A} \pi_i. \end{aligned}$$

The first term $-\left(\sum_{i\in B} \triangle t_i^{g,B} - \sum_{i\in A} \triangle t_i^{g,A}\right)$ is always non-positive by Lemma 2.4.1. However, since $\pi_i \geq 0$ for all $i \in \mathcal{F}$ and $\sum_{i\in B\setminus A} \pi_i \geq 0$, $f(B) - f(A)$ is not necessarily always non-negative or non-positive and thus f is not monotone in general.

Now we can utilize an existing submodular maximization algorithm by [35] which is

shown below as algorithm 1. In this algorithm, we select each flight in the set of flights $\mathcal{F}'$ with a probability. Let X_i and Y_i be two random sets to be updated in each iteration and we initialize $X_0 = \emptyset$ and $Y_0 = \mathcal{F}'$. We also denote the elements in $\mathcal{F}'$ by $1, 2, \ldots, n$.

Algorithm 1: Submodular Maximization($f, \mathcal{F}'$)

Initialize $X_0 \leftarrow \emptyset, Y_0 \leftarrow \mathcal{F}'$
for $i = 1, 2, \ldots, n$ **do**
  $a_i \leftarrow f(X_{i-1} \cup \{i\}) - f(X_{i-1})$
  $b_i \leftarrow f(Y_{i-1} \backslash \{i\}) - f(Y_{i-1})$
  $a'_i \leftarrow \max\{a_i, 0\}, b'_i \leftarrow \max\{b_i, 0\}$
  with probability $a'_i/(a'_i + b'_i)^*$ **do:** ▷ * If $a'_i = b'_i = 0$, we assume $a'_i/(a'_i + b'_i) = 1$
  $X_i \leftarrow X_{i-1} \cup \{i\}, Y_i \leftarrow Y_{i-1}$
  else (with the compliment probability $b'_i/(b'_i + a'_i)$)
  $X_i \leftarrow X_{i-1}, Y_i \leftarrow Y_{i-1} \backslash \{i\}$
end
return X_n *(or equivalently* Y_n*)*

The following theorem establishes a theoretical approximation guarantee of Algorithm 1.

Theorem 2.4.1. *Let $f : 2^{\mathcal{F}'} \mapsto \mathbb{R}$ defined by (2.4.1), and OPT be the true optimal solution of the problem (subMax) and X_n (or equivalently Y_n) is the set returned by the algorithm. If $f(\mathcal{F}') \geq 0$, then $\mathbb{E}(f(X_n)) \geq f(OPT)/2$.*

proof of Theorem 2.4.1. The proof is a very minor modification of what is given in [35] and attached in Appendix A for a reference and completeness. The modification is since we require $f(\mathcal{F}') \geq 0$ instead of $f(A) \geq 0$ for all $A \subseteq \mathcal{F}'$. □

Although it is possible that the condition on f to obtain the theoretical guarantee is not satisfied, the assignment patterns that are generated can still have favorable total net benefits. We discuss later in the numerical experiment section about the implementation of this algorithm and its performance.

2.4.3 Dynamic programming algorithm

Approximation algorithm can be efficient in generating favorable assignment patterns, however, exact optimal solutions to the pricing problems have to be obtained to prove optimality at each node. This would typically be done using a general integer programming solver. However, this can be slow. To reduce overall solution time, we have developed and now present a much more effective dynamic programming algorithm to achieve this task. We divide up the analyses of the dynamic programming algorithm into three cases based on the input type, namely, general input, rational input, and integer input. We note that the analysis in the case of integer input is a simplification of the analysis used in the case of rational input. In addition, although in many cases, the input is not integer, the algorithm developed in that section can serve as another approximation algorithm. Therefore, we defer the details of the analysis in case of rational input to Appendix B.

The general case.

For any gate k, we define an auxiliary function $g_i(t)$ to be the maximum total net benefits from optimally accepting flights in the set $\{i, i+1, \cdots, n\}$ where the smallest indexed accepted flight from $\{i, i+1, \ldots, n\}$ can be accepted any time after time t. To give a formal definition,

$$g_i(t) = \max_{x_{ik}, x_{i+1,k}, \ldots, x_{nk}} \left\{ \sum_{j \geq i} x_{jk} \pi_j - \sum_{j \geq i} (t_j^g - a_j) \mid t_j^g \geq t, \ \forall j \in \{i, i+1, \ldots, n\} \right\}. \tag{2.4.4}$$

As an illustration, we present an example of a set of 8 arrivals and show in Figure 2.1 the functions $g_i(t)$. In particular, the left plot shows the function $g_1(t)$ while the right plot shows the function $g_8(t)$.

We observe that the optimal value of the k^{th} pricing problem is $g_1(0)$. We introduce the notion of processing time of a flight i at gate k which consists of the minimum turn time

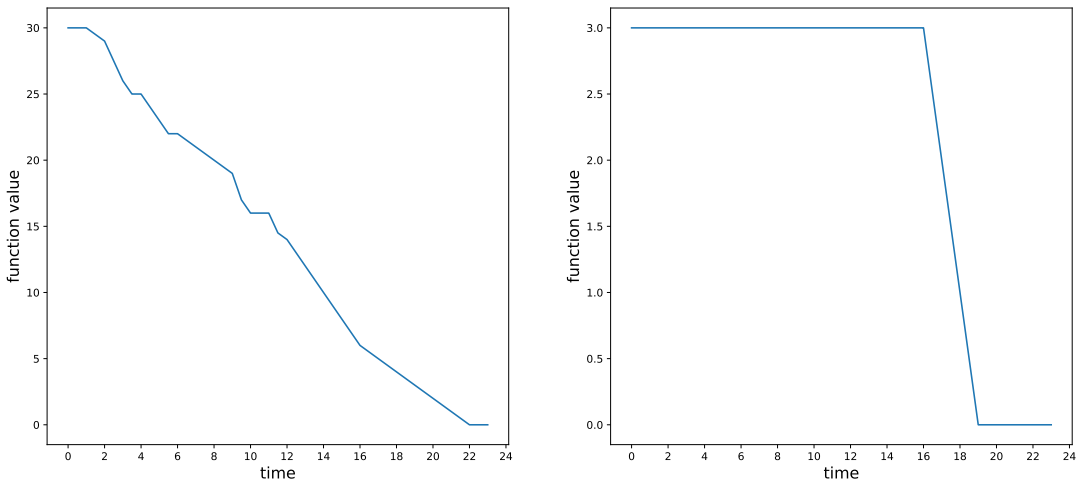

Figure 2.1: Plot of functions $g_1(t)$ and $g_8(t)$.

and the buffer time and denote the processing time of flight i by $p_i := \tau_i + b_k$. In the rest of this section, we drop the index k as we are solving the pricing problem for a fixed gate k. Based on the definition of $g_i(t)$, we can derive the following recursive formula,

Lemma 2.4.3.

$$g_i(t) = \begin{cases} 0 & \text{if } i \geq n+1 \\ g_i(a_i) & \text{if } t \leq a_i \\ g_{i+1}(t) & \text{if } t > a_i + \pi_i \\ \max\{(a_i + \pi_i - t) + g_{i+1}(t + p_i), g_{i+1}(t)\} & \text{if } a_i < t \leq a_i + \pi_i. \end{cases} \tag{2.4.5}$$

proof of Lemma 2.4.3. Similar to the proof of Lemma 2.4.2, we use the expression "the park time of a flight is pushed to the right" to mean the park time of this flight is delayed and consequently this flight experiences a larger arrival delay.

We consider each case separately.

1. This is the terminating case. If $i \geq n+1$, we do not have any flights and thus receive a zero net benefit.

2. Since a flight cannot park earlier than its arrival time, the total net benefits for any

$t \leq a_i$ are equal to $g_i(a_i)$.

3. If the current time $t > a_i + \pi_i$, accepting the flight i does not contribute to the total net benefits as $\pi_i - (t - a_i) < 0$. In addition, the park times of flights in the set $\{i+1, i+2, \cdots, n\}$ are potentially pushed to the right in the presence of flight i and their arrival delays are at least as large as the arrival delays in the absence of flight i. Consequently, we do not accept flight i at this gate. From the above analyses, we see that $a_i + \pi_i$ is the latest time beyond which we do not accept flight i. We denote this time as the end point and a_i as the start point of flight i's acceptance window.

4. In this last case, we compare the total net benefits between accepting and not accepting flight i. In the former case, if we accept flight i at the current time t, the net benefit gained from flight i is $a_i + \pi_i - t$ and earliest possible park time for flight $i+1$ is $t + p_i$. In the latter case, if we do not accept flight i, the earliest possible park time for flight $i+1$ is t.

□

A direct implementation of the formula (2.4.5) recursively considers whether to accept each flight for all flights starting with the first flight. Algorithm 2 below shows an implementation to obtain the value of $g_1(0)$. Note that since we define the function $g_i(t)$ for all continuous values of t, it was not immediately clear whether Algorithm 2 would run in finite time. Thus we verify that Algorithm 2 runs in finite time next.

Proposition 2.4.1. *For any value of i and t, Algorithm 2 runs in finite time. In particular, to evaluate $g_i(t)$ for any t, the procedure **EVALORACLE**(i, t) is recursively called $2^{(n-i+1)}$ times in the worst case.*

Proof. The proof is by backward induction. If $i = n$, evaluating the function $g_n(t)$ for any t is equivalent to the procedure EVALORACLE(i, t) with input n and t which further calls the procedure EVALORACLE(i, t) at most 2 times with input $n+1$ and t or $n+1$ and $t+p_n$.

This gives the base case. Suppose that the statement is true for $i = n, n-1, \ldots, j+1$. Now consider $i = j$ and $g_j(t)$ for any t, the procedure EVALORACLE(i, t) with input j and t further calls the procedure EVALORACLE(i, t) at most 2 times with input $j + 1$ and t or $j + 1$ and $t + p_i$. The former is equivalent to the value of $g_{j+1}(t)$ and the latter is equivalent to the value of $g_{j+1}(t + p_i)$. The inductive step shows that both call requires at most 2^{n-j} further recursive calls to the procedure EVALORACLE(i, t). Therefore, $g_i(t)$ for any t requires a total of $2 \cdot 2^{n-j} = 2^{n-j+1}$ recursive calls, which completes the proof. □

Algorithm 2: ***EvalOracle***(i, t); Evaluation of $g_i(t)$ and optimal assignment S

```
Input: i, t
if i = n + 1 then return 0, S = ∅;
else
    if t ≤ a_i then return EvalOracle(i, a_i);
    else if t ≥ a_i + π_i then return EvalOracle(i + 1, t), S ← S ∪ {0};
    else
        if (a_i + π_i − t)+EvalOracle(i + 1, t + p_i) ≥ EvalOracle(i + 1, t) then
            return (a_i + π_i − t)+EvalOracle(i + 1, t + p_i), S ← S ∪ {1}
        end
        else return EvalOracle(i + 1, t), S ← S ∪ {0};
    end
end
```

The case of integral input data.

We now propose an implementation of the dynamic programming algorithm with a running time of $O(nc)$ in the case of integral input data. In Appendix B, we analyzed the case when the data is rational and we constructed the functions $g_i(t)$ in the interval $[0, c]$ by recursively evaluating the functions at the potential breakpoints from the set

$$\{0, e, 2e, 3e, \ldots, c\} \text{ where } e \in \{1/d, 1/2d, 1/3d, \ldots, 1/(i+1)d\},$$

where d is the common denominator. If we now assume the input data are integral, we can further reduce the set to $\{0, 1, 2, \ldots, c\}$. Formally, we have the following theorem.

Theorem 2.4.2. *Assume all input data are integral, it suffices to evaluate a function $g_i(t)$ at $t \in \{0, 1, 2, \ldots, c\}$ to construct the function.*

Proof. Consider any set of accepted flights, S, and let the flights in A be $1, 2, \ldots, |A|$. Following similar arguments in the subsection 2.4.2, we can assign the park time of the first flight to be a_1 and the gate becomes available for the second flight at time $\min\{a_2, a_1 + p_1\}$. Therefore, we see that the park time for flight i is given by $\sum_{j=2}^{i-1} \min\{a_j, a_{j-1} + p_{j-1}\} + \sum_{j=2}^{i-1} p_j$ for $i \geq 3$. Note that the park time of flight i is the sum of the park time and processing time of flight $i - 1$. Since we assume all input data are integral, the park times for all flights are integral.

Next, we optimally assign the flights $1, 2, \ldots, n$ in the set $\mathcal{F}'$ based on the recursive formula (2.4.5). For any flight j, if we accept j, it can only be accepted at an integral time that is either determined by the accepted flights before j or a_j. If we do not accept j, we move on to optimally assigning the flights $j + 1, j + 2, \ldots, n$ and the arguments for j can be applied. Therefore, decisions on whether to accept the flights all occur at integral times and we only need to evaluate $g_i(t)$ at the integral times in $[0, c]$ to construct that $g_i(t)$. □

Based on the above observation, an implementation of the dynamic programming algorithm with a running time of $O(nc)$ is shown below as Algorithm 3. Note that this is an implementation of the backward version of the dynamic programming algorithm. Lastly note that in the cases where not all input data are integers, this algorithm works as an approximation algorithm and provides an alternative to the submodular maximization approximation algorithm. This new approximation algorithm is referred to as the approximative dynamic programming algorithm (ADP) in the computational experiments.

2.4.4 Alternative reformulation of the pricing problems

Alternatively, the pricing problem can be formulated as a variant of the shortest path problem with time windows known as the shortest path problem with time windows and

Algorithm 3: Alternative implementation of the dynamic programming algorithm

Let $g_i(0 \dots c)$ be a table where $i \in \{1, 2, \cdots, n+1\}$.
for $t \leftarrow c$ *to* 0 **do**
    $g_{n+1}(t) \leftarrow 0$, $S_t \leftarrow \emptyset$
end
for $i \leftarrow n$ *to* 1 **do**
    for $t \leftarrow c$ *to* 0 **do**
        if $t \leq a_i$ **then** $g_i(t) \leftarrow g_i(a_i)$, $S_t \leftarrow S_{a_i}$;
        else if $t \geq a_i + \pi_i$ **then** $g_i(t) \leftarrow g_{i+1}(t)$, $S_t \leftarrow S_t \cup \{0\}$;
        else
            if $a_i + \pi_i - t + g_{i+1}(t + p_i) \geq g_{i+1}(t)$ **then**
              $g_i(t) \leftarrow a_i + \pi_i - t + g_{i+1}(t + p_i)$, $S_t \leftarrow S_t \cup \{1\}$;
            else $g_i(t) \leftarrow g_{i+1}(t)$, $S_t \leftarrow S_t \cup \{0\}$;
        end
    end
end
return $g_1(0)$, S_0

time costs (SPPTWTC) in which there is an additional linear cost associated with the service start time at each node. Such a problem is studied in details in [36]. To formulate the pricing problems as a SPPTWTC, a source and a sink need to be introduced and each of the flights in the set $\mathcal{F}'$ represents a node in the network. The cost associated with each arc between nodes i and j is given by the negative of the corresponding dual variable value, $-\pi_{ij}$. To solve the SPPTWTC problem, a dynamic programming algorithm is proposed in [36]. This dynamic programming algorithm is derived based on the general labelling algorithm for the shortest path problems with time windows. For a detailed discussion of the labeling algorithm and some relevant extensions, we refer the readers to [37] and [38]. One of the key reasons for the effectiveness of this dynamic programming algorithm is the use of upper bounds on the service start times to eliminate labels that are infeasible with respect to the time windows, so that the number of labels created is significantly reduced. However, if we reformulate our pricing problems as SPPTWTC, the lack of upper bounds on the park times leads to an exponential number of labels rendering the dynamic programming algorithm proposed in [36] ineffective. Nonetheless, if one wanted to enforce upper

bounds on the park times, one would definitely be able to leverage the work in [37], [38], and [36]. Therefore, we did not pursue this reformulation further.

2.4.5 Large-sized instances

The total number of arrivals into a busy international airport can be very large and the proposed dynamic programming algorithm can be slow to obtain the optimal assignments. We utilize two ways to further decompose the pricing problem of each gate to tackle large-sized instances. One is a 2-approximation algorithm and the other is a standard rolling horizon method. As the standard rolling horizon method is common in many applications, we defer the details of implementations to Appendix C.

Block decomposition approximation.

When dealing with a large number of arrival flights, the flights can be usually divided into blocks based on the interactions between the flights to reduce the problem sizes. We introduce the idea of an adjacency parameter, denoted by $\overline{\sigma}$. A formal definition of $\overline{\sigma}$ is as follows.

Definition 2.4.1. *For each flight i, let j be the earliest flight after i such that $a_j > a_i + \pi_i + p_i$, and denote $\sigma_i := \min\{j - i, n - i\}$, then $\overline{\sigma} := \max_{1 \leq i \leq n} \sigma_i$.*

Note that this adjacency parameter is likely to differ across the gates and change after a new iteration of the master problem is solved with the updated set P_k. The use of the adjacency parameter also appears in the work of [39]. Now we are ready to give the block decomposition approximation algorithm.

Note that we refer to the discussions in Subsection 2.4.2 for the computation of the total net benefits $f(S)$. We now provide an approximation guarantee for Algorithm 4.

Theorem 2.4.3. *Let S^* and OPT be the true optimal assignment and optimal objective value of the pricing problem respectively and let S be the assignment returned by Algorithm 4 and OPT_S be the objective value associated with S, then $OPT_S \geq OPT/2$.*

Algorithm 4: Block decomposition approximation

1 **Input:** adjacency parameter $\overline{\sigma}$
2 Divide the set $\mathcal{F}'$ into $\lfloor n/\overline{\sigma} \rfloor + 1$ blocks such that each of block has $\overline{\sigma}$ flights and the last block has $n - \lfloor n/\overline{\sigma} \rfloor \cdot \overline{\sigma}$ flights
3 Solve for the optimal assignment in each of the blocks
4 Compute the sum of the total net benefits of the odd-indexed blocks and even-indexed blocks and let B be the collection of blocks with a larger sum of the total net benefits
5 Construct an assignment, S, by cascading the optimal assignments from the blocks in B
6 Compute the total net benefits of the assignment S, $f(S)$
7 **return** S, $f(S)$

proof of Theorem 2.4.3. Label all the blocks by $1, 2, \ldots, \lfloor n/\overline{\sigma} \rfloor, \lfloor n/\overline{\sigma} \rfloor + 1$ and let the optimal objective value of each block be OPT_i for $i \in [\lfloor n/\overline{\sigma} \rfloor + 1]$. By the construction of the blocks, we have that

$$\sum_{i \in [\lfloor n/\overline{\sigma} \rfloor + 1]} OPT_i \geq OPT. \tag{2.4.6}$$

Now without loss of generality, we assume that the sum of the total net benefits of the odd-indexed blocks is larger than that of the even-indexed blocks, then we have that

$$2 \sum_{i \text{ is odd}} OPT_i \geq \sum_{i \text{ is odd}} OPT_i + \sum_{i \text{ is even}} OPT_i \geq OPT. \tag{2.4.7}$$

Now we show that we can construct a valid assignment by just cascading the optimal assignments from the odd-indexed blocks. Let I_1, I_2 be any two consecutive odd-indexed blocks and let f_1 be the last flight in block I_1 and f_2 be the first flight in I_2 respectively. From the construction of the blocks, we have that $f_2 - f_1 > \overline{\sigma}$ since they are from two different blocks. In addition, by the definition of the adjacency parameter, we have $a_{f_2} > a_{f_1} + \pi_{f_1} + p_{f_1}$ and, in fact, $a_{f_2} > a_i + \pi_i + p_i$ for any flight i in block I_1. Recall that $a_i + \pi_i$ is the end point of flight i's acceptance window beyond which i is not accepted. Therefore, the optimal assignment of the flights in the block I_1 does not conflict with whether to accept f_2 in the optimal assignment of the flights in the block I_2 and cascading the optimal

assignments of I_1 and I_2 forms a valid assignment. Repeating this argument for all the odd-indexed blocks, we see cascading the optimal assignments of those blocks form a valid assignment pattern.

Therefore, we have that

$$OPT_S = \sum_{i \text{ is odd}} OPT_i \geq OPT/2, \tag{2.4.8}$$

which completes the proof. □

Note that Theorem 2.4.3 implies that if there exists an assignment pattern with positive total net benefits, the block decomposition approximation algorithm has to return an assignment pattern, S, with a positive OPT_S and thus S is a favorable assignment pattern. Furthermore, Algorithm 4 can be improved as follows. Suppose the odd-indexed blocks have a larger sum of the total net benefits and we construct a valid assignment pattern from the odd-indexed blocks. We can still potentially accept more flights from the even-indexed blocks to contribute more net benefits as long as accepting those flights does not violate constraints (2.3.9) - (2.3.11).

Lastly, as the adjacency parameter, $\overline{\sigma}$, is computed based on the interactions between flights, it is not likely to be very large. Most flights park at the gate to get ready for the next flight leg with a relatively short turn time and thus they have limited interactions with other flights that arrive hours later.

2.5 Feasible solutions and branching scheme

2.5.1 Feasible solutions

After optimality is achieved in the linear programming relaxation of the master problem (2.3.1) - (2.3.4), we solve the master problem with the binary requirements of the decision variables reinstated to obtain a feasible solution. We refer to this problem as the binary program. The objective value of the linear programming relaxation provides a lower bound

on the optimal objective value while the binary program provides an upper bound on the optimal objective value. Subsequently, we can study the quality of the binary program solution by the following,

$$gap_{root} = \frac{UB_{root} - LB_{root}}{LB_{root}}, \tag{2.5.1}$$

where UB_{root} is the upper bound from the binary program and LB_{root} is the lower bound from the linear programming relaxation. In the cases where LB_{root} is zero, we use the absolute gap of $UB_{root} - LB_{root}$ which is denoted by an additional notation of "(a)" after the numerical. We will use the abbreviations of UB and LB to represent the upper bound from the binary program and lower bound from the linear programming relaxation respectively in the computational experiments section.

2.5.2 Branching on the assignment decisions

In many cases, although we can obtain a feasible solution from the binary program, the quality of that solution is poor. Thus, we propose a branching rule on the assignment decisions to obtain better solutions. Formally, let z^* be an optimal solution to the linear programming relaxation of the master problem (2.3.1) - (2.3.4). The corresponding assignment decision for each flight i given by

$$y^*_{ik} := \sum_{p \in P_k} \delta^k_{ip} z^{k*}_p, \quad \forall i \in \mathcal{F},\ k \in \mathcal{G}. \tag{2.5.2}$$

Consider a fractional z^*, as δ^k_{ip} is either 0 or 1, there may exist $0 < y^*_{ik} < 1$ for a particular i and k. When such a fractional assignment decision occurs, we denote the i and k values by $\hat{i}$ and $\hat{k}$. We present a branching scheme on this assignment decision which is adapted from [40]. We force $y_{\hat{i}\hat{k}} = 1$ on the left branch and $y_{\hat{i}\hat{k}} = 0$ on the right branch by adding valid inequalities. In particular, flight $\hat{i}$ can be forced to use gate $\hat{k}$ by the following

inequality,

$$\sum_{p \in P_{\hat{k}}} (1 - \delta_{\hat{i}p}^{\hat{k}}) z_p^{\hat{k}} + \sum_{k \in \mathcal{F} \setminus \hat{k}} \sum_{p \in P_k} \delta_{\hat{i}p}^{k} z_p^{k} \leq 0. \tag{2.5.3}$$

On the other side of the branch where flight $\hat{i}$ can be forced not to use gate $\hat{k}$ by the following inequality,

$$\sum_{p \in P_{\hat{k}}} \delta_{\hat{i}p}^{\hat{k}} z_p^{\hat{k}} \leq 0. \tag{2.5.4}$$

Note that after the branching constraints are added, the objective functions in the pricing problems have to be updated to incorporate the dual information of these constraints. By updating the dual variables of the constraints (2.3.2) and (2.3.3), we can keep the objective functions in the same format. Let the dual variables to the branching constraints be $\lambda_{\hat{i}\hat{k}}$. On the left branches, if $k = \hat{k}$, we have the new objective function as

$$\begin{aligned} &\min \sum_{i \in \mathcal{F}} (t_i^g - a_i) - \sum_{i \in \mathcal{F}} x_{ik} \pi_i - \mu_k - (1 - x_{\hat{i}k}) \lambda_{\hat{i}\hat{k}} \\ \Leftrightarrow &\min \sum_{i \in \mathcal{F}} (t_i^g - a_i) - \left(\sum_{i \in \mathcal{F} \setminus \hat{i}} x_{ik} \pi_i + x_{\hat{i}k} (\pi_{\hat{i}} + \lambda_{\hat{i}\hat{k}}) \right) - (\mu_k + \lambda_{\hat{i}\hat{k}}). \end{aligned} \tag{2.5.5}$$

When $k \neq \hat{k}$, the new objective function becomes

$$\begin{aligned} &\min \sum_{i \in \mathcal{F}} (t_i^g - a_i) - \sum_{i \in \mathcal{F}} x_{ik} \pi_i - \mu_k - x_{\hat{i}k} \lambda_{\hat{i}\hat{k}} \\ \Leftrightarrow &\min \sum_{i \in \mathcal{F}} (t_i^g - a_i) - \left(\sum_{i \in \mathcal{F} \setminus \hat{i}} x_{ik} \pi_i + x_{\hat{i}k} (\pi_{\hat{i}} + \lambda_{\hat{i}\hat{k}}) \right) - \mu_k. \end{aligned} \tag{2.5.6}$$

Now on the right branches, if $k = \hat{k}$, we obtain the same objective function as shown in (2.5.6). When $k \neq \hat{k}$, the objective function remains unchanged.

We want to point out that the dynamic programming and approximation algorithms have to be adapted at a node where some branching constraints have been added, otherwise the assignment patterns that are generated can potentially violate the branching constraints.

In particular, it is easy to modify the pricing problem methods for the right branches. If flight $\hat{i}$ is forced not to use gate $\hat{k}$, we can remove flight $\hat{i}$ from the subset of flights $\mathcal{F}'$ during the pre-processing. However, it is more difficult on the left branches when we force flight $\hat{i}$ to use gate $\hat{k}$. As flight $\hat{i}$ can be accepted at any time in the interval, $[a_{\hat{i}}, a_{\hat{i}} + \pi_{\hat{i}}]$, we need to first determine the optimal time to accept flight $\hat{i}$ after which the rest of the flights in the set $\mathcal{F}'$ can be optimally assigned. This is more difficult to implement because the time in $[a_{\hat{i}}, a_{\hat{i}} + \pi_{\hat{i}}]$ takes continuous values. To deal with this issue, we run the pricing problem algorithms with the constraints implied by the branching decisions by solving an Integer Program directly using a solver to prove optimality.

2.6 Computational experiments

2.6.1 Instance generation and initialization

We test the proposed methods on randomly generated instances and instances derived from real world data. The detailed information about each instance is reported in Table 2.1 and a summary of the real world operational data is presented in Table 2.2. For the randomly generated instances, we generate the inter-arrival time of two consecutive flights based on a uniform distribution such that each gate has an arriving flight every 75 to 180 minutes on average depending on the size of the instances. We use the minimum turn time data from a major U.S. airline. Furthermore, we generate each gate with a type which determines whether it can accommodate heavy aircraft, a set of eligible airlines, and a buffer time. The buffer times are set to be identical in our experiments for simplicity, but they can be adjusted accordingly if needed. The compatibility data, α_{ik}, used in constraints (2.2.6) and (2.3.12) as well as pre-processing step of the pricing problem algorithms, can be computed based on the aircraft type, the gate type, and the set of eligible airlines.

For the real world instances, we use arrivals from multiple days in 2019 before the COVID-19 pandemic from the Atlanta Hartsfield-Jackson Airport, the busiest airport by passenger traffic. The data are obtained from the U.S. Bureau of Transportation Statistics

website. Note that for instances that do not use arrivals in the whole 24 hours' period, we report the time interval during which the arrivals are used. As it is difficult to obtain the precise information about the gates, we generate the gates in a similar way as described above, but with an additional consideration. As very large percentage of arrivals into ATL are Delta Air Lines (DAL) flights and DAL operates many gates at ATL, a large number of gates allow DAL flights.

Once we have the set of arrivals and the set of gates for an instance, we initialize the instance. For each of the test instance, the set of feasible patterns P_k for all gate k have to be initialized to contain at least one pattern to satisfy the availability constraint (2.3.3) in the master problem. Moreover, the union of P_k has to satisfy the covering constraint (2.3.2). Since we start with empty P_k, we provide each gate a feasible assignment pattern by randomly assigning each flight to a compatible gate to have an overall feasible assignment. Consequently, at the beginning of the proposed procedure, there is an assignment pattern for each gate. We note that although this is a simple initialization procedure, we do not expect any more complex initialization procedure could show significant improvements. In addition, our termination criteria is a relative gap of 2% or an absolute gap of 0.5 (in case the optimal objective of the root node LP relaxation is 0.

Table 2.1: Instance information.

no. (name)	size	source	$\|\mathcal{F}\|$	$\|\mathcal{G}\|$	$\|\mathcal{F}\|/\|\mathcal{G}\|$	inter-arr(min.)
1 (f30g5s)	small	synthetic	30	5	6.00	12.07
2 (f30g10s)	small	synthetic	30	10	3.00	9.66
3 (f50g10s)	small	synthetic	50	10	5.00	8.53
4 (f50f20s)	small	synthetic	50	20	2.50	4.37
5 (f100g35s)	small	synthetic	100	35	2.86	2.72
6 (f100g50s)	small	synthetic	100	50	2.00	1.75
7 (f150g50s1)	moderate	synthetic	150	50	3.00	2.01
8 (f150g50s2)	moderate	synthetic	150	50	3.00	2.30
9 (f150g50s3)	moderate	synthetic	150	50	3.00	2.10

Table 2.1 continued

10 (f200g100a1)	moderate	10:00-13:12 08/23/19	200	100	2.00	0.96
11 (f200g100a2)	moderate	13:00-15:51 08/23/19	200	100	2.00	0.86
12 (f300g150s1)	moderate	synthetic	300	150	2.00	0.51
13 (f300g150s2)	moderate	synthetic	300	150	2.00	0.49
14 (f300g150s3)	moderate	synthetic	300	150	2.00	0.48
15 (f300g150a1)	moderate	10:00-14:31 08/23/19	300	150	2.00	0.90
16 (f300g150a2)	moderate	13:00-17:47 08/23/19	300	150	2.00	0.95
17 (f800g200s1)	large	synthetic	800	200	4.00	0.77
18 (f800g200s2)	large	synthetic	800	200	4.00	0.73
19 (f800g200s3)	large	synthetic	800	200	4.00	0.75
20 (f800g200a1)	large	10:00-23:13 08/19/19	800	200	4.00	0.99
21 (f800g200a2)	large	10:00-23:13 08/20/19	800	200	4.00	0.99
22 (f800g200a3)	large	10:00-23:41 08/21/19	800	200	4.00	1.02
23 (f800g200a4)	large	10:00-22:36 08/22/19	800	200	4.00	0.95
24 (f800g200a5)	large	10:00-22:22 08/23/19	800	200	4.00	0.93
25 (f1000g200s1)	large	synthetic	1000	200	5.00	0.93
26 (f1000g200s2)	large	synthetic	1000	200	5.00	0.89
27 (f1000g200s3)	large	synthetic	1000	200	5.00	0.89
28 (f1000g200a1)	large	6:00-21:50 08/19/19	1000	200	5.00	0.95
29 (f1000g200a2)	large	6:00-21:41	1000	200	5.00	0.94

Table 2.1 continued

		08/20/19				
30 (f1000g200a3)	large	6:00-21:37 08/21/19	1000	200	5.00	0.94
31 (f1000g200a4)	large	6:00-21:21 08/22/19	1000	200	5.00	0.92
32 (f1000g200a5)	large	6:00-21:21 08/23/19	1000	200	5.00	0.92
33 (f1113g192a19)	large	08/19/19	1113	192	5.80	1.28
34 (f1095g192a20)	large	08/20/19	1095	192	5.70	1.30
35 (f1092g192a21)	large	08/21/19	1092	192	5.69	1.31
36 (f1125g192a22)	large	08/22/19	1125	192	5.86	1.28
37 (f1125g192a23)	large	08/23/19	1125	192	5.86	1.27

Table 2.2: Summary of arrivals at ATL.

date.	no. of gates	no. of arrivals
Aug 19, 2019	192	1113 (687 Delta, 39 American, 8 United, 379 Other)
Aug 20, 2019	192	1095 (681 Delta, 32 American, 9 United, 373 Other)
Aug 21, 2019	192	1092 (675 Delta, 35 American, 10 United, 372 Other)
Aug 22, 2019	192	1125 (684 Delta, 40 American, 11 United, 390 Other)
Aug 23, 2019	192	1125 (684 Delta, 39 American, 11 United, 391 Other)

2.6.2 Software and hardware

For the implementation, the pricing problem algorithms are implemented in Python. Gurobi (version 9.1) ([41]) is used whenever a commercial solver is needed. The computations are performed on an Unix system with a 12-core CPU and 16GB RAM. We also set a limit of 8 hours for each instance.

2.6.3 Computational results

For a complete comparison, the performance of the compact mixed integer programming (MIP) formulation (2.2.1) - (2.2.8) is considered using the small-sized instances and shown in Table 2.3. The formulation is solved using the Gurobi solver. As we have noted before, the MIP formulation is not ideal for this problem as evidenced in Table 2.3.

Table 2.3: Performances of the MIP formulation.

method	instance no.	time(sec.)	incumbent obj.	best bound
MIP	1 (f30g5s)	339.78	1203.26	1203.26
	2 (f30g10s)	6948.33	258.53	258.53
	3 (f50g10s)	5347.46	74.73	74.73
	4 (f50f20s)	incomplete	18.64	0.00
	5 (f100g35s)	incomplete	1.66	0.00
	6 (f100g50s)	incomplete	1.14	0.00

For the following results, note that we proposed to obtain a feasible solution at the root node by reinstating the binary requirement and compute the quality of the solution by (2.5.1). In addition, we use the following notations to report results in tables:

- ct: computation time in second;
- rt: computation time spent on the root node in second,

in addition to the "LB_{root}", "UB_{root}", and "gap_{root}" introduced previously. Note that these values are obtained at the root node.

Small-sized instances.

It is important to note that we can combine the pricing problem algorithms presented in Section 2.4 in our implementation. For the small-sized instances, we test the following ways:

1. Gurobi solver (S)

2. Dynamic programming algorithm (DP)

3. 70 iterations of submodular maximization followed by the dynamic programming algorithm (SM+DP)

4. 25 iterations of approximative dynamic programming algorithm followed by the dynamic programming algorithm (ADP+DP).

We run a fixed number of iterations of both SM and ADP across different instances as discussed above, and no attempt has been made to tune these parameters. For a particular class of instances, tuning these parameters may provide even better results.

In the solver option (Option 1 above), specifically, we set the Gurobi "PoolSearchMode" parameter to 2 and request the solver to provide up to 75 feasible solutions and add as many as possible to the master problem. The updated solver parameters does not deteriorate the performance of the solver as we observe the solver very rarely performs extra computations after the optimal solution is found to fill the feasible solution pool. These extra feasible solutions are likely assignment patterns with favorable reduced costs and can potentially reduce the number of times the pricing problems are solved.

The detailed computational results for the small-sized instances are shown in Table 2.4. We see that our proposed approaches out-perform the Gurobi solver (S) in all six instances, even with just the dynamic programming algorithm alone (DP). While the approximation algorithms are designed to provide additional improvements relative to the dynamic programming algorithm, however, they do not offer any substantial boost in these small instances and, on the contrary, we see the dynamic programming algorithm alone achieves better computation time. In addition, the binary program solutions obtained with different methods differ in some of instances. This is likely due to three methods, namely, DP, SM, and ADP, are generating different patterns. In general, if we increase the size of the instance, it becomes more difficult to solve. We observe that the flight to gate ratio has

an impact on the time. In particular, although instance 1 is smaller in size than instances 2 and 4 are, it takes longer to solve than both instances 2 and 4 do. The flight to gate ratio determines how congested the gates are. The higher the ratio, the larger the number of flights each gate on average has to accept. Instances with higher ratio require more delicate assignments. Furthermore, in contrast to the worse case theoretical analysis, the dynamic programming algorithm performs well. This is likely due to the fact that the inter-arrival times are much shorter than the processing times and consequently many recursive calls to the evaluation oracle have input t beyond the corresponding flights' acceptance windows.

Table 2.4: Pricing problem methods on small-sized instances.

instance	methods	ct(sec.)	rt(sec.)	LB_{root}	UB_{root}	$gap_{root}(\%)$	final obj.	node(s)
	S	130.71	130.71	1203.26	1203.26	0.00	1203.26	1
1	DP	68.83	68.83	1203.26	1203.26	0.00	1203.26	1
(f30g5s)	SM+DP	48.46	27.52	1203.26	1448.98	20.4	1203.26	3
	ADP+DP	33.64	33.64	1203.26	1203.26	0.00	1203.26	1
	S	26.52	26.52	258.53	258.53	0.00	258.53	1
2	DP	2.15	2.15	258.53	258.53	0.00	258.53	1
(f30g10s)	SM+DP	0.63	0.63	258.53	258.53	0.00	258.53	1
	ADP+DP	3.52	3.52	258.53	258.53	0.00	258.53	1
	S	91.85	91.85	74.73	74.73	0.00	74.73	1
3	DP	39.86	25.52	74.73	77.84	4.00	74.73	5
(f50g10s)	SM+DP	38.60	26.73	74.73	80.79	7.50	74.73	5
	ADP+DP	30.79	30.79	74.73	74.73	0.00	74.73	1
	S	47.63	47.63	18.64	18.64	0.00	18.64	1
4	DP	1.72	1.72	18.64	18.64	0.00	18.64	1
(f50f20s)	SM+DP	1.44	1.44	18.64	18.64	0.00	18.64	1
	ADP+DP	10.07	10.07	18.64	18.64	0.00	18.64	1
	S	1495.42	1495.42	1.66	1.66	0.00	1.66	1
5	DP	189.61	37.92	1.66	1.88	11.70	1.66	26
(f100g35s)	SM+DP	20.67	20.67	1.66	1.66	0.00	1.66	1

Table 2.4 continued

	ADP+DP	46.89	46.89	1.66	1.66	0.00	1.66	1
	S	1893.26	1893.26	1.14	1.14	0.00	1.14	1
6	DP	17.89	17.89	1.14	1.14	0.00	1.14	1
(f100g50s)	SM+DP	10.06	10.06	1.14	1.14	0.00	1.14	1
	ADP+DP	29.67	29.67	1.14	1.14	0.00	1.14	1

Moderate-sized instances.

Next we show the computational results for the moderate-sized instances. As we have seen that the solver option is not as effective as other options are in solving the small-sized instances, only the options of DP, SM+DP, and ADP+DP are considered in this set of experiments. As we pointed out previously, we run 70 iterations of SM and 25 iterations of ADP across all instances.

The results are shown in Table 2.5. We see the same trend of longer time taken to solve an instance of larger size. Since the flight to gate ratios are small for these moderate-sized instances, we do not see any obvious impact of the ratio. The boost in performances from the submodular maximization approximation algorithm is much more obvious in the moderated-sized instances and in some cases, the approximative dynamic programming algorithm improves the computation time. The running time of either approximation algorithm increases linearly with the size of the set of flights. Given a larger set of flights, each iteration of the approximation algorithm takes much less time compared to each iteration of the dynamic programming algorithm. Nonetheless, we observe that the submodular maximization out-performs the approximative dynamic programming in all instances as the assignment patterns that are generated by the submodular maximization are usually of better total net benefits than those that are generated by the approximative dynamic programming. Moreover, we observe small variations in the times taken across instances of the same size and this suggests that besides the sizes of the set of flights and the set of gates, other factors associated with the flights and the gates can complicate the problem. In addition, although we see that even though some of the synthetic instances and real world instances have the same number of flights and gates, the performances are very different

and it is likely due to arrivals in the real world instances are much more complex than those in the synthetic instances.

Table 2.5: Pricing problem methods on moderate-sized instances.

instance	methods	ct(sec.)	rt(sec.)	LB_{root}	UB_{root}	$gap_{root}(\%)$	final obj.	node(s)
7 (f150g50s1)	DP	770.74	479.53	0.0	0.69	0.69(a)	0.0	82
	SM+DP	346.58	346.58	0.0	0.0	0.00(a)	0.0	1
	ADP+DP	985.58	985.58	0.0	0.22	0.22(a)	0.22	1
8 (f150g50s2)	DP	2108.45	1866.29	0.90	1.11	23.3	0.9	23
	SM+DP	486.85	156.04	0.90	5.32	491	0.9	50
	ADP+DP	1034.31	631.49	0.9	1.90	52.6	0.9	50
9 (f150g50s3)	DP	1185.55	1185.55	10.74	10.74	0.00	10.74	1
	SM+DP	439.70	439.70	10.74	10.74	0.00	10.74	1
	ADP+DP	498.51	498.51	10.74	10.74	0.0	10.74	1
10 (f200g100a1)	DP	5002.04	5002.04	0.0	0.0	0.00(a)	0.0	1
	SM+DP	929.45	929.45	0.0	0.0	0.00(a)	0.0	1
	ADP+DP	4689.39	4689.39	0.0	0.0	0.00(a)	0.0	1
11 (f200g100a2)	DP	2883.41	2883.41	0.0	0.0	0.00(a)	0.0	1
	SM+DP	570.75	570.75	0.0	0.0	0.00(a)	0.0	1
	ADP+DP	3168.33	3168.33	0.0	0.0	0.00(a)	0.0	1
12 (f300g150s1)	DP	8918.69	8918.69	349.10	349.10	0.00	349.10	1
	SM+DP	2833.93	2833.93	349.10	349.10	0.00	349.10	1
	ADP+DP	6891.40	6891.40	349.10	349.10	0.00	349.10	1
13 (f300g150s2)	DP	7927.31	7927.31	203.00	203.00	0.00	203.00	1
	SM+DP	3111.97	3111.97	203.10	203.10	0.00	203.10	1
	ADP+DP	6284.43	6284.43	203.10	203.10	0.00	203.10	1
14 (f300g150s3)	DP	7830.18	7830.18	414.12	414.12	0.00	414.12	1
	SM+DP	4570.69	4570.69	414.12	414.12	0.00	414.12	1
	ADP+DP	8734.09	8734.09	414.12	414.12	0.00	414.12	1
15 (f300g150a1)	DP	25132.32	25132.32	0.0	0.0	0.00(a)	0.0	1
	SM+DP	14032.56	14032.56	0.0	0.0	0.00(a)	0.0	1

Table 2.5 continued

	ADP+DP	23449.23	23449.23	0.0	0.0	0.00(a)	0.0	1
16 (f300g150a2)	DP	15249.75	15249.75	0.0	0.0	0.00(a)	0.0	1
	SM+DP	13052.81	13052.81	0.0	0.0	0.00(a)	0.0	1
	ADP+DP	19874.84	19874.84	0.0	0.0	0.00(a)	0.0	1

Large-sized instances.

Following the experiments on the moderate-sized instances, we move on to the large-sized instances. As we discussed in the Subsection 2.4.5, we can utilize the block decomposition approximation and rolling horizon framework to tackle the large-sized instances. The horizon size and window size are important parameters in the rolling horizon framework. We first perform parametric studies to understand how the horizon size and window size affect the performances. We vary the horizon size and window size to test the performances of the rolling horizon method on random large-sized instances. We observe that for any window size, smaller horizon sizes result in better performances. However, very small horizon size does not further reduce the time taken as the quality of the assignment patterns that are generated under small horizon sizes is poor and more iterations are needed to reach the optimality.

For the large-sized instances, we again have a few possible ways to solve the pricing problems because we can either use a fixed or a dynamically determined horizon size for the rolling horizon method and also utilize the approximation algorithms. Here is a detailed breakdown:

1. Rolling horizon method (horizon size: $\overline{\sigma}$, window size: 1) (RHD)
2. Rolling horizon method (horizon size: 20, window size: 1) (RHF)
3. Rolling horizon method (horizon size: $\min(20, \overline{\sigma})$, window size: 1) (RHM)

4. Submodular maximization if $\overline{\sigma} > 60$ in the first 25 iterations and rolling horizon method otherwise (horizon size: $\min(20, \overline{\sigma})$, window size: 1) (SM+RHM)

5. 30 iterations of block decomposition approximation followed by the rolling horizon method (horizon size: $\overline{\sigma}$, window size: 1) (BD+RHD)

where $\overline{\sigma}$ is the adjacency parameter that depends on the values of the dual variables as described in Subsection 2.4.5. Furthermore, submodular maximization can generally provide assignment patterns of good quality and has a linear running time in the size of the set of flights, so its performance remains very effective for these large-sized instances. It can potentially improve the performances and serve as a benchmark for the block decomposition approximation. Note that whenever an algorithm is needed to evaluate the optimal assignment during the rolling horizon process or the block decomposition approximation, the direct implementation of the dynamic programming algorithm (Algorithm 2) is used. Again, for these large-sized instances, we use the same parameters across different instances and there can be potential improvements if parameters are tuned for individual cases. Nonetheless, we present results with a fixed set of parameters which still confirm the strong performances of our proposed approaches.

The results of the computations are shown in Table 2.6. After extensive experiments, we observe that the adjacency parameters, $\overline{\sigma}$, can be very large at beginning and, consequently, the options of RHD and BD+RHD that involve decomposition based on these parameters become very ineffective and unable to obtain a good solution within the allocated time limit. Therefore, we do not report the results of these two options in Table 2.6. From the results in the table, we see the rolling horizon method with a fixed horizon size performs much better than the same method with a horizon size of $\overline{\sigma}$ which struggles to solve these large-sized instances. If we further keep the horizon size at most 20 in the RHM option, we obtain comparable if not slightly better performances in all instances compared to the rolling horizon method with a fixed horizon size. However, submodular maximization algorithm efficiently provides assignment patterns of good quality when $\overline{\sigma}$ is large and improves the performances compared to both RHF and RHM options in all instances. The benefits of using submodular maximization algorithm is especially significant in the cases

derived from ATL arrivals. The reductions in computation time can be as large as about 50%. For instances 33-37, we show that our proposed approach can solve the flight-to-gate problem with arrivals in a single day for the busiest airport in the world within very reasonable amount of time.

Table 2.6: Pricing problem methods on large-sized instances.

instance	methods	ct(sec.)	rt(sec.)	LB_{root}	UB_{root}	$gap_{root}(\%)$	final obj.	node(s)
17 (f800g200s1)	RHF	1160.33	1160.33	0.0	0.0	0.00(a)	0.0	1
	RHM	1147.62	1147.62	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1067.63	1067.63	0.0	0.0	0.00(a)	0.0	1
18 (f800g200s2)	RHF	1166.7	1166.7	0.0	0.0	0.00(a)	0.0	1
	RHM	1076.56	1076.56	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1010.11	1010.11	0.0	0.0	0.00(a)	0.0	1
19 (f800g200s3)	RHF	1015.80	1015.80	0.0	0.0	0.00(a)	0.0	1
	RHM	899.28	899.28	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	975.88	975.88	0.0	0.0	0.00(a)	0.0	1
20 (f800g200a1)	RHF	2313.15	2313.15	0.0	0.0	0.00(a)	0.0	1
	RHM	2381.8	2381.8	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1570.65	1570.65	0.0	0.0	0.00(a)	0.0	1
21 (f800g200a2)	RHF	2033.91	2033.91	0.0	0.0	0.00(a)	0.0	1
	RHM	2138.42	2138.42	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1708.62	1708.62	0.0	0.0	0.00(a)	0.0	1
22 (f800g200a3)	RHF	2658.21	2658.21	0.0	0.0	0.00(a)	0.0	1
	RHM	2673.93	2673.93	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1714.31	1714.31	0.0	0.0	0.00(a)	0.0	1
23 (f800g200a4)	RHF	2457.84	2457.84	0.0	0.0	0.00(a)	0.0	1
	RHM	2444.43	2444.43	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1769.04	1769.04	0.0	0.0	0.00(a)	0.0	1
24 (f800g200a5)	RHF	30932.55	3093.55	0.0	0.0	0.00(a)	0.0	1
	RHM	2989.79	2989.79	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1818.73	1818.73	0.0	0.25	0.25(a)	0.25	1
25 (f1000g200s1)	RHF	1675.46	1675.46	0.0	0.0	0.00(a)	0.0	1

Table 2.6 continued

	RHM	1540.65	1540.65	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1440.66	1440.66	0.0	0.0	0.00(a)	0.0	1
26 (f1000g200s2)	RHF	2286.41	2286.41	0.0	0.0	0.00(a)	0.0	1
	RHM	1959.05	1959.05	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1843.98	1843.98	0.0	0.0	0.00(a)	0.0	1
27 (f1000g200s3)	RHF	1720.46	1720.46	0.0	0.0	0.00(a)	0.0	1
	RHM	1585.0	1585.0	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	1580.16	1580.16	0.0	0.0	0.00(a)	0.0	1
28 (f1000g200a1)	RHF	5858.02	5858.02	0.0	0.0	0.00(a)	0.0	1
	RHM	6354.51	6354.51	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3407.24	3407.24	0.0	0.0	0.00(a)	0.0	1
29 (f1000g200a2)	RHF	7270.22	7270.22	0.0	0.0	0.00(a)	0.0	1
	RHM	7342.51	7342.51	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3206.97	3206.97	0.0	0.0	0.00(a)	0.0	1
30 (f1000g200a3)	RHF	4922.02	4922.02	0.0	0.0	0.00(a)	0.0	1
	RHM	4844.84	4844.84	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3701.92	3701.92	0.0	0.0	0.00(a)	0.0	1
31 (f1000g200a4)	RHF	5897.09	5897.09	0.0	0.0	0.00(a)	0.0	1
	RHM	5715.54	5715.54	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3093.73	3093.73	0.0	0.0	0.00(a)	0.0	1
32 (f1000g200a5)	RHF	4913.89	4913.89	0.0	0.0	0.00(a)	0.0	1
	RHM	4654.29	4654.29	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	2966.66	2966.66	0.0	0.0	0.00(a)	0.0	1
33 (f1113g192a19)	RHF	6305.36	6305.36	0.0	0.0	0.00(a)	0.0	1
	RHM	6177.28	6177.28	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3865.39	3865.39	0.0	0.0	0.00(a)	0.0	1
34 (f1095g192a20)	RHF	5763.05	5763.05	0.0	0.0	0.00(a)	0.0	1
	RHM	6808.54	6808.54	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	3420.51	3420.51	0.0	0.0	0.00(a)	0.0	1
35 (f1092g192a21)	RHF	7156.45	7156.45	0.0	0.0	0.00(a)	0.0	1
	RHM	8568.96	8568.96	0.0	0.0	0.00(a)	0.0	1

Table 2.6 continued

	SM+RHM	3603.5	3603.5	0.0	0.0	0.00(a)	0.0	1
36 (f1125g192a22)	RHF	8384.49	8384.49	0.0	0.0	0.00(a)	0.0	1
	RHM	9347.75	9347.75	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	4279.68	4279.68	0.0	0.0	0.00(a)	0.0	1
37 (f1125g192a23)	RHF	7784.41	7784.41	0.0	0.0	0.00(a)	0.0	1
	RHM	7727.33	7727.33	0.0	0.0	0.00(a)	0.0	1
	SM+RHM	4325.52	4325.52	0.0	0.0	0.00(a)	0.0	1

2.6.4 Summary of the computational results

In summary, it seems that the dynamic programming algorithm together with the submodular maximization offer the best performance for the small-sized and moderate-sized problems. For the large-sized instances, it seems that running the submodular maximization algorithm when $\overline{\sigma}$ is large and using the rolling horizon method while keeping the horizon sizes at most 20 otherwise lead to the best performances. Moreover, the binary program produces feasible solutions of good quality for almost all instances as the gaps are usually very small. The amount of time taken to obtain those feasible solutions is also reasonable in practice.

Finally note that all instance data can be accessed online at the authors' websites.

2.7 Conclusion

In conclusion, we use a column generation approach to solve the flight-to-gate assignment problem aimed at minimizing the total arrival delays. Instead of using the integer program solver for the pricing problem, more efficient approximation algorithms and dynamic programming algorithms are proposed. Among the proposed algorithms, submodular maximization algorithm shows very strong performances on the small and medium-sized instances and together with the rolling horizon framework, it allows us to obtain solutions of

very good quality very efficiently for the large-sized instances.

There are few possible extensions to consider. One of them is to consider the interference between different gates. With different airport layouts, this can be realized by linking constraints in the master problem or a decomposition of the pricing problem by gate groups instead of by each individual gate. Another extension is to take the uncertainties in the arrival times and turn times into consideration. A stochastic programming approach may be suitable to tackle this version of the problem.

2.8 Appendix A. Proof of Theorem 2.4.1

This proof is given in [35] and we modified it to match our assumption.

Let OPT denote an optimal solution and $OPT_i := (OPT \cup X_i) \cap Y_i$, then we have that $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$. Here are two useful lemmas to be used later in the proof of the theorem.

Lemma 2.8.1. *For every* $1 \leq i \leq n$, $a_i + b_i \geq 0$.

Proof. Proof of Lemma 2.8.1. Note that if f is a submodular function with a ground set $\mathcal{F}'$, we have that if $A, B \subset \mathcal{F}'$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.

Now, we see that $(X_{i-1} \cup \{u_i\}) \cup (Y_{i-1} \backslash \{u_i\}) = Y_{i-1}$ and $(X_{i-1} \cup \{u_i\}) \cap (Y_{i-1} \backslash \{u_i\}) = X_{i-1}$. Then by the above definition, we have that

$$a_i + b_i = [f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})] + [f(Y_{i-1} \cup \{u_i\}) - f(Y_{i-1})] \tag{2.8.1}$$

$$= [f(X_{i-1} \cup \{u_i\}) + f(Y_{i-1} \backslash \{u_i\})] - [f(X_{i-1}) + f(Y_{i-1})] \geq 0. \tag{2.8.2}$$

□

Lemma 2.8.2. *For every* $1 \leq i \leq n$,

$$\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2}\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \tag{2.8.3}$$

Proof. Proof of Lemma 2.8.2. It is sufficient to prove (2.8.3) conditioned on any event of the form $X_{i-1} = S_{i-1}$, when $S_{i-1} = \{u_1, \cdots, u_{i-1}\}$ and the probability $X_{i-1} = S_{i-1}$ is non-zero. Hence fix such an event for a given S_{i-1}. The rest of the proof implicitly assumes everything is conditioned on this event. Note due to conditioning, the following quantities become constants,

- $Y_{i-1} = S_{i-1} \cup \{u_i, \cdots, u_n\}$
- $OPT_{i-1} := (OPT \cup X_{i-1}) \cap Y_{i-1} = S_{i-1} \cup (OPT \cap \{u_i, \cdots, u_n\})$
- a_i and b_i.

From Lemma 2.8.1, we have $a_i + b_i \geq 0$, so we only need to consider three cases.

1. $a_i \geq 0$ and $b_i < 0$. In this case, $a_i'/(a_i' + b_i') = 1$. Then we have $Y_i = Y_{i-1}$ and $X_i = S_{i-1} \cup \{u_i\}$. Hence $f(Y_{i-1}) - f(Y_i) = 0$. Also, $OPT_i = (OPT \cup X_i) \cap Y_i = (OPT \cup X_{i-1} \cup \{u_i\}) \cap Y_{i-1} = OPT_{i-1} \cup \{u_i\}$. Then we are left to prove that

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq \frac{1}{2}[f(X_i) - f(X_{i-1})] = \frac{a_i}{2}. \tag{2.8.4}$$

   If $u_i \in OPT_{i-1}$, then the left hand side is zero and this inequality is satisfied. If $u_i \notin OPT_{i-1}$, then we note that

$$OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} \backslash \{u_i\}. \tag{2.8.5}$$

   Next, by the definition of submodularity of f, we have now

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq f(Y_{i-1} \backslash \{u_i\}) - f(Y_{i-1}) = b_i \leq 0 \leq \frac{a_i}{2}. \tag{2.8.6}$$

2. $a_i < 0$ and $b_i \geq 0$. This case is analogous to the previous case.

3. $a_i \geq 0$ and $b_i > 0$. In this case, we have $a'_i = a_i$ and $b'_i = b_i$. Then we can compute the left hand side of (2.8.3) by

$$\begin{aligned}\mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] &= \frac{a_i}{a_i + b_i}[f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})] \\ &\quad + \frac{b_i}{a_i + b_i}[f(Y_{i-1} \cap \{u_i\}) - f(Y_{i-1})] \qquad (2.8.7) \\ &= \frac{a_i^2 + b_i^2}{a_i + b_i}. \qquad (2.8.8)\end{aligned}$$

Next we upper bound $\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)]$,

$$\begin{aligned}\mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] &= \frac{a_i}{a_i + b_i}[f(OPT_{i-1}) - f(OPT_{i-1}) \cup \{u_i\}] \\ &\quad + \frac{b_i}{a_i + b_i}[f(OPT_{i-1}) - f(OPT_{i-1} \backslash \{u_i\})] \qquad (2.8.9) \\ &\leq \frac{a_i b_i}{a_i + b_i}. \qquad (2.8.10)\end{aligned}$$

Last inequality follows from the following cases. Note that $u_i \in Y_{i-1}$ and $u_i \notin X_{i-1}$,

(a) $u_i \notin OPT_{i-1}$, then we note that the second term of the left hand side the last inequality is zero and

$$OPT_{i-1} = (OPT \cup X_{i-1}) \cap Y_{i-1} \subseteq Y_{i-1} \backslash \{u_i\}. \qquad (2.8.11)$$

Next, by the definition of submodularity of f, we have now

$$f(OPT_{i-1}) - f(OPT_{i-1} \cup \{u_i\}) \leq f(Y_{i-1} \backslash \{u_i\}) - f(Y_{i-1}) = b_i. \qquad (2.8.12)$$

(b) $u_i \in OPT_{i-1}$, then we note that the first term of the left hand side the last

inequality is zero and

$$X_{i-1} \subseteq (OPT \cup X_{i-1}) \cap Y_{i-1} \backslash \{u_i\} = OPT_{i-1} \backslash \{u_i\}. \tag{2.8.13}$$

Next, by the definition of submodularity of f, we have now

$$f(OPT_{i-1}) - f(OPT_{i-1} \backslash \{u_i\}) \leq f(X_{i-1} \cup \{u_i\}) - f(X_{i-1}) = a_i. \tag{2.8.14}$$

Then with (2.8.8) and (2.8.10), we are left to verify

$$\frac{a_i b_i}{a_i + b_i} \leq \frac{1}{2} \left(\frac{a_i^2 + b_i^2}{a_i + b_i} \right), \tag{2.8.15}$$

which can be easily verified.

□

With the two Lemmas, we are ready to show the approximation guarantee of the algorithm.

Proof. Proof of Theorem 2.4.1. If we sum all inequalities from lemma 2.8.2 over $1 \leq i \leq n$, we have that

$$\sum_{i=1}^{n} \mathbb{E}[f(OPT_{i-1}) - f(OPT_i)] \leq \frac{1}{2} \sum_{i=1}^{n} \mathbb{E}[f(X_i) - f(X_{i-1}) + f(Y_i) - f(Y_{i-1})] \tag{2.8.16}$$

$$\implies \mathbb{E}[f(OPT_0) - f(OPT_n)] \leq \frac{1}{2} \mathbb{E}[f(X_n) - f(X_0) + f(Y_n) - f(Y_0)]. \tag{2.8.17}$$

Note that $X_0 = \emptyset$, so we have $f(X_0) = 0$ and by our assumption we have $f(Y_0) = f(\mathcal{F}') \geq 0$. In addition, we have $OPT_0 = OPT$ and $OPT_n = X_n = Y_n$, so (2.8.17) gives that

$$f(OPT) \leq \frac{1}{2} \mathbb{E}[2f(X_n) - f(Y_0)] + \mathbb{E}f(X_n) \leq \frac{1}{2} \mathbb{E}[2f(X_n)] + \mathbb{E}f(X_n) \implies \frac{1}{2} f(OPT) \leq \mathbb{E}f(X_n).$$

□

2.9 Appendix B. Rational input data for dynamic programming algorithm

Recall a direct implementation of the dynamic programming algorithm terminates with a worst case 2^n recursive calls as proven in Proposition 2.4.1. We can improve the complexity significantly if we further assume all input data are rationals and run a backward version of the dynamic programming algorithm. Input data includes all input parameters to the pricing problems, namely the arrival times, a_i, the dual information, π_i, and the processing time, p_i. Without loss of generality, we assume the input data are expressed as rationals with a common denominator. We introduce a new notion of last time to consider for the pricing problems, denoted by c. This is the time beyond which no further flights are accepted. It suffices to choose c as $a_n + \max_{1\leq i\leq n} \pi_i$. For any flight j, if we consider the function $g_j(t)$ with $t > c$, we get that $t > a_l + \pi_l$ for $j \leq l \leq n$. Consequently the function value $g_j(t) = g_l(t) = g_{n+1}(t) = 0$ and $x_{lk} = 0$ for $j \leq l \leq n$.

For every $i \in [n]$, we see that the function $g_i(t)$ is piece-wise linear and we label the points where the slope of the function changes as the breakpoints of the function. When we have integral input data, each $g_i(t)$ does not have too many breakpoints. The following theorem gives a formal statement about this observation.

Theorem 2.9.1. *Assume all input data are rational and c is last time to consider. Let the common denominator be d, then each function $g_i(t)$ for $i \in [n]$ has at most $O(n^2dc)$ breakpoints.*

To prove Theorem 2.9.1, we first consider the possible slopes for a particular function $g_{n-i}(t)$.

Lemma 2.9.1. *Given a function $g_{n-i}(t)$, its slope can only take values from the set $\{0, -1, \cdots, -i-1\}$*

proof of Lemma 2.9.1. We prove this lemma by backward induction. We observe that the function $g_n(t)$ is given by

$$g_n(t) = \begin{cases} \pi_n & \text{if } t \leq a_n \\ a_n + \pi_n - t & \text{if } a_n < t \leq a_n + \pi_n \\ 0 & \text{if } t > a_n + \pi_n. \end{cases} \tag{2.9.1}$$

The set of slopes of function $g_n(t)$ is $\{0, -1\}$. This is the base case of n. Suppose the statement is true for n, $n-1$, $\cdots$, $n-i$. Then the set of possible slopes of $g_{n-i}(t)$ is $\{0, -1, \ldots, -i-1\}$. Now consider the function $g_{n-i-1}(t)$, based on the recursive formula (2.4.5),

$$g_{n-i-1}(t) = \begin{cases} g_{n-i-1}(a_i) & \text{if } t \leq a_{n-i-1} \\ g_{n-i}(t) & \text{if } t > a_{n-i-1} + \pi_{n-i-1} \\ \max\{a_{n-i-1} + \pi_{n-i-1} - t + g_{n-i}(t + p_{n-i-1}), g_{n-i}(t)\} & \text{if } a_{n-i-1} < t \leq a_{n-i-1} + \pi_{n-i-1}. \end{cases} \tag{2.9.2}$$

In the second case, the set of possible slopes of $g_{n-i-1}(t)$ is the same as that of $g_{n-i}(t)$. In the third case, $g_{n-i-1}(t)$ can have an additional slope than g_{n-i} does. If $g_{n-i}(t + p_{n-i-1})$ has a slope of $-i-1$, the slope of $g_{n-i-1}(t)$ can be $-i-2$. Therefore, the set of possible slopes of g_{n-i-1} is $\{0, -1, \ldots, -i-2\}$. Thus the statement is also true for $n-i-1$, which completes the proof. □

Intuitively, the slope of the function corresponds to the rate of loss in the total net benefits if we delay the park times of flights. For the flight $n-i$, it can only potentially affect the park times of itself and flights after it, i.e. from the set $\{n-i, n-i+1, \ldots, n\}$ and the corresponding function $g_{n-i}(t)$ can have slopes from the set $\{0, -1, \ldots, -i-1\}$.

Now we are ready to give a proof of theorem 2.9.1.

proof of Theorem 2.9.1. Consider a function $g_{n-i}(t)$, from the recursive formula (2.4.5),

we observe that all the potential breakpoints can be computed by equating the two components in the max expression. The solution of t may constitute a breakpoint of g_{n-i}. As a result, we consider the following linear equation

$$(a_{n-i} + \pi_{n-i} - t) + g_{n-i+1}(t + p_i) = g_{n-i+1}(t). \tag{2.9.3}$$

Functions $g_{n-i+1}(t+p_i)$ and $g_{n-i+1}(t)$ are piece-wise linear in variable t. We can write

$$g_{n-i+1}(t + p_i) = c_1 - k_1 t \text{ and } g_{n-i+1}(t) = c_2 - k_2 t, \tag{2.9.4}$$

where c_1 and c_2 are rational constants and k_1 and k_2 are integer constants. We only consider cases where $k_1 + 1 \neq k_2$, otherwise the two functions on the left hand side and right hand side of (2.9.3) have the same slope and do not constitute a breakpoint. Now equation (2.9.3) can be rewritten as

$$a_{n-i} + \pi_{n-i} - t + c_1 - k_1 t = c_2 - k_2 t \implies t = \frac{|a_{n-i} + \pi_{n-i} + c_1 - c_2|}{|k_1 + 1 - k_2|}. \tag{2.9.5}$$

As shown in lemma 2.9.1, k_1 and k_2 only take values from the set $\{0, -1, \ldots, -i\}$ and thus $|k_1 + 1 - k_2|$ can only take values from the set $\{1, \ldots, i+1\}$. In addition, since the input data are rationals with a common denominator d, the numerator $a_{n-i} + \pi_{n-i} + c_1 - c_2$ can be expressed as C/d where C is an integer. Then all the breakpoints of function $g_{n-i}(t)$ are in the set

$$\{0, e, 2e, 3e, \ldots, c\} \text{ where } e \in \{1/d, 1/2d, 1/3d, \ldots, 1/(i+1)d\}. \tag{2.9.6}$$

The total number of breakpoints is then at most $1dc + 2dc + 3dc + \cdots + (i+1)dc = O(n^2 dc)$, which completes the proof. □

Note that since each of the functions $g_i(t)$ has at most $O(n^2 dc)$ breakpoints, we only

need to evaluate the function at these potential breakpoints to construct the function and consequently we need to run in $O(n^3dc)$ to construct all functions $g_i(t)$ in the interval $[0, c]$. In particular, note that only the function values at the points in the list of the potential breakpoints are needed since the functions are all piece-wise linear. We can first construct the function $g_n(t)$ in the interval $[0, c]$ by evaluating the function at the points in the list of the potential breakpoints given in (2.9.6) in reverse order and then we construct the function $g_j(t)$ for $1 \leq j \leq n-1$ using previously constructed functions $g_l(t)$ for $j+1 \leq l \leq n$ based on the recursive formula (2.4.5).

Although the above analyses provide a simpler complexity in the case of rational input data, it is not very effective in practice as the common denominator d can be large after scaling the input data.

2.10 Appendix C. Rolling horizon method

The rolling horizon framework can also be utilized to decompose the pricing problems to reduce their sizes. As with any implementation of the rolling horizon method, we must decide on two parameters. One is the horizon size and the other is the window size. The horizon size can be either fixed or dynamically determined for each gate and each iteration. The adjacency parameter given in Definition 2.4.1 is an example of possible dynamically determined horizon sizes. We give a brief implementation of the rolling horizon method in Algorithm 5 that we have implemented in our experiments. Similar to the block decomposition approximation algorithm, we refer to the discussions in Subsection 2.4.2 for the computation of the total net benefits $f(S)$ and the time at which the gate becomes available under a set of assigned flights.

Algorithm 5: Rolling horizon

Input: horizon size, l, and window size, w
Initialize $s = 0$, $t = 0$, and $S = \emptyset$
while $s + l \leq n$ **do**
    Solve for the optimal assignment for the flights $\{s + 1, s + 2, \ldots, s + l\}$ at time t
    Fix the assignments of first w flights and add them to S
    $t \leftarrow$ time at which the gate becomes available under the set of assigned flights S
    $s \leftarrow s + w$
end
Solve for the optimal assignment for the flights $\{s + 1, s + 2, \ldots, n\}$ at time t
Fix the assignments of the flights $\{s + 1, s + 2, \ldots, n\}$ and add them to S
Compute the total net benefits of the assignment S, $f(S)$ **return** S, $f(S)$

CHAPTER 3

A DECOMPOSITION FRAMEWORK FOR GAS NETWORK DESIGN

3.1 Introduction

Natural gas is a very important and common resource for both residential and industrial customers around the world. In the United States alone, a total of 27.7 trillion cubic feet of natural gas were delivered to 77.3 million customers in 2020 ([42]). To transport natural gas to meet this demand, a natural gas transportation system has been developed which was worth $\$187.9$ billion in 2020 ([43]). A gas transportation system is usually modeled as a directed graph where the nodes can be customers with demands, manufacturers with supplies, or in-nodes that do not have either demands or supplies, while the arcs represent various system components. Modeling and optimizing gas transportation systems is very challenging due to the complex nature of the physical principles governing the operations of the system components. Generally, these models involve nonlinear and non-convex constraints. Even simple models of the system components lead to challenging problems, as the scale of realistic instances is quite large compared to what state-of-the-art solvers can tackle.

In general, the system components (the arcs) can be divided into pipes, short pipes, resistors, compressors, valves, and control valves. Pipes constitute the majority of the system components. Control valves are sometimes referred to as regulators as well. Each of the system components serves a different role. The components can be grouped into passive components and active components. Pipes, short pipes, and resistors are passive system components and do not have on and off states. Compressors, control valves, and valves are active system components with on and off states. There are several types of gas network optimization problems. Most problems involve the decision on the flowrates in

the arcs and the potentials at the nodes. Commonly used benchmarking instances are based on the Belgian network ([44] and [45]) of size up to 23 nodes and the Gaslib networks of various sizes up to 4197 nodes ([46]).

In this chapter, we study the *design problem*, which considers a given set of pipe locations. The main decisions involve choosing the pipe diameters, the states for valves, compressors, and control valves, and flowrates and potentials to transport gas to satisfy the given demand and supply scenarios while minimizing the network construction costs. We call this version of design problem the *design-from-scratch variant*, different from the reinforcement version; details are provided in Section 3.2. The demand and supply scenarios are commonly referred to as *nominations*.

The main contributions of this chapter are the following. We propose a decomposition framework that involves an iterative procedure of solving a convex integer master problem and a verification subproblem for the solutions obtained from the master problem, and a binary search to minimize the construction cost of the pipes. We use the Gaslib-582 network with 582 nodes to validate our framework. To the best of our knowledge, this is the first study that solves the gas network design problem on such large-scale instances. Previous literature (reviewed below in Section 3.2) on design-from-scratch version of the problem has not considered any instance with over 500 nodes, and these works do not simultaneously consider active elements, discrete diameter choices, and general (non-tree) underlying networks.

The structure of the remainder of this chapter is as follows. We give a summary of the relevant literature in Section 3.2. Section 3.3 presents the technical background and a compact formulation for the design problem while Section 3.4 presents the decomposition framework. Implementation considerations and numerical experiments to validate the decomposition framework are presented in Section 3.5. Lastly, in Section 3.6, we present concluding remarks and future research directions.

3.2 Literature review

Gas network systems have been an important topic of study in the past several decades. As the relevant literature is rather extensive, here we review only works that are most closely related to ours. For a detailed review of the literature, we refer the interested readers to [47] and [48]. In addition, [49] provides an overview on the modeling and common solution approaches in gas network systems.

Among the types of problems studied, we focus below on two relevant problem types, the nomination validation problem and design problem. In the *nomination validation problem*, we assume a nomination is given. In the case where active system components are not considered, the problem aims to evaluate whether the existing network topology is feasible with respect to the given nomination. In the case where active system components are involved, the problem aims to determine whether there exist feasible configurations for the active system components along with rest of the components such that the resulting network is feasible with respect to the nomination. Work in this area includes [50, 51, 52, 53, 45, 54, 55, 56, 57]. In particular, [50] presents a convexification scheme to find the convex hull of a "Y" junction in the network to deal with the non-convex constraints arising from the governing physical principles. The paper by [51] presents four approaches for solving nomination validation problems. The first approach is a piece-wise linear approximation scheme that utilizes the generalized incremental model to linearize the nonlinear constraints; the approximation is improved iteratively by adding more linearization points. The second approach is a spatial branch-and-bound algorithm, which iteratively partitions the feasible region and refines the estimations and relaxations of the original problem in each partition to obtain dual bounds on the solution. The next approach in [51] is called RedNLP, a two-stage procedure, in which heuristics and reformulations are employed to find promising configurations of the active system components and the feasibility of configurations is checked in the second stage. The last approach considered

is called the smoothing procedure, commonly used for mathematical programs with equilibrium constraints, to model the discrete decisions corresponding to the configurations of the active system components with continuous variables. Numerical experiments and comparisons across the four approaches are performed on the Gaslib-582 network. Overall, the spatial branch-and-bound outperforms the other three approaches. The papers by [52] and [53] consider additional constraints of satisfying heat-power demand and supply in the nomination validation problem. In both works, an alternating direction method is applied to a linearized approximation model and numerical experiments are performed on the Gaslib-4197 network.

The design problem can be divided into the reinforcement problem and the design-from-scratch problem. In the *reinforcement problem*, it is assumed that an existing topology is given. The problem considers the options to install additional system components, mostly pipes and compressors, to satisfy a given nomination while minimizing the construction costs of the new system components. The cost of a new pipe is usually a function of its diameter and, as a result, diameter selection becomes a decision for the pipes. The *design-from-scratch problem* assumes no existing pipes in the network and makes decisions on the diameters for all pipes. In both the reinforcement problem and the design-from-scratch problem, the diameter choices can be continuous or discrete. Recent works that study the reinforcement problem include [58, 59, 60, 44, 61]. In particular, [58] considers the reinforcement problem with continuous diameters and utilizes a two-stage formulation. The first-stage problem is a convex nonlinear program to compute favorable diameter choices and flowrates, while the second-stage problem checks whether the first-stage solution is feasible with respect to the nomination by solving for the potential at each node. This convex program was formally introduced in [62] and adapted to solve a network problem in [63]. The numerical experiments were performed on the Belgian network. The work in [59] considers also a reinforcement problem with discrete diameters. To deal with the non-convexity from the governing physical principles, the paper considers reformula-

tions and a convex relaxation which is second-order conic representable. These authors also utilize perspective strengthening that is studied in [64] and [65] to enhance the relaxation. Numerical experiments were performed on both Belgian and Gaslib networks. For the design-from-scratch problem, [60] considers discrete diameter choices without any active components. A bilevel formulation is proposed and in solving the formulation, the discrete variables corresponding to the discrete diameter choices are first transformed to continuous variables. Subsequently, the lower-level problem is reformulated via conjugate duality while a trust region algorithm is developed for the upper-level problem. Numerical experiments were performed using two networks of size up to 14 nodes. The paper by [44] studies a variant of the design-from-scratch problem with continuous diameter choices and solve the model by a bundle method with generalized gradient. Numerical experiments are performed with the Belgian network. Lastly, [61] consider the problem on a tree-shaped network with continuous diameter and "approximate discrete diameter" obtained from the optimal continuous diameter. These authors develop an iterative procedure to solve the problem; their approach contracts the tree (network) converting the original tree into a single equivalent arc. Numerical illustrations of this procedure were performed on networks of size up to 36 nodes.

3.3 Problem description

3.3.1 Technical background

In this section, we provide necessary technical background on gas networks that we will need for our formulations later. For more details, we refer the interested readers to [66, 51, 46, 49]. For the remainder of this chapter, we use a directed graph $G = (\mathcal{V}, \mathcal{A})$ to represent a gas network, where each arc $a \in \mathcal{A}$ represents a system component of the network and each node $v \in \mathcal{V}$ can be a customer, a manufacturer, or an in-node. For each node $v \in \mathcal{V}$ we track its pressure p_v and potential π_v, where the pressure and potential are

related by the equation:

$$\pi_v = p_v^2. \tag{3.3.1}$$

We denote lower and upper bounds on the potential at a node v by $\pi_v^{\min}$ and $\pi_v^{\max}$ respectively.

Pipes: as mentioned earlier, a majority of the arcs in gas networks is pipes. A pipe $a = (v, w)$ is specified by its length l, diameter D, and material properties. The flowrate in arc a, denoted as q_a, is upper bounded by a value $q_a^{\max}$ which is determined by the cross-sectional area, $A := \pi D^2/4$, and material properties. We assume a linear relation between the value of $q_a^{\max}$ and the cross-sectional area, i.e.,

$$q_a^{\max} \propto A. \tag{3.3.2}$$

The gas flow in pipe $a = (v, w)$ is described by a set of partial differential equations derived from conservation of mass and conservation of momentum which, under certain assumptions, can be simplified to

$$\pi_v - \pi_w = p_v^2 - p_w^2 = \alpha_a |q_a| q_a, \tag{3.3.3}$$

where α_a is the pressure loss coefficient. The pressure loss coefficient, α_a, depends on the material and diameter of the pipe and a few properties of the natural gas. As pipes allow bi-directional flow, the sign of the potential drop depends on the direction of the flow resulting in the absolute value of the flowrate variable q_a in the equation.

Short pipes: short pipes are used for modeling purposes to handle complicated contract situations and are modeled as lossless pipes, i.e., a short pipe a is a regular pipe with $\alpha_a = 0$.

Resistors: resistors are commonly used to model pressure or potential drop. In this work, we assume resistors behave in the same way as pipes in terms of potential drop. We

refer to [51] for alternative ways to model resistors.

Compressors: compressors are used to increase potential along an arc. There are many models proposed for compressors. In this chapter, we adopt the model used in [58]. For a compressor $a = (v, w)$, we use a binary variable z_a to indicate the on and off states where $z_a = 1$ indicates the compressor is on and $z_a = 0$ otherwise. When the compressor is on, it allows flow from v to w and increases the potential from v to w. When it is off, it does not allow any flow. As a result, we have the following relations for a compressor:

$$\pi_v - \pi_w \leq 0,\ q_a \geq 0,\ \text{if } z_a = 1 \tag{3.3.4}$$

$$q_a = 0,\ \text{if } z_a = 0. \tag{3.3.5}$$

Additionally, there are limits on potential ratio as follows:

$$\kappa_a^{\min} \pi_v \leq \pi_w \leq \kappa_a^{\max} \pi_v, \tag{3.3.6}$$

where $\kappa_a^{\min} = 1$ and $\kappa_a^{\max} \geq 1$ are typical for a compressor (see [59]). Furthermore, when a compressor $a = (v, w)$ is on, it can impose additional bounds on the potentials at nodes v and w. We denote those bounds by $(\pi_v^{\min})'$ and $(\pi_w^{\max})'$.

Valves: valves are incorporated in the network to join or separate two nodes. They allow bi-directional flow when they are on. A binary variable z_a is used to model the on and off states of valves. For a valve $a = (v, w)$, $z_a = 1$ indicates the valve is on and $z_a = 0$ otherwise. When a valve is on, the potentials at the two end nodes have to be equal. When a valve is off, it does not allow any flow. Formally, the constraints for a valve $a = (v, w)$ are expressed as follows:

$$\pi_v = \pi_w,\ q_a \text{ arbitrary},\ \text{if } z_a = 1 \tag{3.3.7}$$

$$q_a = 0,\ \pi_v, \pi_w \text{ arbitrary},\ \text{if } z_a = 0. \tag{3.3.8}$$

Control valves: contrary to a compressor, the presence of a control valve in the network results in potential relief. We adopt a similar model that is used for compressors. For a control valve $a = (v, w)$, a binary variable z_a is used to indicate its states. When it is on, it allows flow from v to w and causes a potential relief from v to w. When it is off, it does not allow any flow. We have the following model for the control valve:

$$\pi_v - \pi_w \geq 0,\ q_a \geq 0,\ \text{if } z_a = 1 \tag{3.3.9}$$

$$q_a = 0,\ \text{if } z_a = 0. \tag{3.3.10}$$

The limits on the potential relief are given by

$$\kappa_a^{\min} \pi_v \leq \pi_w \leq \kappa_a^{\max} \pi_v, \tag{3.3.11}$$

where $\kappa_a^{\min} > 0$ and $\kappa_a^{\max} \leq 1$ are typical for a control valve (see [59]). A control valve (v, w) can impose additional bounds on the potentials at nodes v and w when it is on. Similar to compressors, we denote those bounds by $(\pi_v^{\min})'$ and $(\pi_w^{\max})'$.

3.3.2 Design problem

We consider the design of a gas network for a given set of pipe locations (arcs), whose diameters we must decide. As discussed in the previous section, we have different system components and thus we divide the arc set $\mathcal{A}$ into $\mathcal{A} = A_p \cup A_{sp} \cup A_r \cup A_{cp} \cup A_v \cup A_{cv}$ where A_p, A_{sp}, A_r, A_{cp}, A_v, and A_{cv} are the set of pipes, short pipes, resistors, compressors, valves, and control valves respectively. We consider discrete diameter choices in our setting and denote the diameter choices by the set $[n] := \{1, 2, \ldots, n\}$. We use binary variables $z_{a,i}$, $a \in A_p$ and $i \in [n]$, to model the discrete diameter choices of the pipes. We further denote the length and diameter of the pipe $a \in A_p$ with the diameter choice $i \in [n]$ by l_a and $D_{a,i}$, respectively, and we use the same cost function, $f_{a,i}$, that is used in [58] and [59],

namely

$$f_{a,i} = l_a(1.04081^{-6} D_{a,i}^{2.5} + 11.2155). \tag{3.3.12}$$

Note a trade-off in the selection of the diameters. A larger diameter, on one hand, leads to a smaller potential drop coefficient and a higher maximum flowrate, while, on the other hand, leads to a larger cost $f_{a,i}$. We introduce a flow direction variable $x_a^{dir} \in \{0, 1\}$ for $a \in A_p \cup A_{sp} \cup A_r \cup A_v$ to account for the bidirectional flow. Recall that compressors and control valves only allow one flow direction. For a pipe $a \in A_p$, as a result of the multiple diameter choices, we have multiple flowrate variables $q_{a,i}$ for each $i \in [n]$. We decompose the flow into positive flow and negative flow, i.e., for $a \in A_p$, $q_{a,i} = q_{a,i}^+ - q_{a,i}^-$ and for $a \in A_{sp} \cup A_r \cup A_v$, $q_a = q_a^+ - q_a^-$. The maximum flowrate limit $q_a^{\max}$ can be defined individually for each diameter choice i as $q_{a,i}^{\max}$ by relation (3.3.2). Similarly, the potential drop coefficients $\alpha_{a,i}$ can be computed for each diameter choice. Furthermore, for a node $v \in \mathcal{V}$, we denote the set of incoming arcs and outgoing arcs by $A_2(v)$ and $A_1(v)$, respectively, i.e., $A_2(v) = \{a \in \mathcal{A} | a = (w, v)\}$ and $A_1(v) = \{a \in \mathcal{A} | a = (v, w)\}$. We use d_v to denote the demand or supply at a node $v \in \mathcal{V}$.

With this notation and technical background, we give a formulation to the design problem:

$$\min \sum_{a \in A_p} \sum_{i \in [n]} f_{a,i} z_{a,i} \tag{3.3.13}$$

$$\text{s.t.} \sum_{a \in A_2(v) \setminus A_p} q_a^+ - \sum_{a \in A_2(v) \setminus A_p \cup A_{cp} \cup A_{cv}} q_a^- - \left(\sum_{a \in A_1(v) \setminus A_p} q_a^+ - \sum_{a \in A_1(v) \setminus A_p \cup A_{cp} \cup A_{cv}} q_a^- \right)$$
$$+ \sum_{i \in [n]} \sum_{a \in A_2(v) \cap A_p} (q_{a,i}^+ - q_{a,i}^-) - \sum_{i \in [n]} \sum_{a \in A_1(v) \cap A_p} (q_{a,i}^+ - q_{a.i}^-) = d_v, \qquad v \in \mathcal{V} \tag{3.3.14}$$

$$\pi_v^{\min} \leq \pi_v \leq \pi_v^{\max}, \qquad v \in \mathcal{V} \tag{3.3.15}$$

$$x_a^{\text{dir}} \in \{0, 1\}, \qquad a \in A_p \cup A_{sp} \cup A_v \cup A_r \tag{3.3.16}$$

$$0 \leq q_{a,i}^-,\ q_{a,i}^+ \leq q_{a,i}^{\max} z_{a,i},\ \forall i \in [n], a \in A_p \tag{3.3.17}$$

$$\pi_v - \pi_w = \sum_{i \in [n]} \alpha_{a,i} (q_{a,i}^+)^2 - \sum_{i \in [n]} \alpha_{a,i} (q_{a,i}^-)^2, \qquad a \in A_p \tag{3.3.18}$$

$$0 \le q_{a,i}^{+} \le q_{a,i}^{\max} x_a^{\text{dir}}, \qquad a \in A_p, i \in [n] \tag{3.3.19}$$

$$0 \le q_{a,i}^{-} \le q_{a,i}^{\max} (1 - x_a^{\text{dir}}), \qquad a \in A_p, i \in [n] \tag{3.3.20}$$

$$\sum_{i \in [n]} z_{a,i} = 1, \qquad a \in A_p \tag{3.3.21}$$

$$\pi_v = \pi_w, \qquad a \in A_{sp} \tag{3.3.22}$$

$$0 \le q_a^{+} \le q_a^{\max} x_a^{\text{dir}}, \qquad a \in A_{sp} \tag{3.3.23}$$

$$0 \le q_a^{-} \le q_a^{\max} (1 - x_a^{\text{dir}}), \qquad a \in A_{sp} \tag{3.3.24}$$

$$\pi_v - \pi_w = \alpha_a (q_a^{+})^2 - \alpha_a (q_a^{-})^2, \qquad a \in A_r \tag{3.3.25}$$

$$0 \le q_a^{+} \le q_a^{\max} x_a^{\text{dir}}, \qquad a \in A_r \tag{3.3.26}$$

$$0 \le q_a^{-} \le q_a^{\max} (1 - x_a^{\text{dir}}), \qquad a \in A_r \tag{3.3.27}$$

$$\kappa_a^{\min} \pi_v - M(1 - z_a) \le \pi_w \le \kappa_a^{\max} \pi_v + M(1 - z_a), \qquad a \in A_{cp} \cup A_{cv} \tag{3.3.28}$$

$$0 \le q_a^{+} \le q_a^{\max} z_a, \qquad a \in A_{cp} \cup A_{cv} \tag{3.3.29}$$

$$(\pi_v^{\min})' - M(1 - z_a) \le \pi_v, \qquad a = (v, w) \in A_{cp} \cup A_{cv} \tag{3.3.30}$$

$$\pi_w \le (\pi_w^{\max})' + M(1 - z_a), \qquad a = (v, w) \in A_{cp} \cup A_{cv} \tag{3.3.31}$$

$$\pi_v - \pi_w \le M(1 - z_a), \qquad a \in A_v \tag{3.3.32}$$

$$\pi_v - \pi_w \ge -M(1 - z_a), \qquad a \in A_v \tag{3.3.33}$$

$$0 \le q_a^{-}, q_a^{+} \le q_a^{\max} z_a, \qquad a \in A_v \tag{3.3.34}$$

$$0 \le q_a^{+} \le q_a^{\max} x_a^{\text{dir}}, \qquad a \in A_v \tag{3.3.35}$$

$$0 \le q_a^{-} \le q_a^{\max} (1 - x_a^{\text{dir}}), \qquad a \in A_v. \tag{3.3.36}$$

In this model, the objective function (3.3.13) minimizes the construction cost of the pipes, also known as the budget. The scalar M represents a large number. For the constraints that involve M, we can alternatively write them in a nonlinear fashion, which eliminates the need for big-Ms. In particular, for constraints (3.3.28) and (3.3.30)-(3.3.31), we have

$$z_a \kappa_a^{\min} \pi_v \le \pi_w, \qquad a \in A_{cp} \cup A_{cv} \tag{3.3.37}$$

$$z_a \pi_w \leq \kappa_a^{\max} \pi_v, \qquad a \in A_{cp} \cup A_{cv} \tag{3.3.38}$$

$$z_a (\pi_v^{\min})' \leq \pi_v, \qquad a \in A_{cp} \cup A_{cv} \tag{3.3.39}$$

$$z_a \pi_w \leq (\pi_w^{\max})', \qquad a \in A_{cp} \cup A_{cv}, \tag{3.3.40}$$

and for constraints (3.3.32)-(3.3.33), we have

$$(\pi_v - \pi_w) z_a = 0, \qquad a \in A_v. \tag{3.3.41}$$

We group the constraints in blocks with block names and provide a summary in Table 3.1. We will refer to the corresponding set of constraints by their block name.

Note that the above formulation can be extended to the reinforcement problem by considering, for each existing pipe, an additional diameter choice with no cost along with potential loss equation (3.3.3) in which the potential loss coefficient is computed based on the diameter of the existing pipe.

3.4 Decomposition framework

We now present a decomposition framework to solve the design problem. The decomposition consists of three major components: primal bound loop, binary search on budget, and initial budget search. Before we present the details on each component, we introduce more background on the convex program introduced in [62] and adapted in [63], which was mentioned briefly in literature review.

3.4.1 CVXNLP

The convex program is called (CVXNLP) in [67]; we adopt the same name. We base the discussions of (CVXNLP) on a gas network in contrast to a water network in [67] in this section for completeness. For a network with only pipes, i.e., $\mathcal{A} = A_p$, (CVXNLP) is

Table 3.1: Constraint blocks

Constraints	Block names	Explanations
(3.3.14)	Flow_conserv	Flow conservation
(3.3.15)-(3.3.16)	Bound	Bounds on potentials (3.3.15); binary directions (3.3.16)
(3.3.17)-(3.3.21)	Pipe	Flow limits on diameter choices (3.3.17); potential drop (3.3.18); flow limits on directions (3.3.19)-(3.3.20); diameter selection (3.3.21)
(3.3.22)-(3.3.24)	Short_pipe	Potential (3.3.22); flow limits on directions (3.3.23)-(3.3.24)
(3.3.25)-(3.3.27)	Resistor	Potential drop (3.3.25); flow limit on directions (3.3.26)-(3.3.27)
(3.3.28)-(3.3.31)	Comp_and_ cont_valve	Depend on on/off states; Potential increase/relief limit (3.3.28); flow limit (3.3.29); additional bounds (3.3.30)-(3.3.31)
(3.3.32)-(3.3.36)	Valve	Depend on on/off states; Potential (3.3.32)-(3.3.33); flow limit (3.3.34); flow limits on directions (3.3.35)-(3.3.36)
(3.3.29), (3.3.37)-(3.3.40)	Comp_and_ cont_valve_nl	Same as Comp_and_cont_valve block
(3.3.34)-(3.3.36), (3.3.41)	Valve_nl	Same as Valve block

closely related to the following set of *network analysis equations*:

$$\pi_v - \pi_w = \text{sgn}(q_a)\phi(|q_a|), \qquad a \in A_p \tag{3.4.1}$$

$$\sum_{a \in A_2(v)} q_a - \sum_{a \in A_1(v)} q_a = d_v, \qquad v \in \mathcal{V}, \tag{3.4.2}$$

where $\text{sgn}(\cdot)$ is the sign function and $\phi(\cdot)$ is the potential loss function. In the network analysis equations, (3.4.1) is the potential drop equation and (3.4.2) is the flow conservation. (CVXNLP) is formally given by

$$\min \sum_{a \in A_p} \Phi(q_a^+) + \Phi(q_a^-) \tag{3.4.3}$$

$$\text{s.t.} \sum_{a \in A_2(v)} (q_a^+ - q_a^-) - \sum_{a \in A_1(v)} (q_a^+ - q_a^-) = d_v, \qquad v \in \mathcal{V} \tag{3.4.4}$$

$$0 \leq q_a^-, q_a^+, \qquad a \in A_p, \tag{3.4.5}$$

where $\Phi(\cdot)$ is defined by

$$\Phi(q) = \int_0^q \phi(q')dq'. \tag{3.4.6}$$

(CVXNLP) is formally linked to the network analysis equations by the following theorem.

Theorem 3.4.1. *If the potential loss function $\phi(\cdot)$ is strictly monotonically increasing function of flowrate, q, with $\phi(0) = 0$, then there exists a solution (π, q) to the network analysis equations if and only if there exists a solution $(\hat{q}^+, \hat{q}^-, \hat{\lambda}, \hat{\mu}^+, \hat{\mu}^-)$ to (CVXNLP) where λ, μ^+, and μ^- are dual variables to the flow conservation constraint (3.4.4) and bounds constraints (3.4.5), respectively.*

Proof. The proof is adapted from a proof in [67] and can be found in Appendix 3.6. □

The monotonicity assumption needed for the theorem to hold is commonly satisfied by gas networks. In addition, as a result of the equivalence between (CVXNLP) and the network analysis equations stated in Theorem 3.4.1, we can solve the convex (CVXNLP)

in lieu of the non-convex network analysis equations and obtain a solution (π, q). As there are no bounds enforced in the network analysis equations for π, we have to perform an additional step to verify that π satisfies the corresponding bounds.

3.4.2 Primal bound loop

The equivalence discussed in Section 3.4.1 motivates us to develop a decomposition procedure that solves a variant of (CVXNLP) as a master problem, while a subproblem is used to check for feasibility. We call this procedure the primal bound loop, which checks whether a budget C is feasible with respect to a nomination. The master problem, denoted by (P_m), is based on (CVXNLP) and is as follows:

$$(P_m) \quad \min \sum_{i\in[n]} \sum_{a\in A_p} \frac{\alpha_{a,i}}{3}(q^+_{a,i})^3 + \frac{\alpha_{a,i}}{3}(q^-_{a,i})^3 + \sum_{a\in A_r} \frac{\alpha_a}{3}(q^+_a)^3 + \frac{\alpha_a}{3}(q^-_a)^3 \tag{3.4.7}$$

$$\text{s.t. } (3.3.14), (3.3.17), (3.3.21), (3.3.29), (3.3.34)$$

$$0 \le q^-_a, q^+_a \le q^{\max}_a, \qquad a \in A_{sp} \cup A_r \tag{3.4.8}$$

$$\sum_{a\in A_p} \sum_{i\in[n]} f_{a,i} z_{a,i} \le C. \tag{3.4.9}$$

In this model, the objective function (3.4.7) extends (CVXNLP) to account for multiple flow variables $q^+_{a,i}$ for $a \in A_p$ and $i \in [n]$. Constraint (3.3.14) is the flow conservation. Constraints (3.3.17) and (3.3.21) on the pipes ensure one diameter choice is selected and the corresponding flow limit is enforced. Constraints (3.3.29) and (3.3.34) on the active system components ensure flows are only allowed when the corresponding binaries are on. Constraint (3.4.8) enforces the flow limit on the short pipes and resistors. The last constraint (3.4.9) is a budget constraint on the construction cost of pipes.

The above model differs from (CVXNLP) mainly in two ways. Firstly, we have introduced the diameter choices, $z_{a,i}$ for $a \in A_p$ and $i \in [n]$ and configurations for the active system components, z_a for $a \in A_{cp} \cup A_{cv} \cup A_v$. If the diameter choices and configurations

of the active system components are fixed, (P_m) resembles the original (CVXNLP). Secondly, we have a constraint to upper bound the construction cost of pipes by the budget C. We hope to obtain favorable diameter choices and active system component configurations from solving this modified (CVXNLP) due to the equivalence between (CVXNLP) and the network analysis equations shown in Theorem 3.4.1. In the solution of (P_m), we denote the optimal diameter choices by $z^*_{a,i}$ for $a \in A_p$ and $i \in [n]$ and the optimal active system configurations by z^*_a for $a \in A_{cp} \cup A_{cv} \cup A_v$. We can then compute the potential loss coefficient and the flow limit of each pipe as follows:

$$\alpha_a = \sum_{i\in[n]} \alpha_{a,i} z^*_{a,i}, \qquad a \in A_p \tag{3.4.10}$$

$$q_a^{\max} = \sum_{i\in[n]} q_{a,i}^{\max} z^*_{a,i}, \qquad a \in A_p. \tag{3.4.11}$$

For the subproblem, denoted by (P_s), since we have additional active system components for which the constraints governing their corresponding potential changes are not included in network analysis equations, we solve a variant of the nomination validation problem, instead of performing simple bound violation verifications, to check if the diameter choices and configurations of the active system components are feasible with respect to the nomination. (P_s) is given by:

$$\begin{aligned} (P_s) \quad & \text{Find } q_a^+, q_a^-, x_a^{\text{dir}}, \pi_v \\ & \text{s.t. } \texttt{Flow\_conserv} \\ & \quad\ \texttt{Bound} \\ & \quad\ \texttt{Pipe} \\ & \quad\ \texttt{Short\_pipe} \\ & \quad\ \texttt{Resistor} \\ & \quad\ \texttt{Comp\_and\_cont\_valve(\_nl)} \end{aligned} \tag{3.4.12}$$

`Valve(_nl).`

In this nomination validation problem, we solve a feasibility problem with seven blocks of constraints that are simplified from the blocks in Table 3.1. We list the changes to the constraints as follows.

`Flow_conserv:` with the diameter choices fixed, we only need two flow variables, q_a^+ and q_a^-, for each pipe $a \in A_p$ and the simplified flow conservation constraint is given by:

$$\sum_{a\in A_2(v)} q_a^+ - \sum_{a\in A_2(v)\setminus A_{cp}\cup A_{cv}} q_a^- - \left(\sum_{a\in A_1(v)} q_a^+ - \sum_{a\in A_1(v)\setminus A_{cp}\cup A_{cv}} q_a^- \right) = d_v, \qquad v \in \mathcal{V}. \tag{3.4.13}$$

`Pipe:` with the diameter choices determined, we compute the potential loss coefficients and flow limits, and we write the potential loss constraints as

$$\pi_v - \pi_w = \alpha_a (q_a^+)^2 - \alpha_a (q_a^-)^2, \qquad a \in A_p, \tag{3.4.14}$$

and use the computed flow limits $q_a^{\max}$ from (3.4.11) as:

$$0 \le q_a^+ \le q_a^{\max} x_a^{\text{dir}}, \qquad a \in A_p \tag{3.4.15}$$

$$0 \le q_a^- \le q_a^{\max} (1 - x_a^{\text{dir}}), \qquad a \in A_p. \tag{3.4.16}$$

`Comp_and_cont_valve(_nl):` we fix the compressor and control valve configurations obtained in (P_m). The constraints are then linear and free of M.

`Valve(_nl):` we fix the valve configurations obtained in (P_m). The constraints are then linear and free of M.

There are no changes to other constraints.

Note that this nomination validation problem is still non-convex due to constraint (3.4.14)

and the potential loss constraint for resistors. There can be two outcomes from solving this subproblem (P_s). If it is infeasible, we can add an integer no-good cut to the master problem (P_m) of the form:

$$\sum_{a\in A_{cp}\cup A_{cv}\cup A_v, z_a^*=0} z_a + \sum_{a\in A_{cp}\cup A_{cv}\cup A_v, z_a^*=1} (1-z_a) + \sum_{i\in[n]} \sum_{a\in A_p, z_{a,i}^*=0} z_{a,i} + \sum_{i\in[n]} \sum_{a\in A_p, z_{a,i}^*=1} (1-z_{a,i}) \geq 1. \tag{3.4.17}$$

If it is feasible, we call budget C a feasible budget with respect to the nomination.

The iterative procedure terminates when we obtain a feasible budget or when the master problem (P_m) becomes infeasible after adding some integer no-good cuts. In the latter case, we call budget C an infeasible budget.

3.4.3 Binary search on budget

As the goal of this design problem is to minimize the construction cost of the network, we propose a binary search procedure to do so. A feasible budget from the primal bound loop provides an upper bound, $\overline{C}$, on the budget while an infeasible budget provides a lower bound, $\underline{C}$, on the budget. We present the binary search in Algorithm 6. For the termination conditions of the binary search in Line 2, we consider a time limit, absolute gap ε_e, i.e, $\overline{C} - \underline{C} < \varepsilon_e$, or relative gap ε_r, i.e., $(\overline{C} - \underline{C})/\underline{C} < \varepsilon_r$.

3.4.4 Initial budget search

To obtain a better initial starting budget for the binary search, we propose the following initial budget search procedure. This procedure is again an iterative procedure involving a master problem and a subproblem. The master problem, denoted by (I_m), is given by

$$(I_m) \quad \min \sum_{a\in A_p} \sum_{i\in[n]} f_{a,i} z_{a,i} \tag{3.4.18}$$

Algorithm 6: Binary search of budget

Input: Initial budget C, upper bound $\overline{C} = \infty$ and valid lower bound $\underline{C}$ by budget initialization ▷ `see Section 3.4.4`
while *not terminated* **do**
  Solve problems (P_m) and (P_s) with budget C ▷ `primal bound loop`
  if C *is a feasible budget* **then**
    if $C < \overline{C}$ **then** $\overline{C} = C$;
    if $C/2 \leq \underline{C}$ **then** $C = (\underline{C} + C)/2$;
    else $C = C/2$;
  else if C *is an infeasible budget* **then**
    if $C > \underline{C}$ **then** $\underline{C} = C$;
    if $2C \geq \overline{C}$ **then** $C = (\overline{C} + \underline{C})/2$;
    else $C = 2C$;
  else ▷ `primal bound loop terminated due to time limit`
    if $2C \geq \overline{C}$ **then** $C = (\overline{C} + C)/2$;
    else $C = 2C$;
  end
end

$$\text{s.t. } (3.3.21)$$

$$z_{a,i} \in \{0,1\}, \qquad a \in A_p, i \in [n] \tag{3.4.19}$$

$$z_a \in \{0,1\}, \qquad a \in A_{cp}. \tag{3.4.20}$$

In this model, the objective function (3.4.18) is the same as the design model which minimizes the construction cost of the pipes. Constraint (3.3.21) allows exactly one diameter choice for each pipe to be selected. In (I_m), we only consider the selection of diameter choices and configurations of the compressors. This integer program aims to obtain the cheapest construction cost of the pipes along with the compressor configurations, and can be solved very quickly due to the much smaller feasible space and simpler structure compared to (P_m). We can similarly compute the potential loss coefficients and flow limits based on the optimal diameter choices as shown in (3.4.10) and (3.4.11).

The subproblem is a variant of the nomination validation problem which includes the configurations of valves and control valves to check if the diameter choices and compressor

configurations are feasible with respect to the nomination. The subproblem, denoted by (I_s), is the same as (P_s), except for the constraints on the control valves and the valves since we do not obtain configurations for them from the master problem (I_m) in contrast to (P_m). We list the changes from (P_s) to obtain (I_s).

Variables: we add the binary variables for the on and off states of control valves and valves.

Constraints: `Comp_and_cont_valve(_nl)`: we fix the configurations of compressors obtained in (I_m) and consequently the constraints for compressors are now linear and free of M. For the control valves, our preliminary studies suggest the use of `comp_and_cont_valve_nl` block.

`Valve(_nl)`: Our preliminary studies suggest the use of `Valve_nl` block for valves.

There are no changes to other constraints.

Solving the subproblem (I_s) has two outcomes. If the cheapest diameter choices and compressor configurations are infeasible with respect to the nomination, we add an integer no-good cut for that set of diameter choices and compressor configurations similar to (3.4.17) to the master problem (I_m) and resolve. Otherwise, we obtain the optimal budget, i.e., the optimal budget for this nomination is the corresponding objective value of (I_m). The initial budget search procedure can be run for a certain time or a fixed number of iterations. It produces an objective value below which there is no feasible budget, thus producing an initial dual bound. This lower bound on the optimal objective function value of the network design problem can be used to initialize the binary search.

3.5 Numerical experiments

3.5.1 Instances

Our numerical experiments are based on the Gaslib library Gaslib-582 network. The size of the problem is given in Tables 3.2 and 3.3. Depending on the nomination, a source node may have zero supply and a sink node may have zero demand.

Table 3.2: Nodes in Gaslib-582 network

Sources	Sinks	In-nodes
31	129	422

Table 3.3: Arcs in Gaslib-582 network

Pipes	Short pipes	Resistors	Compressors	Control valves	Valves
278	269	8	5	23	26

The nominations given with the Gaslib network in [46] are divided into five categories, namely, warm, mild, cool, cold, and freezing, to simulate the temperature conditions. There are two observations about the nominations. Firstly, as temperature conditions change from warm to freezing, the nominations become more demanding. There are more sinks with positive demands and the magnitudes of demands increase. Secondly, the nominations from the same temperature category vary much less compared to nominations across temperature categories. Therefore, we pick one nomination from each temperature category for the experiments. In all experiments, we vary the nomination by stress levels similar to [59]. In particular, we use the stress levels $\{0, 1,\ 0.5,\ 1.0,\ 1.5,\ 2.0\}$ and multiply each stress level by the demand d_v for each $v \in \mathcal{V}$ in a nomination to create an instance. For each pipe, based on the diameter given in the Gaslib network, we use multipliers from the set $\{0.8,\ 1.0,\ 1.3,\ 1.5\}$ to create 4 different diameter choices.

3.5.2 Implementation considerations and settings

We first provide some notes on the implementation of the primal bound loop. As the objective function (3.4.7) in the master problem (P_m) is cubic in the flow variables, there are several possible ways to implement it:

- There are nonlinear mixed integer program solvers that can take the master problem (P_m) as it is, for example, BARON ([68]) and SCIP ([69]).

- The cubic objective function is second-order conic representable. For each of the cubic terms in the flow variable, we can introduce an additional variable. Conse-

quently, we obtain a constraint in the form of $q^3 \leq t$ where q represents the flow variable and t represents the new variable that is an upper bound to q^3 and we can write the second-order conic representation of $q^3 \leq t$ by

$$s \geq 0,\ s+q \geq 0,\ (s+q)^2 \leq w,\ w^2 \leq t(s+q). \tag{3.5.1}$$

The resulting second-order conic program can be handled by specialized solvers such as MOSEK ([70]).

- To take advantage of the Gurobi's ([41]) improved capability in solving quadratic programs, for each of the cubic terms in the flow variable q in the objective function, we introduce an additional variable q_{qua} with $q_{qua} = q^2$. Consequently, we have a bilinear term qq_{qua} in the objective function with an additional constraint. The constraint $q_{qua} = q^2$ can be re-written into the convex constraint $q_{qua} \geq q^2$. Moreover, for the pipes, as we have binary variables corresponding to the diameter choices, the convex constraint $q_{qua} \geq q^2$ can be strengthened to $q_{qua}z \geq q^2$ by perspective strengthening ([64]) where z represents the binary variable for the diameter choice.

We implemented all three methods. Even though the original cubic formulation (used with BARON 22.9.1 ([71]) and SCIP 7.0.1 ([72])) and the second-order conic formulation (used with MOSEK 9.3.10 ([70])) are convex, these solvers tend to be slower due to the presence of the binary variables. On the other hand, Gurobi 9.5.1 ([41]) is able to handle the reformulation of the cubic objective function well. As a result, we decided to use Gurobi 9.5.1 to solve (P_m) and (P_s). This also gives us the opportunity to study the impact of perspective strengthening on the computational speed.

In addition, since we only use the values of the binary variables of the pipe diameter choices and active system component configurations from the master problem (P_m) to fix the corresponding variables in the subproblem (P_s), we do not need to solve the master problem (P_m) to optimality. We can either set a time limit or a non-default optimality gap

and we opt to use a time limit of 60 sec.

We run the experiments on a computer with an Intel i9 CPU (3.70GHz) with 64GB RAM. The computer runs the Ubuntu 20.04 LTS operating system. The framework is coded in Python with Pyomo. We use Gurobi 9.5.1 to solve problems (I_m) and (I_s) as well. Algorithm 7 shows the exact steps we use to solve the problem combining the procedures from the primal bound loop, the binary search on budget, and the initial budget search.

Algorithm 7: Overall procedure

1 Initial budget search (I_m) and (I_s) is run for 10 min. ▷ `Initial budget search phase`
2 **if** *a feasible budget is obtained* **then** terminate with the optimal budget for this nomination;
3 **else** ▷ `Binary search phase`
4 Set starting budget based on the returned value from initial budget search;
5 Binary search is run for 5 hr; for each candidate budget, primal bound loop (P_m) and (P_s) is run for 45 min to check if the budget is feasible.
6 **end**
7 **return** $\overline{C}$ *and* $\underline{C}$ *from binary search*

A note on the time limit for the initial budget search. We performed studies to extend the time limit to longer than 10 min and the improvement on the returned value is not significant. As a result, we decided to limit the initial budget search to 10 min.

3.5.3 Results

In this section, we present the computational results. In addition to the results from our proposed framework, we provide some discussions about the compact formulation and another approach adapted from [59] using computational studies on the nomination warm_31, which comes from the least demanding temperature categories.

We first discuss the performance of the compact formulation with objective function (3.3.13) along with constraint blocks of `Flow_conserv`, `Bound`, `Pipe`, `Short_pipe`, `Resistor`, `Comp_and_cont_valve` or `Comp_and_cont_valve_nl`, and `Valve` or `Valve_nl`. Solvers that support the non-convex integer program are considered. SCIP and

BARON are able to find a lower bound very quickly, but they tend to be slower in finding a feasible solution to close the gap after hours of computation. We also discovered that Gurobi at times returns solutions with large constraint violations for the compact formulation. Note that CPLEX currently only supports non-convexity in the objective function and hence is not applicable.

The papers by [59] and [61] are two recent works on gas network expansion. The work of [61] specifically considers a tree-like network while the Gaslib-582 network contains cycles. Although [59] mainly focuses on the reinforcement problem, the approach can be modified to tackle the design problem. The authors in [59] construct a convex mixed-integer second-order conic (MISOC) relaxation of the reinforcement formulation, and fix all the binary decision variables after solving the relaxation to obtain a nomination validation problem. The resulting nonlinear program is then solved by a solver to obtain a solution if the nonlinear program is feasible. If the nonlinear program is infeasible, then the authors propose to use any feasible solutions to the relaxation. We adapt a similar MISOC relaxation for the pipes and resistors. We leave the details of this relaxation in Appendix 3.6. In our computations, following the procedure and solving the nomination validation problem worked well for instances with up to 40 nodes, but did not yield feasible solutions for larger instances.

Next, we present the results from our framework. We define gap to be

$$\text{gap} = \frac{\overline{C} - \underline{C}}{\underline{C}}. \tag{3.5.2}$$

We present the detailed results for one of the nominations, warm_31, in Table 3.4 and compare the gaps from implementations with and without perspective strengthening for all nominations in Table 3.5 where $\overline{C}$ and $\underline{C}$ are reported in 10^6, and gaps are reported in $\%$. The column "Imp" reports the percentage improvements from perspective strengthening. The detailed results for the rest of the nominations are provided in Appendix 3.6.

From the results, we see that our framework is able to find a feasible budget for all 25 instances. In particular, it provides an optimal budget for 13 instances and a budget with less than 25% gaps for another four instances. There are a few instances where we reached the time limit with large gaps. We mark these instances in bold. These instances are with higher stress levels and/or worse temperature conditions. As we increase the stress level and/or deteriorate the temperature conditions (from warm to freezing) making the nominations more demanding, we observe that it becomes more difficult to find feasible solutions in the primal bound loops to prove a feasible budget and thus close the gap by binary search. The primal bound loops hit the time limit much more often. In addition, for all instances, we are not able to prove infeasible budget from the primal bound loop. While a large number of binary solutions are feasible to the master problem (P_m), each integer no-good cut only invalidates one of them. As a result, the lower bounds on budget, $\underline{C}$, are almost the same across different nominations and stress levels.

Furthermore, the perspective strengthening is shown to be effective in closing the gaps for higher stress levels. There are only two instances (mild_3838 and cold_4218 at stress level of 1.5) for which the implementation without perspective strengthening achieves better gaps than the implementation with perspective strengthening. The average and largest improvements from perspective strengthening are about 31% (excluding the instances that are solved to optimality both with and without perspective strengthening) and 86%, respectively from perspective strengthening. The improvements are all due to obtaining better feasible solutions.

Table 3.4: Computational results with and without perspective strengthening for warm_31

Stress	Without perspective strengthening			With perspective strengthening			Imp(%)
	$\overline{C}$	$\underline{C}$	gap	$\overline{C}$	$\underline{C}$	gap	
0.1	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
0.5	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.0	12620.2	12585.6	0.27	12602.9	12585.6	0.13	50
1.5	14797.9	12585.6	17.57	14355.5	12585.6	14.06	20
2.0	19812.3	12585.6	57.42	18091.8	12585.6	43.75	24

Table 3.5: Gap in (%) without (w/o) and with (w) perspective strengthening

Stress	warm_31		mild_3838		cool_2803		cold_4218		freezing_188	
	w/o	w	w/o	w	w/o	w	w/o	w	w/o	w
0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.0	0.27	0.13	0.00	0.00	4.78	0.68	18.75	14.35	38.27	24.99
1.5	17.57	14.06	32.81	37.50	51.56	24.61	**96.87**	**162.49**	**237.52**	**162.49**
2.0	57.42	43.75	**206.25**	**87.50**	**199.99**	57.42	**250.00**	**162.49**	**287.89**	**206.25**

3.6 Conclusion

In conclusion, we study the gas network design problem, where diameter choices of pipes and active system component configurations are decided. We propose a decomposition framework to solve the problem. In particular, in the primal bound loop of the framework, for a given budget, we modify a convex NLP formulation to construct master problems to obtain favorable diameter choices and active system component configurations, and validate their feasibility in the subproblem. Binary search is performed as an outer loop to minimize the budget. We also proposed a procedure to obtain a good initial budget for the binary search. The proposed framework was tested on the Gaslib-582 network and instances were created from combining nominations under different temperature conditions and stress level multipliers. The computational results show that the framework is able to find an optimal budget in many cases.

There are a few future directions that could be explored. The cost of operating the network may be an interest from an operator's perspective. Our framework can be adapted and applied to incorporate the cost of operations with simple modifications.

Acknowledgement

This work was conducted as part of the Institute for the Design of Advanced Energy Systems (IDAES) with support through the Simulation-Based Engineering, Crosscutting Research Program and the Solid Oxide Fuel Cell Program's Integrated Energy Systems thrust within the U.S. Department of Energy's Office of Fossil Energy and Carbon Management.

Appendix A: Proof of Theorem 3.4.1

Proof. We first consider the if part. Suppose that $(\hat{q}^+, \hat{q}^-, \hat{\lambda}, \hat{\mu}^+, \hat{\mu}^-)$ solves (CVXNLP). Consider the first-order stationary conditions for (CVXNLP) as follows:

$$\phi(\hat{q}_a^+) - \hat{\mu}_a^+ - \hat{\lambda}_v + \hat{\lambda}_w = 0, \qquad a = (v, w) \in A_p \tag{3.6.1}$$

$$\phi(\hat{q}_a^-) - \hat{\mu}_a^- + \hat{\lambda}_v - \hat{\lambda}_w = 0, \qquad a = (v, w) \in A_p \tag{3.6.2}$$

$$\hat{q}_a^+, \hat{\mu}_a^+ \geq 0, \qquad a = (v, w) \in A_p \tag{3.6.3}$$

$$\hat{q}_a^+ \cdot \hat{\mu}_a^+ = 0, \qquad a = (v, w) \in A_p \tag{3.6.4}$$

$$\hat{q}_a^-, \hat{\mu}_a^- \geq 0, \qquad a = (v, w) \in A_p \tag{3.6.5}$$

$$\hat{q}_a^- \cdot \hat{\mu}_a^- = 0, \qquad a = (v, w) \in A_p \tag{3.6.6}$$

$$\sum_{a \in A_2(v)} (\hat{q}_a^+ - \hat{q}_a^-) - \sum_{a \in A_1(v)} (\hat{q}_a^+ - \hat{q}_a^-) = d_v, \ v \in \mathcal{V}. \tag{3.6.7}$$

First, it cannot happen that $\hat{q}_a^+, \hat{q}_a^- > 0$ for any $a \in A_p$, otherwise, we can define

$$\tilde{q}_a^+ = \max\{\hat{q}_a^+ - \hat{q}_a^-, 0\}, \ \tilde{q}_a^- = \max\{0, \hat{q}_a^- - \hat{q}_a^+\}, \tag{3.6.8}$$

where $\tilde{q}_a^+ \leq \hat{q}_a^+$ and $\tilde{q}_a^- \leq \hat{q}_a^-$. The new flow values $\tilde{q}_a^+$ and $\tilde{q}_a^-$ are feasible and because of the strict monotonicity of $\phi(\cdot)$, they result in a smaller objective value which contradicts the optimality of $\hat{q}^+$ and $\hat{q}^-$. Furthermore, the complementary slackness conditions imply that, if $\hat{q}_a^+$ (or $\hat{q}_a^-$) > 0, then $\hat{\mu}_a^+$ (or $\hat{\mu}_a^-$) $= 0$. If $\hat{q}_a^+ = \hat{q}_a^- = 0$ for some a, then adding (3.6.1) and (3.6.2) gives

$$\hat{\mu}_a^+ + \hat{\mu}_a^- = 0 \implies \hat{\mu}_a^+ = \hat{\mu}_a^- = 0. \tag{3.6.9}$$

Consequently, we can simplify (3.6.1) and (3.6.2) by differentiating the cases on q_a^+ and

q_a^- to be

$$\phi(\hat{q}_a^+) - \hat{\lambda}_v + \hat{\lambda}_w = 0, \qquad a = (v, w) : \hat{q}_a^+ > 0 \tag{3.6.10}$$

$$\phi(\hat{q}_a^-) + \hat{\lambda}_v - \hat{\lambda}_w = 0, \qquad a = (v, w) : \hat{q}_a^- > 0 \tag{3.6.11}$$

$$\hat{\lambda}_v - \hat{\lambda}_w = 0, \qquad a = (v, w) : \hat{q}_a^+ = \hat{q}_a^- = 0. \tag{3.6.12}$$

Now define (π, q) as

$$\pi_v = \hat{\lambda}_v, \qquad v \in \mathcal{V} \tag{3.6.13}$$

$$q_a = \hat{q}_a^+ - \hat{q}_a^-, \qquad a \in A_p, \tag{3.6.14}$$

and we see (π, q) satisfies the network analysis equations.

Now we consider the only if part. Suppose that (π, q) solves the network analysis equations. We define the following:

$$\hat{q}_a^+ = \max\{0, q_a\}, \qquad a \in A_p \tag{3.6.15}$$

$$\hat{q}_a^- = |\min\{0, q_a\}|, \qquad a \in A_p \tag{3.6.16}$$

$$\hat{\lambda}_v = \pi_v, \qquad v \in \mathcal{V} \tag{3.6.17}$$

$$\hat{\mu}_a^+ = \max\{0, \pi_w - \pi_v + \phi(\hat{q}_a^+)\}, \qquad a = (v, w) \in A_p \tag{3.6.18}$$

$$\hat{\mu}_a^- = \max\{0, \pi_v - \pi_w + \phi(\hat{q}_a^-)\}, \qquad a = (v, w) \in A_p. \tag{3.6.19}$$

Then $(\hat{q}^+, \hat{q}^-, \hat{\lambda}, \hat{\mu}^+, \hat{\mu}^-)$ satisfies the first-order stationary conditions. To see this, we first verify that, when $q_a \geq 0$, then $\hat{q}_a^+ = q_a \geq 0$ and $\hat{q}_a^- = 0$. From the potential loss equation (3.4.1) in network analysis equations, we have that: $\pi_v - \pi_w = \phi(q_a) = \phi(\hat{q}_a^+)$. Consequently, we have

$$\hat{\mu}_a^+ = \max\{0, \pi_w - \pi_v + \phi(\hat{q}_a^+)\} = 0 \tag{3.6.20}$$

$$\hat{\mu}_a^- = \max\{0, \pi_v - \pi_w + \phi(\hat{q}_a^-)\} = \max\{0, \phi(\hat{q}_a^+) + \phi(0)\} = \phi(\hat{q}_a^+) \geq 0, \tag{3.6.21}$$

and

$$\phi(\hat{q}_a^+) - \hat{\mu}_a^+ - \hat{\lambda}_v + \hat{\lambda}_w = \phi(\hat{q}_a^+) - 0 - \pi_v + \pi_w = 0 \tag{3.6.22}$$

$$\phi(\hat{q}_a^-) - \hat{\mu}_a^- + \hat{\lambda}_v - \hat{\lambda}_w = \phi(0) - \phi(\hat{q}_a^+) + \pi_v - \pi_w = 0. \tag{3.6.23}$$

Similarly, we can verify for $q_a < 0$. Furthermore, the strict monotonically increasing property of $\phi(\cdot)$ implies the convexity of $\Phi(\cdot)$. The constraints in (CVXNLP) are linear and thus (CVXNLP) is convex. The satisfaction of the first-order stationary conditions is necessary and sufficient for (π, q) to be an optimal solution to (CVXNLP) and it is the unique optimal solution due to the convexity. □

Appendix B: Mixed-integer second-order conic (MISOC) relaxation

The relaxations are constructed for the pipes and resistors. For the pipes, instead of decomposing the flow variables $q_{a,i}$ into $q_{a,i}^+$ and $q_{a,i}^-$, we define two binary variables x_a^+ and x_a^- for the flow directions and enforce $x_a^+ + x_a^- = 1$. If $x_a^+ = 1$, then $q_{a,i} \geq 0$ and if $x_a^- = 1$, then $q_{a,i} < 0$. In addition, we create multiple potential variables $\pi_{v,i}$ and $\pi_{w,i}$ for $a = (v, w) \in A_p$ and $i \in [n]$. Now consider a pipe $a = (v, w)$ and a diameter choice i, we can write the potential loss as

$$(x_a^+ - x_a^-)(\pi_{v,i} - \pi_{w,i}) = \alpha_{a,i} q_{a,i}^2. \tag{3.6.24}$$

The left-hand side of (3.6.24) is bilinear. If we define $\gamma_{a,i} = (x_a^+ - x_a^-)(\pi_{v,i} - \pi_{w,i})$, we can write the standard McCormick relaxation for $\gamma_{a,i} = (x_a^+ - x_a^-)(\pi_{v,i} - \pi_{w,i})$ by

$$\gamma_{a,i} \geq \pi_{w,i} - \pi_{v,i} + (\pi_v^{\min} - \pi_w^{\max})(x_a^+ - x_a^- + 1) \tag{3.6.25}$$

$$\gamma_{a,i} \geq \pi_{v,i} - \pi_{w,i} + (\pi_v^{\max} - \pi_w^{\min})(x_a^+ - x_a^- - 1) \tag{3.6.26}$$

$$\gamma_{a,i} \geq \pi_{w,i} - \pi_{v,i} + (\pi_v^{\max} - \pi_w^{\min})(x_a^+ - x_a^- + 1) \tag{3.6.27}$$

$$\gamma_{a,i} \geq \pi_{v,i} - \pi_{w,i} + (\pi_v^{\min} - \pi_w^{\max})(x_a^+ - x_a^- - 1). \tag{3.6.28}$$

With $\gamma_{a,i}$ defined, constraint (3.6.24) can be written as $\gamma_{a,i} = \alpha_{a,i} q_{a,i}^2$ and can be further relaxed to become convex as follows:

$$\gamma_{a,i} \geq \alpha_{a,i} q_{a,i}^2. \tag{3.6.29}$$

Applying perspective strengthening to the relaxed constraint gives

$$z_{a,i}\gamma_{a,i} \geq \alpha_{a,i} q_{a,i}^2. \tag{3.6.30}$$

Now the potential loss constraint (3.3.18) for pipes becomes

$$\pi_v - \pi_w = \sum_{i \in [n]} \gamma_{a,i}. \tag{3.6.31}$$

We can create similar relaxations for the resistors. For a resistor $a = (v, w) \in A_r$, we have

$$\gamma_a \geq \pi_w - \pi_v + (\pi_v^{\min} - \pi_w^{\max})(x_a^+ - x_a^- + 1) \tag{3.6.32}$$

$$\gamma_a \geq \pi_v - \pi_w + (\pi_v^{\max} - \pi_w^{\min})(x_a^+ - x_a^- - 1) \tag{3.6.33}$$

$$\gamma_a \geq \pi_w - \pi_v + (\pi_v^{\max} - \pi_w^{\min})(x_a^+ - x_a^- + 1) \tag{3.6.34}$$

$$\gamma_a \geq \pi_v - \pi_w + (\pi_v^{\min} - \pi_w^{\max})(x_a^+ - x_a^- - 1) \tag{3.6.35}$$

$$\gamma_a \geq \alpha_a q_a^2. \tag{3.6.36}$$

Additionally, the binary variables x_a^{dir} in constraints (3.3.19)-(3.3.20) and (3.3.26)-(3.3.27) that govern flow limits on directions are replaced by x_a^+ and x_a^- correspondingly. We keep

the rest of constraints unchanged and obtain a convex MISOC relaxation of the design problem as a result.

Appendix C: Detailed computational results

All values of $\overline{C}$ and $\underline{C}$ are reported in 10^6 and gaps are reported in $\%$. The column "Imp" reports the percentage improvements from perspective strengthening. Instances marked in bold are those with large gaps at time limit.

Table 3.6: Computational results with and without perspective strengthening for mild_3838

Stress	Without perspective strengthening			With perspective strengthening			Imp(%)
	$\overline{C}$	$\underline{C}$	gap	$\overline{C}$	$\underline{C}$	gap	
0.1	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
0.5	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.0	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.5	16715.4	12585.6	32.81	17305.3	12585.6	37.50	-14
2.0	**38543.6**	**12585.6**	**206.25**	**23598.1**	**12585.6**	**87.50**	58

Table 3.7: Computational results with and without perspective strengthening for cool_2803

Stress	Without perspective strengthening			With perspective strengthening			Imp(%)
	$\overline{C}$	$\underline{C}$	gap	$\overline{C}$	$\underline{C}$	gap	
0.1	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
0.5	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.0	13187.9	12585.6	4.78	12671.6	12585.6	0.68	86
1.5	19075.2	12585.6	51.56	15683.0	12585.6	24.61	52
2.0	**37756.9**	**12585.6**	**199.99**	19812.6	12585.6	57.42	71

Table 3.8: Computational results with and without perspective strengthening for cold_4218

Stress	Without perspective strengthening			With perspective strengthening			Imp(%)
	$\overline{C}$	$\underline{C}$	gap	$\overline{C}$	$\underline{C}$	gap	
0.1	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
0.5	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.0	14945.5	12585.6	18.75	14392.3	12585.6	14.36	23
1.5	**24778.0**	**12585.6**	**96.87**	**33037.2**	**12585.6**	**162.49**	-68
2.0	**44049.9**	**12585.6**	**250.00**	**33037.2**	**12585.6**	**162.49**	35

Table 3.9: Computational results with and without perspective strengthening for freezing_188

Stress	Without perspective strengthening			With perspective strengthening			Imp(%)
	$\overline{C}$	$\underline{C}$	gap	$\overline{C}$	$\underline{C}$	gap	
0.1	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
0.5	12585.2	12585.2	0.00	12585.2	12585.2	0.00	-
1.0	17402.4	12585.6	38.27	15732.0	12585.6	24.99	35
1.5	**42479.6**	**12585.6**	**237.52**	**33037.2**	**12585.6**	**162.49**	32
2.0	**48818.7**	**12585.6**	**287.89**	**38543.6**	**12585.6**	**206.25**	28

CHAPTER 4
OPTIMIZING THE DESIGNS AND OPERATIONS OF WATER NETWORKS: A DECOMPOSITION APPROACH

4.1 Introduction

Water is a vital resource for both residential and industrial usage around the world. It is sourced from various natural sources and requires a distribution network for its transportation. The type of water network used can vary depending on the characteristics of the water sources. In 2019, the total capital spending on water infrastructure was approximately $\$48$ billion. However, it is projected that a total of $\$129$ billion will be needed to ensure continued access to sufficient water in the years to come [73]. Our work explores the challenges associated with the management of the so-called "produced water" that is co-produced when oil and gas are recovered from reservoirs. Reports indicate that the volume of produced water continues to grow while disposal capacities are decreasing [74], necessitating improved treatment and reuse options, as well as more efficient designs and operations of water networks.

We highlight several key characteristics for produced water networks. Firstly, the volume of water produced typically starts high and gradually decreases over time. This temporal trend of the amount of water produced often leads to a multiple time period formulation. Secondly, as produced water is commonly co-produced with oil and gas in a basin where the network is heavily influenced by the elevations of the nodes. Such elevation change between two nodes in the network introduces additional pressure change that has to be included into the problem formulation. Lastly, pump stations have to be installed at some locations of the network to boost the pressure of the water to avoid violations of the bounds of the pressure. There is commonly an upper limit on the number of pump stations that can

be installed. Therefore, the locations of pump stations are design decisions that must be carefully considered.

In addition, we consider the design and operation aspects of water networks in one problem in this chapter. Generally, the design aspect considers pipe sizing and placements of pump stations while the operation aspect considers a multiple time period formulation that accounts for the temporal changes in supply and demand, and considers the scheduling of the installed pump stations.

The remainder of this chapter is organized as follows. We review the relevant literature in Section 4.2. Section 4.3 provides the technical background and a compact formulation for the problem. Section 4.4 discusses two algorithms to obtain primal solutions. In particular, Section 4.4.1 presents a similar decomposition framework to what is proposed in chapter 3 and Section 4.4.2 is based on a time decomposition of the compact formulation. We present the computational experiments and discuss the results in Section 4.5. Lastly, we conclude this chapter and give some future directions in Section 4.6.

4.2 Literature review

The topic of water network design is popular in the literature. For example, [75] provide a comprehensive review of the types of problems in water system design. [76] and [77] focus on the solution methodologies and algorithms used to solve water network design problems. We refer interested readers to these works for broader knowledge.

We first review some relevant literature on the design problem and the operation problem. For the design problem, [78] considers a design problem to select the diameters of the pipes in a purely gravity-fed network with no pump stations or relief valves. They present a Mixed-Integer Nonlinear Programming (MINLP) formulation and focus on reformulations and implementation considerations to optimize the performance of `Bonmin` [79, 80]. The numerical experiments are performed using benchmark networks found in MIPLIB [81]. [67] also considers a design problem on pipe diameter selections without

pump stations or relief valves. A similar MINLP model is presented in which the nonlinear constraints governing the water flow are linearized. The author then uses a linearization-based LP/NLP-BB framework to solve the model and proposes a new leaf-node problem for the framework to correct the discrepancies introduced by linearization. The same set of benchmark networks from [78] is used in the numerical experiments. Later, [82] produces a follow-up study to propose a new formulation on the leaf-node problem from the earlier study [67]. Numerical results from the new formulation show improvements for several networks. [83] studies a slightly different design problem on the optimal placements of relief valves to minimize the average zone pressure. They use a MINLP model to investigate other linear relaxation schemes using a tailored domain reduction procedure to strengthen the relaxations. Computational experiments are performed using benchmark networks and include one real-world network from the UK. For the operation problem, [84] assumes that a fixed topology is given and consider the pump scheduling problem. The authors propose a Lagrangian decomposition approach to decouple the constraints for different periods in a MINLP formulation and allow the smaller subproblems to be solved separately. The authors also propose a simulation-based heuristic to account for additional operation constraints on the pump stations. Numerical results are obtained on two networks with up to 47 nodes. [85] uses a LP/NLP-based BB framework to study the pump scheduling problem. They propose a linear relaxation of the original non-convex formulation to be use in a branch-and-bound framework. They also propose a specialized primal heuristic to repair near-feasible integer solutions from the linear relaxation and improve computational efficiency.

It is common for the operation problem to include the so-called minimum-up and minimum-down constraints that model the technical requirements of the pump stations, sometimes known as the additional operation constraints. The minimum-up (resp. down) constraints indicate once a pump station is switched on (resp. off), it has to stay on (resp. off) for a given minimum number of periods. There have been many detailed studies of

the polytope formed by these constraints in the power system design domain, for example, [86]. A few sets of the minimum-up and minimum-down constraints have been proposed that differ on the number of variables used. [87] studies the "one-variable" variant of the polytope while [88] studies the "two-variables" variant. A discussion and a comparison across the variants can be found in [86].

Lastly, time decomposition can be used to reduce the complexity and size when solving multiple time period problems. In such an approach, the original problem is divided into smaller problems, each considering only a subset of the time periods. Rolling horizon is one of popular time decomposition schemes and a recent study on the properties of rolling horizon for multiple time period optimization can be found in [89].

4.3 Problem description

4.3.1 Technical background

In this section, we review the necessary background information on modeling the water network using only the components in our setting. We denote the set of T time periods by $\{1, \ldots, T\}$. We use a directed graph $G = (\mathcal{V}, \mathcal{A})$ to represent the water network. Each vertex $v \in \mathcal{V}$ can be a customer with demand, a reservoir with supply, or an in-node with neither demand nor supply. There is a pressure variable $p_{v,t}$ associated with the vertex in each time period $t \in \{1, \ldots, T\}$, which are lower and upper bounded by $p_v^{\min}$ and $p_v^{\max}$, respectively. Each arc $a \in \mathcal{A}$ represents a pipe and we use the set A_p to denote the set of pipes, i.e., $\mathcal{A} = A_p$.

Pipes: A pipe $a = (v, w)$ is specified by its length l_a, diameter D_a, and material properties. We use $q_{a,t}$ to denote the volumetric flowrate for the pipe in period $t \in \{1, \ldots, T\}$. The maximum flowrate, $q_a^{\max}$ is proportional to its cross-section area $A = \pi D_a^2/4$. The pipe allows bi-directional water flows, and a constraint reflecting the max flowrate and

bi-directional flow is given by

$$-q_a^{\max} \leq q_{a,t} \leq q_a^{\max}, \quad t \in \{1, \ldots, T\}. \tag{4.3.1}$$

Before we discuss pressure changes across the pipe, recall that the locations of pump stations are design decisions. Additionally, we assume that each pipe has a relief valve. Consequently, in addition to the pressure change due to friction and elevation, additional pressure increase from a pump station or pressure relief from a relief valve can be incurred. We begin the discussion with the pressure loss due to friction. Water flow in a pipe is governed by a set of partial differential equations. Under steady-state flow and other technical assumptions, we can simplify the partial differential equations to a set of nonlinear equations. There are two variants. The first one is the Hazen-Williams equation, which is given by

$$p_{v,t} - p_{w,t} = \frac{10.704 l_a}{C^{1.852} D_a^{4.87}} \rho g |q_{a,t}| q_{a,t}^{0.852}, \quad t \in \{1, \ldots, T\} \tag{4.3.2}$$

where C is the Hazen-Williams constant which is dependent on the material properties of the pipe, ρ is the density of water, and g is the gravitational acceleration constant. The second one is the Darcy equation which is given by

$$p_{v,t} - p_{w,t} = f_D \frac{8 l_a}{\pi^2 g D_a^5} \rho g |q_{a,t}| q_{a,t}, \quad t \in \{1, \ldots, T\} \tag{4.3.3}$$

where f_D is the friction coefficient that is dependent on the Reynolds number of water flow and other material properties of the pipe. We can write both equations in a simpler form as

$$p_{v,t} - p_{w,t} = \alpha_a |q_{a,t}| q_{a,t}^{\eta}, \quad t \in \{1, \ldots, T\} \tag{4.3.4}$$

where we call α_a the pressure loss coefficient and η is 0.852 or 1. Elevation change across a pipe introduces a pressure change that is proportional to the elevation change. Formally, if we denote the elevation at v and w by e_v and e_w respectively, the pressure change induced

by the elevation change is given by

$$\delta_a = (e_v - e_w)\rho g. \tag{4.3.5}$$

Note that δ_a is constant and can be computed before solving the problem.

Next, to model the decision on whether to install a pump station on this pipe, we use a binary variable, $z_{I,a}$, where $z_{I,a} = 1$ if a pump station is installed and 0 otherwise. In addition, we introduce two additional continuous variables, $\Delta_{I,a,t}$ and $\Delta_{R,a,t}$, to indicate the amount of pressure increase induced by a pump station and the amount of pressure relief induced by a relief valve in period $t \in \{1, \ldots, T\}$ respectively. The pressure increase induced by a pump station is upper bounded by $\overline{\Delta}_{I,a}$. Generally, the upper bound on pressure relief induced by a relief valve can be very large and thus we do not have an upper bound on $\Delta_{R,a,t}$.

As a result, the total pressure change across a pipe due to friction, elevation, and additional pressure increase or relief is given by

$$p_{v,t} - p_{w,t} = \alpha_a |q_{a,t}| q_{a,t}^{\eta} + \delta_a - \Delta_{I,a,t} + \Delta_{R,a,t}, \quad t \in \{1, \ldots, T\}. \tag{4.3.6}$$

Constraints on pump stations: We denote the upper bound on the number of pump stations for the whole network by N. We write a constraint as follows:

$$\sum_{a \in A_p} z_{I,a} \leq N. \tag{4.3.7}$$

Next, we write the so-called minimum-up and minimum-down constraints. Specifically, once a pump station $a \in A_p$ is switched on (resp. off), it must stay on (resp. off) for a total of τ_o (resp. τ_f) periods. We decide to the "one-variable" variant of the constraints (see Section 4.2) as we do not consider start-up costs. We use binary variables $\xi_{a,t}$ to indicate the status of the pump station, where $\xi_{a,t}$ equals to 1 if the pump station is on in period t

and 0 otherwise. The minimum-up and minimum-down constraints are given by

$$\xi_{a,t} \leq z_{I,a}, \qquad t \in \{1,\ldots,T\} \tag{4.3.8}$$

$$\xi_{a,t} - \xi_{a,t-1} \leq \xi_{a,\tau}, \qquad t \in \{2,\ldots,T\}, \tau \in \{t+1,\ldots,\min\{t+\tau_o, T\}\} \tag{4.3.9}$$

$$\xi_{a,t-1} - \xi_{a,t} \leq 1 - \xi_{a,\tau}, \quad t \in \{2,\ldots,T\}, \tau \in \{t+1,\ldots,\min\{t+\tau_f, T\}\}. \tag{4.3.10}$$

Additionally, we consider an upper bound, M_a, on the number of periods a pump station is on to simulate the operational cost constraint. Formally, we have the following

$$\sum_{t\in\{1,\ldots,T\}} \xi_{a,t} \leq M_a. \tag{4.3.11}$$

Note that the upper bounds, M_a and $M_{a'}$, can differ for two pipes $a, a' \in A_p$.

Reservoirs: Reservoirs are the common sources for the supply of water into the network. It is common practice to assume that pressures at reservoirs are fixed. Consequently, if we denote the set of reservoirs as $V^{\text{src}} \subset \mathcal{V}$, we have that

$$p_{v,t} = p^{\text{src}}_{v,t}, \quad v \in V^{\text{src}},\ t \in \{1,\ldots,T\}, \tag{4.3.12}$$

where $p^{\text{src}}_{v,t}$ are the fixed pressure values.

4.3.2 A summary on problem formulation

We consider discrete diameter choices for the pipes by the set $[n] := \{1,2,\ldots,n\}$ and use binary variables $z_{a,i}$ for $a \in A_p$ and $i \in [n]$ to indicate the diameter choices. The diameter value corresponding to $z_{a,i}$ is denoted by $D_{a,i}$. We denote the fixed cost of constructing a pipe with a diameter $D_{a,i}$ by $f_{a,i}$. In addition, we create copies of flow variables $q_{a,t}$ for different diameters as $q_{a,t,i}$ and differentiate the maximum flowrate $q^{\text{max}}_{a,i}$ for each diameter choice. As pipes allow bi-directional water flows, we introduce binary flow direction variables $x^{\text{dir}}_{a,t}$ to indicate the flow direction for $a \in A_p$ in time period t

and decompose the flows into positive and negative flows. As a result, the flow variables are $q^+_{a,t,i}$ and $q^-_{a,t,i}$ for pipes. For each vertex $v \in \mathcal{V}$, we denote the set of incoming and outgoing arcs by $A_2(v)$ and $A_1(v)$, respectively, i.e., $A_2(v) = \{a \in \mathcal{A} | a = (w, v)\}$ and $A_1(v) = \{a \in \mathcal{A} | a = (v, w)\}$. Lastly, we use $d_{v,t}$ to denote the demand (a supply can be reflected by a negative demand value) at a vertex $v \in \mathcal{V}$ in period t. With these additional notations, we are ready to give the formulation of the problem.

`Objective` $$\min \sum_{a \in A_p} \sum_{i \in [n]} f_{a,i} z_{a,i} \tag{4.3.13}$$

s.t.

`Flow conserv` $$\sum_{i \in [n]} \sum_{a \in A_2(v)} (q^+_{a,t,i} - q^-_{a,t,i}) - \sum_{i \in [n]} \sum_{a \in A_1(v)} (q^+_{a,t,i} - q^-_{a,t,i}) = d_{v,t},$$

$$v \in \mathcal{V}, t \in \{1, \ldots, T\} \tag{4.3.14}$$

`Pressures` $$p_v^{\min} \le p_{v,t} \le p_v^{\max}, \quad v \in \mathcal{V} \backslash V^{\text{src}}, t \in \{1, \ldots, T\} \tag{4.3.15}$$

$$p_{v,t} = p^{\text{src}}_{v,t}, \quad v \in V^{\text{src}}, t \in \{1, \ldots, T\} \tag{4.3.16}$$

`Pipes` $$0 \le q^-_{a,t,i},\ q^+_{a,t,i} \le q^{\max}_{a,i} z_{a,i}, \quad a \in A_p, i \in [n], t \in \{1, \ldots, T\} \tag{4.3.17}$$

$$p_{v,t} - p_{w,t} = \sum_{i \in [n]} \alpha_{a,i} (q^+_{a,t,i})^{1+\eta} - \sum_{i \in [n]} \alpha_{a,i} (q^-_{a,t,i})^{1+\eta} + \delta_a - \Delta_{I,a,t} + \Delta_{R,a,t},$$

$$a \in A_p, t \in \{1, \ldots, T\} \tag{4.3.18}$$

$$0 \le q^+_{a,t,i} \le q^{\max}_{a,i} x^{\text{dir}}_{a,t}, \qquad a \in A_p, i \in [n], t \in \{1, \ldots, T\} \tag{4.3.19}$$

$$0 \le q^-_{a,t,i} \le q^{\max}_{a,i} (1 - x^{\text{dir}}_{a,t}), \qquad a \in A_p, i \in [n], t \in \{1, \ldots, T\} \tag{4.3.20}$$

$$\sum_{i \in [n]} z_{a,i} = 1, \qquad a \in A_p \tag{4.3.21}$$

`Pump stations` $$\Delta_{I,a,t} \le \overline{\Delta}_{I,a} \xi_{a,t}, \quad a \in A_p, t \in \{1, \ldots, T\} \tag{4.3.22}$$

$$\xi_{a,t} \le z_{I,a}, \quad a \in A_p, t \in \{1, \ldots, T\} \tag{4.3.23}$$

$$\sum_{a \in A_p} z_{I,a} \le N \tag{4.3.24}$$

$$\sum_{t \in \{1, \ldots, T\}} \xi_{a,t} \le M_a, \quad a \in A_p \tag{4.3.25}$$

$$\xi_{a,t} - \xi_{a,t-1} \leq \xi_{a,\tau}, \quad a \in A_p, t \in \{2, \ldots, T\}, \tau \in \{t+1, \ldots, \min\{t+\tau_o, T\}\} \tag{4.3.26}$$

$$\xi_{a,t-1} - \xi_{a,t} \leq 1 - \xi_{a,\tau}, \quad a \in A_p, t \in \{2, \ldots, T\}, \tau \in \{t+1, \ldots, \min\{t+\tau_f, T\}\}. \tag{4.3.27}$$

Note that η is 0.852 or 1 depending on the choices of Hazen-Williams equation or Darcy equation. For the remainder of this chapter, we use Hazen-Williams equation as it is the more common choice in literature and in practice, i.e., $\eta = 0.852$. This results in a general nonlinear and non-convex formulation. We group the objective function and the constraints into blocks. A summary of the blocks is given in Table 4.1.

There are a few solvers that take the formulation directly. In particular, we consider BARON ([68]) and SCIP ([69]). In our preliminary study, we notice that both solvers are able to improve the dual bounds constantly, but it can be hard to obtain primal (feasible) solutions in some instances. This observation motivates us to investigate means to obtain primal solutions and provide them to the solvers.

4.4 Primal solutions

This section presents the two algorithms to obtain primal solutions. The first algorithm is adapted from the decomposition algorithm presented in Section 3.4 in chapter 3 and is based on the convex program (CVXNLP) (see [67] and Section 3.4.1 for its properties). The main components in this algorithm is a primal bound loop and an initial budget search. The second algorithm is based on time decomposition. In particular, we propose a rolling horizon type algorithm to obtain primal solutions to the problem.

Table 4.1: Constraint blocks

References	Block names	Explanations
(4.3.13)	`Objective`	Objective function to minimize the total cost of constructions (the budget)
(4.3.14)	`Flow conserv`	Flow conservation (to satisfy demand and supply)
(4.3.15)-(4.3.16)	`Pressures`	Non-source node pressure bounds (4.3.15); source node pressures (4.3.16)
(4.3.17)-(4.3.21)	`Pipes`	Flow limits on diameter choices (4.3.17); pressure change (4.3.18); flow limits on directions (4.3.19)-(4.3.20); diameter selection (4.3.21)
(4.3.22)-(4.3.27)	`Pump stations`	Pressure increase limit (4.3.22); pump stations can be on only when installed (4.3.23); resource limit on number of pump stations (4.3.24); resource limit on number of periods a pump station is on (4.3.25); minimum-up (4.3.26); minimum-down (4.3.27)

4.4.1 CVXNLP based decomposition

Primal bound loop

The primal bound loop consists of a master problem and a subproblem. The primal bound loop checks if, for a given budget C, there exists a set of feasible flows and pressures to satisfy the demand and supply along with a set of feasible diameter choices, locations of pump stations, and scheduling of the pump stations and relief valves. The master problem is based on (CVXNLP) and is given as

$$(P_m) \quad \min \sum_{t\in\{1,\dots,T\}} \sum_{i\in[n]} \sum_{a\in A_p} \frac{\alpha_{a,i}}{1+\eta}(q^+_{a,t,i})^{1+\eta} + \delta_a q^+_{a,t,i} + \frac{\alpha_{a,i}}{1+\eta}(q^-_{a,t,i})^{1+\eta} + \delta_a q^-_{a,t,i}$$
$$- \sum_{t\in\{1,\dots,T\}} \sum_{v\in V^{\text{src}}} p^{\text{src}}_{v,t} \left[\sum_{i\in[n]} \sum_{a\in A_2(v)} (q^+_{a,t,i} - q^-_{a,t,i}) - \sum_{i\in[n]} \sum_{a\in A_1(v)} (q^+_{a,t,i} - q^-_{a,t,i}) \right] \tag{4.4.1}$$

$$\text{s.t.} \sum_{i\in[n]} \sum_{a\in A_2(v)} (q^+_{a,t,i} - q^-_{a,t,i}) - \sum_{i\in[n]} \sum_{a\in A_1(v)} (q^+_{a,t,i} - q^-_{a,t,i}) = d_{v,t}, \quad v \in \mathcal{V}\backslash V^{\text{src}}, t \in \{1,\dots,T\} \tag{4.4.2}$$

$$0 \le q^-_{a,t,i}, q^+_{a,t,i} \le q^{\max}_{a,i} z_{a,i}, \quad a \in A_p, i \in [n], t \in \{1,\dots,T\} \tag{4.4.3}$$

$$\sum_{i\in[n]} z_{a,i} = 1, \ \forall a \in A_p \tag{4.4.4}$$

$$\sum_{a\in A_p} \sum_{i\in[n]} f_{a,i} z_{a,i} \le C. \tag{4.4.5}$$

Note that in the objective function (4.4.1), we do not include the pressure increase $\Delta_{I,a,t}$ or the pressure relief $\Delta_{R,a,t}$. The reasons are two-fold. Firstly, we do not consider the decisions of locations of the pump stations in the master problem (P_m). Secondly, we can rewrite the pressure change equation (4.3.17) as

$$p_{v,t} - p_{w,t} + \Delta_{I,a,t} - \Delta_{R,a,t} = \sum_{i\in[n]} \alpha_{a,i}(q^+_{a,t,i})^{1+\eta} - \sum_{i\in[n]} \alpha_{a,i}(q^-_{a,t,i})^{1+\eta} + \delta_a. \tag{4.4.6}$$

As a result, we can think of the values of $\Delta_{I,a,t}$ and $\Delta_{R,a,t}$ as part of the pressure $p_{v,t}$ or

$p_{w,t}$.

From (P_m), we can obtain a set of binary solutions for diameter choices, $z^*_{a,i}$, for $a \in A_p$ and $i \in [n]$. Consequently, we can compute the pressure loss coefficient and the max flowrate for each pipe $a \in A_p$ by

$$\alpha_a = \sum_{i \in [n]} \alpha_{a,i} z^*_{a,i} \tag{4.4.7}$$

$$q^{\max}_a = \sum_{i \in [n]} q^{\max}_{a,i} z^*_{a,i}. \tag{4.4.8}$$

Then, we can fix the corresponding binary variables and obtain a subproblem (P_s) that differs from the compact formulation (4.3.13) - (4.3.27) only in a few blocks. Formally, we describe the changes in each of the blocks.

- `Objective` Instead of minimizing the total construction costs, we have a feasibility objective function as

$$\text{Find} \quad q^+_{a,t}, q^-_{a,t}, x^{\text{dir}}_{a,t}, p_{v,t}, z_{I,a}, \xi_{a,t}, \Delta_{I,a,t}, \Delta_{R,a,t} \tag{4.4.9}$$

- `Flow conserv` Once we obtain the diameter choices, we only have one set of the flowrate variables for each pipe $a \in A_p$ in each time period $t \in \{1, \ldots, T\}$, denoted by $q^+_{a,t}$ and $q^-_{a,t}$. We can then write the flow conservation constraint as

$$\sum_{a \in A_2(v)} (q^+_{a,t} - q^-_{a,t}) - \sum_{a \in A_1(v)} (q^+_{a,t} - q^-_{a,t}) = d_{v,t}, \quad v \in \mathcal{V}, t \in \{1, \ldots, T\} \tag{4.4.10}$$

- `Pressures` block remains unchanged.

- `Pipes` Similar to the `Flow conserv` block, we only need one set of the flowrate variables, $q^+_{a,t}$ and $q^-_{a,t}$, for $a \in A_p$ and $t \in \{1, \ldots, T\}$. We simplify the `Pipes` block

as follows:

$$p_{v,t} - p_{w,t} = \alpha_a (q^+_{a,t})^{1+\eta} - \alpha_a (q^-_{a,t})^{1+\eta} + \delta_a - \Delta_{I,a,t} + \Delta_{R,a,t}, \quad a \in A_p, t \in \{1, \ldots, T\} \tag{4.4.11}$$

$$0 \leq q^+_{a,t} \leq q^{\max}_a x^{\text{dir}}_{a,t}, \qquad a \in A_p, i \in [n], t \in \{1, \ldots, T\} \tag{4.4.12}$$

$$0 \leq q^-_{a,t} \leq q^{\max}_a (1 - x^{\text{dir}}_{a,t}), \qquad a \in A_p, i \in [n], t \in \{1, \ldots, T\}. \tag{4.4.13}$$

- `Pump stations` block remains unchanged.

If the subproblem (P_s) is infeasible, we can add an integer no-good cut to the master problem (P_m) of the form,

$$\sum_{i \in [n]} \sum_{a \in A_p, z^*_{a,i}=0} z_{a,i} + \sum_{i \in [n]} \sum_{a \in A_p, z^*_{a,i}=1} (1 - z_{a,i}) \geq 1, \tag{4.4.14}$$

and resolve the master problem (P_m). On the other hand, if (P_s) is feasible, we then obtain a set of primal solution to the original formulation with an objective value of $\sum_{a \in A_p} \sum_{i \in [n]} f_{a,i} z^*_{a,i}$. The primal bound loop terminates when we obtain a primal solution or when it reaches a pre-set time limit. In the latter case, we double the budget C and re-run the primal bound loop.

Initial budget search

To obtain an initial starting budget for the primal bound loop. We propose an initial budget search procedure which is a loop consisting of a master problem and subproblem. The master problem (I_m) is given as follows:

$$(I_m) \quad \min \sum_{a \in A_p} \sum_{i \in [n]} f_{a,i} z_{a,i} \tag{4.4.15}$$

$$\text{s.t.} \sum_{i \in [n]} z_{a,i} = 1, \ \forall a \in A_p \tag{4.4.16}$$

$$z_{a,i} \in \{0,1\}, \qquad a \in A_p, i \in [n]. \tag{4.4.17}$$

We only consider the diameter choices in (I_m) and pick the cheapest available diameter choices. This problems can be solved very quickly compared to (P_m). Once we obtain a set of solution $z^*_{a,i}$ for $a \in A_p$ and $i \in [n]$, we use (4.4.7) and (4.4.8) to compute the pressure loss coefficients and max flowrates respectively. The resulting subproblem is the same as (P_s). If (P_s) is feasible, we conclude that the original problem has been solved and the optimal cost is $\sum_{a\in A_p} \sum_{i\in[n]} f_{a,i} z^*_{a,i}$, otherwise, we add the integer no-good cut of the form (4.4.14) and resolve (I_m). We run this search with a time limit and return the objective value of (I_m) in the last iteration before termination if no feasible budget is obtained.

Summary of the algorithm

Algorithm 8 presents the overall procedure with both components from the previous sections.

Algorithm 8: CVXNLP based decomposition

```
Initial budget search is run for 10 min. ▷ (iteratively solving (I_m)
  and (P_s))
if a feasible budget is obtained then
    Terminate with the optimal construction costs for this set of demand and supply
end
else
    Set starting budget based on the returned value from initial budget search;
    Primal bound loop is run for 45 min for each budget. ▷ (iteratively
     solving (P_m) and (P_s))
end
return Primal solutions or no primal solution found
```

4.4.2 Time decomposition

In this section, we discuss the rolling horizon type decomposition. Consider a $\tilde{T} < T$ such that, if we restrict the original problem to $\{1, \ldots, \tilde{T}\}$, the problem can be solved to a

target gap ε relatively efficiently. As we discussed in Section 4.1, one of the characteristics of produced water is that the volume of water produced typically is high at the beginning and gradually decreases over time. Such a characteristic means that the diameter choices from solving the problem with restricted time periods $\{1, \ldots, \tilde{T}\}$ are very likely to be feasible with respect to the volume of water produced in $\{\tilde{T}+1, \ldots, T\}$. Additionally, we argue that a pump station is more likely needed when the amount of water produced is high. Consider a pipe $a = (v, w)$ and the pressure change equation without a pump station or a relief valve for a single diameter choice in a time period t. We also include the bounds on the pressure variables and we have

$$p_{v,t} - p_{w,t} = \alpha_a {q_{a,t}^+}^{1+\eta} - \alpha_a {q_{a,t}^-}^{1+\eta} \tag{4.4.18}$$

$$p_v^{\min} \leq p_{v,t} \leq p_v^{\max} \tag{4.4.19}$$

$$p_w^{\min} \leq p_{w,t} \leq p_w^{\max}. \tag{4.4.20}$$

When the volume of water is higher, there is a larger frictional pressure loss leading to either a higher pressure value at v or a smaller pressure value at w. Therefore, the upper bound at v or the lower bound at w could be violated. A pump station on this pipe is the only way to allow the pressures at v and w to be feasible by reducing the frictional pressure loss at the same flowrate.

With the two observations, we now present the rolling horizon type algorithm. For a given $\tilde{T}$, we first modify the constraint (4.3.25) to be

$$\sum_{t \in \{1,\ldots,\tilde{T}\}} \xi_{a,t} \leq M_a \frac{\tilde{T}}{T}, \quad a \in A_p, \tag{4.4.21}$$

and solve the resulting restricted problem for the time periods $\{1, \ldots, \tilde{T}\}$ to a target gap ε. We can then obtain a set of binary solutions for diameter choices $z_{a,i}^*$ and the locations of the pump stations $z_{I,a}^*$ for $a \in A_p$ and $i \in [n]$. Additionally, we obtain a set of binary

solutions for the status of the pump stations $\xi^*_{a,t}$ for $a \in A_p$ and $t \in \{1, \ldots, \tilde{T}\}$. Once we fix the diameter choices and locations of the pump stations, we obtain a feasibility problem (P_r) for the remaining time periods $t \in \{\tilde{T}+1, \ldots, T\}$ by modifications to the objective and constraint blocks as follows,

- `Objective` Instead of minimizing the total cost on pipes, we have a feasibility objective function as

$$\text{Find} \quad q^+_{a,t}, q^-_{a,t}, x^{\text{dir}}_{a,t}, p_{v,t}, z_{I,a}, \xi_{a,t}, \Delta_{I,a,t}, \Delta_{R,a,t} \tag{4.4.22}$$

- `Pressures` block remains unchanged.

- `Flow conserv` Once we obtain the diameter choices, we only have one set of the flowrate variables for each pipe $a \in A_p$ in each time period $t \in \{\tilde{T}+1, \ldots, T\}$, denoted by $q^+_{a,t}$ and $q^-_{a,t}$. We can then write the flow conservation constraint as

$$\sum_{a \in A_2(v)} (q^+_{a,t} - q^-_{a,t}) - \sum_{a \in A_1(v)} (q^+_{a,t} - q^-_{a,t}) = d_{v,t}, \quad v \in \mathcal{V}, t \in \{\tilde{T}+1, \ldots, T\} \tag{4.4.23}$$

- `Pipes` Similar to the `Flow conserv` block, we only need one set of the flowrate variables, $q^+_{a,t}$ and $q^-_{a,t}$, for $a \in A_p$ and $t \in \{\tilde{T}+1, \ldots, T\}$. We simplify the `Pipes` block as follows:

$$p_{v,t} - p_{w,t} = \alpha_a (q^+_{a,t})^{1+\eta} - \alpha_a (q^-_{a,t})^{1+\eta} + \delta_a - \Delta_{I,a,t} + \Delta_{R,a,t},$$
$$a \in A_p, t \in \{\tilde{T}+1, \ldots, T\} \tag{4.4.24}$$
$$0 \le q^+_{a,t} \le q^{\max}_a x^{\text{dir}}_{a,t}, \qquad a \in A_p, i \in [n], t \in \{\tilde{T}+1, \ldots, T\} \tag{4.4.25}$$
$$0 \le q^-_{a,t} \le q^{\max}_a (1 - x^{\text{dir}}_{a,t}), \qquad a \in A_p, i \in [n], t \in \{\tilde{T}+1, \ldots, T\}. \tag{4.4.26}$$

- `Pump stations` Once we fix the locations of the pump stations, we simplify this

block as follows,

$$\Delta_{I,a,t} \leq \overline{\Delta}_{I,a}\xi_{a,t}, \quad a \in A_p, t \in \{\tilde{T}+1, \ldots, T\} \tag{4.4.27}$$

$$\xi_{a,t} \leq z^*_{I,a}, \quad a \in A_p, t \in \{\tilde{T}+1, \ldots, T\} \tag{4.4.28}$$

$$\sum_{t \in \{\tilde{T}+1,\ldots,T\}} \xi_{a,t} \leq M_a(1 - \frac{\tilde{T}}{T}), \quad a \in A_p \tag{4.4.29}$$

$$\xi_{a,t} - \xi_{a,t-1} \leq \xi_{a,\tau}, \quad a \in A_p, t \in \{\pi_{a,\tilde{T},o}, \ldots, T\}, \tau \in \{t+1, \ldots, \min\{t+\tau_o, T\}\} \tag{4.4.30}$$

$$\xi_{a,t-1} - \xi_{a,t} \leq 1 - \xi_{a,\tau}, \quad a \in A_p, t \in \{\pi_{a,\tilde{T},f}, \ldots, T\}, \tau \in \{t+1, \ldots, \min\{t+\tau_f, T\}\}, \tag{4.4.31}$$

where $\pi_{a,\tilde{T},o}$ and $\pi_{a,\tilde{T},f}$ are two parameters that depend on the values of $\xi^*_{a,t}$ for $a \in A_p$ and $t \in \{1, \ldots, \tilde{T}\}$ and note we may need to fix the variables $\xi_{a,t}$ for $t \in \{\tilde{T}+1, \ldots, T\}$ for the pipes that have pump stations installed due to minimum-up constraint (4.3.26) and minimum-down constraint (4.3.27). Formally, if we denote set of pipes where the pump stations are installed by $\tilde{A}$, we have the fixing procedure in Algorithm 9.

If the problem (P_r) is feasible, we then combine the variable values from the restricted problem for time periods $\{1, \ldots, \tilde{T}\}$ and (P_r) to obtain a primal solution to the original problem with the same objective value as the restricted problem. Otherwise, we can pick a different $\tilde{T}$ or target gap ε and repeat the algorithm.

Note that we can adapt this algorithm when the volume of water produced fluctuates across the time periods by constructing a restricted problem for a total of $\tilde{T}$ periods with the highest volume of water produced and then solving a similar feasibility problem for the remaining time periods. Additionally, the fixing procedure in Algorithm 9 needs to have minor modifications.

Algorithm 9: Fixing procedure

for $a \in \tilde{A}$ **do**
  Initialize $t_o = \tilde{T}$
  while $t_o \geq \tilde{T} - \tau_o + 1$ *and* $t_o \geq 2$ **do**
    if $\xi^*_{a,t_o} - \xi^*_{a,t_o-1} = 1$ **then**
      Fix $\xi_{a,\tau} = 1$ for $\tau \in \{\tilde{T} + 1, \ldots, t_o + \tau_o\}$
      Set $\pi_{a,\tilde{T},o} = t_o + \tau_o + 1$
    end
    $t_o = t_o - 1$
  end
  Initialize $t_f = \tilde{T}$
  while $t_f \geq \tilde{T} - \tau_f + 1$ *and* $t_f \geq 2$ **do**
    if $\xi^*_{a,t_f-1} - \xi^*_{a,t_f} = 1$ **then**
      Fix $\xi_{a,\tau} = 0$ for $\tau \in \{\tilde{T} + 1, \ldots, t_f + \tau_f\}$
      Set $\pi_{a,\tilde{T},f} = t_f + \tau_f + 1$
    end
    $t_f = t_f - 1$
  end
  if $\pi_{a,\tilde{T},o}$ *has not been set* **then** Set $\pi_{a,\tilde{T},o} = \tilde{T} + 2$;
  if $\pi_{a,\tilde{T},f}$ *has not been set* **then** Set $\pi_{a,\tilde{T},f} = \tilde{T} + 2$;
end

Summary of the algorithm

We now give a summary of the algorithm. Note that there is a trade-off in selecting $\tilde{T}$ and ε. A combination of small $\tilde{T}$ and large ε leads to a restricted problem that can be solved fast, however, the diameter choices and locations of pump stations obtained from such a restricted problem may not result in a feasible (P_r). In the implementation of this algorithm, we can select $\tilde{T}$ from a set $\mathcal{T}$ and ε from a set $\mathcal{E}$. Consequently, we may obtain a set of primal solutions. The algorithm is shown in Algorithm 10.

Algorithm 10: Time decomposition

```
Initialize Sol = ∅
for T̃ ∈ 𝒯 do
    for ε ∈ ℰ do
        Solve the restricted problem for time periods {1, ..., T̃} to a target gap of ε
          by SCIP
        Solve the problem (P_r) for time periods {T̃ + 1, ..., T} after the fixing
          procedure (Algorithm 9)
        if (P_r) is feasible then
            Add the primal solution and objective value to Sol
        end
    end
end
if Sol ≠ ∅ then return The primal solution with smallest objective value;
else return No primal solution found;
```

4.5 Numerical experiments

4.5.1 Instances

Our numerical experiments are performed on a network that is derived from The Produced Water Optimization Initiative (PARETO) strategic case study ([1]). The characteristics of the network are given in Table 4.2 and we refer the readers to PARETO's website for more details of the network. We consider two values of T with $T = 24$ and $T = 53$. The base demand and supply scenario follows the same temporal trend as those given in

PARETO's strategic case study. In addition, similar to the numerical experiments in chapter 3, we use stress factors to create additional demand and supply scenarios. Specifically, the stress factors are selected from the set $\{0.1, 0.5, 1.5, 2.0\}$ and directly multiplied to the base demand and supply values of each individual node to create new demand and supply scenarios. In this way, the base scenario has a stress factor of 1.0. We consider 4 different diameter choices for each pipe and we assume the fixed construction cost of each pipe is proportional to the circumference of the pipe.

Table 4.2: Characteristics of the network

Sources	Sinks	In-nodes	Pipes
19	7	29	58

4.5.2 Implementation settings

We run the experiments on a computer with an Intel i9 CPU (3.70GHz) with 64GB RAM. The computer runs the Ubuntu 20.04 LTS operating system. The formulation is coded in Python with Pyomo. We use SCIP ([69]) via GAMS to solve all optimization tasks. Algorithm 11 shows the exact steps we use to solve the problem starting from obtaining the first primal solution to providing it to SCIP as the initial point and running SCIP to improve the dual bounds and in some cases obtain better primal solutions. For a comparison, we also use SCIP to solve the MINLP formulation directly for 5 hours.

Algorithm 11: Overall procedure

```
Run CVXNLP based decomposition (Algorithm 8) or time decomposition
  (Algorithm 10)
if a primal solution is obtained then
    Record the amount of time taken to obtain the primal solution;
    Provide the primal solution to SCIP;
    Run SCIP for the remainder of 5 hours
end
return Primal bound C̄ and dual bound C
```

4.5.3 Results

In this section, we present the computational results. We compare the performances of solving the MINLP formulation by SCIP directly, Algorithm 11 with CVXNLP based composition, and Algorithm 11 with time decomposition. We report the primal bound $\overline{C}$, lower bound $\underline{C}$, and percentage gap which is computed by

$$\text{gap} = \frac{\overline{C} - \underline{C}}{\underline{C}}. \tag{4.5.1}$$

The first set of results for $T = 24$ is in Table 4.3, where $\overline{C}$ and $\underline{C}$ are reported in 10^4. We see that SCIP is only able to find primal solutions for stress factors 1.0 and 2.0 while Algorithm 11 with either CVXNLP based decomposition or time decomposition, we are able to find primal solutions for all stress factors with a largest optimality gap of about 25%. We observe further improvement from Algorithm 11 with time decomposition from CVXNLP based decomposition. Additionally, we report that SCIP obtained new primal solutions for stress factors 0.5 and 1.0 when Algorithm 11 with CVXNLP based decomposition. The final primal solution reported has about 37.6% and 38.5% improvements from the primal solutions provided to SCIP for these two stress factors, respectively. For all other demand and supply scenarios, the final primal solutions reported are obtained from Algorithm 11. Furthermore, from the comparisons of the dual bounds $\underline{C}$, we see that SCIP has the best dual bounds as Algorithm 11 takes time to obtain the primal solutions. The better dual bounds for Algorithm 11 with time decomposition indicate that time decomposition outperforms CVXNLP based decomposition. The general expectation is that as we increase the stress factor, the demand and supply scenario becomes more challenging to solve. However, this trend is not obvious in using SCIP directly. For stress factors 0.5 and 1.0, SCIP obtained the primal solutions via built-in heuristics. We believe with some stress factors, the problem may present structures that could trigger certain heuristics in SCIP while in general, it remains very challenging to obtain primal solutions using SCIP directly.

Table 4.3: Computational results for $T = 24$

Stress	MINLP by SCIP			Algorithm 11; CVXNLP			Algorithm 11; time decomp		
	$\overline{C}$	$\underline{C}$	gap(%)	$\overline{C}$	$\underline{C}$	gap (%)	$\overline{C}$	$\underline{C}$	gap (%)
0.1	-	2443.39	-	2443.39	2443.39	0.00	2443.39	2443.39	0.00
0.5	-	2498.78	-	2682.88	2493.39	7.60	2616.23	2496.77	4.78
1.0	3070.02	2632.23	16.63	2892.44	2627.61	10.08	2789.74	2631.38	6.02
1.5	-	2805.03	-	3219.64	2793.41	15.26	3023.93	2804.21	7.84
2.0	3276.69	2937.63	11.54	3683.82	2918.45	26.23	3199.12	2937.07	8.92

The next set of results for $T = 53$ is in Table 4.4 and similarly $\overline{C}$ and $\underline{C}$ are reported in 10^4. With more time periods, the problems become harder to solve. SCIP now is not able to obtain a primal solution for any of the stress factors. Algorithm 11 with CVXNLP based decomposition is able to obtain primal solutions for all stress factors except 2.0 while Algorithm 11 with time decomposition is able to obtain primal solutions for all stress factors with a largest optimality gap of less than 15%. The results for $T = 53$ confirm the expectated trend for increasing the stress factors. Unlike a few instances with $T = 24$, all primal solutions reported in this table are obtained from Algorithm 11. We note that CVXNLP based decomposition struggles to find good primal solutions for stress factors 1.0 and 1.5 with large optimality gaps. Moreover, we see that the dual bounds for Algorithm 11 with CVXNLP based decomposition and with time decomposition are comparable and this indicates with $T = 53$, the time taken to obtain a primal solution from time decomposition is much longer than the time taken with $T = 24$. If we consider an even larger network or more time periods, solving the restricted problem for $\{1, \ldots, \tilde{T}\}$ (Step 4 of Algorithm 10) directly with a solver may not be ideal for the computation time needed. Instead, we can potentially use the CVXNLP based decomposition to solve the restricted problem much more efficiently.

4.6 Conclusion

In chapter 4, we study a water network problem which considers the design problem and operation problem at the same time. The resulting compact formulation is a non-

Table 4.4: Computational results for $T = 53$

Stress	MINLP by SCIP			Algorithm 11; CVXNLP			Algorithm 11; time decomp		
	$\overline{C}$	$\underline{C}$	gap(%)	$\overline{C}$	$\underline{C}$	gap (%)	$\overline{C}$	$\underline{C}$	gap (%)
0.1	-	2443.39	-	2443.39	2443.39	0.00	2443.39	2443.39	0.00
0.5	-	2493.29	-	2721.64	2493.29	9.16	2645.06	2493.29	6.09
1.0	-	2599.72	-	4822.70	2583.69	86.66	2828.82	2582.85	9.52
1.5	-	2748.69	-	5288.98	2715.45	94.77	3054.41	2711.14	12.66
2.0	-	2857.61	-	-	-	-	3212.51	2829.65	13.53

convex MINLP and we observe that either BARON or SCIP is able to constantly improve on the dual bounds of the formulation, but is slow in finding primal solutions. We propose two algorithms to obtain primal solutions. One algorithm is based on the convex program CVXNLP and similar to the framework proposed in chapter 3 and the other is a rolling horizon type algorithm. We perform computational studies using a network derived from the PARETO strategic case study and results show that the algorithms proposed can obtain good primal solutions in the majority of the demand and supply scenarios and the algorithm based on time decomposition outperforms the algorithm based on CVXNLP. Nonetheless, there is a potential that combines the two algorithms for even larger network or more time periods.

There are a few future directions that can be pursued. One is to consider the operation costs of the pump stations and the relief valves. Both algorithms proposed can be adapted for these additional cost components.

Acknowledgement

We gratefully acknowledge support from the U.S. Department of Energy, Office of Fossil Energy and Carbon Management, through the Environmentally Prudent Stewardship Program.

Disclaimer

Attribution

Team KeyLogic's contributions to this work were funded by the National Energy Technology Laboratory under the Mission Execution and Strategic Analysis contract (DE-FE0025912) for support services.

CHAPTER 5
MODELING AND SOLVING CASCADING FAILURES ACROSS INTERDEPENDENT INFRASTRUCTURE SYSTEMS

5.1 Introduction

Physical infrastructure systems provide crucial resources such as power, water, and medical treatments for both residential and industrial activities. An infrastructure system generally consists of multiple types of facilities and these infrastructure facilities can be damaged and disabled in an event of natural disaster. A hurricane, for example, could leave particular infrastructure facilities in-operable (i.e., disabled) costing significant damage. In addition, physical infrastructure systems are generally interdependent. A cascading failure happens when one in-operable facility propagates the failure to other facilities that depend on it. Such failures, if not handled properly, could propagate further and lead to an overall system collapse. This motivates us to study an interdependent and interconnected infrastructure system and identify the most severe contingencies, or a combination of failures of facilities in terms of their impact on the service level of the network, and in particular, how cascading failures affect the overall system. The identified set of infrastructure is essential to the network's functionality and deserves special attentions in the events of natural disasters. We propose a model that does not assume a given detailed description of the connectivity of the infrastructure system, instead, we use a probabilistic model to create likely connectivity patterns. We pose this problem as a bilevel interdiction problem in the attacked-defender (AD) framework where the attacker (i.e., natural disaster) seeks a set of infrastructure that maximizes the disruption to the network and the defender (i.e., network operator) maximizes the remaining service level of the network given the attacked infrastructures and cascade of failures.

The rest of this Chapter is organized as follows. Section 5.2 provides the literature review on relevant work. Section 5.3 presents the bilevel interdiction model while Section 5.4 demonstrates the decomposition algorithm. Section 5.5 presents the numerical experiments. Lastly Section 5.6 concludes the chapter.

5.2 Literature review

Studies of the cascading failures in networks can be considered as part of the studies of resilient networks. For a detailed review on the resilience of networks, we refer to [90].

We first give a summary of the ways to model this problem. Relevant work can be divided into two sets. The first set of work focuses on abstract network model and studies the effect of cascading failures in network via tools other than optimization. The work [91] studies the propagation of cascading failures in networks of different characteristics and presents a study accounting for the dynamics of the nodes in the network and analyzing the impact of various degree of disruptions to the nodes in the network. [92] considers a similar model for the cascading condition and proposes an algorithm to select a set of more instrumental nodes in propagating the failure process. In addition, they consider additional parameters that affect cascading failures and run simulation to validate the proposed algorithm. [93] proposes a new way to model the overloads due to cascading failures in the risk and reliability assessment of complex infrastructure systems. An illustration on the effect of the proposed model is performed on power systems. The second set of work utilizes optimization tools to tackle the problems and usually focuses on power or electric systems. This set of work includes [94, 96, 97, 98, 99, 100, 101, 102, 103, 95]. Most of these work use the attacker-defender framework, a bilevel formulation, to model the problem on power or electric network. In general, in the bilevel formulation, the attacker (i.e., leader) decides on which set of nodes to disable and the defender (i.e., follower) decides on the operation of the network to minimize the disruption of the given attack. In particular, [98] presents the general frameworks of attacker-defender which results in a bilevel formulation

and defender-attacker which could extend the bilevel formulation into a trilevel formulation with fortification by the defender before the decisions of the attacker. The authors consider applications in power systems and petroleum reserve system. Some work consider different features in the follower problem which is commonly an optimal power flow problem. The work [97] considers additional elements, recovery transformers, in the power grids and studies the values of these transformers to the networks against attack. Another work [96] incorporates an additional operation, line switching, into the follower problem. The authors perform the vulnerability analysis of the power grids and study the mitigation effect of having line switching operations. To the best of our knowledge, there has not been a work that studies the general infrastructure network and considers the effect of cascading failure. In addition, the previous work generally assume the network is deterministic and given.

There are many methods and algorithms proposed to solve bilevel optimization problems. As, in most cases, the attacker decisions are modeled by binary variables, the resulting problem is a bilevel mixed-integer problem. We give a summary of relevant techniques and refer to the work [104] for more details. In general, there are two types of algorithms. The first is reformulating the bilevel problems into single level problems. A common reformulation of a bilevel problem is the value function reformulation and the associated so-called high point relaxation where the reformulation keeps all the constraints and relaxes the optimality condition for the follower problem objective function. The solutions that are infeasible with respect to the optimality condition are then cut off iteratively. The work [105] proposes a new class of intersection cuts that is proven to be effective in cutting off these infeasible points and reducing the size of the branch-and-bound tree from the high point relaxation. The authors validate the performance by considering common bilevel benchmark instances. Another single level reformulation utilizes the KKT conditions or the duality theory to rewrite the follower problem. The work [100] considers both to reformulate the bilevel problem into a single level problem. This reformulation can be

effective when the follower problem is relatively easy, for example, a linear program. The next type of algorithm is decomposition and iterative algorithm. The work [106] constructs an approximate leader problem from previously explored leader problem solutions and proposes an iterative algorithm. When the follower problem satisfies certain assumptions, the proposed algorithm can be further improved. The authors demonstrate the effectiveness of the algorithm with min-max 0-1 knapsack problem and min-max clique problem. The work [107] follows up on [106] and further improves algorithm. In [95], the authors propose a novel relaxation that leads to certain problem structures that scale well computationally with problem size. Benders decomposition algorithm is adapted by [94]. The work [94] computes a cut for each explored leader problem solution. The authors consider power or electric system as an application of the proposed procedure. In addition, there are work that study the strategy on exploring the leader problem feasible space, such as [108]. Metaheuristics can also be applied. A modified Binary Particle Swarm Optimization (BPSO) is proposed in [102] to solve the bilevel formulation in power systems.

5.3 Problem formulation

Our goal is to identify the most severe contingencies, or combination of failures of facilities in terms of their impact on the service level of the network. We formulate this problem as a bilevel interdiction problem, or Stackelberg game. In our model, the leader (e.g., the hurricane) selects facilities that it disables so that the remaining service of the network is minimized. The follower (e.g., network operator) tries to maximize the service subject to the disabled facilities, taking the cascade of failures through the network into account. The follower problem is itself a multistage optimization problem, where each stage models a progression of the cascade. We assume the progression of the cascade to be synchronized for simplicity.

5.3.1 Notations

We use the convention that sets are denoted by calligraphic parameters; parameters and constants are denoted by upper-case characters; optimization variables are denoted by lower-case characters.

- **Sets:**
  - $\mathcal{F}$ is the set of facilities (nodes).
  - $\mathcal{A} \subset \mathcal{F} \times \mathcal{F}$ is the set of directed arcs that indicate dependencies, i.e., $(s, f) \in \mathcal{A}$ indicates that facility f depends on facility s.
  - $W_f \geq 0$ is the importance weight of facility $f \in \mathcal{F}$.
  - $P_{s,f}$ is the probability variable of an arc $(s, f) \in \mathcal{A}$ that connects facility s to facility f. We use $\boldsymbol{P_f}$ to denote the vector that contains the probability variables $P_{s,f}$.
  - $\mathcal{P}_f$ denotes the family of arc probability distribution for a facility $f \in \mathcal{F}$. A generic polyhedron description of $\mathcal{P}_f$ is given as follows,

$$\mathcal{P}_f := \{P_{s,f} \geq 0 : \sum_{s \in \mathcal{F}:(s,f)\in\mathcal{A}} P_{s,f} = 1, \sum_{s \in \mathcal{F}:(s,f)\in\mathcal{A}} U^f_{l,s} P_{s,f} \geq u^f_l,\ l \in L\}, \tag{5.3.1}$$

    where $U^f_{l,s}$ and u^f_l are given constants and L is the index set of the constraints.

- **Parameters:**
  - $N := \text{card}(\mathcal{F})$ is the total number of facilities.
  - $N_c \leq N$ is the maximum number of facilities that can be disabled by the leader.
  - $N_s \geq 1$ is the number of stages that we model in our cascade in the follower problem.

- **Decision variables:**

- $x_f \in \{0, 1\}$ are the upper-level decisions, where x_f equals to 1 if facility f is disabled and 0 otherwise. We use $\boldsymbol{x}$ to denote the vector that contains all x_f.
- $y_{f,i} \in [0, 1]$ are the decision variables in stage i for $i \in \{1, 2, \ldots, N_s\}$; we interpret the variables $y_{f,i}$ as the fractional capacity which is determined by the arc probability between dependent facilities and decision variables from previous stages. We use $\boldsymbol{y_i}$ to denote the vector that contains all $y_{f,i}$ for stage i.

Given an facility *type*, we assume that we know a deterministic *hypergraph* $\mathcal{H}$ that describes the dependency relationships between facilities of different types. Below are some notations for the dependency relationships.

- $\mathcal{T}$ is the set of facility types.
- $T_f \in \mathcal{T}$ is the facility type of a particular facility f.
- $\mathcal{A}^{\mathcal{H}} \subset \mathcal{T} \times \mathcal{T}$ denotes the set of arcs in $\mathcal{H}$; an arc $(T_s, T_f) \in \mathcal{A}^{\mathcal{H}}$ means that facilities of type T_f are dependent on facilities of type T_s.

5.3.2 Optimization model

With the notations, we define the leader and follower problems in turn. Consider a particular facility f, an arc probability $P_{s,f} \in \mathcal{P}_f$ and x_s for $s \in \mathcal{F}$, we can compute the total loss of capacities from the upstream dependencies as

$$\sum_{t \in \mathcal{T}:(t,T_f) \in \mathcal{A}^{\mathcal{H}}} \sum_{s \in \mathcal{F}:T_s = t} P_{s,f} x_s = \sum_{s:(s,f) \in \mathcal{A}} P_{s,f} x_s. \tag{5.3.2}$$

Next, assuming linear degradation of capacity and the worst-case realization of $\boldsymbol{P_f}$, we have that then

$$y_{f,1} \leq \min_{\boldsymbol{P_f} \in \mathcal{P}_f} \phi_f(\boldsymbol{P_f}, \boldsymbol{x}) := \left[1 - \sum_{s:(s,f) \in \mathcal{A}} P_{s,f} x_s \right]^+, \tag{5.3.3}$$

where $[\cdot]^+ := \max\{\cdot, 0\}$. With these notations, we are ready to give our multistage formulation of the problem,

$$\min_{\boldsymbol{x}\in\{0,1\}^N} \max_{0\le \boldsymbol{y}\le 1} \quad \sum_{f\in\mathcal{F}} \sum_{i=1}^{N_s} W_f y_{f,i} \tag{5.3.4}$$

$$\text{subject to} \quad e^\top \boldsymbol{x} \le N_c \tag{5.3.5}$$

$$y_{f,1} \le 1 - x_f, \quad f \in \mathcal{F} \tag{5.3.6}$$

$$y_{f,1} \le \min_{\boldsymbol{P_f}\in\mathcal{P}_f} \phi_f(\boldsymbol{P_f}, \boldsymbol{x}), \quad f \in \mathcal{F} \tag{5.3.7}$$

$$y_{f,i} \le y_{f,i-1}, \quad f \in \mathcal{F}, i \in \{2, \ldots, N_s\} \tag{5.3.8}$$

$$y_{f,i} \le \min_{\boldsymbol{P_f}\in\mathcal{P}_f} \phi_f(\boldsymbol{P_f}, \boldsymbol{y_{i-1}}), \quad f \in \mathcal{F}, i \in \{2, \ldots, N_s\}, \tag{5.3.9}$$

where functions ϕ_f in constraints (5.3.7) and (5.3.9) have the form shown in (5.3.3) and we replace $\boldsymbol{x}$ and $y_{f,1}$ with the corresponding $1 - \boldsymbol{y_{i-1}}$ and $y_{f,i}$ expressions in (5.3.9) respectively. We use e to represent the vector of all ones and we refer the constraint $e^\top \boldsymbol{x} \le N_c$ as the budget constraint for the leader.

Furthermore, we denote the follower problem $Q(\bar{\boldsymbol{x}})$ with a fixed value of $\boldsymbol{x} = \bar{\boldsymbol{x}}$ as follows,

$$Q(\bar{\boldsymbol{x}}) := \max_{0\le \boldsymbol{y}\le 1} \quad \sum_{f\in\mathcal{F}} \sum_{i=1}^{N_s} W_f y_{f,i} \tag{5.3.10}$$

$$\text{subject to} \quad y_{f,1} \le 1 - \bar{x}_f, \quad f \in \mathcal{F} \tag{5.3.11}$$

$$y_{f,1} \le \min_{\boldsymbol{P_f}\in\mathcal{P}_f} \phi_f(\boldsymbol{P_f}, \bar{\boldsymbol{x}}), \quad f \in \mathcal{F} \tag{5.3.12}$$

$$y_{f,i} \le y_{f,i-1}, \quad f \in \mathcal{F}, i \in \{2, \ldots, N_s\} \tag{5.3.13}$$

$$y_{f,i} \le \min_{\boldsymbol{P_f}\in\mathcal{P}_f} \phi_f(\boldsymbol{P_f}, \boldsymbol{y_{i-1}}), \quad f \in \mathcal{F}, i \in \{2, \ldots, N_s\}, \tag{5.3.14}$$

Consider constraints (5.3.12), the right-hand-side minimization problem can be rewritten

as the following problem and we indicate the dual variables in the brackets,

$$\min \quad \xi \tag{5.3.15}$$

$$\text{subject to} \quad \xi \geq 1 - \sum_{s:(s,f)\in\mathcal{A}} P_{s,f}\bar{x}_s \quad (\lambda_{f,1}) \tag{5.3.16}$$

$$\xi \geq 0 \tag{5.3.17}$$

$$P_{s,f} \geq 0 \tag{5.3.18}$$

$$\sum_{s\in\mathcal{F}} P_{s,f} = 1 \quad (q_{f,1}) \tag{5.3.19}$$

$$\sum_{s\in\mathcal{F}} U^f_{l,s} P_{s,f} \geq u^f_l, \quad l \in L \quad (v_{f,1,l}), \tag{5.3.20}$$

and the dual of (5.3.15)-(5.3.20) is given by

$$\max \quad q_{f,1} + \boldsymbol{u^f}^\top \boldsymbol{v_{f,1}} + \lambda_{f,1} \tag{5.3.21}$$

$$\text{subject to} \quad q_{f,1} + \boldsymbol{U_s^f}^\top \boldsymbol{v_{f,1}} + \lambda_{f,1}\bar{x}_s \leq 0, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.22}$$

$$q_{f,1} + \boldsymbol{U_s^f}^\top \boldsymbol{v_{f,1}} \leq 0, \quad s \in \mathcal{F} : (T_s, T_f) \notin \mathcal{A}^{\mathcal{H}} \tag{5.3.23}$$

$$0 \leq \boldsymbol{v_{f,1}}, 0 \leq \lambda_{f,1} \leq 1, \tag{5.3.24}$$

where $\boldsymbol{u^f}$, $\boldsymbol{v_{f,1}}$, and $\boldsymbol{U_s^f}$ of length L are the vectors that contain the entries of u^f_l, $v_{f,1,l}$, and $U^f_{l,s}$ for $l \in L$ respectively. Similarly, for constraints (5.3.14), we can write the corresponding dual for each $i \in \{2, \ldots, N_s\}$,

$$\max \quad q_{f,i} + \boldsymbol{u^f}^\top \boldsymbol{v_{f,i}} + \lambda_{f,i} \tag{5.3.25}$$

$$\text{subject to} \quad q_{f,i} + \boldsymbol{U_s^f}^\top \boldsymbol{v_{f,i}} + \lambda_{f,i}(1 - y_{s,i-1}) \leq 0, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.26}$$

$$q_{f,i} + \boldsymbol{U_s^f}^\top \boldsymbol{v_{f,i}} \leq 0, \quad s \in \mathcal{F} : (T_s, T_f) \notin \mathcal{A}^{\mathcal{H}} \tag{5.3.27}$$

$$0 \leq \boldsymbol{v_{f,i}}, 0 \leq \lambda_{f,i} \leq 1. \tag{5.3.28}$$

Consequently, we arrive at an equivalent reformulation of the problem as follows (see [109]

Section 3.3.3),

$$\min_{\boldsymbol{x}\in\{0,1\}^N} \max \quad \sum_{f\in\mathcal{F}}\sum_{i=1}^{N_s} W_f y_{f,i} \tag{5.3.29}$$

$$\text{subject to} \quad e^\top \boldsymbol{x} \le N_c \tag{5.3.30}$$

$$y_{f,1} \le 1 - x_f, \quad f \in \mathcal{F} \tag{5.3.31}$$

$$y_{f,1} \le q_{f,1} + \boldsymbol{u^f}^\top \boldsymbol{v_{f,1}} + \lambda_{f,1}, \quad f \in \mathcal{F} \tag{5.3.32}$$

$$(5.3.22) - (5.3.24), \quad f \in \mathcal{F}$$

$$y_{f,i} \le y_{f,i-1}, \quad f \in \mathcal{F}, i \in \{2,\dots,N_s\} \tag{5.3.33}$$

$$y_{f,i} \le q_{f,i} + \boldsymbol{u^f}^\top \boldsymbol{v_{f,i}} + \lambda_{f,i}, \quad f \in \mathcal{F}, i \in \{2,\dots,N_s\} \tag{5.3.34}$$

$$(5.3.26) - (5.3.28), \quad f \in \mathcal{F}$$

$$0 \le y_{f,i} \le 1, \quad f \in \mathcal{F}, i \in \{1,\dots,N_s\}, \tag{5.3.35}$$

as well as a reformulation of the follower problem $Q(\bar{\boldsymbol{x}})$. We can utilize the McCormick envelopes to linearize the bilinear constraints (5.3.26) by $\zeta_{f,s,i} = \lambda_{f,i} y_{s,i-1}$ and we have the following set of constraints for (5.3.26),

$$q_{f,i} + U_{s,f}^\top v_{f,i} + \lambda_{f,i} - \zeta_{f,s,i} \le 0, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.36}$$

$$\zeta_{f,s,i} \ge 0, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.37}$$

$$\zeta_{f,s,i} \ge \lambda_{f,i} + y_{s,i-1} - 1, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.38}$$

$$\zeta_{f,s,i} \le \lambda_{f,i}, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A} \tag{5.3.39}$$

$$\zeta_{f,s,i} \le y_{s,i-1}, \quad s \in \mathcal{F}, (s,f) \in \mathcal{A}. \tag{5.3.40}$$

5.4 Decomposition approach

In this section, we present a Benders type decomposition algorithm.

First, for the ease of exposition, we rewrite the problem in a compact form, where in

addition to $\boldsymbol{x}$ and $\boldsymbol{y}$, we write $\boldsymbol{q}$, $\boldsymbol{\lambda}$, $\boldsymbol{v}$ as the vector forms of the corresponding variables respectively, as

$$\min_{\boldsymbol{x}\in X} \max_{0\leq \boldsymbol{y}\leq 1, \boldsymbol{q}, \boldsymbol{\lambda}, \boldsymbol{v}} W^{\top}\boldsymbol{y} \tag{5.4.1}$$

$$\text{s.t. } g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q}, \boldsymbol{\lambda}, \boldsymbol{v}) \leq 0, \tag{5.4.2}$$

where X is the feasible region for $\boldsymbol{x}$ and same as defined before. $g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{q}, \boldsymbol{\lambda}, \boldsymbol{v}) \leq 0$ denotes the constraints of the follower problem. We first present the master problem (MP),

$$(\text{MP}) \quad \min z \tag{5.4.3}$$

$$\text{s.t. } z \geq Q(\boldsymbol{x^i}) - \sum_{f\in\mathcal{F}:x_f^i=0} \alpha_f(x_f - x_f^i), \quad i \in I \tag{5.4.4}$$

$$\boldsymbol{x} \in X, \tag{5.4.5}$$

where I is the index set for the previously explored leader solution $\boldsymbol{x^i}$ and $Q(\boldsymbol{x^i})$ is the objective value of the follower problem for $\boldsymbol{x^i}$. Constraint (5.4.4) are the optimality cuts and provide lower bounds to the objective value z. The coefficients α_f for $f \in \mathcal{F}$ computes the reduction in the service capacity of the network if facility f is disabled, i.e., x_f changes from 0 to 1. We compute these coefficients for $f \in \mathcal{F}$ in Algorithm 12. We use $[N_s]$ to denote the set $\{1, \ldots, N_s\}$ and $\overline{P}_{f,s}$ to represent upper bounds on the arc probabilities $P_{f,s}$ for $f,\ s \in \mathcal{F}$. Since Algorithm 12 relies on the network structure and number of stages, it only needs to be executed once before the first iteration. Note that the upper bounds on the arc probability $\overline{P}_{f,s}$ depends on the description of the arc probability distribution set $\mathcal{P}_s$.

Next, from the construction of the master problem (MP), the subproblem is exactly the follower problem, $Q(\boldsymbol{x^*})$, to solve for the maximum remaining service capacity of the network given the leader solution x^*. Using our compact form, the subproblem (SP) is

Algorithm 12: Computation of α_f

Input: W_f for $f \in \mathcal{F}$, $\mathcal{A}$, $\overline{P}_{f,s}$ for $f,\ s \in \mathcal{F}$ such that $(f,s) \in \mathcal{A}$
Output: α_f for $f \in \mathcal{F}$
Let $\alpha_{f,i}$ be a table for $f \in \mathcal{F}$ and $i \in [N_s]$
for $f \in \mathcal{F}$ **do**
    $\alpha_{f,1} = W_f + \sum_{s \in \mathcal{F}:(f,s)\in\mathcal{A}} W_s \overline{P}_{f,s}$
    for $i \in \{2, \ldots, N_s\}$ **do**
        $\alpha_{f,i} = W_f + \sum_{s \in \mathcal{F}:(f,s)\in\mathcal{A}} \alpha_{s,i-1} \overline{P}_{f,s}$
    end
end
for $f \in \mathcal{F}$ **do**
    $\alpha_f = \sum_{i \in [N_s]} \alpha_{f,i}$
end
return α_f *for* $f \in \mathcal{F}$

given by,

$$\text{(SP)} \quad Q(\boldsymbol{x}^*) = \max_{0 \le \boldsymbol{y} \le 1, \boldsymbol{q}, \boldsymbol{\lambda}, \boldsymbol{v}} W^\top \boldsymbol{y} \tag{5.4.6}$$

$$\text{s.t. } g(\boldsymbol{x}^*, \boldsymbol{y}, \boldsymbol{q}, \boldsymbol{\lambda}, \boldsymbol{v}) \le 0. \tag{5.4.7}$$

We are now ready to present the overall procedure in Algorithm 13 with a target gap ε in percentage. In practice, we can use any feasible solution of the master problem (MP) to construct a subproblem (SP) and obtain a cut. However, only the objective values associated with optimal solutions are valid lower bounds to update $\underline{z}$ in step 7 of Algorithm 13.

Next, we present a modification of the algorithm. We consider the strengthening of cut (5.4.4) in the form of

$$z \ge Q(\boldsymbol{x^i}) - \sum_{f \in \mathcal{F}: x_f^i = 0} \alpha_f (x_f - x_f^i) + \sum_{f \in \mathcal{F}: x_f^i = 1} \beta_f(\boldsymbol{x^i})(x_f - x_f^i). \tag{5.4.8}$$

The coefficients $\beta_f(\boldsymbol{x^i})$ computes the release of service capacity of network if a facility f that is disabled in $\boldsymbol{x^i}$ is instead operational. Unlike the coefficients of α_f, $\beta_f(\boldsymbol{x^i})$ depends on a particular leader solution $\boldsymbol{x^i}$ and we can utilize auxiliary problems to compute them.

Algorithm 13: Decomposition algorithm

Input: Target gap ε
Initialize $\overline{z} = \infty$, $\underline{z} = 0$, $i = 0$
Compute the cut coefficients α_f
while $(\overline{z} - \underline{z})/\underline{z} > \varepsilon$ **do**
    $i = i + 1$
    Solve the master problem (MP) to obtain the solution $\boldsymbol{x^i}$ and z^i
    if $z^i > \underline{z}$ **then** update $\underline{z} = z^i$;
    Fix $\boldsymbol{x} = \boldsymbol{x^i}$ and solve the subproblem (SP)
    if $Q(\boldsymbol{x^i}) < \overline{z}$ **then** update $\overline{z} = Q(\boldsymbol{x^i})$;
    Add cut (5.4.4) to (MP)
end
return *ε-optimal solution* x^i

Consider a particular $\boldsymbol{x^i}$ and f such that $x^i_f = 1$, we use $\tilde{\boldsymbol{x}}$ to denote the new vector obtained from changing x^i_f to 0 while keeping the rest of components unchanged. We solve the subproblem (SP) $Q(\tilde{\boldsymbol{x}})$ and set $\beta_f(\boldsymbol{x^i})$ to be $Q(\tilde{\boldsymbol{x}}) - Q(\boldsymbol{x^i})$. We repeat the process for all facilities that are disabled in x^i to obtain a strengthened cut of (5.4.8). Depending on the value of N_c, this may require solving a few auxiliary problems.

5.5 Computational results

5.5.1 Instances

In the computational experiments, we use synthetic networks to validate the proposed algorithm as it is generally difficult and sensitive to obtain real-world data on the locations and types of infrastructure facilities. Our synthetic instances are generated based on frequency and importance data provided by Decision and Infrastructure Science (DIS) division at Argonne National Laboratory, USA. We present the characteristics of the facility types and importance weights in Table 5.1.

The frequency column in Table 5.1 indicates the distribution of facilities. We first generate the facilities according to their frequencies. To generate the dependency graph $\mathcal{A}$, for facilities s and f such that $(T_s, T_f) \in \mathcal{A}^{\mathcal{H}}$, we use the Euclidean distance, denoted by

Table 5.1: Facility information

Facility type	Upstream dependencies	Frequency	W
Cell tower	Substation	0.0438	23
EMS	Substation, cell tower, water, health care	0.0501	20
Thermal generation	Transport, cell tower	0.00887	113
Renewable generation	Cell tower	0.0162	62
Substation	Thermal or renewable generation, cell tower	0.178	6
Health care	Transport, EMS, substation, cell tower, water	0.0391	26
Pharmacies	Transport, substation, cell tower, water	0.527	2
Transport	Cell tower, substation, water	0.0318	31
Water	Substation, transport, cell tower	0.105	10

d_{sf} and Hoffman model to compute a base value for the probability $\tilde{P}_{s.f}$, i.e.,

$$\tilde{P}_{s,f} = \frac{1}{d_{sf}^2}. \tag{5.5.1}$$

We also set a threshold value below which we set $\tilde{P}_{s,f}$ to be zero since those facilities are too far away from each other to have the dependent relationship. This also allows us to keep the formulation simple and consistent as we do not need to differentiate between s and $\tilde{s}$ with $P_{s,f} = 0$ and $P_{\tilde{s},f} > 0$ in constraints (5.3.22) and (5.3.26). The differences are reflected by the coefficients in the description of $\mathcal{P}_f$. Base on the value of $\tilde{P}_{s,f}$, we construct the set $\mathcal{P}_f$ as

$$\mathcal{P}_f = \{P_{s,f} \geq 0, \sum_{s \in \mathcal{F}:(s,f)\in\mathcal{A}} P_{s,f} = 1,\ |P_{s,f} - \tilde{P}_{s,f}| \leq \delta \tilde{P}_{s,f}, \quad s, f \in \mathcal{F}, (s,f) \in \mathcal{A}\}, \tag{5.5.2}$$

with $\delta = 0.1$. Consequently, we can obtain upper bounds, $\overline{P}_{s,f}$, for $P_{s,f}$ used in Algorithm 12 by $(1+\delta)\tilde{P}_{s,f}$.

For our experiments, we generate 2 networks that vary in the number of facilities. Table 5.2 shows a breakdown on the size of the instances.

We want to point out that many larger infrastructure networks can be mapped into smaller networks of sizes that are comparable to our synthetic networks by grouping in-

Table 5.2: Instance information

Instance name	f128	f210
No. of facilities	128	210
No. of arcs in $\mathcal{A}$	1044	2033

frastructure facilities by their geographical proximity. This is valid because often natural disaster events can disable multiple facilities located in an area rather than a single facility. The dependency graph can be constructed by considering the functionalities of the facilities in the groups. As a result, we are interested in identifying most important facility groups in the networks.

5.5.2 Computational settings

We run the experiments on a computer with an Intel i9 CPU with 64GB RAM. The computer runs the Ubuntu 20.04 LTS operating system. The algorithm is coded in Julia 1.6.1 ([110]) and JuMP. Based on our preliminary experiments, we decided to use Gurobi 9.5.1 ([41]) to solve both master problem (MP) and subproblem (SP). For each instance, we set an 1 hour time limit. We set the target gap to be 1% while using 10 feasible solutions including the optimal solution from the master problem (MP) to generate optimality cuts.

5.5.3 Results

In this section, we compare the performances between the nonlinear formulation and the McCormick envelopes as well as across the variants of the algorithm. We denote Algorithm 13 with optimality cuts (5.4.4) by `Benders` and with strengthened optimality cuts (5.4.8) by `Benders-S` respectively. As strengthening requires solving auxiliary problems, it is not practical to perform strengthening for all feasible solutions and we only strengthen the optimality cut for the optimal solution from the master problem (MP).

We first compare the performances from using the nonlinear formulation and the McCormick envelopes for the follower problem (SP). In particular, the nonlinear formulation

for the follower problem utilizes constraint (5.3.26) while the McCormick envelopes utilize the constraints (5.3.36)-(5.3.40). We opt to use the variant `Benders` to solve the problem and report the detailed results in Table 5.3. The column "Iter." reports the number of iterations.

Table 5.3: Comparison of nonlinear formulation and McCormick envelopes

Instance name	Method	N_s	N_c	Time (s.)	Iter.	Optimal obj	Disabled facilities
`f128`	Nonlinear	3	5	176.87	24	3754.69	1, 17, 33, 67, 97
	McCormick	3	5	14.36	24	3754.69	1, 17, 33, 67, 97
`f210`	Nonlinear	3	5	490.21	30	6052.88	1, 43, 85, 106, 148
	McCormick	3	5	39.94	29	6052.88	1, 43, 85, 106, 148

In this set of experiments, we choose the number of stages of cascade (N_s) and number of disabled facilities (N_c) to be such that both the nonlinear formulations and McCormick envelopes can be solved within the time limit. We observe that using McCormick envelopes yields the same optimal objective values and solutions (i.e., disabled facilities) as using the nonlinear formulations does, but the computation times with McCormick envelopes are significantly shorter. The average improvement in computation time from McCormick envelopes is about 91.8%. In addition, the number of iterations needed to achieve the target gap do not differ significantly for both formulations.

Next, we compare the performances between the variants `Benders` and `Benders-S`. As the McCormick envelopes outperform the nonlinear formulation significantly in computation time, in this set of experiments, we use the McCormick envelopes for the follower problem (SP). We first present the plots for the progress of upper bound $\overline{z}$ and the lower bound $\underline{z}$ as well as the gaps as computed by

$$\text{Gap} = \frac{\overline{z} - \underline{z}}{\underline{z}} \times 100\%. \tag{5.5.3}$$

Figures 5.1 to 5.3 show the plots for instance `f210` with 3 stages of cascade ($N_s = 3$) and 8 disabled facilities ($N_c = 8$). We observe that the gap between $\overline{z}$ and $\underline{z}$ reduces very quickly in the first 30 iterations for all both variants. After approximately 80 iterations,

the upper bound $\overline{z}$ remains unchanged while computations are carried out to close the gaps from improving the lower bound $\underline{z}$. This observation suggests that the optimal solution and objective values are likely to be found early in the execution of the algorithm while proving optimality is a much more challenging task. In addition, the number of iterations needed to reach the target gap for `Benders-S` is also significantly smaller than that for `Benders`. We want to point out that `Benders` terminates with a gap of 1.48% in the given time limit while `Benders-S` achieves the target gap in about 40 minutes. This shows the effectiveness of the strengthened optimality cuts despite, in each iterations, additional auxiliary problems have to be solved in the `Benders-S` variant.

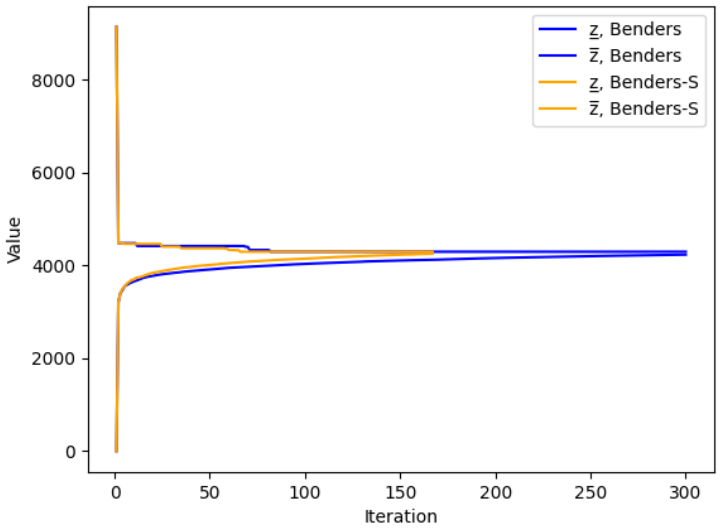

Figure 5.1: Full progress of $\overline{z}$ and $\underline{z}$

The next set of results report scalability of the algorithm. In particular, we vary the number of stages of cascade (N_s) and number of disabled facilities (N_c) for both networks. Similar to the previous set of experiments, we use McCormick envelopes for the follower problem (SP). In Tables 5.4 and 5.5, we report the detailed computational results. From both tables, we see that with smaller number of stages of cascade ($N_s = 1$ or 2) in the follower problem, both variants of the algorithms solve the problem to optimality in a relatively short amount of time. As we increase the number of stages of cascade, the problems

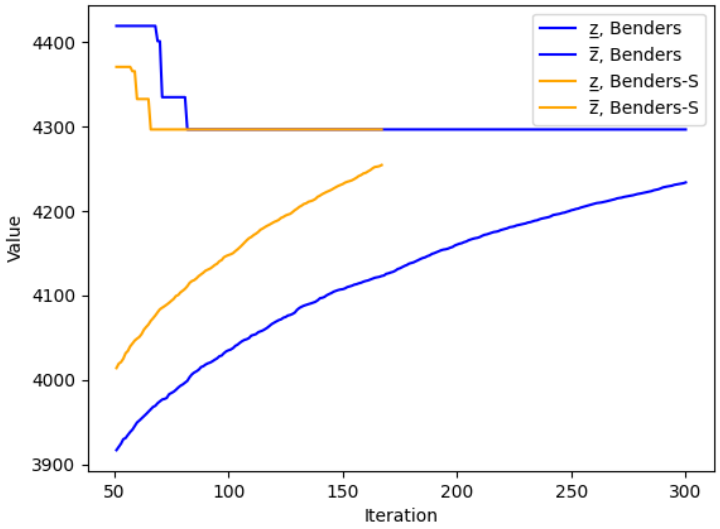

Figure 5.2: Progress of $\overline{z}$ and $\underline{z}$ from iteration 50 onwards

become much more challenging to solve. When we consider 4 stages of cascade ($N_s = 4$), we see that in few cases we reach the time limit with non-zero gaps. Nonetheless, we observe similar progresses of the $\overline{z}$ and $\underline{z}$ as well as gaps to what are shown in Figures 5.1 to 5.2. The algorithm is able to find feasible solutions in these cases, but more computations are needed to improve the lower bounds to close the gap and prove the feasible solutions are optimal. Even with the non-zero gaps, these feasible solutions have their practical uses. In addition, we see that the increasing the number of stages of cascade (N_s) has a more significant impact than increasing the number of disabled facilities (N_c) on the computation. This observation aligns with we expected. As we increase the number of stages of cascade (N_s), the follower problem (SP) becomes larger with more variables on the expected fractional capacity, while as we increase the number of disabled facilities (N_c), we do not change the size of the follower problem (SP). When the problems are relatively easy with small number of stages of cascade (N_s) or disabled facilities (N_c), we see `Benders` outperforms `Benders-S` that requires solving additional auxiliary problems in the process. The effectiveness of the strengthened optimality cuts becomes obvious when the problems are challenging to solve with larger number of stages of cascade (N_s) or disabled

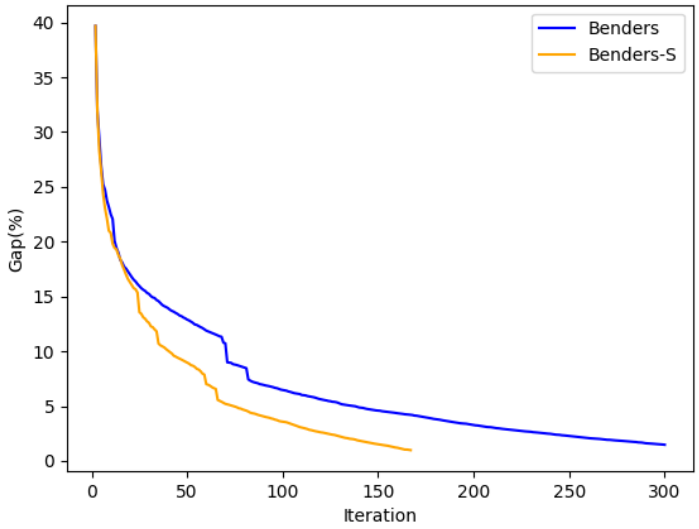

Figure 5.3: Progress of gap

facilities (N_c). There are few instances where `Benders` terminates with non-zero gaps while `Benders-S` is able to close the gaps despite the time taken per each iteration with `Benders-S` is longer.

5.5.4 Comparison with existing work

As we discussed in Section 5.2, there are two categories of approaches existed in the literature. One relies on reformulation and relaxation of the original problem to obtain a single-level problem. As we consider worse-case scenario from the probabilistic network, the resulting follower problem is a nonlinear multistage problem. Even in the case of using McCormick envelopes in place of the nonlinear constraints, it is difficult to reformulate the bilevel problem into a single-level problem. The size of the final problem could also be too large for performances. The other approach is decomposition and iterative procedures. We first discuss the iterative algorithm proposed in [106] and [107] that leverages an approximate leader problem constructed from previously explored leader problem solutions. We have implemented the the basic version of the algorithm in our study, but we observed that the algorithm explores every single feasible leader problem solution before it

terminates. Due to the large number of feasible leader problem solutions in our setting, the algorithm cannot obtain the optimal solution with the target gap of 1% even for instance `f128` with one stage of cascade ($N_s = 1$) in the given time limit. The main procedure in the improved version is to construct a new follower problem solution from a sequence of previously explored solutions. The new follower problem solution has to be feasible to all constraints and then a valid lower bound can be obtained. However, the construction of such follower problem assumes certain structures on the follower problem and our follower problem does not share such structures. Next, we note that `Benders` shares some similarity with the algorithm proposed in [94] and [97], nonetheless, the strengthened variant `Benders-S` proposed in this chapter has further improvements in its performances.

5.6 Conclusion

In conclusion, we study the effect of cascading failures in an interdependent infrastructure network. In particular, we try to identify the most severe contingencies in the network considering an event of natural disaster which could disable some of facilities to cause failures and the failures can propagate further to other dependent facilities. We propose a bilevel interdiction model for this problem. Due the complexity of the follower problem, it is difficult to reformulate the bilevel problem into a single level problem. We instead propose a Benders type decomposition algorithm and study a strengthened variant of it. We validate the proposed algorithm using two synthetic networks and vary the number of stages of cascade and number of disabled facilities. The proposed algorithm is shown to be able to obtain the optimal solutions in most cases, while, in some challenging cases, feasible solutions of practical usefulness.

For the future directions, we could explore other ways to strengthen the optimality cuts and study other classes of valid inequalities to help close the gaps for the challenging cases when the number of stages of cascade is large. Furthermore, we can study and take advantages of the structures in the network and dependency graph to further reduce the

feasible space in the leader problem. One example is to utilize any symmetries in the network and dependency graph to generate more valid inequalities in each iteration of the algorithm.

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694.

Table 5.4: Computational results for instance f128

N_s	N_c	Benders			Benders-S		
		Time (s.)	Iteration	Gap(%)	Time (s.)	Iteration	Gap(%)
1	2	1.02	2	0.0	1.13	2	0.0
	3	1.14	3	0.0	1.24	3	0.0
	4	1.17	3	0.0	1.28	3	0.0
	5	1.16	3	0.0	1.28	3	0.0
	6	1.33	4	0.0	1.65	4	0.0
	7	1.71	6	0.0	1.89	5	0.0
	8	3.05	12	0.0	2.67	8	0.0
	9	4.06	16	0.0	3.89	10	0.0
	10	4.68	18	0.0	4.54	11	0.0
2	2	1.91	2	0.0	1.99	2	0.0
	3	1.97	3	0.0	2.25	3	0.0
	4	2.40	4	0.0	2.62	4	0.0
	5	2.55	5	0.0	3.09	5	0.0
	6	3.37	8	0.0	3.79	6	0.0
	7	6.62	16	0.0	5.91	11	0.0
	8	13.87	28	0.0	13.36	19	0.0
	9	70.29	54	0.0	59.3	35	0.0
	10	269.23	97	0.0	230.65	60	0.0
3	2	3.12	3	0.0	3.29	3	0.0
	3	3.51	4	0.0	3.550	3	0.0
	4	6.12	9	0.0	5.52	7	0.0
	5	14.36	24	0.0	12.05	15	0.0
	6	29.55	37	0.0	25.47	27	0.0
	7	134.47	69	0.0	103.71	44	0.0
	8	460.04	124	0.0	367.74	77	0.0
	9	2105.39	250	0.0	1436.06	149	0.0
	10	3600	271	7.36	3600	206	4.85
4	2	5.73	3	0.0	5.94	3	0.0
	3	7.51	6	0.0	7.74	5	0.0
	4	20.16	19	0.0	19.97	14	0.0
	5	49.96	41	0.0	46.00	29	0.0
	6	202.77	88	0.0	170.64	59	0.0
	7	585.16	145	0.0	515.25	99	0.0
	8	2854.24	305	0.0	2147.91	202	0.0
	9	3600	288	11.49	3600	222	9.16
	10	3600	232	43.81	3600	173	38.12

Table 5.5: Computational results for instance `f210`

N_s	N_c	Benders			Benders-S		
		Time (s.)	Iteration	Gap(%)	Time (s.)	Iteration	Gap(%)
1	2	3.89	2	0.0	4.04	2	0.0
	3	3.93	2	0.0	4.1	2	0.0
	4	4.15	3	0.0	4.56	3	0.0
	5	4.15	3	0.0	4.58	3	0.0
	6	4.31	3	0.0	4.55	3	0.0
	7	4.43	4	0.0	4.71	3	0.0
	8	5.22	6	0.0	6.31	6	0.0
	9	7.37	12	0.0	8.05	9	0.0
	10	15.21	25	0.0	133.66	15	0.0
2	2	6.24	2	0.0	6.5	2	0.0
	3	6.59	3	0.0	7.18	3	0.0
	4	6.76	3	0.0	7.23	3	0.0
	5	6.79	3	0.0	7.29	3	0.0
	6	7.89	5	0.0	8.36	4	0.0
	7	8.54	6	0.0	9.25	5	0.0
	8	27.29	30	0.0	16.09	11	0.0
	9	99.83	70	0.0	87.18	42	0.0
	10	480.63	129	0.0	297.42	71	0.0
3	2	9.59	3	0.0	10.06	3	0.0
	3	10.49	4	0.0	11.33	4	0.0
	4	15.13	9	0.0	15.07	7	0.0
	5	39.94	29	0.0	31.93	17	0.0
	6	100.55	61	0.0	73.56	33	0.0
	7	410.82	123	0.0	246.05	69	0.0
	8	3600	300	1.48	2086.63	167	0.0
	9	3600	275	5.37	3600	193	3.97
	10	3600	240	9.77	3600	166	9.08
4	2	17.33	4	0.0	15.92	3	0.0
	3	39.31	12	0.0	30.60	8	0.0
	4	158.56	50	0.0	95.98	25	0.0
	5	444.04	108	0.0	301.88	63	0.0
	6	1277.96	184	0.0	882.34	110	0.0
	7	3600	282	2.78	3339.4	215	0.0
	8	3600	258	10.45	3600	185	9.12
	9	3600	233	20.99	3600	163	19.79
	10	3600	216	36.48	3600	144	35.13

CHAPTER 6
CONCLUSION

We study four integer problems from the domain of transportation, energy systems, and general infrastructure networks in this thesis. We propose a decomposition framework for each of the problem and computational experiments show that our proposed decomposition frameworks can solve majority of the instances to optimality or with small gaps. On one hand, the column generation framework in chapter 2 and the Benders type decomposition algorithm in chapter 5 belong to the category of general-purpose algorithms. In fact, Benders decomposition is a row generation framework. In addition, the time decomposition, specifically, rolling horizon, in chapter 2 and chapter 4 is also a general-purpose algorithm. It is commonly used in problems with multiple time periods to decompose the problems into smaller problems based on the time periods. On the other hand, the decomposition framework based on the convex program, CVXNLP, in chapters 3 and 4 is a specialized algorithm that requires the problems to have certain structures. Nonetheless, this type of framework can be generalized to a general-purpose two-phase decomposition framework where we solve for the values of some variables in the first phase via reformulations or heuristics and solve for the remaining variables after fixing the values of the variables that are solved in the first phase.

We summarize the future research directions that are applicable to the problems we study in this thesis and many more applications of MILP and MINLP. The first one is better strategies for the exploration of the binary (integer) variable space and cut generation. In chapters 3, 4, and 5, we have a lot of binary variables creating a large feasible space to explore. Both the integer no-good cuts and the Benders optimality cuts are constructed based on a single solution. There are techniques such as [111, 112] that consider stronger cuts which are constructed from many previously explored solutions. The second one is

the incorporation of additional problem features. This will improve on the accuracy of the models and the results obtained from solving the models. The consideration of the gate interference in chapter 2, and additional cost components in chapters 3 and 4, are examples in this regard.

REFERENCES

[1] M. G. Drouven, A. J. Calderon, M. A. Zamarripa, and K. Beattie, "Pareto: An open-source produced water optimization framework," *Optimization and Engineering*, 2022.

[2] Y. Li, J.-P. Clarke, and S. S. Dey, "Using submodularity within column generation to solve the flight-to-gate assignment problem," *Trnasportation Research Part C: Emerging Technology*, vol. 129, no. 103217, 2021.

[3] Y. Li, S. S. Dey, and N. V. Sahinidis, "A decomposition framework for gas network design," 2023.

[4] Y. Li, S. S. Dey, and N. V. Sahinidis, "Optimizing the designs and operations of water networks: A decomposition approach," 2023.

[5] Y. Li, K. Kim, S. Leyffer, and M. Menickelly, "Modeling and solving cascading failures across interdependent infrastructure systems," 2023.

[6] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, Wiley, 1988.

[7] M. Conforti, G. Cornuejols, and G. Zambelli, *Integer Programming*. Springer Publishing Company, Incorporated, 2014, ISBN: 3319110071.

[8] IATA, *Iata pressroom*, https://www.iata.org/en/pressroom/pr/2016-10-18-02, Accessed: Feb 13, 2020, 2016.

[9] FAA, *Air traffic by the numbers*, Federal Aviation Administration, 2019.

[10] A. Bouras, M. A. Ghaleb, U. S. Suryahatmaja, and A. M. Salem, "The airport gate assignment problem: A survey," *The Scientific World Journal*, vol. 2014, p. 923 859, 2014.

[11] G. S. Das, F. Gzara, and T. Stutzle, "A review on airport gate assignment problems: Single versus multi objective approaches," *Omega*, vol. 92, p. 102 146, 2020.

[12] S. Mokhtarimousavi, D. Talebi, and H. Asgari, "A non-dominated sorting genetic algorithm approach for optimization of multi-objective airport gate assignment problem," *Transportation Research Record*, vol. 2672, no. 23, pp. 59–70, 2018.

[13] C.-H. Cheng, S. C. Ho, and C.-L. Kwan, "The use of meta-heuristics for airport gate assignment," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 430–12 437, 2012.

[14] S. Yan and C.-H. Tang, "A heuristic approach for airport gate assignments for stochastis flight delays," *European Journal of Operations Research*, vol. 180, no. 2, pp. 547–567, 2007.

[15] S. H. Kim, E. Feron, and J.-P. Clarke, "Gate assignment to minimize passenger transit time and aircraft taxi time," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 59–70, 2013.

[16] I. Kaliszewski, J. Micoforidis, and J. Stanczak, "Multiobjective optimization in the airport gate assignment problem, exact versus evolutionary multiobjective optimization," *Computer Science*, vol. 18, no. 1, p. 41, 2007.

[17] Y. Cheng, "A knowledge-based airport gate assignment system integrated with mathematical programming," *Computers & Industrial Engineering*, vol. 32, no. 4, pp. 837–852, 1997.

[18] F. Jaehn, "Solving the flight gate assignment problem using dynamic programming," *Z Betriebswirtsch*, vol. 80, pp. 1027–1039, 2010.

[19] U. M. Neuman and J. A. D. Atkin, "Airport gate assignment considering ground movement," in *Computational Logistics*, D. Pacino, S. Voß, and R. M. Jensen, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 184–198, ISBN: 978-3-642-41019-2.

[20] U. Dorndorf, F. Jaehn, and E. Pesch, "Flight gate assignment and recovery strategies with stochastic arrival and departure times," *OR Spectrum 39, 65-93*, vol. 39, pp. 65–93, 2017.

[21] V. P. Kumar and M. Bielaire, "Multi-objective airport gate assignment problem in planning and operations," *Journal of Advanced Transportation*, vol. 48, pp. 902–926, 2013.

[22] S. H. Kim, E. Feron, J.-P. Clarke, A. Marzuoli, and D. Delahaye, "Airport gate scheduling for passengers, aircraft, and operations," *Journal of Air Transporation, AIAA*, vol. 25, no. 4, pp. 109–144, 2017.

[23] U. Dorndort, F. Jaehn, and E. Pesch, "Flight gate scheduling with respect to a reference schedule," *Annals of Operations Research*, vol. 194, pp. 177–187, 2012.

[24] G. S. Das, "New multi objective models for the gate assignment problem," *Computers & industrial Engineering*, vol. 109, pp. 347–356, 2017.

[25] C.-H. Tang and W.-C. Wang, "Airport gate assignments for airline-specific gates," *Journal of Air Transport Management*, vol. 30, pp. 10–16, 2013.

[26] M. Seker and N. Noyan, "Stochastic optimization models for the airport gate assignment problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 2, pp. 438–459, 2012.

[27] S. Yan and C.-M. Huo, "Optimization of multiple objective gate assignments," *Transportation Research Part A: Policy and Practice*, vol. 35, pp. 413–432, 2001.

[28] C. Yu, D. Zhang, and H. Lau, "Mip-based heuristics for solving robust gate assignment problems," *Computers & Industrial Engineering*, vol. 93, pp. 171–191, 2016.

[29] C. Yu, D. Zhang, and H. Lau, "An adaptive large neighborhood search heuristic for solving a robust gate assignment problem," *Expert Systems with Applications*, vol. 84, pp. 143–154, 2017.

[30] U. Benlic, E. K. Burke, and J. R. Woodward, "Breakout local search for the multi-objective gate allocation problem," *Computes & Operations Research*, vol. 78, pp. 80–93, 2017.

[31] W. Deng, H. Zhao, X. Yang, J. Xiong, M. Sun, and B. Li, "Study on an improved adaptive pso algorithm for solving multi-objective gate assignment," *Applied Soft Computing (59) October 2017, Pages 288-302*, vol. 59, pp. 288–302, 2017.

[32] J. Xu and G. Bailey, "The airport gate assignment problem: Mathematical model and a tabu search algorithm," *Proceedings of the 34th Hawaii International Conference on System Sciences*, vol. 3, p. 10, Feb. 2001.

[33] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. 316–329, 1996.

[34] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time window," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

[35] N. Buchbinder, M. Feldman, J. (Naor, and R. Schwartz, "A tight linear time (1/2)-approximation for unconstrained submodular maximization," *SIAM J. COMPUT.*, vol. 44, no. 5, pp. 1384–1402, 2012.

[36] I. Ioachim, S. Gerlinas, F. Soumis, and J. Desrosiers, "A dynamic programming algorithms for the shortest path problem with time windows and linear node costs," *Networks, Vol.31, pp. 193-204*, vol. 31, pp. 193–204, 1998.

[37] M. Desrochers and F. Soumis, "A generalized permanent labeling algorithm for the shortest path problem with time windows," *INFOR*, vol. 26, pp. 191–212, 1988.

[38] Y. Dumas, F. Soumis, and J. Desrosiers, "Technical note-optimizing the schedule for a fixed vehicle path with convex inconvenience costs," *Transportation Science 24(2):145-152*, vol. 24, no. 2, pp. 145–152, 1990.

[39] D. Buitendijk, "First order stability flexible gate scheduling," 2014, Thesis.

[40] D. Ryan and B. Foster, "An integer programming approach to scheduling," *Operations Research Quarterly (1970-1977)*, vol. 27, no. 2, pp. 367–384, 1981.

[41] L. Gurobi Optimization, *Gurobi optimizer reference manual*, 2022.

[42] U.S. EIA, *Natural gas explained*, https://www.eia.gov/energyexplained/natural-gas/natural-gas-pipelines.php, Accessed: 04-21-2022, 2021.

[43] S. Sullivan and S. Dlin, *Year in pipelines: Growth in us natural gas pipeline assets slowed again in 2020*, https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/year-in-pipelines-growth-in-us-natural-gas-pipeline-assets\-slowed-again-in-2020-66386728, Accessed: 04-21-2022, 2021.

[44] D. D. Wolf and Y. Smeers, "Optimal dimensioning of pipe networks with application to gas transmission networks," *Operations Research*, vol. 44, no. 4, pp. 596–608, 1996.

[45] D. D. Wolf and Y. Smeers, "The gas transmission problem solved by an extension of the simplex algorithm," *Management Science*, vol. 46, no. 11, pp. 1454–1465, 2000.

[46] M. Schmidt *et al.*, "Gaslib - a library of gas network instances," *data*, vol. 2, no. 4, p. 40, 2015.

[47] R. Z. Rio-Mercado and C. Borraz-Sanchez, "Optimization problems in natural gas transportation systems: A state-of-the-art review," *Applied Energy*, vol. 147, pp. 536–555, 2015.

[48] Q. P. Zheng, S. Rebennack, N. A. Iliadis, and P. Pardalos, in *Handbook of Power Systems I*, P. M. Pardalos, S. Rebennack, M. V. F. Pereira, and N. A. Iliadis, Eds. Heidebergy: Springer Berlin, 2010, ch. Optimization Models in the Natural Gas Industry.

[49] F. M. Hante and M. Schmidt, "Gas transport network optimization: Mixed-integer nonlinear models," *Optimization-online*, 2023.

[50] F. Liers, A. Martin, M. Merkert, N. Mertens, and D. Michaels, "Solving mixed-integer nonlinear optimization problems using simultaneous convexification–a case

study for gas networks," *Journal of Global Optimization*, vol. 80, pp. 307–340, 2021.

[51] M. E. Pfetsch *et al.*, "Validation of nominations in gas network optimization: Models, methods, and solutions," *Optimization Methods and Software*, vol. 30, no. 1, pp. 15–53, 2015.

[52] B. Geibler, A. Morsi, L. Schewe, and M. Schmidt, "Solving power-constrained gas transportation problems using an mip-based alternating direction method," *Computers & Chemical Engineering*, vol. 82, pp. 303–317, 2015.

[53] B. Geibler, A. Morsi, L. Schewe, and M. Schmidt, "Solving highly detailed gas transport minlps: Block separability and penalty alternating direction methods," *INFORMS Journal on Computing*, vol. 30, no. 2, pp. 309–323, 2018.

[54] R. Burlacu, B. Geissler, and L. Schewe, "Solving mixed-integer nonlinear programs using adaptively refined mixed-integer linear programmes," *Optimization Methods and Software*, vol. 35, no. 1, pp. 37–64, 2020.

[55] T. Koch, B. Hiller, M. E. Pfetsch, and L. Schewe, Eds., *Evaluating Gas Network Capacities*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2015.

[56] D. Rose, M. Schmidt, M. C. Steinbach, and B. M. Willert, "Computational optimization of gas compressor stations: Minlp models versus continuous reformulations," *Math Meth Oper Res*, vol. 83, pp. 409–444, 2016.

[57] M. Schmidt, M. C. Steinbach, and B. M. Willert, "High detail stationary optimization models for gas networks: Validation and results," *Optim Eng*, vol. 17, pp. 437–472, 2016.

[58] F. Babonneau, Y. Nesterov, and J.-P. Vial, "Design and operations of gas transmission networks," *Operations Research*, vol. 60, no. 1, pp. 34–47, 2012.

[59] C. Borraz-Sanchez, R. Bent, S. Backhaus, H. Hijazi, and P. V. Hentenryck, "Convex relaxations for gas expansion planning," *INFORMS Journal on Computing*, vol. 28, pp. 645–656, 2016.

[60] J. Zhang and D. Zhu, "A bilevel programming method for pipe network optimization," *SIAM J. Optimization*, vol. 6, no. 3, pp. 838–857, 1996.

[61] N. Shiono and H. Suzuki, "Optimal pipe-sizing problem of tree-shaped gas distribution networks," *European Journal of Operational Research*, vol. 252, no. 2, pp. 550–560, 2016.

[62] C. Cherry, "Some general theorems for non-linear systems possessing reactance," *Philos. Mag.*, vol. 42, no. 7, pp. 1161–1177, 1951.

[63] M. Collins, L. Cooper, R. Helgason, J. Kennington, and L. LeBlanc, "Solving the pipe network analysis problem using optimization techniques," *Management Science*, vol. 24, no. 7, pp. 747–760, 1978.

[64] A. Frangioni and C. Gentile, "Perspective cuts for a class of convex 0-1 mixed integer programs," *Math. Program.*, vol. 106, pp. 225–236, 2006.

[65] O. Günlük and J. Linderoth, "Perspective reformulation of mixed integer nonlinear programs with indicator variables," *Math. Program.*, vol. 124, pp. 183–205, 2010.

[66] J. Humpola, "Gas network optimization by minlp," 2014, PhD Dissertation.

[67] A. U. Raghunathan, "Global optimization of nonlinear network design," *SIAM J. OPTIM*, vol. 23, no. 1, pp. 268–295, 2013.

[68] M. Tawarmalani and N. V. Sahinidis, "A polyhedral branch-and-cut approach to global optimization," *Math. Program.*, vol. 103, pp. 225–249, 2 2005.

[69] T. Achterberg, "Scip: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[70] M. ApS, *The mosek optimization toolbox for python manual. version 9.3.10.* 2022.

[71] N. V. Sahinidis, *Baron 22.9.1: Global optimization of mixed-integer nonlinear programs,* user's manual, 2022.

[72] G. Gamrath *et al.*, "The scip optimization suite 7.0," Optimization Online, Technical Report, Mar. 2020.

[73] A. S. of Civil Engineers, "The economic benefits of investing in water infrastructure," Value of water campaign.

[74] J. S. o. G. University of Texas at Austin, "Hydraulic fracturing water cycle," 2020, https://news.utexas.edu/2020/02/20/water-reuse-could-be-key-for-future-of-hydraulic-fracturing/.

[75] H. Mala-Jetmarova, N. Sultanova, and D. Savic, "Lost in optimisation of water distribution systems? a literature review of system design," *Environmental Modelling & Software*, vol. 93, pp. 209–254, 2017.

[76] C. D'Ambrosio, A. Lodi, S. Wiese, and C. Bragalli, "Mathematical programming techniques in water network optimization," *European Journal of Operational Research*, vol. 243, pp. 774–788, 2015.

[77] P. M. Castro and J. P. Teles, "Comparison of global optimization algorithms for the design of water-using networks," *Computers and Chemical Engineering*, vol. 52, pp. 249–261, 2013.

[78] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth, "On the optimal design of water distribution networks: A practical minlp approach," *Optim Eng*, vol. 13, pp. 219–246, 2012.

[79] B. P and L. J, *Bonmin users' manul. tech rep*, 2006.

[80] B. P. B. L *et al.*, "An algorithmic framework for convex mixed integer nonlinear programs," *Discrete Optim*, vol. 5, pp. 186–204, 2008.

[81] A. Gleixner *et al.*, "MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library," *Mathematical Programming Computation*, 2021.

[82] S. Rajagopalan, "Design and maintenance planning problems in commodity distribution and chemical site networks," 2018, PhD Dissertation.

[83] F. Pecci, E. Abraham, and I. Stoianov, "Global optimality bounds for the placement of control valves in water supply networks," *Optimization and Engineering*, vol. 20, pp. 457–495, 2019.

[84] B. Ghaddar, J. Naoum-Sawaya, A. Kishimoto, N. Taheri, and B. Eck, "A lagrangian decomposition approach for the pump scheduling problem in water networks," *European Journal of Operational Research*, vol. 241, pp. 490–501, 2015.

[85] G. Bonvin, S. Demassey, and A. Lodi, "Pump scheduling in drinking water distribution networks with an lp/nlp-bbased branch and bound," *Optimization and Engineering*, vol. 22, pp. 1275–1313, 2021.

[86] J. Ostrowski, M. F. Anjos, and A. Vannelli, "Tight mixed integer linear programming formulations for the unit commitment problem," *IEEE transactions on power systems*, vol. 27, no. 1, 2012.

[87] J. Lee, J. Leung, and F. Margot, "Min-up/min-down polytops," *Discrete Optimization*, vol. 1, pp. 77–85, 2004.

[88] D. Rajan and S. Takriti, "Minimum up/down polytopes of the unit commitment problem with start-up costs," *IBM Research Report*, 2005, RC23628 (W0506-050).

[89] L. Glomb, F. Liers, and F. Rosel, "A rolling-horizon approach for multi-period optimization," *European Journal of Operational Research*, vol. 300, pp. 189–206, 2022.

[90] D. K. Mishra, M. J. Ghadi, A. azizivahed, L. Li, and J. Zhang, "A review on resilience studies in active distribution systems," *Renewable and Sustainable Energy Reviews*, vol. 135, p. 110 201, 2021.

[91] D. Duan *et al.*, "Universal behavior of cascading failures interdependent networks," *Applied Physical Sciences*, vol. 116, no. 45, pp. 22 452–22 457, 2019.

[92] A. Smolyak, O. Levy, I. Vodenska, S. Buldyrev, and S. Havlin, "Mitigation of cascading failures in complex networks," *Scientific Reports*, vol. 10, p. 16 124, 2020.

[93] L. Duenas-Osorio and S. M. Vemuru, "Cascading failures in complex infrastructure systems," *Structural Safety*, vol. 31, no. 2, pp. 157–167, 2009.

[94] J. Salmeron, K. Wood, and R. Baldick, "Worst-case interdiction analysis of large-scale electric power grids," *IEEE Transactions on Power Systems*, vol. 24, no. 1, pp. 96–104, 2009.

[95] E. Johnson and S. S. Dey, "A scalable lower bound for the worst-case relay attack problem on the transmission grid," *INFORMS Journal on Computing*, vol. 34, no. 4, pp. 2296–2312, 2022.

[96] L. Zhao and B. Zeng, "Vulnerability analysis of power grids with line switching," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2727–2736, 2013.

[97] J. Salmeron and R. K. Wood, "The value of recovery transformers in protecting an electric transmission grid against attack," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2396–2403, 2015.

[98] G. Brown, M. Carlyle, J. Salmeron, and K. Wood, "Defending critical infrastructure," *Interfaces*, vol. 36, no. 6, pp. 530–544, 2006.

[99] K. Sundar, S. Misra, R. Bent, and F. Pan, "Credible interdiction for transmission systems," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 2, pp. 738–748, 2021.

[100] J. Arroyo, "Bilevel programming applied to power system vulnerability analysis under multiple contingencies," *IET Gener. Transm. Distrib.*, vol. 4, no. 2, pp. 178–190, 2010.

[101] R. K. Wood, "Bilevel network interdiction models:formulations and solutions," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[102] A. Z. G. Seyyedi, M. J. Armand, S. Shahmoradi, S. M. Rashid, E. Akbari, and A. J. K. Al-Hassanawy, "Iterative optimization of a bi-level formulation to identify severe contingencies in power transmission systems," *International Journal of Electrical Power and Energy Systems*, vol. 145, p. 108 670, 2023.

[103] Y. Wang and R. Baldick, "Interdiction analysis of electric grids combining cascading outage and medium-term impacts," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2160–2168, 2014.

[104] T. Kleinert, M. Labbe, I. Ljubic, and M. Schmidt, "A survey on mixed-integer programming techniques in bilevel optimization," *EURO Journal on Computational Optimization*, vol. 9, p. 100 007, 2021.

[105] M. Fischetti, I. Ljubie, M. Monaci, and M. Sinnl, "A new general-purpose algorithm for mixed-integer bilevel linear programs," *Operations Research*, vol. 65, no. 6, pp. 1615–1637, 2020.

[106] Y. Tang, J.-P. P. Richard, and J. C. Smith, "A class of algorithms for mixed-integer bilevel min-max optimization," *Journal of Global Optimization*, vol. 66, pp. 225–262, 2016.

[107] K. Taninmis, N. Aras, and I. K. Altinel, "Improved x-space algorithm for min-max bilevel problems with an application to misinformation spread in social networks," *European Journal of Operational Research*, vol. 297, no. 1, pp. 40–52, 2022.

[108] L. Lozano and J. C. Smith, "A backward sampling framework for interdiction problems with fortification," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 123–139, 2022.

[109] A. Nemirovski, *Introduction to linear optimization*, 2016.

[110] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[111] G. Angulo, S. Ahmed, S. S. Dey, and V. Kaibel, "Forbidden vertices," *Mathematics of Operations Research*, vol. 40, no. 2, pp. 350–360, 2015.

[112] G. Angulo, S. Ahmed, and S. S. Dey, "Improving the integer l-shaped method," *INFORMS Journal on Computing*, vol. 28, no. 3, pp. 483–499, 2016.