DECOMPOSITION METHODS IN COLUMN GENERATION AND DATA-DRIVEN STOCHASTIC OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

By

Mohamed Ali El-Moghazi El Tonbari

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

Georgia Institute of Technology

December 2021

DECOMPOSITION METHODS IN COLUMN GENERATION AND DATA-DRIVEN STOCHASTIC OPTIMIZATION

Thesis committee:

Dr. George Nemhauser, Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

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

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

Dr. Kibaek Kim
Mathematics and Computer Science Division
Argonne National Laboratory

Dr. Mathieu Dahan
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Shabbir Ahmed, Late Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Date approved: November 22, 2021

ACKNOWLEDGMENTS

I would first like to thank my late advisor, Shabbir Ahmed - a great researcher and a great mentor who was passionate about research and cared about his students. I really enjoyed our discussions and working through problems and proofs on the whiteboard with you. I am grateful for all the opportunities you have given me, and for teaching me how to be a researcher. You were one of the best teachers I've had the fortune to work with. Thank you for being an amazing advisor. I would like to thank my advisor, George Nemhauser, for always being there for me throughout my PhD. I could not have asked for a better advisor. I admire you as a researcher and as a person. You taught me the importance of thinking about the big picture. Thank you for all the guidance, support, and patience. I would like to thank Natashia Boland for guiding me at the start of my last chapter, for all the advice and the support. It was great working with you as we figured out how to frame the problem. Last but not least, thank you, Alejandro Toriello, for being a great advisor in my last year. I really enjoyed discussing and exploring ideas, working through the last chapter and seeing it to the end.

I am very grateful to my supervisor during my internships at Argonne National Laboratory, Kibaek Kim, and my collaborator Anirudh Subramanyam. Thank you, Kibaek, for introducing me to the topic of Distributionally Robust Optimization. I have learned a lot through my time working with you about computational aspects of stochastic optimization and power systems. Thank you for your patience and support. To Anirudh, thank you for being a great collaborator and mentor. I will always cherish the countless hours we spent on the board working out the problems and bouncing off ideas. It was some of the most fun I've had in research.

I would like to acknowledge the remaining members of my committee, Santanu Dey and Mathieu Dahan. Thank you for your support and kindness. Thank you, Mathieu and Pinar Keskinocak, for your valuable input at the start of our project on natural disaster management. Thank you to PACE for providing me with computing resources. Thank you, Amanda Ford, for your patience throughout the years as you answered many of my questions.

Thank you, Amir Ali Ahmadi, for being an incredible mentor throughout the years. You were always there for me to answer questions and give me advice. My interest in optimization started with your amazingly taught course. I remember the times after office hours where you went out of your way to make sure I understood the concepts, and I am grateful for that.

Thank you, Jana Boerger, for always being there for me, for all the support and for the amazing times, and hope for many more trips and fun times with you. I am grateful for the many fellow students and friends I have met along the way who have made this a memorable journey. Thank you, Émile Trottier, Jakob Schoeffer, Devesh Kedia, Jen Coppola, Daniel Baller. I will always cherish our times together, and hope for many more to come. To Emma Johnson, Kirthana Hampapur, Paritosh Ramanan, Alexandre Velloso: I will always remember our lunches, food trips, and countless interesting discussions we've had. It would not have been the same without you. I would also like to thank my friends and collegues at Georgia Tech for the fun times and interesting discussions as we worked through the many assignments and projects: Andrew ElHabr, Ramon Auad, Adrian Rivera, Reem Khir, Alejandro Carderera, Yuliia Lut, Lingquan Ding, Jimmy Zhang, Alex Forney, Hassan Mortagy, Daniela Hurtado, Georgios Kotsalis, Tyler Perini, Alfredo Torrico, Adolfo Rocco, Tony Yaacoub, Yassine Ridouane, Seyma Gurkan, Digvijay Boob, Sait Cakmak, Di Liu, Ana María Estrada Gómez, Sebastian Perez-Salazar, Beste Basciftci. Thank you to all of you, and anyone I am missing in this list!

Most importantly, I am deeply grateful to my amazing family, my parents and my brother. This journey would not have been possible without your constant support. Thank you for always believing in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, we are focused on tackling large-scale problems arising in two-stage stochastic optimization and the related Dantzig-Wolfe decomposition. We start with a deterministic setting, where we consider linear programs with a block-structure, but data cannot be stored centrally due to privacy concerns or decentralized storage of large data sets. The larger portion of the thesis is dedicated to the stochastic setting, where we study two-stage distributionally robust optimization under the Wasserstein ambiguity set to tackle problems with limited data.

Chapter 2 is joint work with Shabbir Ahmed and is based on the paper in [1]. In this work, we propose a fully distributed Dantzig-Wolfe decomposition (DWD) algorithm using the Alternating Direction Method of Multipliers (ADMM) method. DWD is a classical algorithm used to solve large-scale linear programs whose constraint matrix is a set of independent blocks coupled with a set of linking rows. In a typical implementation, the algorithm alternates between solving a master problem centrally and a set of independent subproblems in parallel. In certain cases, solving the master problem centrally is undesirable or infeasible, due to privacy concerns or decentralized storage of data. In the former, the independent blocks represent agents who desire privacy of information. In the latter, data is stored in decentralized servers due to memory limitations, or as a protection against attackers or failures. To this end, we develop a consensus-based Dantzig-Wolfe decomposition algorithm, where the dual of the master problem is solved using consensus-based ADMM at each iteration. We discuss the benefits of using ADMM over other consensus methods and working on the dual over the primal problem, and detail the computational and algorithmic challenges. We provide bounds on the optimality gap and feasibility violation, and perform extensive computational experiments on instances of the cutting stock problem and synthetic instances using a Message Passing Interface (MPI) implementation, where we obtain high-quality solutions in reasonable time. Although our main contribu-

tion is to tackle decentralized storage of data and privacy, our method also shows a potential computational benefit for instances with a large number of variables.

In Chapter 3 and 4, we turn our focus to stochastic optimization, specifically applications where data is scarce and the underlying probability distribution is difficult to estimate. We study two-stage distributionally robust optimization (DRO) under the Wasserstein ambiguity set in different settings, where the Wasserstein set is a ball in the space of probability distributions centered at an empirical distribution obtained from historical data.

Chapter 3 is joint work with Anirudh Subramanyam and Kibaek Kim and is based on the paper in [2]. Here, we consider two-stage conic DRO under the Wasserstein set with zero-one uncertainties. We are motivated by problems arising in network optimization, where binary random variables represent failures of network components. We are interested in applications where such failures are rare and have a high impact, making it difficult to estimate failure probabilities. Due to our support set being non-convex, we cannot use typical duality tools to obtain a tractable convex program. By using ideas from bilinear programming and penalty methods, we reformulate our two-stage DRO model by decomposing the inner maximization into a maximization problem for each sampled scenario over mixed-integer conic sets. We use Lovász-Schrijver approximations to get an outer description of the convex hulls of the inner problems which can be iteratively improved, permitting us to dualize and obtain a model which can be solved using commercial solvers. We illustrate the computational and out-of-sample performance of our method on the optimal power flow problem with random transmission line failures and a multi-commodity network design problem with random node failures.

In Chapter 4, joint work with Alejandro Toriello and George Nemhauser, we study a two-stage model which arises in natural disaster management applications, where the first stage is a facility location problem, deciding where to open facilities and pre-allocate resources, and the second stage is a fixed-charge transportation problem, routing resources to affected areas after a disaster. We solve a two-stage DRO model under the Wasserstein

set to deal with the lack of available data. The presence of binary variables in the second stage significantly complicates the problem. We develop a column-and-constraint generation algorithm, where we generate scenarios as needed, and leverage the structure of our support set and second-stage value function to efficiently generate new scenarios. More specifically, we show that the optimal cost of the second-stage fixed-charge transportation problem is concave with respect to a subset of the uncertainty, leading to an efficient line search algorithm in the scenario generation step. We show our results extend to the case where the second stage is a fixed-charge network flow problem. We provide a detailed discussion on our implementation, and end the chapter with computational experiments on synthetic instances and a case study of hurricane threats on the coastal states of the United States.

CHAPTER 1
INTRODUCTION

Decision-making under uncertainty has witnessed a surge in research and interest over the last decades in many fields, such as reinforcement learning, control theory, and operations research. Although there are many similarities in the theory and algorithms used, the wide variety of applications across the fields is a testament to the value of integrating uncertainty in the decision-making process. Indeed, many methods such as dynamic programming, multistage stochastic programming or reinforcement learning methods rely on the theory of Markov Decision Processes and the Bellman equations, for example. Examples of applications where certain parameters are unknown include transportation problems where travel times might be uncertain, long-term network design planning where future demands are uncertain, and energy applications where transmission line failures or market prices are random, to name a few.

1.1 Two-Stage Stochastic Programming

A popular approach in decision-making under uncertainty is two-stage stochastic programming which is a special case of multistage stochastic programming where decisions are made in two stages, before and after observing the uncertainty. The second-stage problem is often referred to as the recourse problem. In this paradigm, the objective is to minimize the sum of the first-stage cost and the expected value of the second-stage cost. Such problems have the form:

$$\min_{\boldsymbol{x} \in X} \quad \boldsymbol{c}^\top \boldsymbol{x} + \mathbb{E}_{\mathbb{P}} \left[\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}) \right], \tag{1.1.1}$$

where $\boldsymbol{x}$ corresponds to the first-stage decision vector belonging to a set X, and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is the optimal value of the second-stage problem, defined as

$$\begin{aligned} \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}) := \min_{\boldsymbol{y}} \quad & \boldsymbol{q}(\boldsymbol{\xi})^\top \boldsymbol{y} \\ \text{s.t.} \quad & W\boldsymbol{y} \geq h(\boldsymbol{\xi}) - T(\boldsymbol{\xi})\boldsymbol{x}. \end{aligned} \tag{1.1.2}$$

Variable $\boldsymbol{y}$ corresponds to the second-stage decisions and $\boldsymbol{\xi}$ is a random vector belonging to some support set Ξ with an underlying probability distribution $\mathbb{P}$. Different variations of (1.1.1) can be considered, where the uncertainty $\boldsymbol{\xi}$ might only be present in the objective, on the right-hand side, in the matrix $T(\boldsymbol{\xi})$, or in any combination of the three. A two-stage model where $W(\boldsymbol{\xi}) = W$ for all $\boldsymbol{\xi} \in \Xi$ as in (1.1.2) is referred to as a problem with fixed recourse.

Consider the case where the support Ξ is a finite set of scenarios $\{\boldsymbol{\xi}^1, \boldsymbol{\xi}^2, \dots\}$ indexed by $s \in \mathcal{S}$, and assume the associated probability distribution $\{p_s\}_{s\in\mathcal{S}}$ is known. We can then explicitly write the expectation of the second-stage value function $\mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})\right]$. Popular methods include Benders decomposition [3], where $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is represented by its epigraph whose approximation is iteratively improved by generating cuts using dual information. Alternatively, we can write an extensive reformulation of (1.1.1) as

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{y}} \quad & \boldsymbol{c}^\top \boldsymbol{x} + \sum_{s\in\mathcal{S}} p_s \boldsymbol{q}(\boldsymbol{\xi}^s)^\top \boldsymbol{y}^s \\ \text{s.t.} \quad & \boldsymbol{x} \in X, \\ & T(\boldsymbol{\xi}^s)\boldsymbol{x} + W\boldsymbol{y}^s \geq h(\boldsymbol{\xi}^s), \forall s \in \mathcal{S} \end{aligned} \tag{1.1.3}$$

where we make a copy of the second-stage problem for each scenario. For larger sets of scenarios, this model can be computationally challenging to solve. An important observation is the L-shaped structure of the constraints, since problem (1.1.3) is coupled by the first-stage decision vector $\boldsymbol{x}$. A common decomposition technique is to make a copy of the first-stage vector for each scenario, and add the so-called non-anticipativity constraints, forcing all copies of x to be equal to each other. Such a reformulation opens the door to decomposition techniques which leverage distributed computing to more efficiently solve (1.1.3). The Dual Decomposition method, first proposed by Carøe and Schultz in [4], relaxes the non-anticipativity constraints to obtain a problem which decomposes by scenario, and solves the Lagrangian dual instead. In [5], Rockafellar and Wets propose the Progressive Hedging method, which, closely related to the Alternating Direction Method of Multipliers (ADMM) algorithm, solves the augmented Lagrangian dual by adding a quadratic penalty.

1.2 Dantzig-Wolfe Decomposition

In Chapter 2, we start with a deterministic setting, where we are interested in a related problem to the Benders decomposition algorithm. It is well known that the dual of the Benders problem also leads to a special block structure in the constraints. Dantzig-Wolfe decomposition, first proposed by Dantzig and Wolfe [6, 7], is a column generation algorithm which leverages such structure, more specifically large-scale linear programs whose constraint matrix involves a set of independent blocks coupled with a set of linking rows. The algorithm starts with a reformulation involving a subset of the columns, and alternates between solving a master problem centrally and a set of independent subproblems to generate new columns.

We are specifically interested in the case where the data related to each block cannot be centrally located, either due to privacy concerns or decentralized storage of data. In the former, the independent blocks might represent local optimization problems of independent agents who may not wish to share sensitive data. In the latter, data might be stored in decentralized servers for security measures to hedge against attackers or failures, or due to the size of the data sets. We provide examples of such applications in Chapter 2. In both cases, the master problem cannot be solved centrally. To this end, we develop a fully distributed Dantzig-Wolfe decomposition algorithm by solving the dual of the master problem using consensus-based ADMM. We discuss the benefits of using ADMM over other consensus-based methods and working on the dual over the primal problem, and detail the computational and algorithmic challenges. We provide bounds on the optimality and feasibility gap resulting from solving the master in a distributed fashion, and perform extensive computational experiments on instances of the cutting stock problem and synthetic instances.

1.3 Data-Driven Stochastic Optimization

In Chapter 3 and 4, we turn our attention back to two-stage stochastic programming. Recall that in problem (1.1.3), we assume we know the probability distribution of the uncertainty. In many applications, however, the distribution $\mathbb{P}$ is unknown and the uncertainty is only observable through historical data. Given a set of samples, Monte Carlo based methods such as the Sample Average Approximation (SAA) (see [8, 9]) use the obtained empirical distribution to approximate the expected second-stage value function. SAA is also commonly used for continuous support sets Ξ. SAA is known to converge to the optimal solution of (1.1.1) as the number of samples goes to infinity under mild conditions, and generally has nice computational performance, which accounts for its popularity [10].

An alternative paradigm in optimization under uncertainty is to minimize the worst-case second-stage cost with respect to the uncertainty. This is known as robust optimization (RO). RO is a distribution-free method which can be used to obtain a conservative solution, protecting the decision-maker against the worst-case scenario, or providing a solution which is feasible for all scenarios. Generally, the maximization of the second-stage cost is done with respect to random vectors belonging to a constructed uncertainty set. For example, in the $N - k$ security criterion problem in power systems, the goal is to guarantee the system is operable even if up to k of N components fail. A budget constraining the number of failtures to k is then added in the uncertainty set. Different uncertainty sets have been considered in the literature to control for the level of conservatism and their tractability studied [11].

There exists other methodologies in optimization under uncertainty where risk measures are incorporated, or where chance-constraints are included to ensure constraints involving uncertainty are satisfied with at least a pre-defined probability. A closely related paradigm is distributionally robust optimization (DRO) [12]. Indeed, it has been shown that under certain conditions, risk measures admit an equivalent DRO model (see [13]). In DRO, we seek to minimize the worst-case expected cost with respect to probability distributions belonging to some ambiguity set. In the context of two-stage models, we minimize the sum of the first-stage cost and the worst-case expected second-stage cost. Note the distinction between the uncertainty set in RO defined in the space of the uncertainty, and the ambiguity set in DRO defined in the space of probability distributions.

1.3.1 Distributionally Robust Optimization Under The Wasserstein Ambiguity Set

Priot Work and Motivation

There is a wide variety of ambiguity sets that have been studied in the literature. Ambiguity sets can be roughly split into one of two categories: moment-based ambiguity sets and balls in the space of probability distributions centered at some nominal distribution. In the former, the ambiguity sets include constraints on the moments of the distribution [14, 15, 16]. Given point estimates of the moments, the ambiguity set can include constraints such that moments match the estimates or are some distance away from the estimates to account for the uncertainty in the latter. In [17], the authors propose an ambiguity set which constrains the mean and the second-order moments to lie in an ellipsoid centered at the estimate of the mean and in the intersection of two positive semi-definite cones, respectively. Moment-based ambiguity sets have gained popularity for their computational benefits [18]. On the other hand, balls centered at a nominal distribution are defined using a metric defined on probability distributions. Popular metrics include the ϕ-divergence [19, 20], the Kullback-Leibler divergence [21, 22], which is a special case of the ϕ-divergence, and the Wasserstein metric [23, 18, 24, 25, 26].

The Wasserstein set is a ball in the space of probability distributions defined on the Wasserstein metric and centered at a nominal distribution, typically an empirical distribution obtained from a set of samples. The Wasserstein ambiguity set has recently received great attention due to its finite-sample guarantees and out-of-sample performance [23, 18, 27, 28, 24]. It has been shown that the DRO problem can be reformulated as a tractable convex program for various classes of value functions $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ and support sets [18, 23]. In [29], Weijun Xie provides tractability results for zero-one uncertainty under the type-∞ Wasserstein set. Most work has focused on one-stage problems involving continuous random variables where the support set is a polytope. Problems where the support set is finite generally rely on a combination of cutting plane and colum generation procedures (e.g. [26, 30], where two-stage problems are considered).

The robust optimization literature is rich in tackling two-stage problems. A common approximation in RO involves decision rules [31, 32]. Such methods tend to lead to a more tractable model, and are even optimal in certain applications [33]. Examples of such techniques are affine policies, where the second-stage variables are restricted to an affine policy with respect to the uncertainty [33, 34].

Given a finite training data set or if data is scarce, SAA can lead to poor out-of-sample performance [18, 10, 35]. Whereas SAA can lead to an optimistic solution in such situations, DRO hedges against overfitting the data. In [24], Daniel Kuhn et al. show that regularization techniques in classification and regression are equivalent to estimating the predictors using DRO under the Wasserstein ball. On the other hand, RO can lead to a very conservative and high cost solution. Using a ball centered at the empirical distribution as an ambiguity set leads to a nice generalization of SAA and RO. If the radius of the ball is 0, then we get an equivalent problem to SAA as the ambiguity set is a singleton corresponding to the empirical distribution, wheras if the radius is large enough, then the ball includes all probability distributions which assign a weight of one to a scenario and zero to the rest, and we thus get an equivalent RO problem. The radius of the ball provides decision-makers with a tangible control over risk-aversness.

Finally, we note that the Wasserstein ambiguity set does not share some of the shortcomings of other sets, which makes it an attractive choice. In DRO, we want the ambiguity set to be rich enough to include the true distribution, but not too large to exclude pathological ones. A discussion of shortcomings of the ϕ-divergence metric that are not shared by the Wasserstein metric can be found in [23], where the former can exclude the true distribution while including pathological ones. Moreover, when using the Kullback-Leibler divergence metrics, the worst-case distribution can only have a support on observed scenarios, thus not protecting against unobserved scenarios. In contrast, the Wasserstein set can assign a non-zero probability to any scenario in the support set. This is especially advantageous in applications where we wish to protect ourselves against unobserved scenarios.

Preliminaries

We present some preliminaries for two-stage distributionally robust optimization under the Wasserstein ambiguity set and some background which will be useful in Chapter 3 and Chapter 4. Assume we have a set of N samples $\{\hat{\boldsymbol{\xi}}^{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{\xi}}^{\boldsymbol{N}}\}$ indexed by $n \in \mathcal{N}$ (as in SAA) whose empirical distribution is defined as $\mathbb{P}_N = \frac{1}{N}\sum_n \delta_{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}}$, where $\delta_{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}}$ is the dirac distribution assigning unit mass to $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$, i.e. each sample is assigned equal probability. The two-stage DRO problem can be written as

$$\min_{\boldsymbol{x} \in X} \quad \boldsymbol{c}^\top \boldsymbol{x} + \max_{\mathbb{P} \in \mathcal{B}(\mathbb{P}_N, \theta)} \mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})\right] \tag{DRO}$$

where $\mathcal{B}_W(\mathbb{P}_N, \theta)$ is the Wasserstein ball of radius $\theta \geq 0$ centered at the empirical distribution $\mathbb{P}_N$. The Wasserstein ball is defined as

$$\mathcal{B}_W(\mathbb{P}_N, \theta) = \{\mathbb{P} \in \mathcal{M}(\Xi) : \mathrm{d}_W(\mathbb{P}, \mathbb{P}_N) \leq \theta\}.$$

where $\mathcal{M}(\Xi)$ is the set of all distributions supported on Ξ. Given a distribution $\mathbb{P}$ defined by a vector $\{p_s\}_{s\in\mathcal{S}}$ and a valid metric $\mathrm{d}(\cdot,\cdot)$ defined on Ξ, the Wasserstein distance $\mathrm{d}_W(\mathbb{P},\mathbb{P}_N)$ between $\mathbb{P}$ and $\mathbb{P}_N$ corresponds to the following minimization problem:

$$\begin{aligned}
\min_{\boldsymbol{\pi}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \mathrm{d}(\boldsymbol{\xi^s},\hat{\boldsymbol{\xi}}^{\boldsymbol{n}})\pi_{sn} \\
\text{s.t.} \quad & \sum_{n\in\mathcal{N}} \pi_{sn} = p_s, \qquad s\in\mathcal{S}, \\
& \sum_{s\in\mathcal{S}} \pi_{sn} = \frac{1}{N}, \qquad n\in\mathcal{N}, \\
& \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \pi_{sn} = 1, \\
& \pi_{sn} \geq 0, \qquad\qquad s\in\mathcal{S},\ n\in\mathcal{N}
\end{aligned} \tag{W}$$

Let $q_s^n = N\pi_{sn}$ be the conditional probability of $\boldsymbol{\xi^s}$ given that we have observed $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$ (the sampled scenario). It is convenient to write (W) as:

$$\begin{aligned}
\min_{\boldsymbol{q}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N} q_s^n\, \mathrm{d}(\boldsymbol{\xi^s},\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}) \\
\text{s.t.} \quad & \sum_{n\in\mathcal{N}} \frac{1}{N} q_s^n = p_s, \forall s\in\mathcal{S}, \\
& \sum_{s\in\mathcal{S}} q_s^n = 1, \qquad \forall n\in\mathcal{N}, \\
& q_s^n \geq 0, \qquad\qquad s\in\mathcal{S},\ n\in\mathcal{N}
\end{aligned} \tag{W'}$$

where $\{\boldsymbol{\xi^s}\}_{s\in\mathcal{S}}$ is the set of all scenarios and $\{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}\}_{n\in\mathcal{N}}$ is the set of sampled scenarios. The following theorem presents a known extensive reformulation of (DRO) which we will use in Chapter 3 and 4.

Theorem 1. *Problem* (DRO) *is equivalent to*

$$\begin{aligned}
\min_{\boldsymbol{x}, \boldsymbol{\gamma}, \alpha} \quad & \boldsymbol{c}^\top \boldsymbol{x} + \theta\alpha + \frac{1}{N}\sum_{n\in\mathcal{N}} \gamma_n \\
\text{s.t.} \quad & \gamma_n \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) - \alpha\, \mathrm{d}(\boldsymbol{\xi^s}, \boldsymbol{\hat{\xi}^n}),\, s \in \mathcal{S}, n \in \mathcal{N}, \\
& \boldsymbol{x} \in X, \\
& \alpha \geq 0
\end{aligned}$$

Proof. Using (W'), the inner problem of (DRO) can be written as

$$\begin{aligned}
\max_{\boldsymbol{p}} \quad & \sum_{s\in\mathcal{S}} p_s \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) \\
\text{s.t.} \quad & \min_{\boldsymbol{q}\geq 0} \left\{ \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N}\, \mathrm{d}(\boldsymbol{\xi^s}, \boldsymbol{\hat{\xi}^n}) q_s^n : \begin{array}{l} \sum_{s\in\mathcal{S}} q_s^n = 1,\ \forall n \in \mathcal{N} \\ \sum_{n\in\mathcal{N}} \frac{1}{N} q_s^n = p_s,\ \forall s \in \mathcal{S} \end{array} \right\} \leq \theta
\end{aligned}$$

which is equivalent to

$$\begin{aligned}
\max_{\boldsymbol{q}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N} q_s^n \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) \\
\text{s.t.} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N}\, \mathrm{d}(\boldsymbol{\xi^s}, \boldsymbol{\hat{\xi}^n}) q_s^n \leq \theta, \\
& \sum_{s\in\mathcal{S}} q_s^n = 1 \qquad\qquad \forall n \in \mathcal{N}, \\
& q_s^n \geq 0, \qquad\qquad \forall s \in \mathcal{S},\ \forall n \in \mathcal{N}.
\end{aligned} \tag{1.3.1}$$

Its dual is

$$\begin{aligned}
\min_{\boldsymbol{\gamma}, \alpha} \quad & \theta\alpha + \frac{1}{N}\sum_{n\in\mathcal{N}} \gamma_n \\
\text{s.t.} \quad & \gamma_n \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) - \alpha\, \mathrm{d}(\boldsymbol{\xi^s}, \boldsymbol{\hat{\xi}^n}), \forall s \in \mathcal{S},\ \forall n \in \mathcal{N}, \\
& \alpha \geq 0.
\end{aligned}$$

We then have

$$
\begin{aligned}
\min_{\boldsymbol{x}, \boldsymbol{\gamma}, \alpha} \quad & \boldsymbol{c}^\top \boldsymbol{x} + \theta\alpha + \frac{1}{N}\sum_{n\in\mathcal{N}} \gamma_n \\
\text{s.t.} \quad & \gamma_n \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) - \alpha\,\mathrm{d}(\boldsymbol{\xi^s}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}),\ s \in \mathcal{S}, n \in \mathcal{N}, \\
& \boldsymbol{x} \in X, \\
& \alpha \geq 0
\end{aligned}
$$

□

In the reformulated inner maximization (1.3.1), the conditional probabilities q_s^n can be interpreted as the hedging strategy, where we are moving weight from the sampled scenario $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$ to scenario $\boldsymbol{\xi^s}$, at a cost of $\frac{1}{N}\,\mathrm{d}(\boldsymbol{\xi^s}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}})$, and the total weight moved from sample $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$ to all scenarios $\{\boldsymbol{\xi^s}\}_{s\in\mathcal{S}}$ has to equal one.

1.3.2 Data-Driven Two-Stage Conic Optimization with Rare High-Impact Zero-One Uncertainties

In Chapter 3, we study two-stage distributionally robust conic optimization with zero-one uncertainties under the Wasserstein ambiguity set. We are motivated by problems arising in network optimization where the binary random variables represent failures of components in the network. More specifically, we are interested in applications where failure events are rare and have a high impact. In such applications, approximating the failure probability is difficult. We are dealing with a very large set of scenarios which increases exponentially in the number of network components, the failure events are unlikely to be independent of each other due to phenomena like the cascading effect (i.e. component failures might trigger additional failures of nearby components), and the historical data is not rich enough to accurately estimate failure probabilities. This motivates solving a DRO model instead.

Due to our support set being a binary non-convex set, we cannot use existing reformulation techniques which rely on a continuous, convex support set. By using ideas from bilinear programming and penalty methods, we reformulate our problem by decomposing the inner maximization problem into maximization problems for each sampled scenario, where the latter is a maximization of a linear function over the convex-hull of a mixed-integer conic set. We then use Lovász-Schrijver approximations to get outer descriptions of the convex hulls which can be iteratively improved, permitting us to use duality and solve the model using commercial solvers. We demonstrate the benefits of our method in computational and out-of-sample performance compared to SAA and RO on challenging optimal power flow problems where transmission lines are subject to random failures, and a multi-commodity network design problem with random node failures.

1.3.3 Binary Distributionally Robust Optimization Under the Wasserstein Set: A Disaster Relief Application

In Chapter 4, we extend our previous work to consider binary variables in both stages, where we are dealing with a finite but large set of scenarios. We are specifically interested in a two-stage problem arising in natural disaster management, where the first stage is a facility location problem, deciding on where to open facilities and how much resources to pre-allocate (e.g. medical kits or food), and the second stage is a fixed-charge transportation problem, routing the resources to affected areas after a disaster. Similar to the applications considered in Chapter 3, natural disasters are relatively rare events and have a high impact. Thus, we cannot reliably estimate the probability distribution of the uncertain parameters [36, 37]. The presence of binary variables in the second stage significantly complicates the problem, as we cannot use duality tools to reformulate our two-stage model as a tractable convex program without relying on approximations or relaxations. We instead develop a column-and-constraint generation (CCG) algorithm, where we generate scenarios as needed. To circumvent the computational difficulties which are typically faced in CCG algorithms, we leverage the structure of our support set and second-stage value function, leading to an efficient scenario generating procedure. We also show how our results extend to the case where the second stage is a fixed-charge network flow problem. We end the chapter with computational experiments illustrating the strong computational performance of our proposed method, and analyze the solutions obtained from DRO and SAA on a case study of hurricane threats in the coastal states of the United States, along the Gulf of Mexico and Atlantic Ocean.

1.4 Contributions

We summarize the contributions of this thesis in this section. The start of the thesis is dedicated to a deterministic setting where we tackle large-scale linear programs with a special structure that can be solved using Dantzig-Wolfe decomposition. More specifically, we are motivated by problems where either privacy is of concern, or data is stored in a decentralized fashion. In such cases, a classical Dantzig-Wolfe decomposition algorithm cannot be used as the master problem would be solved centrally. To this end, we develop a fully-distributed, consensus-based Dantzig-Wolfe decomposition algorithm where privacy of information is satisfied, and data is not required to be available centrally at any time. The only information that agents exchange are dual vectors. We provide a theoretical analysis of our algorithm, deriving bounds on the optimality gap and feasibility violation. We perform computational experiments to illustrate the performance of our method, where we retrieve high quality solutions in competitive time.

Following Chapter 2, we turn our attention back to two-stage stochastic programming. We are specifically interested in applications where data is limited, and the underlying probability distribution is difficult to estimate.

In Chapter 3, we tackle two-stage distributionally robust conic optimization under the Wasserstein ambiguity sets, where we are interested in applications affected by rare high-impact zero-one uncertainties. Such applications are common in network optimization problems. In this work, we derive tractable approximations to the two-stage distributionally robust model which can be iteratively improved, where we exploit ideas from penalty methods and bilinear programming to get a single global optimization problem. Using a reformulation based on a penalty method, we are able to transform a problem with right-hand side uncertainty to a problem with only objective uncertainty. We show that we can exactly compute a finite penalty parameter. We then demonstrate the computational and out-of-sample performance on the optimal power flow problem with random transmission line failures and a multi-commodity network design problem with random node failures.

We extend our work to consider second-stage binary variables in Chapter 4, in which case $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is no longer convex. We narrow our focus to a two-stage model which often arises in natural disaster management, where the first stage is a facility location problem and the second stage is a fixed-charge transportation problem. We develop a support set tailored to the application, and leverage the structure of our support set and second-stage value function $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ to design an efficient column-and-constraint generation algorithm. We also provide extensions to a fixed-charge network flow second-stage problem. We demonstrate the strong computational performance of our method over classical column-and-constraint generation algorithms on synthetic instances, and present a case study from the literature on hurricane threats on coastal states of the United States, analyzing the solution and out-of-sample performance of our DRO model and SAA. We show that the distributionally robust model outperforms SAA in out-of-sample cost in many instances, and consistently satisfies significantly more demand on average after a disaster occurs.

CHAPTER 2
CONSENSUS-BASED DANTZIG-WOLFE DECOMPOSITION

2.1 Introduction

2.1.1 Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition (DWD) [7] is a classical algorithm for solving large-scale linear programs whose constraint matrix involves a set of independent blocks coupled with a set of linking rows. This class of problems are of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \sum_{n=1}^{N} \boldsymbol{c_n}^\top \boldsymbol{x_n} \\ \text{s.t.} \quad & \sum_{n=1}^{N} A_n \boldsymbol{x_n} = \boldsymbol{t}, \\ & \boldsymbol{x_n} \in X_n, \qquad n = 1, \ldots, N \end{aligned} \tag{P}$$

where N is the number of blocks, the set X_n denotes the feasible set of the n-th block (or subproblem) with the decision vector $\boldsymbol{x_n}$, and $\sum_{n=1}^{N} A_n \boldsymbol{x_n} = t$ denotes the system of linking constraints. The DWD method decomposes (P) into a master problem and a set of N independent subproblems. Throughout this chapter, we assume that the sets X_n are polytopes (i.e. bounded polyhedra) for all n. In this case, the master problem is a reformulation where each vector $\boldsymbol{x_n}$ is replaced by a convex combination of the extreme points of X_n. From Minkowski's theorem, the master problem can be written as

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} \boldsymbol{c_n}^\top \boldsymbol{x_n^i} \lambda_{ni} \\
\text{s.t.} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} A_n \boldsymbol{x_n^i} \lambda_{ni} = t, \\
& \sum_{i \in I_n} \lambda_{ni} = 1, \qquad && n = 1, \ldots, N, \\
& \lambda_{ni} \geq 0, && \forall i \in I_n,\ n = 1, \ldots, N
\end{aligned} \tag{2.1.1}$$

where $\{\boldsymbol{x_n^i}\}_{i \in I_n}$ and I_n correspond to the extreme points of X_n and their index set, respectively; the variable λ_{ni} is the convex multiplier associated with extreme point i of subproblem n.

Since the number of extreme points of the subproblem polytopes can be exponentially large, a restricted master problem (RMP) is solved instead, using only a subset of the extreme points (or columns). Using an optimal dual solution of the RMP, each block n solves a pricing subproblem independently to find a new extreme point of X_n with negative reduced cost. This process (called column generation) is repeated until no columns with negative reduced costs can be found. The optimal solution of RMP in the last iteration then provides an optimal solution to (P).

2.1.2 Motivation

DWD has been used to solve a variety of problems, many arising in energy and transportation, and has received great attention in solving mixed-integer programs to obtain tighter relaxation bounds, leading to the popular branch-and-price algorithm. See [38, 39, 40, 41] for details and applications of DWD and column generation.

In each iteration of DWD, the subproblems can be solved independently in a distributed manner but the RMP is solved centrally. In certain settings, solving the master problem centrally is undesirable or infeasible. With increasingly large amounts of data available, we are seeing an upward trend in decentralized storage of data, either as a protection against attacks, or due to memory limitations, in which case the data is not available centrally to solve the master problem [42]. Alternatively, if we interpret the variables of each block as the decision variables of independent agents, then solving the RMP centrally requires the agents to share data of their constraints, objective, and decisions, potentially violating privacy. In such applications, agents have local constraints and a local objective function but share common constraints (e.g. a resource), making DWD an appropriate method [43]. One example is in collaborative logistics, where companies, potentially competitors, share routes and resources to merge loads and cut overall costs, but such collaborations might be infeasible if corporations are unwilling to share data [43]. Privacy concerns are also prevalent in energy, such as in smart-grid optimization, for example. In order to optimize power generation, smart meters provide sensitive and private energy consumption data of the household to the utility provider [44, 45]. Another example is in the healthcare industry with Accountable Care Organizations (ACOs), where a collection of hospitals or clinics form a group to give coordinated care to patients. To drive costs down and improve patient care, sensitive patient data would link the optimization problem, causing concerns for privacy violations among ACOs [46, 47]. Developing a consensus-based Dantzig-Wolfe algorithm would permit the use of DWD in such scenarios, while handling decentralized storage of data and/or privacy concerns.

2.1.3 Contribution

In this chapter, we propose a consensus-based DWD algorithm which relies on solving the dual of the master problem using a consensus-based Alternating Direction Method of Multipliers (ADMM) algorithm. We address the computational challenges and theoretical questions that arise from solving the master problem in a distributed fashion. To the best of our knowledge, a fully distributed consensus-based technique to solve the master problem has not been studied within the context of column generation or Dantzig-Wolfe decomposition where privacy is fully preserved.

We illustrate the benefits of working on the dual and using ADMM as a consensus-based method. To reduce computation time, we dynamically adjust the tolerances, where we first aim for inaccurate dual solutions to speed up ADMM convergence, similar to [48, 49], and lower the tolerances as the algorithm progresses. This significantly decreases the number of times we need to solve ADMM to high accuracy, yielding computational benefits. ADMM is known to be slow to converge to high accuracy, but requires only a few iterations to achieve modest accuracy [42], and thus synergizes well with solving for inaccurate duals at first. Moreover, ADMM has stronger convergence properties than subgradient methods [42]. By solving the dual of (2.1.1) using ADMM, the master problem is solved in a distributed fashion and privacy of information is guaranteed to be preserved. Indeed, each agent need only share their dual variables with a central coordinator, as is common in a privacy preserving setting. Moreover, we note that we circumvent primal-recovery issues present in subgradient methods and avoid the need for ergodic sequences [50, 51]. We show we can easily recover a primal solution to the original problem that is close to feasible and close to optimal using the Lagrangian multiplers associated with the constraints in the ADMM subproblems. Finally, while most stability techniques in DWD rely on suboptimal duals, our method also handles infeasible duals, as the ADMM approach provides ϵ-optimal and δ-feasible dual solutions. We prove bounds on the feasibility violation and optimality gap at the recovered primal solution.

We provide preliminary computational results for the proposed algorithm using a Message Passing Interface (MPI) implementation on cutting stock instances from the literature and synthetic instances where we obtain high quality solutions. Although our main contribution is to tackle decentralized storage of data and privacy, our method also shows a potential computational benefit for instances with a large number of variables and in many of the cutting stock instances.

2.1.4 Prior Work

Solving the dual of RMP in a distributed manner leads to approximate dual solutions used in the pricing subproblems, as opposed to standard DWD where exact optimal dual solutions are readily available. Several stability techniques proposed in the literature use suboptimal but feasible dual solutions to solve the pricing subproblems to circumvent unstable behavior resulting from using optimal dual solutions [48]. One method is to add a penalty term to reduce the variation of obtained dual solutions [52, 53]; another involves using primal-dual interior point method to solve for suboptimal dual solutions which are well-centered, where the optimality tolerance of the interior point method is dynamically adjusted to reduce the computation time of solving the RMP [48, 53, 49]. In these methods, suboptimal dual solutions are used to solve the pricing subproblems, where tolerances are adjusted to satisfy a specified duality gap, but solve the RMP centrally. In the context of dual decomposition in stochastic integer programming, Lubin et al. discuss the computational benefits of using tailored interior-point methods such as PIPS-IPM to solve the master problem in [54]. Such solvers leverage dual block angular structures by parallelizing the linear algebra. Closest in spirit to our work is [55], where a consensus-based cutting-plane procedure is developed. This work addresses a different setting where communication is asynchronous, but agents are required to share all their cutting planes with neighbors, thus potentially violating privacy. An example application to Dantzig-Wolfe decomposition is also presented, where the consensus-based cutting-plane procedure is applied to the dual of the problem. Besides the potential privacy violation, although the authors consider more general convex objective functions, a final recovery step is necessary to obtain a primal solution if the objective function is not strongly convex, where the master problem involving columns generated by all agents is solved centrally.

In privacy-preserving optimization, common techniques include random matrix transformations or perturbations. In [56], the author proposes to multiply the constraints by a randomly generated matrix in the case of equality constraints. A similar method to handle inequality constraints is proposed in [57]. In [43], the authors apply Dantzig-Wolfe decomposition to a transformed problem to preserve privacy, where the constraint matrices of each agent are multiplied by a random matrix, but requires all the data to be centrally located to solve the master problem. Many other methods rely on distributed algorithms to preserve privacy of information [45, 58].

Consensus problems and distributed optimization algorithms have been heavily studied. A classic method is dual decomposition where the linking constraints are relaxed, and the problem becomes separable. The algorithm alternates between solving local subproblems independently and a central step which updates the dual variables as in dual ascent [42]. Many variants of dual decomposition have been proposed, such as using subgradients to update the dual iterates when optimizing nonsmooth functions [59]. Many of the distributed methods are optimized over a network, where agents, treated as nodes, only share limited information with neighbors according to a transition matrix [60, 61, 62]. In [63], Nedic and Ozdaglar give a survey on dual decomposition techniques for distributed optimization and consensus problems. Dual decomposition is known to suffer from weak convergence properties, which led to the augmented Lagrangian method. It provides stronger convergence properties and requires fewer assumptions than dual ascent (the method behind dual decomposition), but leads to an optimization problem that cannot be solved in a distributed way. ADMM has been developed to leverage the decomposability of dual decomposition, and the nice convergence properties of the augmented Lagrangian method [42]. Theoretical results and convergence properties of ADMM have been thoroughly studied in [42, 64, 65, 66, 67]. Studies on parameter tuning have also been done, notably the penalty parameter [68, 69].

The remainder of the chapter is organized as follows. Section 2.2 formally defines the problem structure we are interested in and establishes the notation used throughout. In Section 2.3, we give a brief overview of consensus-based ADMM and discuss traditionally used stopping criteria. In Section 2.4, we describe our algorithm and prove bounds on the optimality gap and feasibility violation. We include numerical results to illustrate our method in Section 2.5.

2.2 Preliminaries

We are interested in problems of the form of (P) where $\boldsymbol{c_n}$ and X_n are the cost vector and local constraints of block $n = 1, \ldots, N$, and A_n is the constraint matrix of block n in the linking constraints. To simplify notation, we rewrite the master problem as

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} c_n^i \lambda_{ni} \\
\text{s.t.} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} \boldsymbol{A_n^i} \lambda_{ni} = \boldsymbol{t}, \\
& \sum_{i \in I_n} \lambda_{ni} = 1, \qquad n = 1, \ldots, N, \\
& \lambda_{ni} \geq 0, \qquad n = 1, \ldots, N
\end{aligned} \tag{MP}$$

where $c_n^i = \boldsymbol{c_n}^\top \boldsymbol{x_n^i}$ and $\boldsymbol{A_n^i} = A \boldsymbol{x_n^i}$ for all $i \in I_n$ and for all n.

We assume each X_n to be a non-empty polytope, so that there exists $L_n > 0$ such that $\left\| \boldsymbol{x_n^i} \right\|_2 \leq L_n$ for all extreme points $\boldsymbol{x_n^i}$ of X_n. We further assume problem (MP) to be feasible and to have an optimal solution. The dual of (MP) is

$$\begin{aligned}
\max_{\boldsymbol{\pi}, \boldsymbol{u}} \quad & \boldsymbol{t}^\top \boldsymbol{\pi} + \sum_{n=1}^{N} u_n \\
\text{s.t.} \quad & A_n^{i\top} \boldsymbol{\pi} + u_n \leq c_n^i, \quad \forall i \in I_n, \ n = 1, \ldots, N
\end{aligned}$$

where $\boldsymbol{\pi}$ are dual variables associated with the linking constraints and u_n are dual variables associated with the convexity constraints $\sum_{i \in I_n} \lambda_{ni} = 1$ for all n. We restrict the presentation of our method and analysis to the case of linking equality constraints without loss of generality.

We refer to the optimal objective values of the master problem and its dual by z_{MP}^*

and z^*_{DM}. When considering a subset of the extreme points, we refer to the restricted primal and dual problems as RMP and RDM, and their optimal values by z^*_{RMP} and z^*_{RDM}, respectively. Approximate solutions and their objective values are denoted by a hat. The notations $\|\cdot\|$ and $\|\cdot\|_F$ refer to the ℓ_2-norm and Frobenius norm, respectively. Vectors are printed in bold. Finally, the terms agent and block will be used interchangeably.

2.3 ADMM Overview

We give a brief overview of consensus-based ADMM. We present the algorithm and known convergence conditions. Detailed discussion of the ADMM method and its convergence properties can be found in [42].

2.3.1 Consensus-Based ADMM

Consensus-based ADMM is well-suited for problems of the form

$$\begin{aligned} \max_{\boldsymbol{x}} \quad & \sum_{n=1}^{N} f_n(\boldsymbol{x}) \\ \text{s.t.} \quad & A_n\boldsymbol{x} \le \boldsymbol{b_n} \quad \forall n = 1,\dots,N \end{aligned} \tag{A1}$$

where $f_n : \mathbb{R}^d \to \mathbb{R}$ are convex, proper and closed functions, and $A \in \mathbb{R}^{m\times d}$. The objective function and constraints are linked solely by the variable $\boldsymbol{x}$. We can equivalently rewrite (A1) as

$$\begin{aligned} \max_{\boldsymbol{x_n},\boldsymbol{x}} \quad & \sum_{n=1}^{N} f_n(\boldsymbol{x_n}) \\ \text{s.t.} \quad & A_n\boldsymbol{x_n} \le \boldsymbol{b_n} \quad n = 1,\dots,N, && \text{(2.3.1a)} \\ & \boldsymbol{x_n} = \boldsymbol{x} \qquad\;\; n = 1,\dots,N. && \text{(2.3.1b)} \end{aligned}$$

Let $\boldsymbol{\lambda_n} \in \mathbb{R}^m$ and $\boldsymbol{\alpha_n} \in \mathbb{R}^d$ be the Lagrangian multipliers of (2.3.1a) and (2.3.1b), respectively, for $n = 1, \dots, N$. Taking the augmented Lagrangian of (2.3.1) gives

$$\begin{aligned} \max_{\boldsymbol{x_n}, \boldsymbol{x}} \quad & \sum_{n=1}^{N} \left[f_n(\boldsymbol{x_n}) + \alpha_n^\top (\boldsymbol{x} - \boldsymbol{x_n}) - \frac{\rho}{2} \|\boldsymbol{x} - \boldsymbol{x_n}\|^2 \right] \\ \text{s.t.} \quad & A_n \boldsymbol{x_n} \le \boldsymbol{b_n} \quad n = 1, \dots, N \end{aligned} \tag{AL}$$

where $\rho > 0$ is a predetermined penalty parameter. Define the objective of (AL) as

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{x_1}, \dots, \boldsymbol{x_N}, \boldsymbol{\alpha_1}, \dots, \boldsymbol{\alpha_N}) = \sum_{n=1}^{N} \left[f_n(\boldsymbol{x_n}) + \boldsymbol{\alpha_n}^\top (\boldsymbol{x} - \boldsymbol{x_n}) - \frac{\rho}{2} \|\boldsymbol{x} - \boldsymbol{x_n}\|^2 \right]$$

The ADMM method consists of alternating between maximizing the function $\mathcal{L}_\rho$ over $(\boldsymbol{x}, \boldsymbol{x_1}, \dots, \boldsymbol{x_N})$ and minimizing over $(\boldsymbol{\alpha_1}, \dots, \boldsymbol{\alpha_N})$, where the maximization step is done sequentially, so that we first maximize over $(\boldsymbol{x_1}, \dots, \boldsymbol{x_N})$ before maximizing over $\boldsymbol{x}$. This allows to solve the former in a distributed fashion. The ADMM steps at an iteration k can be summarized as follows:

$$\boldsymbol{x_n^{k+1}} \leftarrow \operatorname*{argmax}_{\boldsymbol{x_n}} \mathcal{L}_\rho(\boldsymbol{x^k}, \boldsymbol{x_1}, \dots, \boldsymbol{x_N}, \boldsymbol{\alpha_1^k}, \dots, \boldsymbol{\alpha_N^k}), \ \forall n = 1, \dots, N \tag{2.3.2a}$$

$$\boldsymbol{x^{k+1}} \leftarrow \operatorname*{argmax}_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{x_1^{k+1}}, \dots, \boldsymbol{x_N^{k+1}}, \boldsymbol{\alpha_1^k}, \dots, \boldsymbol{\alpha_N^k}) \tag{2.3.2b}$$

$$\boldsymbol{\alpha_n^{k+1}} \leftarrow \boldsymbol{\alpha_n^k} - \rho(\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}), \ \forall n = 1, \dots, N. \tag{2.3.2c}$$

Note that (2.3.2c) is a gradient step where the step size is the penalty parameter ρ and (2.3.2b) is an unconstrained maximization problem for which there exists a closed form solution. We have:

$$\begin{aligned} \nabla_x \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{x_1^{k+1}}, \dots, \boldsymbol{x_N^{k+1}}, \boldsymbol{\alpha_1^k}, \dots, \boldsymbol{\alpha_N^k}) = 0 \Rightarrow & \sum_{n=1}^{N} \left[\boldsymbol{\alpha_n^k} - \rho(\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}) \right] = 0 \\ \Rightarrow & \boldsymbol{x^{k+1}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x_n^{k+1}} + \frac{1}{N\rho} \sum_{n=1}^{N} \boldsymbol{\alpha_n^k}. \end{aligned}$$

2.3.2 Convergence and Stopping Criteria

We first note that dual feasibility in (2.3.1) is equivalent to $\nabla_{\boldsymbol{x_n}} f_n(\boldsymbol{x_n}) - A_n^\top \boldsymbol{\lambda_n} - \boldsymbol{\alpha_n} = 0$. From the optimality conditions of (AL), we have at iteration k:

$$\begin{aligned}
&\nabla_{\boldsymbol{x_n}} f_n(\boldsymbol{x_n^{k+1}}) - A_n^\top \boldsymbol{\lambda_n^{k+1}} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{x^k} - \boldsymbol{x_n^{k+1}}) = 0 \\
\Rightarrow &\nabla_{\boldsymbol{x_n}} f_n(\boldsymbol{x_n^{k+1}}) - A_n^\top \boldsymbol{\lambda_n^{k+1}} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{x^k} - \boldsymbol{x_n^{k+1}} + \boldsymbol{x^{k+1}} - \boldsymbol{x^{k+1}}) = 0 \\
\Rightarrow &\nabla_{\boldsymbol{x_n}} f_n(\boldsymbol{x_n^{k+1}}) - A_n^\top \boldsymbol{\lambda_n^{k+1}} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}) + \rho(\boldsymbol{x^k} - \boldsymbol{x^{k+1}}) = 0 \\
\Rightarrow &\nabla_{\boldsymbol{x_n}} f_n(\boldsymbol{x_n^{k+1}}) - A_n^\top \boldsymbol{\lambda_n^{k+1}} - \boldsymbol{\alpha_n^{k+1}} = \rho(\boldsymbol{x^{k+1}} - \boldsymbol{x^k}).
\end{aligned}$$

Thus, dual feasibility in (2.3.1) amounts to having $\rho(\boldsymbol{x^{k+1}} - \boldsymbol{x^k}) = 0$.

Let $\boldsymbol{\alpha}$ be the vertical concatenation of vectors $\{\boldsymbol{\alpha_n}\}_{n=1}^N$. As proven in [42], under the assumption that the functions f_n in (A1) are convex, proper and closed, and assuming that $\mathcal{L}_0(\boldsymbol{x}, \boldsymbol{x_1}, ..., \boldsymbol{x_N}, \boldsymbol{\alpha})$, where $\rho = 0$, has a saddle point, then as $k \to \infty$, we have the following:

(i) Primal feasibility violation vanishes: $\sqrt{\sum_{n=1}^N \left\|\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}\right\|^2} \to 0,\ n = 1, ..., N$

(ii) Dual feasibility violation vanishes: $\rho\left\|\boldsymbol{x^{k+1}} - \boldsymbol{x^k}\right\| \to 0$

(iii) Optimality gap vanishes: $\left\|f(\boldsymbol{x^*}) - f(\boldsymbol{x^k})\right\| \to 0$

(iv) Dual vector $\boldsymbol{\alpha}$ converges to an optimal dual solution: $\left\|\boldsymbol{\alpha^k} - \boldsymbol{\alpha}^*\right\| \to 0$

In (i), we define primal feasibility violation to be

$$\left\|\begin{matrix} \boldsymbol{x^{k+1}} - \boldsymbol{x_1^{k+1}} \\ \vdots \\ \boldsymbol{x^{k+1}} - \boldsymbol{x_N^{k+1}} \end{matrix}\right\| = \sqrt{\sum_{n=1}^N \left\|\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}\right\|^2}$$

This implies that we reach consensus as $k \to \infty$, i.e $\left\|\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}\right\| \to 0$ for all n.

For our purposes, we also assume the functions f_n to be differentiable. This assumption is satisfied in our case since we are dealing with linear cost functions. As suggested in [42], it is reasonable to terminate ADMM once we reach primal and dual feasibility within some tolerance. Given specified tolerances ϵ_p and ϵ_d, we terminate ADMM once $\sqrt{\sum_{n=1}^{N}\left\|\boldsymbol{x^{k+1}} - \boldsymbol{x_n^{k+1}}\right\|^2} \leq \epsilon_p$ and $\rho\left\|\boldsymbol{x^{k+1}} - \boldsymbol{x^k}\right\| \leq \epsilon_d$.

2.4 Consensus-Based Dantzig-Wolfe Algorithm

We first present the consensus-based Dantzig-Wolfe decomposition (CDWD) algorithm before deriving error bounds on the optimality gap and feasibility violation.

2.4.1 CDWD Algorithm

We define k to be the ADMM iteration counter and ℓ to be the Dantzig-Wolfe outer iteration counter. To solve the restricted master problem (RMP) in a distributed fashion, we solve a reformulation of the dual of RMP. The reformulation permits us to perform consensus-based ADMM. We split the dual vector $\boldsymbol{\pi}$ associated with the linking constraints into N copies as in (2.3.1) to get the following equivalent formulation:

$$\begin{aligned} \max_{\boldsymbol{\pi}, \boldsymbol{\pi_n}, u_n} \quad & \sum_{n=1}^{N} \left[\frac{1}{N}\boldsymbol{t}^\top \boldsymbol{\pi_n} + u_n\right] & \\ \text{s.t.} \quad & \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n} + u_n \leq c_n^i & \forall i \in I_n,\ n = 1, \ldots, N, \\ & \boldsymbol{\pi_n} = \boldsymbol{\pi} & n = 1, ..., N. \end{aligned} \tag{DM}$$

If there exists linking inequality constraints, then we can simply add appropriate non-negativity or non-positivity constraints on $\boldsymbol{\pi_n}$ for all n, and the remaining steps of the algorithm follow.

Note that the dual of the master problem is linked by dual variables $\boldsymbol{\pi}$. Performing ADMM on (DM) leads to a natural consensus-based algorithm with a guaranteed convergence, where the first ADMM block corresponds to solving for each $\boldsymbol{\pi_n}$ independently, and the second block corresponds to optimizing with respect to $\boldsymbol{\pi}$. Indeed, performing ADMM with more than two blocks requires additional conditions and correction steps to ensure convergence [70, 71, 72]. As a result, working on (DM) avoids any potential violation of privacy, as agents need only share dual vectors with a central coordinator, and does not require any data to be available centrally. Although one could solve (MP) using a dual decomposition method with subgradients while maintaining decomposability, two problems arise. First, privacy could be violated when computing the subgradients (i.e. the residuals of the linking constraints) [73, 74]. Second, using an averaging scheme would be necessary to recover an approximate primal solution. Indeed, it is well-known that simply solving the Lagrangian dual at the optimal dual solution does not necessarily return a feasible primal solution [75, 42]. Although out of the scope of this chapter and further experiments are required, we note that averaging schemes did not perform well for simple instances in our setting. We show that by performing ADMM on (DM), we circumvent primal-recovery issues and can easily retrieve high quality primal solutions via the Lagrangian multipliers associated with the constraints in $(ARDM_n)$ (defined below).

We denote the restricted problem of (DM), i.e. one involving constraints corresponding to only a subset of the columns, by RDM and its optimal value by z^*_{RDM}. We take the augmented Lagrangian of RDM by relaxing the copy constraints as in (AL) and get a separable problem with respect to variables $(\boldsymbol{\pi_n}, u_n)$:

$$\begin{aligned} \max_{\boldsymbol{\pi_n},\, u_n} \quad & \sum_{n=1}^{N} \left[\frac{1}{N} \boldsymbol{t}^\top \boldsymbol{\pi_n} + u_n + \boldsymbol{\alpha_n}^\top (\boldsymbol{\pi} - \boldsymbol{\pi_n}) - \frac{\rho}{2} \|\boldsymbol{\pi} - \boldsymbol{\pi_n}\|^2 \right] \\ \text{s.t.} \quad & \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n} + u_n \le c_n^i, \forall i \in I_n,\ n = 1, \ldots, N. \end{aligned}$$

At iteration k of ADMM and using current iterates $\boldsymbol{\pi^k}$ and $\boldsymbol{\alpha_n^k}$, each agent n solves

$$\begin{aligned} \max_{\boldsymbol{\pi_n},\, u_n} \quad & \frac{1}{N} \boldsymbol{t}^\top \boldsymbol{\pi_n} + u_n + \boldsymbol{\alpha_n^k}^\top (\boldsymbol{\pi^k} - \boldsymbol{\pi_n}) - \frac{\rho}{2} \left\|\boldsymbol{\pi^k} - \boldsymbol{\pi_n}\right\|^2 \\ \text{s.t.} \quad & \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n} + u_n \le c_n^i, \forall i \in I_n^\ell \end{aligned} \qquad (ARDM_n)$$

where $I_n^\ell \subseteq I_n$ is the index set of extreme points of block n at outer iteration ℓ. From (2.3.2), the steps to solving RDM can be summarized as follows:

1. Each agent solves $(ARDM_n)$ and collects optimal solutions $(\boldsymbol{\pi_n^{k+1}}, u_n^{k+1})$

2. $\boldsymbol{\pi^{k+1}} \leftarrow \frac{1}{N}\sum_{n=1}^N(\boldsymbol{\pi_n^{k+1}}) + \frac{1}{N\rho}\sum_{n=1}^N \boldsymbol{\alpha_n^k}$

3. $\boldsymbol{\alpha_n^{k+1}} = \boldsymbol{\alpha_n^k} - \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}})$, $\forall n = 1, \ldots, N$.

First note that $\boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} + u_n^{k+1} \leq c_n^i$ is satisfied for all $i \in I_n^\ell$ and for all n, since $(\boldsymbol{\pi_n^{k+1}}, u_n^{k+1})$ is a solution of $(ARDM_n)$. Thus, $\boldsymbol{\pi_n^{k+1}} = \boldsymbol{\pi^{k+1}}$ are the only violated constraints. To avoid confusion, we refer to $\sqrt{\sum_{n=1}^N \left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\right\|^2}$ as the dual feasibility violation, and $\rho\left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k}\right\|$ as the primal feasibility violation. Note that this is the opposite of what is defined in Section 2.3 because we are performing ADMM on the dual problem here. We then perform steps 1-3 until $\sqrt{\sum_{n=1}^N \left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\right\|^2} \leq \epsilon_d$ and $\rho\left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k}\right\| \leq \epsilon_p$, where ϵ_d and ϵ_p are dual and primal feasibility tolerances, respectively. Each agent n then solves a pricing subproblem to look for an extreme point with negative reduced cost:

$$z_{SEP}^n = \min_{\boldsymbol{x_n}}\{\boldsymbol{c_n}^\top \boldsymbol{x_n} - \boldsymbol{\pi^{k+1}}^\top A_n \boldsymbol{x_n} - u_n^{k+1} : \boldsymbol{x_n} \in X_n\}.$$

Let $\boldsymbol{x_n^*}$ be an optimal solution. In standard DWD, we would add $\boldsymbol{x_n^*}$ as a new column if $z_{SEP}^n < 0$. However, the dual solution $(\boldsymbol{\pi^{k+1}}, \{u_n^{k+1}\}_{n=1}^N)$ is ϵ-optimal and only close to feasible for the current RMP. It is possible that we find a column whose reduced cost is negative and close to 0 when evaluated at the approximate dual solutions, but is in fact already in the current RMP. It is also possible that at the (unavailable) optimal dual solution, the reduced cost is actually positive and the extreme point should not be added. To ensure a finite algorithm, agent n only adds $\boldsymbol{x_n^*}$ as a new extreme point if $z_{SEP}^n < -\max_{i\in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d$. In Lemma 1, we show that ADMM terminates with $c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1} \geq -\left\|\boldsymbol{A_n^i}\right\|\epsilon_d$ for all i and n. Thus, if $-\max_{i\in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d \leq z_{SEP}^n = \boldsymbol{c_n}^\top \boldsymbol{x_n^*} - \boldsymbol{\pi^{k+1}}^\top A_n \boldsymbol{x_n^*} - u_n^{k+1} < 0$, then we cannot guarantee that $\boldsymbol{x_n^*}$ is a necessary extreme point. In other words, we can only trust z_{SEP}^n within $\max_{i\in I_n^\ell}\{\|\boldsymbol{A_n^i}\|\}\epsilon_d$. This is necessary for the analysis in Section 2.4.2.

Lemma 1. *At outer-iteration ℓ, if ADMM terminates with $\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\| \leq \epsilon_d$ for all n, we have*

$$c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1} \geq -\|\boldsymbol{A_n^i}\|\epsilon_d$$

for all $i \in I_n^\ell, n = 1, ..., N$.

Proof. We have

$$\left\| \begin{matrix} \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_1^{k+1}} \\ \vdots \\ \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_N^{k+1}} \end{matrix} \right\| \leq \epsilon_d \Rightarrow \sum_{n=1}^{N} \|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\|^2 \leq \epsilon_d^2$$

$$\Rightarrow \|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\|^2 \leq \epsilon_d^2,\ \forall n = 1, \ldots, N$$

$$\Rightarrow \|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\| \leq \epsilon_d,\ \forall n = 1, \ldots, N.$$

For any n, computing the distance between $c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1}$ and $c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} - u_n^{k+1}$ gives us

$$\begin{aligned} \left\|c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1} - c_n^i + \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} + u_n^{k+1}\right\| &= \left\|\boldsymbol{A_n^i}^\top (\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}})\right\| \\ &\leq \|\boldsymbol{A_n^i}\|\epsilon_d,\ \forall i \in I_n^\ell \end{aligned} \tag{2.4.1}$$

Since $c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} - u_n^{k+1} \geq 0$ for all $i \in I_n^\ell$, (2.4.1) implies $c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1} \geq -\|A_n^i\|\epsilon_d$ for all $i \in I_n^\ell$ and n. □

Once the columns are added to the RMP, we use the solutions of the last iterates $\boldsymbol{\pi^{k+1}}$ and $\boldsymbol{\alpha_n^{k+1}}$ as warm starts for $\boldsymbol{\pi^1}$ and $\boldsymbol{\alpha_n^1}$ for all n in the next outer iteration $\ell + 1$. If $z_{SEP}^n \geq -\max_{i\in I_n^{\ell+1}}\{\|\boldsymbol{A_n^i}\|\}\epsilon_d$ for all n, we terminate the algorithm and retrieve approximate primal solutions $\boldsymbol{\hat{x}_n} \leftarrow \sum_{i\in I_n^{\ell+1}} \lambda_{ni}^{k+1}\boldsymbol{x_n^i}$ for all $n = 1, ..., N$, where λ_{ni}^{k+1} are the Lagrangian multipliers associated with the constraints in ($ARDM_n$), $i \in I_n^{\ell+1}, n = 1, ..., N$. The CDWD algorithm is summarized in Algorithm 1. At each ADMM iteration, the master (or central) node calls the `BROADCAST()` function to send the current estimate of $\boldsymbol{\pi}$ to each processor, and the `RECEIVE()` function to collect each processor's dual solution $\boldsymbol{\pi_n}$ obtained from solving ($ARDM_n$).

Note that for computational benefits, we repeatedly run Algorithm 1 where we start with loose tolerances and dynamically tighten them to achieve a desired accuracy, but omit this discussion here for ease of exposition as it does not affect the analysis, and provide more detail in Section 2.5.

2.4.2 Convergence

We now prove the convergence of CDWD and provide bounds on the optimality gap and feasibility violation. The quality of the dual solutions obtained by the consensus ADMM algorithm directly affects the quality of the recovered primal solution. We are able to reduce the optimality gap and feasibility violation by tweaking the primal and dual infeasibility tolerances ϵ_p and ϵ_d. Recall since we are solving the dual of (MP) using ADMM, we refer to the Lagrangian multipliers $\boldsymbol{\alpha_n}$ in the objective of ($ARDM_n$) and the multipliers λ_{ni} associated with the constraints as primal variables, and $\boldsymbol{\pi}, \boldsymbol{\pi_n}$ and u_n as dual variables; we refer to $\sqrt{\sum_{n=1}^{N} \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}} \right\|^2}$ as the dual feasibility violation and $\rho \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k} \right\|$ as the primal feasibility violation. Moreover, recall that z_{MP}^* and z_{DM}^* refer to the optimal objective values of the master problem (MP) and its dual, respectively; z_{RMP}^* and z_{RDM}^* refer to the optimal values of their restrictive counterparts; objective values and solutions resulting from the CDWD algorithm are denoted by a hat such as $\hat{z}_{RDM}$.

As shown in [42] and other sources in the literature, given tolerances $\epsilon, \epsilon_p, \epsilon_d > 0$, we can assume that ADMM terminates with $z_{RDM}^* - \hat{z}_{RDM} \leq \epsilon$, $\rho \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k} \right\| \leq \epsilon_p$ and $\sqrt{\sum_{n=1}^{N} \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}} \right\|^2} \leq \epsilon_d$. The following lemmas will be helpful in proving the error bounds.

Algorithm 1 CDWD Algorithm

```
1:  Input: tolerances $\epsilon_p, \epsilon_d \geq 0$, penalty parameter $\rho > 0$
2:  Let $I_n^1$ be the initial set of columns for each block $n$
3:  Initialize $\boldsymbol{\pi^1}, \boldsymbol{\alpha_n^1}$ for all $n$ and $\ell \leftarrow 0$
4:  while $I_n^{\ell+1} \neq I_n^{\ell}$ for some $n$ do
5:      $\ell \leftarrow \ell + 1$
6:      Initialize primal and dual residuals $r_p = \infty$ and $r_d = \infty$
7:      /*Solve RDM using consensus-based ADMM*/
8:      $k \leftarrow 0$
9:      BROADCAST($\boldsymbol{\pi^1}$)
10:     while $r_d > \epsilon_d$ and $r_p > \epsilon_p$ do
11:         $k \leftarrow k + 1$
12:         for each agent $n = 1, ..., N$ do
13:             Solve $(ARDM_n)$
14:             Collect optimal solutions $(\boldsymbol{\pi_n^{k+1}}, u_n^{k+1})$ and Lagrangian multipliers $\boldsymbol{\lambda_n^{k+1}}$
15:         end for
16:         RECEIVE($\{\boldsymbol{\pi_n^{k+1}}\}_{n=1}^N$)
17:         $\boldsymbol{\pi^{k+1}} \leftarrow \frac{1}{N}\sum_n(\boldsymbol{\pi_n^{k+1}}) + \frac{1}{N\rho}\sum_n \boldsymbol{\alpha_n^k}$
18:         BROADCAST($\boldsymbol{\pi^{k+1}}$)
19:         $\boldsymbol{\alpha_n^{k+1}} \leftarrow \boldsymbol{\alpha_n^k} - \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}})$, $\forall n = 1, \ldots, N$
20:         $r_d \leftarrow \sqrt{\sum_{n=1}^N \left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}\right\|^2}$
21:         $r_p \leftarrow \rho\left\|\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k}\right\|$
22:     end while

23:     /*Solve pricing subproblems*/
24:     for each agent $n = 1, ..., N$ do
25:         $z_{SEP}^n \leftarrow \min_{\boldsymbol{x_n}}\{\boldsymbol{c}_n^\top \boldsymbol{x_n} - \boldsymbol{\pi^{k+1}}^\top A_n \boldsymbol{x_n} - u_n^{k+1} : \boldsymbol{x_n} \in X_n\}$
26:         Let $\boldsymbol{x_n^i}$ be the optimal solution
27:         if $z_{SEP}^n < -\max_{i \in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d$ then
28:             Add extreme point $\boldsymbol{x_n^i}$: $I_n^{\ell+1} \leftarrow I_n^\ell \cup \{i\}$
29:         else
30:             $I_n^{\ell+1} \leftarrow I_n^\ell$
31:         end if
32:     end for
33:     $\boldsymbol{\pi^1} \leftarrow \boldsymbol{\pi^{k+1}}$
34:     $\boldsymbol{\alpha_n^1} \leftarrow \boldsymbol{\alpha_n^{k+1}}$, $\forall n = 1, \ldots, N$
35: end while

36: /*Each agent $n$ retrieves primal solution*/
37: $\hat{\boldsymbol{x}}_{\boldsymbol{n}} \leftarrow \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} \boldsymbol{x_n^i}$, $\forall n = 1, \ldots, N$
```

Lemma 2. *After the first iteration of CDWD, the Lagrangian multipliers $\boldsymbol{\alpha_n^k}$ associated with the copy constraints are primal feasible for all n, i.e for $k \geq 0$, we have $\sum_{n=1}^N \boldsymbol{\alpha_n^{k+1}} = 0$.*

Proof. From the updates, we have with $k \geq 0$:

$$\boldsymbol{\pi^{k+1}} = \frac{1}{N}\sum_{n=1}^N \boldsymbol{\pi_n^{k+1}} + \frac{1}{N\rho}\sum_{n=1}^N \boldsymbol{\alpha_n^k} \Rightarrow \sum_{n=1}^N \boldsymbol{\alpha_n^k} = \rho\left(N\boldsymbol{\pi^{k+1}} - \sum_{n=1}^N \boldsymbol{\pi_n^{k+1}}\right)$$

and

$$\boldsymbol{\alpha_n^{k+1}} = \boldsymbol{\alpha_n^k} - \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}),\ \forall n = 1, ..., N.$$

Summing over n, we get

$$\begin{aligned}\sum_{n=1}^N \boldsymbol{\alpha_n^{k+1}} &= \sum_{n=1}^N \boldsymbol{\alpha_n^k} - \rho\left(N\boldsymbol{\pi^{k+1}} - \sum_{n=1}^N \boldsymbol{\pi_n^{k+1}}\right) \\ &= \rho\left(N\boldsymbol{\pi^{k+1}} - \sum_{n=1}^N \boldsymbol{\pi_n^{k+1}}\right) - \rho\left(N\boldsymbol{\pi^{k+1}} - \sum_{n=1}^N \boldsymbol{\pi_n^{k+1}}\right) \\ &= 0.\end{aligned}$$

□

Theorem 2 establishes the feasibility violation at the recovered primal solution.

Theorem 2 (Feasibility Violation)**.** *Given a primal feasibility tolerance $\epsilon_p > 0$, CDWD terminates with a solution $\boldsymbol{\hat{x}_n} = \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} \boldsymbol{x_n^i}$ such that:*

$$\left\|\sum_{n=1}^N A_n \boldsymbol{\hat{x}_n} - t\right\| \leq N\epsilon_p$$

$$\sum_{i \in I_n^\ell} \lambda_{ni}^{k+1} = 1,\ \forall n$$

Proof. At outer iteration ℓ and iteration k of ADMM, let the Lagrangian functions of $(ARDM_n)$ for each n be:

$$Q_n(\boldsymbol{\pi_n}, \boldsymbol{\alpha_n^k}, \boldsymbol{\pi^k}, \boldsymbol{\lambda}) = \frac{1}{N}\boldsymbol{t}^\top \boldsymbol{\pi_n} + u_n + \boldsymbol{\alpha_n^k}^\top(\boldsymbol{\pi^k} - \boldsymbol{\pi_n}) - \frac{\rho}{2}\left\|\boldsymbol{\pi^k} - \boldsymbol{\pi_n}\right\|_2^2 + \sum_{i \in I_n^\ell} \lambda_{ni}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n} - u_n)$$

where $\{\lambda_{ni}\}_{i \in I_n^\ell}$ are the multipliers of the constraints in $(ARDM_n)$.

We have the following optimality conditions in $(ARDM_n)$:

$$\lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} - u_n^{k+1}) = 0,\ \forall i \in I_n^\ell \qquad \text{(Complementary Slackness)}$$

$$\lambda_{ni}^{k+1} \geq 0,\ \forall i \in I_n^\ell \qquad \text{(Dual Feasibility)}$$

$$\begin{aligned} \nabla_{\boldsymbol{\pi_n}} Q_n &= \frac{1}{N}\boldsymbol{t} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{\pi^k} - \boldsymbol{\pi_n^{k+1}}) - \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} = 0 \\ \nabla_{u_n} Q_n &= 1 - \sum_{i \in I_n^\ell} \lambda_{ni}^{k+1} = 0. \end{aligned} \qquad \text{(Stationarity)}$$

Thus, the convexity constraints $\sum_{i \in I_n^\ell} \lambda_{ni}^{k+1} = 1$ in the RMP are satisfied for all n and $\lambda_{ni}^{k+1} \geq 0$ for all n and i.

We rewrite the stationarity condition with respect to $\boldsymbol{\pi_n}$ as

$$\begin{aligned} \nabla_{\boldsymbol{\pi_n}} Q_n &= \frac{1}{N}\boldsymbol{t} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{\pi^k} - \boldsymbol{\pi_n^{k+1}}) - \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \\ &= \frac{1}{N}\boldsymbol{t} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{\pi^k} - \boldsymbol{\pi^{k+1}} + \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}) - \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \\ &= \frac{1}{N}\boldsymbol{t} - \boldsymbol{\alpha_n^k} + \rho(\boldsymbol{\pi^k} - \boldsymbol{\pi^{k+1}}) + \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}}) - \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \\ &= \frac{1}{N}\boldsymbol{t} - \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^k}) - \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} - \boldsymbol{\alpha_n^{k+1}} \end{aligned} \qquad (2.4.2)$$

where the last equality holds from $\boldsymbol{\alpha_n^{k+1}} = \boldsymbol{\alpha_n^k} - \rho(\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi_n^{k+1}})$.

Summing $\nabla_{\boldsymbol{\pi_n}} Q_n$ over $n = 1, ..., N$, we get

$$\begin{aligned}
\sum_{n=1}^{N} \nabla_{\boldsymbol{\pi_n}} Q_n &= \boldsymbol{t} - \rho \sum_{n=1}^{N} (\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^{k}}) - \sum_{n=1}^{N} \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} - \sum_{n=1}^{N} \boldsymbol{\alpha_n^{k+1}} \\
&= \boldsymbol{t} - \rho N (\boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^{k}}) - \sum_{n=1}^{N} \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \\
&= 0
\end{aligned}$$

where the second equality follows because $\sum_{n=1}^{N} \boldsymbol{\alpha_n^{k+1}} = 0$ from Lemma 2.

Then

$$\begin{aligned}
&\left\| \sum_{n=1}^{N} \nabla_{\boldsymbol{\pi_n}} Q_n \right\| = 0 \geq \left\| \boldsymbol{t} - \sum_{n=1}^{N} \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \right\| - N\rho \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^{k}} \right\| \\
\Rightarrow &\left\| \boldsymbol{t} - \sum_{n=1}^{N} \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni}^{k+1} \right\| \leq N\rho \left\| \boldsymbol{\pi^{k+1}} - \boldsymbol{\pi^{k}} \right\| \\
\Rightarrow &\left\| \boldsymbol{t} - \sum_{n=1}^{N} A_n \boldsymbol{\hat{x}_n} \right\| \leq N\epsilon_p
\end{aligned}$$

where $\boldsymbol{\hat{x}_n} = \sum_{i \in I_n^\ell} \lambda_{ni}^{k+1} \boldsymbol{x_n^i} = \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} \boldsymbol{x_n^i}$, since $I_n^{\ell+1} = I_n^\ell$ for all n when CDWD terminates. $\square$

Before deriving error bounds on the optimality gap, we first introduce bounds on $z_{DM}^* - \hat{z}_{RDM}$ and $\hat{z}_{RMP} - \hat{z}_{RDM}$. Adding these two will then give us bounds on $\hat{z}_{RMP} - z_{DM}^*$, or equivalently $\hat{z}_{RMP} - z_{MP}^*$. The following is a known relationship between $\hat{z}_{RDM}$, z_{DM}^* and z_{RDM}^* (cf. [40]).

Lemma 3. *After terminating ADMM, we have* $\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z_{SEP}^n\} \leq z_{DM}^* \leq z_{RDM}^*$.

Proof. If $z_{SEP}^n < 0$ for some n, then we can set $\hat{u}_n' = \hat{u}_n + z_{SEP}^n$. Doing so for each n, we get a feasible solution $(\hat{\boldsymbol{\pi}}, \{\hat{u}'_n\}_{n=1}^N)$ to (DM) with objective value

$$\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z_{SEP}^n\}$$

which is bounded above by z_{DM}^*. Moreover, $z_{DM}^* \leq z_{RDM}^*$ since RDM is a relaxation of (DM). □

Proposition 1. *Given that ADMM terminates with $z_{RDM}^* - \hat{z}_{RDM} \leq \epsilon$ and we terminate CDWD when $z_{SEP}^n \geq -\max_{i \in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d$ for all n, we have*

$$-\epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n \leq z_{DM}^* - \hat{z}_{RDM} \leq \epsilon$$

where L_n is a bound on the ℓ_2-norm of all extreme points of the set X_n, defining the local constraints of block n.

Proof. By Lemma 3, $\hat{z}_{RDM} + \sum_{n=1}^N \min\{0, z_{SEP}^n\} \leq z_{DM}^* \leq z_{RDM}^*$. Since $z_{RDM}^* - \hat{z}_{RDM} \leq \epsilon$ and $z_{SEP}^n \geq -\max_{i \in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d$ for all n after terminating ADMM, we have:

$$\begin{aligned}
&\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z_{SEP}^n\} \leq z_{DM}^* \leq z_{RDM}^* \\
\Rightarrow &\hat{z}_{RDM} - \hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z_{SEP}^n\} \leq z_{DM}^* - \hat{z}_{RDM} \leq z_{RDM}^* - \hat{z}_{RDM} \\
\Rightarrow &-\sum_{n=1}^{N} \max_{i \in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d \leq z_{DM}^* - \hat{z}_{RDM} \leq \epsilon \\
\Rightarrow &-\epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n \leq z_{DM}^* - \hat{z}_{RDM} \leq \epsilon
\end{aligned}$$

where the last set of inequalities holds from our assumption that $\left\|\boldsymbol{x_n^i}\right\| \leq L_n$ for all extreme points of block n: $\left\|\boldsymbol{A_n^i}\right\| = \left\|A_n \boldsymbol{x_n^i}\right\| \leq \|A_n\|_F L_n$. □

Since we assume that (MP) (and thus (P)) is feasible and has an optimal solution, the dual of (MP) is clearly bounded and has a bounded optimal solution. Let G be a bound on the absolute values of the components of $\boldsymbol{\pi}$. Note that such a bound may be computed in practice for problems with a special structure. For example, in the cutting stock problem with multiple stock lengths and vehicle routing problem with time windows, we can drop the convexity constraint (see [48]), in which case the dual of (MP) only involves variables $\boldsymbol{\pi}$. In both problems, the extreme points $\boldsymbol{x_n^i}$ are binary vectors, and vectors $\boldsymbol{t}$ and $\boldsymbol{A_n^i}$ are non-negative. Thus, the maximum component in the cost vector $\boldsymbol{c}$ is a valid bound. Note that G is not needed in practice, and is only needed for the purpose of the analysis.

Proposition 2. *Terminating ADMM with primal and dual feasibility tolerances ϵ_p and ϵ_d, respectively, we have at any outer iteration ℓ*

$$|\hat{z}_{RMP} - \hat{z}_{RDM}| \leq \epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n + mGN\epsilon_p$$

where m is the number of linking constraints, G is an upperbound on the absolute values of the components of π, and N is the number of blocks.

Proof. The complementary slackness conditions for RDM are

$$\lambda_{ni}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi} - u) = 0,\ \forall i \in I_n^\ell,\ n = 1, ..., N \tag{2.4.3}$$

$$\boldsymbol{\pi}^\top(\sum_{n=1}^{N} \sum_{i \in I_n^\ell} \boldsymbol{A_n^i} \lambda_{ni} - t) = 0,\ n = 1, ..., N \tag{2.4.4}$$

$$u_n(\sum_{i \in I_n^\ell} \lambda_{ni} - 1) = 0,\ n = 1, ..., N. \tag{2.4.5}$$

Note that $\lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} - u_n^{k+1}) = 0$ from the optimality conditions in $(ARDM_n)$.

Thus, plugging $\boldsymbol{\pi^{k+1}}, u_n^{k+1}, \lambda_{ni}^{k+1}$ into (2.4.3), we get for each n:

$$\begin{aligned}\left|\lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1})\right| &= \Big|\lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1}) \\ &\qquad - \lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi_n^{k+1}} - u_n^{k+1})\Big| \\ &= \left|\lambda_{ni}^{k+1}\boldsymbol{A_n^i}^\top(\boldsymbol{\pi_n^{k+1}} - \boldsymbol{\pi^{k+1}})\right| \\ &\le \left|\lambda_{ni}^{k+1}\right| \left\|\boldsymbol{A_n^i}\right\|\epsilon_d,\ \forall i \in I_n^\ell.\end{aligned}$$

Summing over $i \in I_n^\ell$, we get

$$\begin{aligned}\sum_{i\in I_n^\ell}\left|\lambda_{ni}^{k+1}(c_n^i - \boldsymbol{A_n^i}^\top \boldsymbol{\pi^{k+1}} - u_n^{k+1})\right| &\le \sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}\left\|\boldsymbol{A_n^i}\right\|\epsilon_d \\ &\le \max_{i\in I_n^\ell}\{\left\|\boldsymbol{A_n^i}\right\|\}\epsilon_d \\ &\le \|A_n\|_F L_n \epsilon_d.\end{aligned}$$

The first inequality follows because $\left|\lambda_{ni}^{k+1}\right| = \lambda_{ni}^{k+1}$ and the second inequality holds because $\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1} = 1$. Summing over n gives us

$$\begin{aligned}&\left|\sum_{n=1}^N\sum_{i\in I_n^\ell}c_n^i\lambda_{ni}^{k+1} - \sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(\boldsymbol{A_n^i}^\top\boldsymbol{\pi^{k+1}} + u_n^{k+1})\right| \le \sum_{n=1}^N\|A_n\|_F L_n\epsilon_d \\ \Rightarrow &\left|\hat{z}_{RMP} - \left[\sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(\boldsymbol{A_n^i}^\top\boldsymbol{\pi^{k+1}} + u_n^{k+1})\right]\right| \le \sum_{n=1}^N\|A_n\|_F L_n\epsilon_d. \qquad (2.4.6)\end{aligned}$$

Moreover, using Theorem 2, (2.4.4) becomes

$$\left|\boldsymbol{\pi^{k+1}}^\top(\sum_{n=1}^N\sum_{i\in I_n^\ell}\boldsymbol{A_n^i}\lambda_{ni}^{k+1} - \boldsymbol{t})\right| \le \left\|\boldsymbol{\pi^{k+1}}\right\|N\epsilon_p \le mGN\epsilon_p$$

Since $u_n^{k+1}(\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}-1)=0$ is satisfied for all n, we have

$$\begin{aligned}
&\left|\boldsymbol{\pi^{k+1}}^\top(\sum_{n=1}^{N}\sum_{i\in I_n^\ell}\boldsymbol{A_n^i}\lambda_{ni}^{k+1}-t)+\sum_{n=1}^{N}u_n^{k+1}(\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}-1)\right|\le mGN\epsilon_p\\
\Rightarrow&\left|\left[\sum_{n=1}^{N}\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(\boldsymbol{A_n^i}^\top\boldsymbol{\pi^{k+1}}+u_n^{k+1})\right]-\left[\boldsymbol{\pi^{k+1}}^\top\boldsymbol{t}+\sum_{n=1}^{N}u_n^{k+1}\right]\right|\le mGN\epsilon_p\\
\Rightarrow&\left|\left[\sum_{n=1}^{N}\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(\boldsymbol{A_n^i}^\top\boldsymbol{\pi^{k+1}}+u_n^{k+1})\right]-\hat{z}_{RDM}\right|\le mGN\epsilon_p. \qquad (2.4.7)
\end{aligned}$$

Adding (2.4.6) and (2.4.7) gives us

$$|\hat{z}_{RMP}-\hat{z}_{RDM}|\le\epsilon_d\sum_{n=1}^{N}\|A_n\|_F L_n+mGN\epsilon_p.$$

□

Theorem 3 (Optimality Gap)**.** *CDWD terminates with a solution $\hat{\boldsymbol{x}}$ such that:*

$$-\epsilon-\gamma\epsilon_d-mGN\epsilon_p\le\hat{z}_{RMP}-z^*_{MP}\le 2\gamma\epsilon_d+mGN\epsilon_p$$

where $\gamma=\sum_{n=1}^{N}\|A_n\|_F L_n$.

Proof. By Proposition 1,

$$-\epsilon\le\hat{z}_{RDM}-z^*_{DM}\le\epsilon_d\sum_{n=1}^{N}\|A_n\|_F L_n \qquad (2.4.8)$$

and by Proposition 2,

$$-\epsilon_d\sum_{n=1}^{N}\|A_n\|_F L_n-mGN\epsilon_p\le\hat{z}_{RMP}-\hat{z}_{RDM}\le\epsilon_d\sum_{n=1}^{N}\|A_n\|_F L_n+mGN\epsilon_p. \qquad (2.4.9)$$

Letting $\gamma = \sum_{n=1}^{N} \|A_n\|_F L_n$ and adding (2.4.8) and (2.4.9), we get

$$\begin{aligned} & -\epsilon - \gamma\epsilon_d - mGN\epsilon_p \leq \hat{z}_{RMP} - z^*_{DM} \leq 2\gamma\epsilon_d + mGN\epsilon_p \\ \Rightarrow & -\epsilon - \gamma\epsilon_d - mGN\epsilon_p \leq \hat{z}_{RMP} - z^*_{MP} \leq 2\gamma\epsilon_d + mGN\epsilon_p \end{aligned}$$

where the second line of inequalities follows from strong duality, i.e $z^*_{DM} = z^*_{MP}$. $\square$

2.5 Computational Experiments

In this section, we present preliminary computational experiments where we solve cutting stock instances from the literature and synthetic instances. CDWD and DWD are implemented in Python using a Message Passing Interface (MPI) package called mpi4py [76, 77, 78] and Gurobi is used as the backbone solver. In DWD, only the subproblems are parallelized. The quadratic programs resulting from the augmented Lagrangian ($ARDM_n$) are solved using the barrier method. The master problem in DWD and all subproblems are solved using Gurobi's concurrent method. The cutting stock instances were run on a 3.6 GHz Linux machine with 6 cores, 2 threads per core and 16 GB of RAM, and the synthetic instances were run on a 3.00 GHz Amazon server running on Linux, with 16 cores, 2 threads per core and 8 GB of RAM. We however limit ourselves to only using one thread per core to avoid the overhead of hyperthreading. An open-source implementation of the algorithm and data generation of the synthetic instances is available at https://github.com/mtonbari/ddw.

2.5.1 ADMM Parameters

To pick the penalty parameter ρ, we follow the guidelines provided in [42], where we dynamically adjust ρ according to the primal and dual residuals, so that they are a factor of μ away from each other, by either multiplying or dividing ρ by positive scalars τ^{inc} or τ^{dec}, respectively. At the end of iteration k, we update ρ as follows:

$$\rho^{k+1} \leftarrow \begin{cases} \tau^{inc}\rho^k & \text{if } \|r_d\| > \mu\|r_p\| \\ \frac{\rho^k}{\tau^{dec}} & \text{if } \|r_p\| > \mu\|r_d\| \\ \rho^k & \text{otherwise.} \end{cases}$$

Intuitively, increasing ρ would put more weight on the terms $\|\pi - \pi_n\|^2$, thus reducing dual feasibility violation, and decreasing ρ would put more weight on dual optimality, reducing primal feasibility violation.

We also dynamically adjust the tolerances, similar to [49]. We solve the pricing subproblems with increasingly accurate dual solutions, where we first solve CDWD with high ADMM tolerances, then divide the tolerances by 10. We repeat the process until we reach desired target tolerances. This significantly reduces computation time, as we reduce the number of times we solve ADMM to high accuracy. We note that the threshold used to add a new column depends on the dual tolerance ϵ_d (see Section 2.4.1 and Lemma 1). When starting with high dual tolerances at the beginning of the algorithm, the bound derived in Lemma 1 can be too loose, making the condition to add new columns too harsh. We found it computationally beneficial to be more lenient in adding columns by setting the threshold according to the target dual tolerance and not adapting the threshold according to the current dual tolerance used in solving CDWD.

In our experiments, we notice achieving desired dual tolerances to be harder than reducing feasibility violation. To this end, we pick $\tau^{dec} < \tau^{inc}$ to drive ρ up more easily (thus reducing dual feasibility violation). Moreover, feasibility violations tend to be low, so we pick slightly higher primal feasibility tolerances.

As in classical Dantzig-Wolfe Decomposition (DWD), the RMP might be infeasible if the starting set of extreme points is too small. We circumvent this by adding upper and lower bounds $-G_n \leq \boldsymbol{\pi_n} \leq G_n$ for each block n to ensure a bounded dual problem. As discussed in Section 2.4.2, it is possible to obtain such bounds for certain applications. Note that alternatively, one could apply CDWD to the auxiliary problem where slacks are added to the linking constraints, and the cost is replaced by the sum of the absolute values of the slacks. This would result in an initial set of columns such that the RMP is feasible. Thus, setting bounds G_n is not necessary but simplifies the implementation of CDWD. In our experiments, we set G_n to be a scalar multiple of cost vectors $\|\boldsymbol{c_n}\|$ for simplicity. For our DWD implementation, we first apply DWD to the aforementioned auxiliary problem to get an initial set of columns such that the RMP is feasible.

We report the optimality gap, computed as $\frac{|\hat{z}_{RMP}-z^*_{MP}|}{|z^*_{MP}|}$, where $\hat{z}_{RMP}$ is the objective value of the RMP evaluated at the recovered primal solution of CDWD and z^*_{MP} is the optimal objective value of the instance. We note that the cutting stock model can only be solved using a column generation algorithm, and many of the larger synthetic instances could not be solved by Gurobi as the process is killed due to the size of the problem. We thus use the objective value obtained from DWD for z^*_{MP}. To compute the feasibility violation, we compute the differences between the left and right-hand sides for each violated linking constraint, normalize by the right-hand side, and report the maximum value. We provide more details as we present the instances.

2.5.2 Cutting Stock Problem

In the cutting stock problem (CSP) with multiple stock lengths, we are given K types of rolls of different lengths L_k and costs c_k, and P pieces with demands d_p and lengths ℓ_p. Indexing the rolls of each type by $\{1, ..., N_k\}$, where N_k is an upperbound on the number of stocks of type k needed, the CSP with multiple stock lengths can be modeled as:

$$\begin{aligned}
\min_{x,y} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N_k} c_k y_{kn} \\
\text{s.t.} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N_k} x_{knp} \geq d_p, \quad p = 1, \dots, P, && \text{(2.5.1a)}\\
& \sum_{p=1}^{P} \ell_p x_{knp} \leq L_k y_{kn}, \; n = 1, \dots, N_k, \; k = 1, ..., K, && \text{(2.5.1b)}\\
& y_{kn} \in \{0, 1\}, \qquad \forall n, \, \forall k, && \text{(2.5.1c)}\\
& x_{knp} \in \mathbb{Z}_+, \qquad \forall p, \, \forall n, \, \forall k && \text{(2.5.1d)}
\end{aligned}$$

where y_{kn} is one if roll $n \in \{1, \dots, N_k\}$ of type k is used, and x_{knp} is the number of pieces of type p cut from roll $n \in \{1, \dots, N_k\}$ of type k. Constraints (2.5.1a) correspond to the linking constraints, ensuring demand satisfaction. For each $n \in \{1, \dots, N_k\}$ and $k \in \{1, \dots, K\}$, constraints (2.5.1b)-(2.5.1d) correspond to a block's constraints, ensuring the sum of the lengths of the pieces cut from a roll do not exceed the roll's length. There are then $\sum_{k=1}^{K} N_k$ blocks.

Reformulation

Given extreme points $\{x_{kn}^i, y_{kn}^i\}_{i \in I_{kn}}$ for each block (k, n), the master problem can be written as

$$\begin{aligned}
\min_{x,y} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{i \in I_{kn}} c_k \lambda_{kni} y_{kn}^i \\
\text{s.t.} \quad & \sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{i \in I_{kn}} \lambda_{kni} x_{knp}^i \geq d_p, \; p = 1, \dots, P, \\
& \sum_{i \in I_{kn}} \lambda_{kni} = 1, \qquad n = 1, \dots, N_k, \; k = 1, ..., K, \\
& \lambda_{kni} \geq 0 \qquad \forall i \in I_{kn}, \, \forall n, \, \forall k.
\end{aligned}$$

which is a relaxation of the original CSP. For a fixed stock type k, the resulting subproblems for all $n \in \{1, ..., N_k\}$ are the same and thus return the same column. As in [48], we aggregate variables such that $\lambda_{ki} = \sum_{n=1}^{N_k} \lambda_{kni}$ for all $i \in I_k$, so that we have one subproblem per stock type k. The resulting master problem is

$$\begin{aligned}
\min_{x,\, y} \quad & \sum_{k=1}^{K} \sum_{i \in I_k} c_k \lambda_{ki} y_k^i \\
\text{s.t.} \quad & \sum_{k=1}^{K} \sum_{i \in I_k} \lambda_{ki} x_{kp}^i \geq d_p, \; p = 1, \ldots, P, \\
& \sum_{i \in I_k} \lambda_{ki} = N_k, \qquad k = 1, ..., K, \\
& \lambda_{ki} \geq 0 \qquad \forall i \in I_k, \; \forall k,
\end{aligned} \tag{CSP}$$

and the resulting subproblem for block k is

$$\begin{aligned}
\min_{x} \quad & c_k - \sum_{p=1}^{P} \pi_p^* x_p \\
\text{s.t.} \quad & \sum_{p=1}^{P} \ell_p x_p \leq L_k, \\
& x_p \in \mathbb{Z}_+, \qquad p = 1, \ldots, P,
\end{aligned} \tag{CSP_k}$$

where $\boldsymbol{\pi}^*$ corresponds to the optimal dual variables associated with the linking constraints. To add a new column, the solution $y_k^i = 1$ and $x_{kp}^i = x_p^*$ is added as a new extreme point i, where $\boldsymbol{x}^*$ is the optimal solution of (CSP_k). The extreme points $\boldsymbol{x_k^i}$ are feasible cutting patterns of a stock of type k and variables $\boldsymbol{\lambda}$ are selecting patterns such that demands are satisfied in (CSP).

Finally, we note that since the solution $x_p = 0$ for all p is feasible in (CSP_k) and has cost zero, we can relax the convexity constraints to $\sum_{i \in I_k} \lambda_{ki} \leq N_k$ and since N_k is an upperbound on the number of stocks of type k needed, we can omit the convexity constraints in (CSP) (see [48] for more details). This omission is accounted for in (CSP_k).

Table 2.1: CSP Results: optimality and feasibility gaps of CDWD, and runtimes of CDWD and DWD for various number of roll types K and number of items P.

P Range	K	Optimality Gap	Feasibility Gap	CDWD Time - GM (sec)	DWD Time - GM (sec)	CDWD Time - Median (sec)	DWD Time - Median (sec)
$[38, 40]$	5	1.00e-02	2.46e-04	1.3	2.19	1.15	2.23
$[146, 150]$	4	1.00e-02	8.85e-03	12.63	17.53	11.39	19.06
$[194, 200]$	4	1.00e-02	8.90e-03	22.32	29.99	21.21	31.36
$[289, 299]$	4	9.60e-03	9.43e-03	64.52	58.23	58.66	60.84
$[385, 395]$	4	1.03e-02	9.38e-03	138.83	101.5	116.03	103.26

Computational Results

We solve CSP instances with multiple stock lengths obtained from the CaPaD library [79](http://www.math.tu-dresden.de/~capad/). We solve up to 50 instances with 4 and 5 stock lengths, and approximately 40, 150, 200, 300, and 400 items. For a set of instances with P items and K stock lengths, certain instances had fewer than K stock types. We ignore such instances resulting in fewer than 50 instances solved for certain combinations of P and K. The number of items also slightly vary around P throughout the instances. We solve instances with $P \approx 40$ and $K = 5$, with $P \approx 150$ and $K = 4$, with $P \approx 200$ and $K = 4$, with $P \approx 300$ and $K = 4$, and with $P \approx 400$ and $K = 4$. For the ADMM parameters, we pick $\mu = 50$, $\tau^{inc} = 2$, $\tau^{dec} = 1.5$ and $\rho^0 = 100$. For tolerances, we start at $\epsilon_p = 5$ and $\epsilon_d = 50$, and end at target tolerances $\epsilon_p = 5 \times 10^{-2}$ and $\epsilon_d = 5 \times 10^{-3}$.

In Table 2.1, we include the range of the number of items P across the instances, the number of stock lengths K, the geometric means (GM) of the optimality gaps, the feasibility violations and the runtimes, and the medians of the runtimes. Given a recovered solution $\hat{x}$, the feasibility violation is computed as

$$\max_p \left\{ \frac{d_p - \sum_{k=1}^K \hat{x}_{kp}}{d_p}, 0 \right\}.$$

CDWD recovers high quality solutions with low optimality gaps and low feasibility violations. The geometric mean of optimality gaps is consistently at about 10^{-2} and the geometric mean of feasibility violations is close to 10^{-4} for $P = 40$ and about 10^{-2} for the remaining sets of experiments. We note that CDWD runs faster than DWD in most instances in the first three sets of experiments, with the geometric means of CDWD's runtimes being about 40%, 28% and 26% lower than DWD's geometric means, and the medians of CDWD's runtimes being about 48%, 40%, 32% lower than DWD's medians, respectively. For instances with about 200 items, the geometric means are close, with DWD's being slightly lower, but CDWD runs faster in over 50% of the instances. With about 400 items, DWD runs faster in most instances, with a geometric mean that is about 27% lower and a median that is about 11% lower. As we increase the number of items, the difference between the runtimes decreases, where DWD eventually runs faster in most instances with P close to 400. We note that the number of linking constraints is equal to the number of items, thus increasing linearly as the CSP instances get larger. As the number of linking constraints gets larger, ADMM convergence starts to slow down, explaining our observations. To this end, we turn to synthetic instances to more easily perform sensitivity analysis with respect to the number of blocks, variables and linking constraints.

2.5.3 Synthetic Instances

Instance Generation

The synthetic instances are of the form

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} \boldsymbol{c_n}^\top \boldsymbol{x_n} \\ \text{s.t.} \quad & \sum_{n=1}^{N} A_n \boldsymbol{x_n} \geq \boldsymbol{t} \\ & B_n \boldsymbol{x_n} \leq \boldsymbol{b_n}, \ \forall n = 1, ..., N \\ & 0 \leq \boldsymbol{x_n} \leq \boldsymbol{u_n}, \ \forall n = 1, ..., N \end{aligned}$$

where the coefficients of the matrices A_n and B_n are from the discrete uniform distribution $\mathcal{U}\{-10, 20\}$, and the components of the cost vector are from $\mathcal{U}\{-10, 30\}$. Let ℓ_i be the sum of the entries in row i of the linking constraints, i.e $\ell_i = \sum_{n,j}(A_n)_{ij}$, where $(A_n)_{ij}$ is component (i, j) of A_n; similarly let β_i^n be the sum of the entries of row i of B_n. The vectors $\boldsymbol{t}$ and $\boldsymbol{b_n}$ were generated according to the sum of each row of the constraint matrix. We construct component i of $\boldsymbol{t}$ as follows:

$$\begin{cases} t_i \sim \mathcal{U}\{2\ell_i, 3\ell_i\}, & \text{if } \ell_i > 0 \\ t_i \sim \mathcal{U}\{3\ell_i, 2\ell_i\}, & \text{if } \ell_i < 0 \\ t_i = 0, & \text{if } \ell_i = 0 \end{cases} i = 1, ..., m$$

where m is the number of linking constraints. Similarly, component i of $\boldsymbol{b_n}$ is constructed as

$$\begin{cases} (b_n)_i \sim \mathcal{U}\{2\beta_i^n, 3\beta_i^n\} & \text{if } \beta_i^n > 0 \\ (b_n)_i \sim \mathcal{U}\{3\beta_i^n, 2\beta_i^n\}, & \text{if } \beta_i^n < 0 \\ (b_n)_i = 0, & \text{if } \beta_i^n = 0 \end{cases} i = 1, ..., m_n$$

where m_n is the number of constraints in block n. Moreover, to ensure a bounded region, we add upper and lower bounds to the variables, where $u_n = 30$ for all n.

Computational Results

We perform four sets of experiments, each set involving 1, 2, 5, and 10 linking constraints. For each set of experiment, we vary the number of blocks $N \in \{2, 4, 8, 10, 15\}$ and the total number of variables across all blocks $n_v \in \{100, 1000, 5000, 10000, 20000, 25000, 50000, 100000\}$. The results for $n_v >= 10000$ are reported in Tables 2.2-2.5. We define m_n to be the number of block constraints. For simplicity, each block has approximately the same number of variables. We limit the number of blocks to 15 to avoid the overhead of hyperthreading. We note that we have also solved the instances as a single LP using Gurobi but omit the results, as Gurobi's runtimes, although lower for the smaller instances, quickly become greater than CDWD and DWD as the instances get larger. Moreover, Gurobi failed to solve instances with $n_v \geq 50000$. The feasibility violations are computed as

$$\max_i \left\{ \frac{t_i - \sum_{n,j} (A_n)_{ij} (\hat{x}_n)_j}{|t_i|}, 0 \right\}.$$

For the ADMM parameters, we pick $\mu = 100$, $\tau^{inc} = 2$, $\tau^{dec} = 1.5$ and $\rho^0 = 100$. For tolerances, we start at $\epsilon_p = 50$ and $\epsilon_d = 50$, and end at target tolerances $\epsilon_p = 5 \times 10^{-2}$ and $\epsilon_d = 5 \times 10^{-3}$.

Across all instances, feasibility violations are very close to 0, with the largest violation being in the order of 10^{-4}; optimality gaps are also very low, with most instances hovering between the order of 10^{-6} and 10^{-2}, with only eight instances reaching optimality gaps as high as 10^{-2}.

For smaller instances, DWD runs faster than CDWD. For a fixed number of linking constraints and blocks, the gap between the runtimes closes as the number of variables increases, with CDWD eventually running faster in many instances with 25000 variables or more, especially for smaller values of m and N. For example, with five linking constraints and ten blocks, DWD is over an order of magnitude faster for $n_v = 100$ and $n_v = 1000$, and is about 50% faster for $n_v = 5000$; the runtimes are almost identical for larger values of n_v, until CDWD becomes faster at 100000 total variables. We observe this overall trend in many cases, but note that the gap between the runtimes levels out, and further experiments may be required to confirm it. We plot the ratios of DWD to CDWD runtimes in Figure 2.1 (a value under 1 means DWD is faster). The ratios approach one fast as we increase n_v to 25000, after which CDWD runs faster in more and more instances. Fixing n_v to 10000, 20000, 25000 and 50000, we observe that CDWD runs faster in 15%, 30%, 25%, and 50% of all instances, respectively. Between 50000 and 100000 variables, the ratios either stay close to one, slightly decrease in some cases or slightly increase in others, with 70% of the instances having a ratio greater than 0.95 when $n_v = 100000$. The ratio also approaches one at a slower rate as we increase N and, to some extent, the number of linking constraints. This is expected as increasing the number of blocks and linking constraints slows down the convergence of ADMM, thus requiring larger instances in terms of number of variables for CDWD to catch up to DWD and potentially run faster.

Although our main focus is dealing with privacy issues and data that cannot be stored in a central location, either for security reasons or physical limitations, we show that the computational price of handling these concerns approaches zero as the number of variables increases, and even see a potential computational benefit in many cases, while sacrificing very little in terms of solution quality.

Table 2.2: Optimality gaps and feasibility violations of CDWD, and runtimes of CDWD and DWD in synthetic instances with 1 linking constraint for N blocks, n_v total variables, and m_n block constraints.

N	n_v	m_n	Optimality Gap	Feasibility Gap	CDWD Time (sec)	DWD Time (sec)
2	10000	2500	2.89e-02	0	57.6	71.6
	20000	2500	1.07e-02	0	123.2	159.3
	25000	2500	5.86e-03	0	155.3	206.7
	50000	2500	9.87e-03	0	357.6	512.0
	100000	2500	2.41e-02	0	801.4	1132.9
4	10000	2500	1.16e-06	1.11e-07	37.6	34.8
	20000	2500	4.60e-06	0	82.3	82.9
	25000	2500	2.75e-02	0	82.9	104.1
	50000	2500	2.17e-02	0	173.7	230.2
	100000	2500	1.05e-05	0	531.2	516.9
8	10000	1250	2.89e-05	0	9.99	9.04
	20000	2500	1.57e-05	8.13e-09	50.1	50.0
	25000	2500	7.96e-06	0	69.8	65.1
	50000	2500	5.76e-06	0	148.7	144.2
	100000	2500	1.30e-05	1.92e-07	316.6	311.6
10	10000	1000	9.41e-06	3.32e-06	9.32	7.51
	20000	2000	4.75e-06	0	43.4	43.4
	25000	2500	1.48e-05	7.61e-08	75.3	76.9
	50000	2500	1.97e-05	0	164.6	162.4
	100000	2500	1.31e-05	0	370.6	356.0
15	10000	666	4.05e-06	0	3.96	3.02
	20000	1333	9.40e-06	0	19.4	17.0
	25000	1666	9.55e-06	5.28e-07	31.0	29.4
	50000	2500	9.55e-06	0	120.2	118.2
	100000	2500	5.08e-06	0	265.8	255.2

Table 2.3: Optimality gaps and feasibility violations of CDWD, and runtimes of CDWD and DWD in synthetic instances with 2 linking constraints for N blocks, n_v total variables, and m_n block constraints.

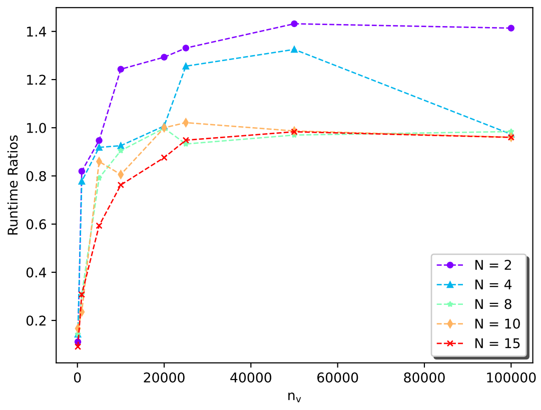
N	n_v	m_n	Optimality Gap	Feasibility Gap	CDWD Time (sec)	DWD Time (sec)
2	10000	2500	1.66e-04	0	89.5	90.6
	20000	2500	3.37e-02	0	105.1	219.8
	25000	2500	5.45e-06	5.07e-09	276.1	298.5
	50000	2500	3.69e-05	8.52e-09	690.2	758.5
	100000	2500	3.03e-02	0	841.6	1791.3
4	10000	2500	2.47e-05	0	47.0	45.2
	20000	2500	6.06e-06	0	102.4	78.9
	25000	2500	2.66e-04	1.33e-07	141.5	139.1
	50000	2500	4.16e-05	1.60e-07	317.0	325.0
	100000	2500	3.64e-05	6.39e-08	793.9	804.0
8	10000	1250	6.34e-06	9.05e-07	13.1	11.9
	20000	2500	2.30e-05	1.97e-07	74.1	67.7
	25000	2500	6.61e-06	3.54e-07	100.2	94.9
	50000	2500	2.32e-05	0	205.6	218.3
	100000	2500	1.22e-05	6.43e-08	447.8	447.7
10	10000	1000	1.37e-05	1.44e-06	9.85	8.37
	20000	2000	9.50e-06	0	57.6	57.5
	25000	2500	2.73e-05	6.04e-07	102.1	99.7
	50000	2500	4.82e-06	0	251.2	223.9
	100000	2500	5.69e-06	0	528.5	510.2
15	10000	666	1.38e-05	0	4.36	3.77
	20000	1333	1.47e-05	1.16e-06	24.5	22.4
	25000	1666	8.19e-06	1.05e-06	40.4	40.0
	50000	2500	4.75e-06	0	141.6	143.7
	100000	2500	3.31e-06	4.27e-08	411.9	347.0

Table 2.4: Optimality gaps and feasibility violations of CDWD, and runtimes of CDWD and DWD in synthetic instances with 5 linking constraints for N blocks, n_v total variables, and m_n block constraints.

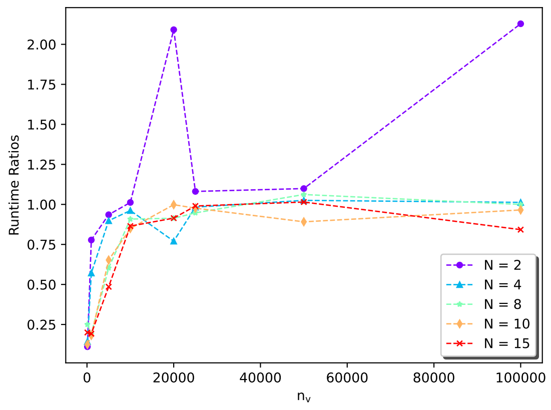
N	n_v	m_n	Optimality Gap	Feasibility Gap	CDWD Time (sec)	DWD Time (sec)
2	10000	2500	1.69e-05	1.36e-08	238.1	232.1
	20000	2500	3.46e-05	2.58e-07	572.6	566.3
	25000	2500	1.97e-05	7.51e-08	766.0	809.1
	50000	2500	5.35e-05	6.67e-08	2040.1	1862.4
	100000	2500	3.45e-05	7.14e-09	3202.5	3323.3
4	10000	2500	3.23e-05	6.96e-07	100.4	100.6
	20000	2500	4.65e-06	4.45e-07	213.6	244.9
	25000	2500	3.08e-05	2.61e-07	331.3	289.5
	50000	2500	2.59e-05	4.69e-08	876.7	929.3
	100000	2500	1.52e-05	9.98e-11	2564.8	2434.8
8	10000	1250	5.31e-05	2.46e-06	28.2	24.2
	20000	2500	1.40e-05	8.65e-08	127.9	151.0
	25000	2500	3.43e-05	6.57e-08	193.8	178.4
	50000	2500	9.62e-06	1.55e-08	416.0	404.2
	100000	2500	1.20e-05	7.08e-08	1029.3	1018.3
10	10000	1000	4.91e-06	1.05e-07	21.4	19.4
	20000	2000	1.38e-05	1.49e-06	121.2	108.6
	25000	2500	1.59e-05	4.81e-07	222.7	217.3
	50000	2500	7.98e-06	1.14e-07	430.4	416.6
	100000	2500	4.02e-05	0	948.9	978.0
15	10000	666	2.08e-05	1.34e-06	11.1	6.88
	20000	1333	1.00e-05	1.53e-06	42.8	30.5
	25000	1666	9.55e-06	5.05e-07	85.5	74.5
	50000	2500	7.55e-06	2.49e-07	300.6	333.9
	100000	2500	7.63e-06	7.41e-08	813.5	820.9

Table 2.5: Optimality gaps and feasibility violations of CDWD, and runtimes of CDWD and DWD in synthetic instances with 10 linking constraints for N blocks, n_v total variables, and m_n block constraints.

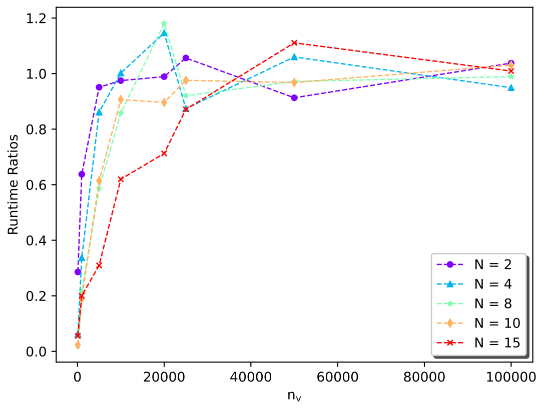
N	n_v	m_n	Optimality Gap	Feasibility Gap	CDWD Time (sec)	DWD Time (sec)
2	10000	2500	4.23e-05	3.54e-07	568.5	482.1
	20000	2500	2.11e-05	6.73e-08	1339.1	1332.4
	25000	2500	2.05e-05	2.14e-07	2334.7	1895.1
	50000	2500	2.74e-05	8.39e-09	5412.5	5122.9
	100000	2500	4.91e-05	5.20e-09	15681.6	14124.8
4	10000	2500	3.45e-05	8.81e-07	359.1	348.8
	20000	2500	1.60e-05	2.18e-07	631.4	664.2
	25000	2500	9.95e-06	8.11e-08	879.6	856.4
	50000	2500	1.98e-05	1.50e-07	1820.5	2118.9
	100000	2500	1.61e-05	5.15e-08	5044.7	4836.2
8	10000	1250	1.38e-05	1.16e-06	50.2	46.7
	20000	2500	1.67e-05	4.49e-07	329.8	302.4
	25000	2500	3.14e-05	2.60e-07	372.9	353.1
	50000	2500	1.20e-05	3.54e-08	867.2	913.8
	100000	2500	9.61e-06	1.11e-07	2659.3	2149.4
10	10000	1000	1.61e-05	1.49e-06	44.1	28.9
	20000	2000	2.27e-05	6.12e-07	209.1	175.7
	25000	2500	1.19e-05	3.32e-07	412.5	381.5
	50000	2500	6.69e-06	2.76e-07	845.3	795.6
	100000	2500	7.34e-06	4.53e-08	2601.6	2215.6
15	10000	666	1.78e-05	9.22e-07	22.3	10.2
	20000	1333	2.12e-05	1.27e-06	99.4	73.3
	25000	1666	1.63e-05	3.63e-07	163.9	128.4
	50000	2500	9.94e-06	7.10e-07	559.7	532.5
	100000	2500	7.16e-06	5.97e-08	1176.1	1088.9



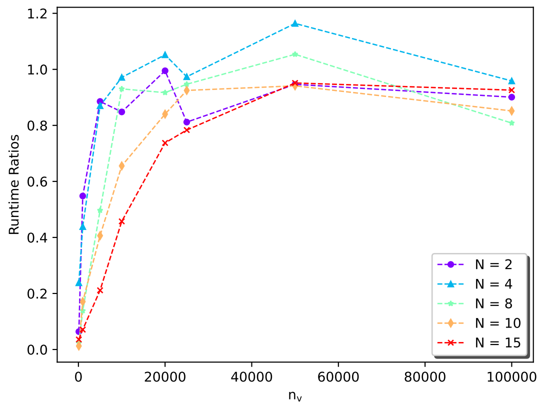
(a) $m = 1$



(b) $m = 2$



(c) $m = 5$



(d) $m = 10$

Figure 2.1: DWD to CDWD runtime ratios in synethetic instances for various number of blocks N and linking constraints m.

2.5.4 Parallel Efficiency and Scalability

To measure how our Python implementation scales as we increase the number of blocks and available cores, we use two common metrics as in [80]. The first one measures the speedup gained by using the available cores. The second metric measures core utilization and time lost in communication and synchronization. We compute the two metrics for instances with 9000, 18000 and 36000 total variables. For each set of instances, we experiment with 5, 10, 20, 36 and 72 blocks. As before, each block contains the same number of variables. Note that in these experiments, we potentially pay the price of hyperthreading. The main objective is to showcase the slowdowns that can be caused by idle cores, indicating poten-

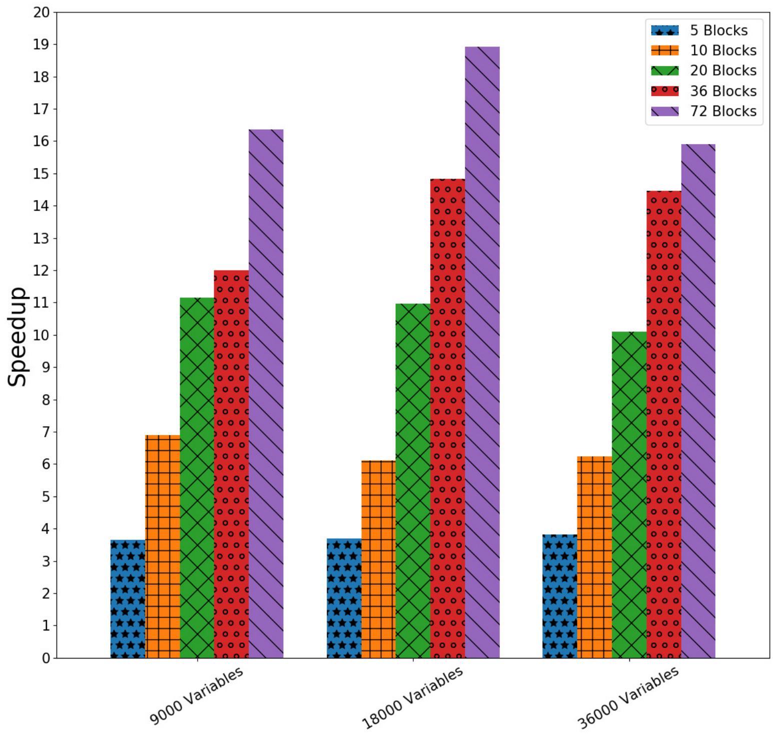


Figure 2.2: Ratio of runtimes between serial and parallel implementations

tial benefits in an asynchronous implementation of our algorithm.

Parallel Speedup Let t_p be the time it takes for CDWD to terminate using p cores. We compute the ratio $\frac{t_p}{t_1}$ for each experiment and report results in Figure 2.2. We observe similar trends for different number of total variables. The computational gain from parallelizing decreases as we increase the number of blocks. This is mainly due to cores sitting idle, waiting on other processes to finish, as well as communication overhead increasing with the number of cores used. This is confirmed by our analysis on core utilization.

Core Utilization To estimate core utilization, we measure total time spent doing useful computations, communication time, and synchronization time where a core is sitting idle waiting on others to finish their computations. For each core, if we define these three values as T_u, T_c and T_s, respectively, then core utilization can be estimated as $\frac{T_u}{T_u+T_c+T_s}$ [80].

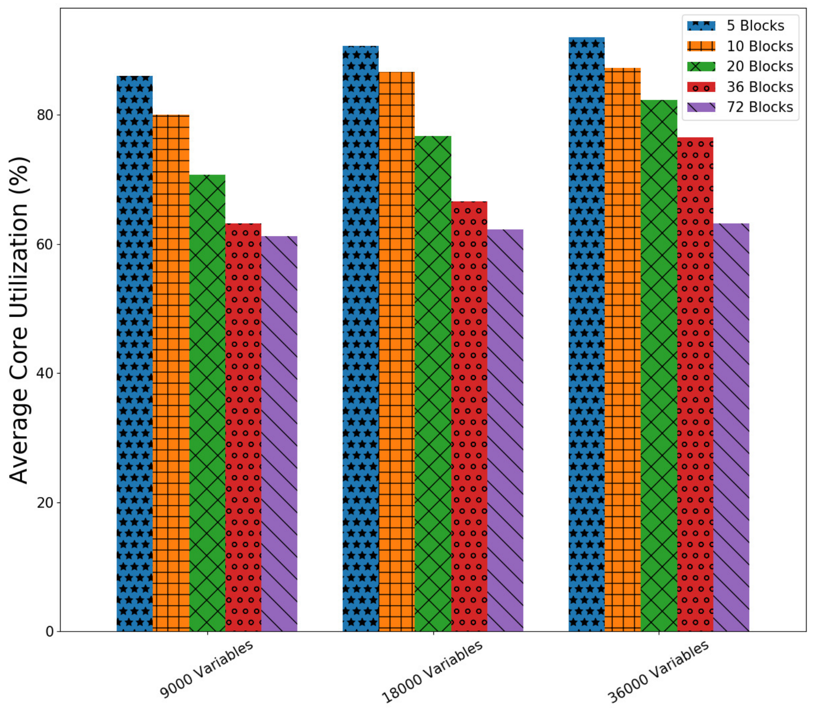


Figure 2.3: Average core utilization

Figure 2.3 reports average core utilization for each instance. We again see diminishing returns where average utilization decreases as the number of blocks and cores used increases. However, it seems that the average utilization is slightly better as we increase the number of total variables.

2.6 Conclusions

In this chapter, we proposed a consensus-based Dantzig-Wolfe decomposition algorithm for loosely coupled large-scale linear programs. As opposed to the standard Dantzig-Wolfe algorithm, we solve the master problem using consensus-based ADMM in a distributed fashion, thus handling circumstances where data is not available centrally and preserving privacy of information of the blocks, and addressed the resulting computational and theoretical challenges. We proved convergence of the algorithm and provided error bounds on the feasibility and optimality gaps. We illustrated our method using an MPI implementation on cutting stock and synthetic instances, and showed that we are able to achieve high accuracy in reasonable time. Although the main objective of our method is to handle decentralized storage of data and privacy concerns, we illustrated potential computational benefits for instances with a large number of variables and in many of the cutting stock instances. To further improve computation time, it is possible to use other algorithms or more sophisticated versions of ADMM to solve the consensus problem. As the difficulty and size of the problems for each block increases, the cost per ADMM iteration can become prohibitive. Certain workarounds involve linearizing the objective of the augmented Lagrangian, yielding computational benefits [42]. Other interesting consensus algorithms include a distributed interior point method which might converge faster than first-order distributed methods [81]. Finally, as suggested by our experiments, an asynchronous implementation of CDWD has the potential to improve computation times.

CHAPTER 3
DATA-DRIVEN TWO-STAGE CONIC OPTIMIZATION WITH RARE HIGH-IMPACT ZERO-ONE UNCERTAINTIES

3.1 Motivation

In this chapter, we are motivated by problems arising in network optimization affected by zero-one uncertainties. In such applications, binary random variables represent failures of network components such as edges or nodes. We are interested in the case where failure events are rare but have a high impact. In particular, we are motivated by applications in which the decision-relevant random events consist of high-dimensional binary outcomes. For example, in electric power networks, random node and edge failures have been used to model losses of physical components such as substations, transmission lines, generators, and transformers [82, 83]. Similarly, random zero-one vectors can represent failures of compressors and gas pipelines in natural gas [84], antennas in wireless communication [85], and road links in transportation networks [86].

The challenges in modeling and solving such uncertainty-affected optimization problems are threefold. First, the number of scenarios grows exponentially as the number of network components increases, up to 2^n scenarios for n components. Second, we are focused on applications where network failures are rare but critical, and historical records are often not rich enough to include observations for every possible failure state. Third, failures of individual network elements are unlikely to be independent of each other. For example, transmission line failures in electric power systems often have a cascading effect which triggers the failure of other transmission lines. As a result, the true underlying distribution of the random parameters is often unknown and difficult to reliably estimate given limited data.

In this high-dimensional context, we focus on two-stage stochastic programming. Let $\tilde{\boldsymbol{\xi}} \in \Xi$ be a random zero-one vector of dimension M, where $\Xi \subseteq \{0,1\}^M$ is the support set. Recall that if the distribution $\mathbb{P}$ is known, then the two-stage problem takes the form

$$\min_{\boldsymbol{x} \in X} \; c(\boldsymbol{x}) + \mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{x}, \tilde{\boldsymbol{\xi}})\right],$$

where $\boldsymbol{x}$ represents the first-stage decisions that must be made before the realization of the random parameters, $X \subseteq \mathbb{R}^{N_1}$ is a convex compact set of feasible first-stage decisions, $c : X \mapsto \mathbb{R}$ is a convex function representing the deterministic cost associated with $\boldsymbol{x}$, and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is the optimal value of the second-stage problem given first-stage decision vector $\boldsymbol{x}$ and a fixed realization $\boldsymbol{\xi} \in \Xi$. We assume that the second-stage problem is a convex conic problem which can be written as

$$\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}) = \inf_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \boldsymbol{q}(\boldsymbol{\xi})^\top \boldsymbol{y} : \boldsymbol{W}(\boldsymbol{\xi}) \boldsymbol{y} \geq \boldsymbol{T}(\boldsymbol{x}) \boldsymbol{\xi} + \boldsymbol{h}(\boldsymbol{x}) \right\}, \tag{3.1.1}$$

where $\boldsymbol{y}$ denotes the second-stage decisions made after observing $\boldsymbol{\xi}$; $\mathcal{Y} \subseteq \mathbb{R}^{N_2}$ is a *proper* (closed, convex, pointed, and full-dimensional) cone; $\boldsymbol{q} : \Xi \mapsto \mathbb{R}^{N_2}$ and $\boldsymbol{W} : \Xi \mapsto \mathbb{R}^{L \times N_2}$ are vector- and matrix-valued affine functions respectively, and $\boldsymbol{h} : X \mapsto \mathbb{R}^L$ and $\boldsymbol{T} : X \mapsto \mathbb{R}^{L \times M}$ are componentwise closed, proper, convex vector- and matrix-valued functions, respectively. We allow uncertainty to affect only the affine constraints of the problem and, similarly, the first- and second-stage decisions to interact only via the affine part.

Since the true underlying distribution $\mathbb{P}$ is unknown, this two-stage optimization formulation is ill-posed. Nevertheless, $\mathbb{P}$ is typically observable through a finite amount of historical data. We assume that we have access to N such independent and identically distributed observations, which we denote by $\{\hat{\boldsymbol{\xi}}^1, \ldots, \hat{\boldsymbol{\xi}}^N\}$. We also assume that generating additional data (e.g., via Monte Carlo computer simulations) is either costly or impossible.

A popular approach to approximate $\mathbb{E}\left[\mathbb{Q}(\boldsymbol{x}, \boldsymbol{\xi})\right]$ is the Sample Average Approximation (SAA) [87]. This approach replaces the true distribution with the empirical distribution

$\mathbb{P}_N = \frac{1}{N}\sum_{i=1}^{N}\delta_{\hat{\boldsymbol{\xi}}^i}$, where $\delta_{\hat{\boldsymbol{\xi}}^i}$ denotes the Dirac distribution at $\hat{\boldsymbol{\xi}}^i$. In the context of rare events, however, obtaining accurate estimates of the true distribution and, hence, the optimal solution of the true two-stage problem may require unrealistically large amounts of data. Indeed, given limited data, SAA is known to lead to an optimistic solution which generalizes poorly to unobserved data, leading to poor out-of-sample performance [18].

3.2 Distributionally Robust Approach for Discrete Rare Events

The high dimensionality and rare occurrence of failure events make estimating the underlying failure probabilities difficult. To this end, we adopt a distributionally robust approach and construct an ambiguity set $\mathcal{P}$ of possible distributions that are consistent with the observed data. We then minimize the worst-case expected costs over all distributions in the ambiguity set. Specifically, we consider two-stage conic distributionally robust optimization problems of the form

$$\min_{\boldsymbol{x}\in X}\ c(\boldsymbol{x}) + \sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{x},\tilde{\boldsymbol{\xi}})\right]. \tag{3.2.1}$$

The ambiguity set $\mathcal{P}$ must be chosen such that it contains the true distribution with high confidence or, at the very least, distributions that assign nonzero probability to the rare events. We focus on the Wasserstein ambiguity set, i.e. $\mathcal{P} = \mathcal{B}_W(\mathbb{P}_N, \theta)$. Recall that the Wasserstein set is a ball in the space of probability distributions centered at the empirical one $\mathbb{P}_N$ and is defined as

$$\mathcal{B}_W(\mathbb{P}_N,\theta) = \left\{\mathbb{P}\in\mathcal{M}(\Xi) : \mathrm{d}_W(\mathbb{P},\hat{\mathbb{P}}_N)\le\theta\right\}, \tag{3.2.2}$$

where $\mathcal{M}(\Xi)$ denotes the set of all distributions supported on Ξ, $\theta \geq 0$ is the radius of the Wasserstein ball. Given a valid metric $\mathrm{d}(\cdot,\cdot)$ on the support set Ξ, the Wasserstein distance

$\mathrm{d}_W(\mathbb{P}, \mathbb{P}')$ between two distributions $\mathbb{P}, \mathbb{P}'$ is

$$\mathrm{d}_W(\mathbb{P}, \mathbb{P}') = \min_{\Pi \in \mathcal{M}(\Xi \times \Xi)} \left\{ \sum_{\boldsymbol{\xi} \in \Xi} \sum_{\boldsymbol{\xi}' \in \Xi} \mathrm{d}(\boldsymbol{\xi}, \boldsymbol{\xi}') \Pi(\boldsymbol{\xi}, \boldsymbol{\xi}') : \ \Pi \text{ is a coupling of } \mathbb{P} \text{ and } \mathbb{P}' \right\}.$$

We discuss the choice of the metric $\mathrm{d}(\cdot, \cdot)$ and the radius θ of the Wasserstein ball in Section 3.2.1.

3.2.1 Choice of the Underlying Metric and Radius of the Wasserstein Ball

In the definition of the Wasserstein ambiguity set, the choice of the underlying metric $\mathrm{d}(\cdot, \cdot)$ of the support set can play an important role from both a modeling and algorithmic perspective. Although we take advantage of this observation in Chapter 4, we assume throughout this chapter that the metric $\mathrm{d}(\cdot, \cdot)$ is an arbitrary norm. We note, however, that our results also apply when $\mathrm{d}(\cdot, \cdot)$ is any mixed-integer conic-programming-representable metric.

For a given choice of the underlying metric, the radius θ of the ambiguity set allows us to control the level of risk-aversness. Specifically, given a confidence level $\beta \in (0, 1)$, one can choose the radius as a function of β and the number of observations N such that the true distribution $\mathbb{P}$ is contained in the ambiguity set with high probability:

$$\mathbb{P}\left[\mathrm{d}_W(\mathbb{P}, \mathbb{P}_N) \leq \theta_N(\beta)\right] \geq 1 - \beta. \tag{3.2.3}$$

It was shown in [18] that (3.2.3) holds if we select $\theta_N(\beta) = c_0 \left(N^{-1} \log \beta^{-1}\right)^{1/M}$, where c_0 is a problem-dependent constant. Since this choice can lead to unnecessarily large values for the radius, the authors suggest solving the two-stage problem (3.2.1) for several fixed choices of θ and then using k-fold cross-validation to select a radius. In this technique, the data set is split into k folds, where $k - 1$ of the folds are used as the training set, and the remaining one as the test set. This process is repeated k times and the average of the radii obtained from each of the k runs is used as the final radius (see [18]). However, this approach is not expected to work well when addressing rare event uncertainties as the training and testing data sets used for cross-validation are likely to be unbalanced in such cases (e.g., the testing data set may not contain any rare events leading to a trivial value of $\theta = 0$). Although one can circumvent this issue by using techniques such as stratified cross validation, doing so would require solving the two-stage problem (3.2.1) repeatedly for various choices of θ, which can become computationally expensive. Instead, we use the following theorem to guide our choice of the radius θ which provides tighter values by exploiting the finiteness of the support Ξ.

Theorem 4 (Finite sample guarantee)**.** *For every fixed sample size $N > 0$, and confidence level $\beta \in (0, 1)$, the probabilistic guarantee* (3.2.3) *holds whenever*

$$\theta_N(\beta) \geq D\sqrt{(2N)^{-1}\left(|\Xi| \log(N + 1) + \log \beta^{-1}\right)}, \tag{3.2.4}$$

where $D := \max_{\boldsymbol{\xi},\boldsymbol{\xi}' \in \Xi} \mathrm{d}(\boldsymbol{\xi}, \boldsymbol{\xi}')$ is the diameter of Ξ with respect to the metric d.

Proof. We refer the reader to [2] for the proof. $\square$

The right-hand side of (3.2.4) indicates that for a choice of a confidence level β, $\theta_N(\beta)$ is roughly proportional to $\sqrt{N^{-1}\log(N + 1)}$. For such choices, the optimal value of the corresponding two-stage distributionally robust problem (3.2.1) can be expected to provide an upper bound on the true (unknown) out-of-sample cost. We empirically verify this upper bound in Section 3.6, where we vary $\theta = \nu\sqrt{N^{-1}\log(N + 1)}$ as a function of a scalar ν. Theorem 4 also indicates that $\theta_N(\beta) \to 0$ as the sample size becomes large ($N \to \infty$), and that $\theta_N(\beta) \to \infty$ if we are overly conservative ($\beta \to 0$). This is in line with our statement that DRO under the Wasserstein ambiguity set is a generalization of stochastic and robust optimization.

Remark 1 (Reduction to two-stage stochastic and robust optimization)**.** *The two-stage distributionally robust problem* (3.2.1) *reduces to the classical sample average approximation whenever the radius of the ambiguity set $\theta = 0$, since $\mathcal{B}_W(\mathbb{P}_N, \theta) = \{\hat{\mathbb{P}}_N\}$ reduces to a* *singleton in this case. Similarly, it reduces to a classical two-stage robust optimization problem whenever* $\theta \geq \max_{\boldsymbol{\xi},\boldsymbol{\xi}' \in \Xi} d(\boldsymbol{\xi},\boldsymbol{\xi}')$ *since* Ξ *is compact and* $\mathcal{B}_W(\mathbb{P}_N,\theta)$ *contains all Dirac distributions* $\delta_{\boldsymbol{\xi}}$, $\boldsymbol{\xi} \in \Xi$, *in this case. Therefore, the worst-case expectation in* (3.2.1) *reduces to* $\max_{\boldsymbol{\xi} \in \Xi} \mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$.

3.2.2 Contributions

This chapter addresses the relatively unexplored topic of rare high-impact uncertainties through the lens of data-driven distributionally robust optimization. Existing methods for addressing rare high-impact uncertainties [88, 89] are few, and they are all based on variants of Monte Carlo methods (e.g., importance sampling), which require the existence of a probability distribution that can be sampled to generate additional observations.

In this chapter, we extend the state of the art in data-driven optimization by studying two-stage conic programs with a particular focus on high-dimensional zero-one uncertainties. This is crucial because existing reformulations and algorithms for distributionally robust optimization with finitely supported distributions [90, 19, 27, 26] scale with the size of the support set $|\Xi|$, which can grow exponentially large in such cases. We circumvent this exponential growth by utilizing tractable conservative approximations inspired by lift-and-project convexification techniques in global optimization [91, 92, 93].

Closest in spirit to our work are the papers of [94, 95, 96] who consider the case where the second-stage value function $\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$ is the optimal value of a linear program with uncertain right-hand sides and the support set Ξ is a polytope. In this setting, the authors in [94, 95] reformulate (3.2.1) as a copositive cone program which they approximate using semidefinite programming, whereas [96] provide approximations by leveraging reformulation-linearization techniques from bilinear programming. Although some extensions of these approaches to the case of zero-one support sets Ξ have been made [97, 98], problems where $\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$ is the optimal value of a conic program have not been addressed. This is partly because such extensions lead to so-called generalized copositive programs or set-semidefinite programs (e.g., see [99, 100]), and relatively little is known about their tractable approximations. In contrast, generalizations of lift-and-project techniques to mixed zero-one conic problems are fairly well known (e.g., see [101, 102]), and we exploit these to derive tractable approximations for distributionally robust optimization. The relationship between convexification hierarchies based on copositive programming and lift-and-project techniques (for specific problem classes) has been explored in [103, 104].

We highlight the following main contributions:

1. By exploiting ideas from penalty methods and bilinear programming, we develop reformulations of the two-stage distributional robust problem (3.2.1) under the Wasserstein set, reducing its solution to optimization problems over the convex hulls of mixed-integer conic representable sets.

2. We prove the existence of a finite penalty parameter such that the two-stage distributionally robust problem (3.2.1) with right-hand side uncertainty is equivalent to a two-stage problem with objective uncertainty using a penalty method reformulation. This significantly reduces the size of the resulting reformulations compared to using McCormick inequalities.

3. By using lift-and-project hierarchies to approximate the convex hull of the mixed-integer conic representable sets, we derive tractable conservative approximations of the distributionally robust two-stage problem (3.2.1), and we provide practical guidelines to compute them efficiently. The approximations are tractable, and they become exact as the Wasserstein radius θ shrinks to zero.

4. We demonstrate the practical viability of our method and its out-of-sample performance on challenging nonlinear optimal power flow and multi-commodity network design problems that are affected by rare network contingencies, and we study its behavior as a function of the rarity and impact of these contingencies, illustrating

improvements over classical sample average and two-stage robust optimization formulations.

The rest of the chapter is organized as follows. Section 3.3 derives the mixed-integer conic programming representation of interest, Section 3.4 derives their lift-and-project approximations, and Section 3.6 reports numerical results.

Notation. Vectors and matrices are printed in bold lowercase and bold uppercase letters, respectively, while scalars are printed in regular font. The set of non-negative integers and reals are denoted by $\mathbb{Z}_+$ and $\mathbb{R}_+$, respectively. For any positive integer N, we define $[N]$ as the index set $\{1, \ldots, N\}$. We use $\mathbf{e}_k$ to denote the k^{th} unit basis vector, $\mathbf{e}$ to denote the vector of ones, $\mathbf{I}$ to denote the identity matrix, and $\mathbf{0}$ to denote the vector or matrix of zeros, respectively; their dimensions will be clear from the context. For a matrix $\boldsymbol{A}$, we use $\text{vec}(\boldsymbol{A})$ to denote the vector obtained by stacking the columns of $\boldsymbol{A}$ in order. The inner product between two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$ is denoted by $\langle \boldsymbol{A}, \boldsymbol{B} \rangle := \sum_{i\in[m]} \sum_{j\in[n]} A_{ij}B_{ij}$. We use $\mathcal{C}^n = \{(\boldsymbol{x}, t) \in \mathbb{R}^{n-1} \times \mathbb{R} : \|\boldsymbol{x}\| \leq t\}$ to denote the norm cone associated with the norm $\|\cdot\|$. For a logical expression $\mathcal{E}$, we define $\mathbb{I}[\mathcal{E}]$ as the indicator function which takes a value of 1 if $\mathcal{E}$ is true and 0 otherwise. Throughout the chapter, we refer to an optimization problem as tractable if it can be solved in polynomial time in the size of its input data, and intractable if it is NP-hard.

3.3 Mixed-Integer Conic Representations

We assume that the two-stage distributionally robust problem (3.2.1) satisfies the assumptions of complete and sufficiently expensive recourse.

(A1) For every realization $\boldsymbol{\xi} \in \Xi$, there exists $\boldsymbol{y}^+ \in \text{int}(\mathcal{Y})$ such that $\boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}^+ > 0$.

(A2) For every first-stage decision $\boldsymbol{x} \in X$ and every realization $\boldsymbol{\xi} \in \Xi$, the second-stage loss function $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is bounded.

A natural way to ensure these assumptions hold is to add slack variables in the formulation of the second-stage problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ and penalize them in the objective function. Whenever the assumptions are satisfied, they imply that *(i)* $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is always strictly feasible and bounded, *(ii)* the dual of $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$, given in the following, is always feasible, and *(iii)* strong conic duality holds between the second-stage problem and its dual, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}) = \mathcal{Q}_d(\boldsymbol{x}, \boldsymbol{\xi})$, where

$$\mathcal{Q}_d(\boldsymbol{x}, \boldsymbol{\xi}) := \sup_{\boldsymbol{\lambda} \in \mathbb{R}^L_+} \left\{ \left[\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{h}(\boldsymbol{x})\right]^\top \boldsymbol{\lambda} : \boldsymbol{q}(\boldsymbol{\xi}) - \boldsymbol{W}(\boldsymbol{\xi})^\top \boldsymbol{\lambda} \in \mathcal{Y}^* \right\}. \tag{3.3.1}$$

Here, $\mathcal{Y}^*$ denotes the dual cone of $\mathcal{Y}$. We assume that the uncertain vectors and matrices in (3.1.1) are affine and can be represented as $\boldsymbol{q}(\boldsymbol{\xi}) = \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi}$ and $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}_0 + \sum_{j \in [M]} \xi_j \boldsymbol{W}_j$, where $\boldsymbol{Q} \in \mathbb{R}^{N_2 \times M}$, and for each $j \in \{0, 1, \ldots, M\}$, we have $\boldsymbol{W}_j \in \mathbb{R}^{L \times N_2}$.

A consequence of the above assumptions is the following lemma, which states that computing the worst-case expectation in the two-stage problem (3.2.1) is equivalent to averaging N worst-case values of the loss function $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ over $\boldsymbol{\xi} \in \Xi$, each regularized by one of the training samples.

Lemma 4. *The distributionally robust two-stage problem* (3.2.1) *admits the following reformulation:*

$$\min_{\boldsymbol{x} \in X, \alpha \geq 0} \; c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N} \sum_{i=1}^{N} \max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \, \mathrm{d}(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}^{(i)}) \right\}. \tag{3.3.2}$$

Proof. The proof directly follows from Theorem 1. □

The remainder of this section establishes that the inner maximization in (3.3.2) is equivalent to optimizing a linear function over the convex hull of the feasible region of a mixed-integer conic program (MICP). The following theorem is key to establishing this result.

Theorem 5 (Convex hull reformulation)**.** *The distributionally robust two-stage problem*

(3.2.1) *admits the following convex hull reformulation:*

$$\min_{\boldsymbol{x}\in X,\alpha\geq 0} \quad c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i=1}^{N} Z_i(\boldsymbol{x},\alpha), \tag{3.3.3}$$

where, for each $i \in [N]$*, we define the function* $Z_i : X \times \mathbb{R}_+ \mapsto \mathbb{R}$ *and the set* $\mathcal{Z}_i$ *as follows:*

$$Z_i(\boldsymbol{x},\alpha) = \max_{(\boldsymbol{\xi},\boldsymbol{\lambda},\boldsymbol{\Lambda},\tau)\in \operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)} \left\{ \langle \boldsymbol{T}(\boldsymbol{x}), \boldsymbol{\Lambda}\rangle + \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\lambda} - \alpha\tau \right\} \tag{3.3.4a}$$

$$\mathcal{Z}_i = \left\{ \begin{array}{l} (\boldsymbol{\xi},\boldsymbol{\lambda}) \in \Xi \times \mathbb{R}^L_+, \\ (\boldsymbol{\Lambda},\tau) \in \mathbb{R}^{L\times M} \times \mathbb{R}_+ \end{array} : \begin{array}{l} \boldsymbol{\Lambda} = \boldsymbol{\lambda}\boldsymbol{\xi}^\top,\ (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i, \tau) \in \mathcal{C}^{M+1} \\ \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} - \sum_{j\in[M]} \boldsymbol{W}_j^\top \boldsymbol{\Lambda}\mathbf{e}_j \in \mathcal{Y}^* \end{array} \right\}. \tag{3.3.4b}$$

Proof. See Appendix A.2. □

The inner optimization problem (3.3.4a) is over the closed convex hull of the set $\mathcal{Z}_i$, which couples the binary uncertain parameters $\boldsymbol{\xi}$ with the continuous dual variables $\boldsymbol{\lambda}$ via the bilinear equation $\boldsymbol{\Lambda} = \boldsymbol{\lambda}\boldsymbol{\xi}^\top$. The sets $\mathcal{Z}_i$ are thus not the feasible regions of MICPs. We propose two approaches to ensure MICP representability. In Section 3.3.1, we linearize the bilinear equation $\boldsymbol{\Lambda} = \boldsymbol{\lambda}\boldsymbol{\xi}^\top$ using McCormick inequalities, which requires knowledge of upper bounds on the dual variables $\boldsymbol{\lambda}$. In Section 3.3.2, we reformulate the value function $\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$ using ideas from penalty methods so that the uncertainty $\boldsymbol{\xi}$ appears only in the objective function, which circumvents any bilinear terms. We compare the two approaches and summarize their merits in Section 3.3.3.

3.3.1 Linearized Reformulation

The decision variables $\boldsymbol{\lambda}$ of the dual problem $\mathcal{Q}_d(\boldsymbol{x},\boldsymbol{\xi})$ must be necessarily bounded for any fixed value of $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$. Indeed, under assumptions (A1) and (A2), the value of $\mathcal{Q}_d(\boldsymbol{x},\boldsymbol{\xi})$ is bounded for any fixed $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$; and since X and Ξ are compact sets, the variables $\boldsymbol{\lambda}$ must also be necessarily bounded. Suppose that $\bar{\boldsymbol{\lambda}} \in \mathbb{R}^L_+$ are known upper bounds (independent of $\boldsymbol{x}$ and $\boldsymbol{\xi}$) on these variables. Such bounds may be analytically known whenever we explicitly add slack variables to ensure feasibility of the second-stage problem or if the latter has some structure (e.g., see [105]). Whenever such bounds are known, we can exactly linearize the bilinear equation $\boldsymbol{\Lambda} = \boldsymbol{\lambda}\boldsymbol{\xi}^\top$ using McCormick inequalities since $\boldsymbol{\xi}$ is binary-valued (e.g., see [106]), and reformulate the set $\mathcal{Z}_i$ in (3.3.4b) as the feasible region of an MICP:

$$\mathcal{Z}_i = \left\{ \begin{array}{l} (\boldsymbol{\xi}, \boldsymbol{\lambda}) \in \Xi \times \mathbb{R}^L_+, \\ (\boldsymbol{\Lambda}, \tau) \in \mathbb{R}^{L\times M}_+ \times \mathbb{R}_+ \end{array} : \begin{array}{l} (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i, \tau) \in \mathcal{C}^{M+1} \\ \boldsymbol{\Lambda} - \boldsymbol{\lambda}\mathbf{e}^\top + \bar{\boldsymbol{\lambda}}(\mathbf{e} - \boldsymbol{\xi})^\top \in \mathbb{R}^{L\times M}_+ \\ \boldsymbol{\lambda}\mathbf{e}^\top - \boldsymbol{\Lambda} \in \mathbb{R}^{L\times M}_+, \ \bar{\boldsymbol{\lambda}}\boldsymbol{\xi}^\top - \boldsymbol{\Lambda} \in \mathbb{R}^{L\times M}_+ \\ \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} - \sum_{j\in[M]} \boldsymbol{W}_j^\top \boldsymbol{\Lambda}\mathbf{e}_j \in \mathcal{Y}^* \end{array} \right\}. \tag{3.3.4b-ℓ}$$

The MICP representation (3.3.4b-ℓ) adds $O(ML)$ variables and constraints for each $\mathcal{Z}_i$, $i \in [N]$, which can be prohibitively large. Moreover, analytical upper bounds $\bar{\boldsymbol{\lambda}} \in \mathbb{R}^L_+$ on the dual variables, which are independent of (and valid for all) $\boldsymbol{x}$ and $\boldsymbol{\xi}$, may be unavailable. We circumvent this by using a penalty reformulation.

3.3.2 Penalty Reformulation

We now show that we can avoid adding McCormick inequalities to linearize the bilinear terms by considering a penalty reformulation, where the uncertainty in the second-stage problem is moved to the objective. By doing so, we immediately get a MICP in the inner optimization problem (3.3.4a), thus significantly reducing the number of additional constraints and variables, and the need to estimate bounds on the dual variables when using McCormick inequalities. In the following corollary to Theorem 5, we re-write the resulting MICPs when uncertainty is only present in the objective of $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$.

Corollary 1 (Convex hull reformulation for objective uncertainty)**.** *Suppose that only the* *objective function of the second-stage problem* $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ *is uncertain:* $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}_0$ *and* $\boldsymbol{T}(\boldsymbol{x}) = \mathbf{0}$*. Then the distributionally robust two-stage problem* (3.2.1) *admits reformulation* (3.3.3) *where, for each* $i \in [N]$*, we define the function* $Z_i : X \times \mathbb{R}_+ \mapsto \mathbb{R}$ *and the MICP-representable set* $\mathcal{Z}_i$ *as follows:*

$$Z_i(\boldsymbol{x}, \alpha) = \max_{(\boldsymbol{\xi},\boldsymbol{\lambda},\tau)\in \operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)} \left\{\boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\lambda} - \alpha\tau\right\} \tag{3.3.5a}$$

$$\mathcal{Z}_i = \left\{(\boldsymbol{\xi}, \boldsymbol{\lambda}, \tau) \in \Xi \times \mathbb{R}^L_+ \times \mathbb{R}_+ : (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i, \tau) \in \mathcal{C}^{M+1},\ \ \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} \in \mathcal{Y}^*\right\}. \tag{3.3.5b}$$

For the remainder of the chapter, we make the following additional assumption of fixed recourse which will be key to achieving our goal.

(A3) For every realization $\boldsymbol{\xi} \in \Xi$ and every first-stage decision $\boldsymbol{x} \in X$, we have $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}_0$ and $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{T}_0$, respectively.

Under this additional assumption, the following theorem states that the second-stage problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ with constraint uncertainty can be equivalently reformulated as one with objective uncertainty.

Theorem 6 (Penalty reformulation of the loss function)**.** *There exists a sufficiently large, yet finite, penalty parameter* $\rho > 0$ *such that the second-stage value function* $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ *in* (3.1.1) *is equivalent to*

$$\mathcal{Q}^\rho(\boldsymbol{x}, \boldsymbol{\xi}) := \inf_{\boldsymbol{y}\in\mathcal{Y}, \boldsymbol{z}\in[0,1]^M} \left\{\boldsymbol{q}(\boldsymbol{\xi})^\top \boldsymbol{y} + \rho\left((\mathrm{e} - 2\boldsymbol{\xi})^\top \boldsymbol{z} + \mathrm{e}^\top \boldsymbol{\xi}\right) : \boldsymbol{W}_0 \boldsymbol{y} \geq \boldsymbol{T}_0 \boldsymbol{z} + \boldsymbol{h}(\boldsymbol{x})\right\}. \tag{3.3.6}$$

Proof. The z variable in $\mathcal{Q}^\rho(\boldsymbol{x}, \boldsymbol{\xi})$ is an auxiliary variable representing $\boldsymbol{\xi}$. The idea is to penalize any components of z_i not matching ξ_i by ρ. First note that:

$$\begin{aligned}\|\boldsymbol{z}-\boldsymbol{\xi}\|_1 \le 0 &\iff \sum_{i\in[M]} |z_i-\xi_i| \le 0 \\ &\iff \sum_{i\in[M]} [\xi_i(1-z_i)+(1-\xi_i)z_i] \le 0 \\ &\iff (\mathbf{e}-2\boldsymbol{\xi})^\top \boldsymbol{z} + \mathbf{e}^\top\boldsymbol{\xi} \le 0,\end{aligned}$$

where the second equivalence holds because $\boldsymbol{\xi}\in\{0,1\}^M$ and $\boldsymbol{z}\in[0,1]^M$. Thus, enforcing $(\mathbf{e}-2\boldsymbol{\xi})^\top\boldsymbol{z}+\mathbf{e}^\top\boldsymbol{\xi}\le 0$ is equivalent to ensuring $\boldsymbol{z}=\boldsymbol{\xi}$. We then have that $\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$ is equivalent to

$$\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi}) = \inf_{\boldsymbol{y}\in\mathcal{Y},\boldsymbol{z}\in[0,1]^M} \left\{\boldsymbol{q}(\boldsymbol{\xi})^\top\boldsymbol{y} : \boldsymbol{W}_0\boldsymbol{y} \ge \boldsymbol{T}_0\boldsymbol{z}+\boldsymbol{h}(\boldsymbol{x}),\ (\mathbf{e}-2\boldsymbol{\xi})^\top\boldsymbol{z}+\mathbf{e}^\top\boldsymbol{\xi}\le 0\right\}. \tag{3.3.7}$$

Let ρ be the Lagrangian multiplier associated with the added inequality, and let $\mathcal{Q}^\rho(\boldsymbol{x},\boldsymbol{\xi})$ be the resulting Lagrangian function after relaxing the added inequality. Note that the second-stage problem is convex, and under Assumption (A1) and Assumption (A2), $\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})$ is bounded and feasible. Thus, we have strong duality:

$$\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi}) = \sup_{\rho\ge 0} \mathcal{Q}^\rho(\boldsymbol{x},\boldsymbol{\xi}).$$

It suffices to pick ρ to be the optimal Lagrangian multiplier associated with $(\mathbf{e}-2\boldsymbol{\xi})^\top\boldsymbol{z}+\mathbf{e}^\top\boldsymbol{\xi}\le 0$ in (3.3.7), and the result follows. In fact, since $\mathcal{Q}^\rho(\boldsymbol{x},\boldsymbol{\xi})$ is concave and non-decreasing with respect to ρ (since $(\mathbf{e}-2\boldsymbol{\xi})^\top\boldsymbol{z}+\mathbf{e}^\top\boldsymbol{\xi}\ge 0$ whenever $\boldsymbol{z}\in[0,1]^M$ and $\boldsymbol{\xi}\in\{0,1\}^M$), for a fixed choice of $\boldsymbol{x}\in X$ and $\boldsymbol{\xi}\in\Xi$, it suffices to choose any value of ρ that is greater than or equal to the optimal Lagrangian multiplier of the last constraint in (3.3.7). □

In conjunction with Corollary 1, Theorem 6 implies that the two-stage distributionally robust problem (3.2.1) admits a convex hull reformulation of the form (3.3.3), where the function $Z_i : X \times \mathbb{R}_+ \mapsto \mathbb{R}$ and the MICP-representable set $\mathcal{Z}_i$ for each $i \in [N]$ are given as follows:

$$Z_i(\boldsymbol{x}, \alpha) = \max_{(\boldsymbol{\xi},\boldsymbol{\lambda},\boldsymbol{\mu},\tau)\in \operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)} \left\{ \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\lambda} + \rho(\mathrm{e}^\top \boldsymbol{\xi}) - \mathrm{e}^\top \boldsymbol{\mu} - \alpha\tau \right\} \tag{3.3.4a-ρ}$$

$$\mathcal{Z}_i = \left\{ (\boldsymbol{\xi},\boldsymbol{\lambda},\boldsymbol{\mu},\tau) \in \Xi \times \mathbb{R}^L_+ \times \mathbb{R}^M_+ \times \mathbb{R}_+ : \begin{array}{l} (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{(i)}, \tau) \in \mathcal{C}^{M+1} \\ \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} \in \mathcal{Y}^* \\ \rho(\mathrm{e} - 2\boldsymbol{\xi}) + \boldsymbol{T}_0^\top \boldsymbol{\lambda} + \boldsymbol{\mu} \in \mathbb{R}^M_+ \end{array} \right\}. \tag{3.3.4b-ρ}$$

In contrast to the linearized reformulation (3.3.4b-ℓ), the MICP representation (3.3.4b-ρ) adds only $O(M + L)$ variables and constraints for each $\mathcal{Z}_i$, $i \in [N]$.

The reformulation (3.3.6) is motivated by penalty methods in nonlinear programming. However, in contrast to the latter, which suffers from numerical issues because the penalty parameter ρ must be driven to ∞, a finite value for ρ can be precomputed in our case. Indeed, Theorem 6 shows that a finite ρ exists. We devise a method to obtain an approximate value of ρ, and later present a procedure to compute an exact penalty parameter. The following proposition will guide our choice of ρ.

Proposition 3. *The optimal value of the distributionally robust two-stage problem* (3.2.1) *where we replace the second-stage value function $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ with its penalty reformulation $\mathcal{Q}^\rho(\boldsymbol{x}, \boldsymbol{\xi})$ as defined in* (3.3.6) *is equivalent to*

$$\max_{\rho \geq 0} \min_{\boldsymbol{x}\in X, \alpha \geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N} \sum_{i\in[N]} \max_{\boldsymbol{\xi}\in\Xi} \left\{ \mathcal{Q}^\rho(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \right\}$$

The proof of Proposition 3 relies on the following technical lemma.

Lemma 5. *For each $i \in [N]$, let $f_i : \mathbb{R}_+ \mapsto \mathbb{R}$ be a non-decreasing function such that the* *supremum* $\sup_{\rho \geq 0} f_i(\rho)$ *is achieved for some finite* ρ. *Then, the following equality holds:*

$$\sum_{i \in [N]} \max_{\rho \geq 0} f_i(\rho) = \max_{\rho \geq 0} \sum_{i \in [N]} f_i(\rho). \tag{3.3.8}$$

Proof. We have that $\sum_{i \in [N]} \max_{\rho \geq 0} f_i(\rho) \geq \max_{\rho \geq 0} \sum_{i \in [N]} f_i(\rho)$ is clearly true. To show the other direction, we use the fact that the functions f_i are monotone non-decreasing. Let $\rho^* \in \arg\max_{\rho \geq 0} \sum_{i \in [N]} f_i(\rho)$. Assume that $\sum_{i \in [N]} \max_{\rho \geq 0} f_i(\rho) < \sum_{i \in [N]} f_i(\rho^*)$. Then we must have $\rho^* \notin \arg\max_{\rho \geq 0} f_{i'}(\rho)$ for some $i' \in [N]$, and there exists $\hat{\rho} > \rho^*$ such that $f_{i'}(\hat{\rho}) > f_{i'}(\rho^*)$. It follows from their monotonicity that $f_j(\hat{\rho}) \geq f_j(\rho^*)$ for all $j \in [N] \setminus \{i'\}$. This implies that $\sum_{i \in [N]} f_i(\hat{\rho}) > \sum_{i \in [N]} f_i(\rho^*)$, contradicting that ρ^* is a maximizer of the right-hand side. $\square$

We now prove Proposition 3.

Proof. From Theorem 6, we have

$$\begin{aligned}
&\min_{\boldsymbol{x} \in X, \alpha \geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N} \sum_{i \in [N]} \max_{\boldsymbol{\xi} \in \Xi} \left\{ \max_{\rho \geq 0} \mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}} \right\| \right\} \\
&= \min_{\boldsymbol{x} \in X, \alpha \geq 0} c(\boldsymbol{x}) + \alpha\theta + \max_{\rho \geq 0} \frac{1}{N} \sum_{i \in [N]} \max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}} \right\| \right\} \\
&= \min_{\boldsymbol{x} \in X, \alpha \geq 0} \max_{\rho \geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N} \sum_{i \in [N]} \max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}} \right\| \right\}
\end{aligned}$$

where we assume the infimum and the suprema over ρ and Ξ are attained. The second equality holds because, with respect to ρ, $\mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi})$ is monotone non-decreasing, and therefore, $\max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}} \right\| \right\}$ is also monotone non-decreasing. Thus, Lemma 5 applies. Using the fact that X is convex and compact, and the objective is convex in $(\boldsymbol{x}, \alpha)$

and quasi-concave in ρ, we apply Sion's minimax theorem to get:

$$\begin{aligned}
&\min_{\boldsymbol{x}\in X,\alpha\geq 0}\max_{\rho\geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i\in[N]}\max_{\boldsymbol{\xi}\in\Xi}\left\{\mathcal{Q}^{\rho}(\boldsymbol{x},\boldsymbol{\xi}) - \alpha\left\|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}}\right\|\right\} \\
=&\max_{\rho\geq 0}\min_{\boldsymbol{x}\in X,\alpha\geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i\in[N]}\max_{\boldsymbol{\xi}\in\Xi}\left\{\mathcal{Q}^{\rho}(\boldsymbol{x},\boldsymbol{\xi}) - \alpha\left\|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{\boldsymbol{i}}\right\|\right\} \qquad (3.3.9)
\end{aligned}$$

□

We note that for any $\theta \geq 0$, the inner minimization in (3.3.9) is bounded above by the optimal value of the classical robust optimization problem for any support set $\Xi^0 \supseteq \Xi$. In fact, we have the following relationship

$$\max_{\rho\geq 0}\min_{\boldsymbol{x}\in X} c(\boldsymbol{x}) + \max_{\boldsymbol{\xi}\in\Xi^0}\mathcal{Q}^{\rho}(\boldsymbol{x},\boldsymbol{\xi}) \qquad (3.3.10)$$

$$\geq \max_{\rho\geq 0}\min_{\boldsymbol{x}\in X,\alpha\geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i\in[N]}\max_{\boldsymbol{\xi}\in\Xi}\left\{\mathcal{Q}^{\rho}(\boldsymbol{x},\boldsymbol{\xi}) - \alpha\left\|\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{i}\right\|\right\} \qquad \forall\theta \geq 0. \qquad (3.3.11)$$

Let $\mathcal{L}(\theta,\rho)$ be the optimal cost of the inner minimization of (3.3.11) for a fixed ρ and θ. Note that a maximizer ρ^r of (3.3.10), the classical robust problem over Ξ^0, does not necessarily maximize (3.3.11), but we expect ρ^r to be a reliable upper bound to the true optimal ρ^θ of the distributionally robust problem. First note that given an optimal penalty parameter ρ^θ for the distributionally robust problem, we can pick any $\rho \geq \rho^\theta$ to get an equivalent problem since $\mathcal{L}(\theta,\rho)$ is non-decreasing in ρ. Moreover, we intuitively expect that for the robust problem, a high value of ρ is needed to compensate for the high costs incurred in the worst-case scenario to ensure $\boldsymbol{z} = \boldsymbol{\xi}$, i.e. we expect ρ^r to be greater than ρ^θ. The benefit of using ρ^r as an approximation is its computation only requires solving a tractable deterministic problem in many real-world applications. We first detail how to practically compute ρ^r. In Section 3.5, we present a procedure to check whether ρ^r is indeed large enough for the distributionally robust problem for a radius θ, and a simple iterative procedure to update it if not optimal.

Computing ρ^r

This process only requires solving the classical robust optimization formulation over any support $\Xi^0 \supseteq \Xi$ such that the robust optimization problem is "easy" to solve. More specifically, to compute a finite ρ, we solve the robust optimization problem over a support set $\Xi^0 \supseteq \Xi$ and record an optimal first-stage solution $\boldsymbol{x}^{\boldsymbol{r}}$. We then compute $\boldsymbol{\xi}^r \in \arg\max_{\boldsymbol{\xi}\in\Xi^0} \mathcal{Q}(\boldsymbol{x}^{\boldsymbol{r}}, \xi)$, and solve the second-stage problem $\mathcal{Q}(\boldsymbol{x}^{\boldsymbol{r}}, \boldsymbol{\xi}^{\boldsymbol{r}})$ given by (3.3.7). We then set ρ^r to be the optimal Lagrangian multiplier of the last inequality in (3.3.7).

The robust optimization problem is tractable if we choose $\Xi^0 = \{0,1\}^M$ and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ exhibits a down-monotone (or up-monotone) property with respect to the random parameters $\boldsymbol{\xi}$; that is, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}') \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ whenever $\boldsymbol{\xi}' \geq \boldsymbol{\xi}$. Indeed, in many applications in network optimization, removing network components is never advantageous. In such cases, the robust optimization problem reduces to a deterministic one and becomes tractable. For example, consider a problem where a set of edges are subject to random failures. The worst-case realization is then to disrupt all edges. In other words, $\boldsymbol{\xi}^{\boldsymbol{r}}$ corresponds to the vector of all ones. Given $\boldsymbol{\xi}^{\boldsymbol{r}}$, we formulate and solve the following deterministic problem:

$$\begin{aligned}
\min_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}} \quad & c(\boldsymbol{x}) + \boldsymbol{q}(\boldsymbol{\xi}^{\boldsymbol{r}})^\top \boldsymbol{y} \\
\text{s.t.} \quad & \boldsymbol{x} \in X, \\
& \boldsymbol{y} \in \mathcal{Y}, \\
& \boldsymbol{z} \in [0,1]^M, \\
& \boldsymbol{W}_0 \boldsymbol{y} \geq \boldsymbol{T}_0 \boldsymbol{z} + \boldsymbol{h}(\boldsymbol{x}), \\
& (\mathrm{e} - 2\boldsymbol{\xi}^{\boldsymbol{r}})^\top \boldsymbol{z} + \mathrm{e}^\top \boldsymbol{\xi}^{\boldsymbol{r}} \leq 0.
\end{aligned} \tag{3.3.12}$$

We then set ρ^r to be the optimal Lagrangian multiplier of the last constraint in (3.3.12). We generalize this observation in the following proposition.

Proposition 4. *Assume without loss of generality that $\mathcal{Y} \subseteq \mathbb{R}_+^{N_2}$. Suppose also that for all $j \in [M]$, we have either $\boldsymbol{T}_0 \mathrm{e}_j, \boldsymbol{Q}\mathrm{e}_j \in \mathbb{R}_+^L$ or $-\boldsymbol{T}_0 \mathrm{e}_j, -\boldsymbol{Q}\mathrm{e}_j \in \mathbb{R}_+^L$. Then, the classical robust optimization problem $\min_{\boldsymbol{x}\in X} \left\{c(\boldsymbol{x}) + \max_{\boldsymbol{\xi}\in\{0,1\}^M} \mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})\right\}$ reduces to a deterministic problem $\min_{\boldsymbol{x}\in X} \{c(\boldsymbol{x}) + \mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi}^r)\}$, where for each $j \in [M]$ we have $\xi_j^r = 1$ if $\boldsymbol{T}_0 \mathrm{e}_j, \boldsymbol{Q}\mathrm{e}_j \in \mathbb{R}_+^L$ and $\xi_j^r = 0$ otherwise.*

Proof. See Appendix A.2. □

Penalty reformulation for indicator constraints

We note that we can avoid introducing the auxiliary variable $\boldsymbol{z} \in [0,1]^M$ in the penalty reformulation $\mathcal{Q}^\rho(\boldsymbol{x},\boldsymbol{\xi})$ in (3.3.6) by exploiting a common structure in network optimization problems. In many applications, the binary random variables represent on and off switches such as $f_j(\boldsymbol{y}) \le \bar{f}_j(1-\xi_j)$, where $f_j : \mathcal{Y} \mapsto \mathbb{R}$ is an affine function for $j \in [M]$ and $\bar{f}_j$ is an appropriate upper bound of $f_j(\boldsymbol{y})$. For example, in a network, $\xi_j = 1$ might indicate that link j has failed and the corresponding flow $f_j(\boldsymbol{y}) = y_j$ on this link must be set to 0. Such a second-stage problem can be written as

$$\mathcal{Q}_{\text{ind}}(\boldsymbol{x},\boldsymbol{\xi}) = \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{q}(\boldsymbol{\xi})^\top \boldsymbol{y} : \begin{array}{c} \boldsymbol{W}_0 \boldsymbol{y} \ge \boldsymbol{h}(\boldsymbol{x}) \\ f_j(\boldsymbol{y}) \le \bar{f}_j(1-\xi_j),\ j \in [M] \\ f_j(\boldsymbol{y}) \ge 0,\ j \in [M] \end{array} \right\}, \tag{3.3.13}$$

In such cases, we need only penalize nonzero values of $f_j(\boldsymbol{y})$ if $\xi_j = 1$. To move the uncertainty to the objective, we add penalty terms $\rho\xi_j f_j(\boldsymbol{y})$ for each $j \in [M]$ in the objective, remove the associated constraints $f_j(\boldsymbol{y}) \le \bar{f}_j(1-\xi_j)$, and keep the non-negativity constraints $f_j(\boldsymbol{y}) \ge 0$. For large enough values of ρ, the penalty term drives $f_j(\boldsymbol{y})$ to 0 if $\xi_j = 1$. With this transformation, we avoid introducing M auxiliary variables z_j and avoid having to estimate upper bounds $\bar{f}_j$. Obtaining tight estimates on $\bar{f}_j$ may be non-trivial and lead to numerical issues if the estimates are too large.

Corollary 2 (Penalty reformulation of the loss function with indicator constraints)**.** *There exists a sufficiently large, yet finite, penalty parameter $\rho > 0$ such that the second-stage value function $\mathcal{Q}_{\text{ind}}(\boldsymbol{x}, \boldsymbol{\xi})$ in* (3.3.13) *is equivalent to*

$$\mathcal{Q}^{\rho}_{\text{ind}}(\boldsymbol{x}, \boldsymbol{\xi}) = \inf_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \boldsymbol{q}(\boldsymbol{\xi})^\top \boldsymbol{y} + \rho \boldsymbol{\xi}^\top \boldsymbol{f}(\boldsymbol{y}) : \boldsymbol{W}_0 \boldsymbol{y} \geq \boldsymbol{h}(\boldsymbol{x}),\ \boldsymbol{f}(\boldsymbol{y}) \geq \boldsymbol{0} \right\}. \tag{3.3.14}$$

Our previous results continue to be valid as long as we replace each occurrence of the penalty term $(\mathrm{e} - 2\boldsymbol{\xi})^\top \boldsymbol{z} + \mathrm{e}^\top \boldsymbol{\xi}$ that multiplies ρ in the objective function of (3.3.6) with this modification. For example, suppose that $\boldsymbol{f}(\boldsymbol{y}) = \boldsymbol{f}_0 + \boldsymbol{F}\boldsymbol{y}$. Then, Corollaries 1 and 2 imply that the two-stage distributionally robust problem (3.2.1) admits a convex hull reformulation of the form (3.3.3), where the function $Z_i : X \times \mathbb{R}_+ \mapsto \mathbb{R}$ and the MICP-representable set $\mathcal{Z}_i$ for each $i \in [N]$ are given as follows:

$$Z_i(\boldsymbol{x}, \alpha) = \underset{(\boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \tau) \in \operatorname{cl} \operatorname{conv}(\mathcal{Z}_i)}{\text{maximize}} \left\{ \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\lambda} + \rho \boldsymbol{f}_0^\top \boldsymbol{\xi} - \boldsymbol{f}_0^\top \boldsymbol{\mu} - \alpha \tau \right\}$$

$$\mathcal{Z}_i = \left\{ \begin{array}{ll} (\boldsymbol{\xi}, \boldsymbol{\lambda}) \in \Xi \times \mathbb{R}^L_+, & (\boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i, \tau) \in \mathcal{C}^{M+1} \\ (\boldsymbol{\mu}, \tau) \in \mathbb{R}^M_+ \times \mathbb{R}_+ & \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} + \rho \boldsymbol{F}^\top \boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} - \boldsymbol{F}^\top \boldsymbol{\mu} \in \mathcal{Y}^* \end{array} \right\}.$$

The procedure to compute ρ^r is exactly the same, except we replace the constraint $(\mathrm{e} - 2\boldsymbol{\xi}^r)^\top \boldsymbol{z} + \mathrm{e}^\top \boldsymbol{\xi} \leq 0$ with $(\boldsymbol{\xi}^r)^\top f(\boldsymbol{y}) \leq 0$ in (3.3.7). Equivalently, if $\boldsymbol{\xi}^r$ is easily known as in Proposition 4, then we replace the former constraint with the latter in the robust optimization problem (3.3.12).

3.3.3 Summary and Comparison

Table 3.1 summarizes the main differences between the linearized and penalty-based MICP reformulations of the sets $\mathcal{Z}_i$, $i \in [N]$ appearing in the convex hull reformulation (3.3.4a)–(3.3.4b). Notably, the penalty reformulation adds far fewer variables and constraints. However, it also requires additional assumptions and computations. In particular, it requires computing a value for the penalty parameter ρ, which may further entail the solution of a classical robust optimization problem. We do not expect this to be a limitation, however, because the latter will likely reduce to a deterministic optimization problem for most practical applications.

Table 3.1: Summary of the MICP representations of $\mathcal{Z}_i$ based on the linearized and penalty reformulations.

Reformulation	Size	$\boldsymbol{W}(\boldsymbol{\xi})$	$\boldsymbol{T}(\boldsymbol{x})$	Requirements
Linearized (3.3.4b-ℓ)	$\mathcal{O}(ML)$	affine	convex	a priori bounds on $\boldsymbol{\lambda}$ in $\mathcal{Q}_d(\boldsymbol{x}, \boldsymbol{\xi})$
Penalty (3.3.4b-ρ)	$\mathcal{O}(M+L)^*$	constant	constant	computation of penalty parameter†

*Can be further reduced; see Corollary 2

†Approximate penalty parameter can be practically computed; see Proposition 4 and preceding discussion.

3.4 Lift-and-Project Approximations

The key challenge in solving the convex hull reformulation (3.3.3) is the inner optimization (3.3.4a) over the convex hull of the MICP-representable set $\mathcal{Z}_i$, $i \in [N]$. Therefore, Appendix A.1 presents a Benders scheme, similar to the ones proposed in [107, 108], to tackle the convex hull constraints. This scheme iteratively refines an inner approximation of the MICP representation of $\mathcal{Z}_i$. An alternative to solving the convex hull reformulation (3.3.3) is direct solution of the original reformulation (3.3.2) using a column-and-constraint generation scheme [109]. In contrast to the Benders scheme, the latter models the second-stage problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ via explicit second-stage variables and constraints by implicitly enumerating $\boldsymbol{\xi} \in \Xi$. In both schemes, however, each iteration requires the solution of N global MICP problems and therefore, can become computationally prohibitive. Moreover, intermediate solutions obtained from early termination provide no guarantees since they bound the optimal value of the distributionally robust problem (3.2.1) from below, which itself is an upper bound on the true (unknown) optimal value.

These observations motivate the development of tractable outer approximations of the convex hulls of $\mathcal{Z}_i$, $i \in [N]$, that provide not only *(i)* upper bounds on the optimal value of (3.2.1) but also *(ii)* guarantees of polynomial time solvability. Our approximations are based on the following key observations. First, if we suppose that $\Xi = \{\boldsymbol{\xi} \in \mathbb{Z}_+^M : \boldsymbol{E\xi} \leq \boldsymbol{f}\}$ has a given outer description, then Section 3.3 establishes that each of the sets $\mathcal{Z}_i$, $i \in [N]$, can be represented as the feasible region of an MICP as follows:

$$\mathcal{Z} = \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{Az} - \boldsymbol{b} \in \mathcal{K}, \;\; z_j \in \{0,1\} \; j \in [M]\}\,, \tag{3.4.1}$$

where, for ease of exposition, we have dropped the subscript i and included the bounds, $\boldsymbol{z} \geq \boldsymbol{0}$ and $1 \geq z_j := \xi_j$, $j \in [M]$ in $\boldsymbol{Az} - \boldsymbol{b} \in \mathcal{K}$. For example, in case of (3.3.4b-ℓ), we have $\boldsymbol{z} = (\boldsymbol{\xi}, \boldsymbol{\lambda}, \operatorname{vec}(\boldsymbol{\Lambda}), \tau)$, $n = M + L + ML + 1$, $\mathcal{K} = \tilde{\mathcal{K}} \times \mathbb{R}_+^n$, where $\tilde{\mathcal{K}} = \mathbb{R}_+^F \times_{i=1}^3 \mathbb{R}_+^{LM} \times \mathcal{C}^{M+1} \times \mathcal{Y}^*$ and F is the dimension of $\boldsymbol{f} \in \mathbb{R}^F$, and $\boldsymbol{A} = [\tilde{\boldsymbol{A}}^\top \, \mathbf{I}]^\top$ and $\boldsymbol{b} = [\tilde{\boldsymbol{b}}^\top \, \mathbf{0}^\top]^\top$ model the right-hand half of (3.3.4b-ℓ). Similarly, in case of (3.3.4b-ρ), we have $\boldsymbol{z} = (\boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \tau)$, $n = 2M + L + 1$, $\mathcal{K} = \tilde{\mathcal{K}} \times \mathbb{R}_+^n$, where $\tilde{\mathcal{K}} = \mathbb{R}_+^F \times \mathbb{R}_+^M \times \mathcal{C}^{M+1} \times \mathcal{Y}^*$, and $\boldsymbol{A} = [\tilde{\boldsymbol{A}}^\top \, \mathbf{I}]^\top$, $\boldsymbol{b} = [\tilde{\boldsymbol{b}}^\top \, \mathbf{0}^\top]^\top$ for suitable matrices $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{b}}$.

Second, given an MICP representation such as the above, its convex hull can be approximated by a hierarchy of increasingly tight convex relaxations,

$$\mathcal{Z}^0 \supseteq \mathcal{Z}^1 \supseteq \ldots \supseteq \mathcal{Z}^M = \operatorname{cl}\operatorname{conv}(\mathcal{Z}),$$

that converge to the convex hull in M iterations. Here, $\mathcal{Z}^0 := \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{Az} - \boldsymbol{b} \in \mathcal{K}\}$ is the continuous relaxation of $\mathcal{Z}$. Several such sequential convexification hierarchies are known, the most popular ones being those of [91, 110, 92, 93]. They are based on the concept of lift-and-project and represent $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$ as the projection of another convex set lying in a higher-dimensional space. These hierarchies were originally proposed for (pure or mixed-) integer linear sets and later extended to mixed-integer convex sets in [101, 102]. Our proposal is to use an intermediate relaxation $\mathcal{Z}^t$ of any such hierarchy to outer approximate $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$, which results in an outer approximation of the convex hull reformulation (3.3.3) and, hence, a conservative approximation of the distributionally robust two-stage problem (3.2.1). The approximation can be refined, if desired, by using higher values of t.

Third, the approximation of $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$ when used in the convex hull reformulation allows us to dualize the inner optimization in (3.3.4a) using conic duality. The result is a single-stage convex conic optimization model that can be solved using off-the-shelf solvers. Notably, we can prove that the resulting approximations of the distributionally robust two-stage problem (3.2.1), obtained by replacing $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$ with any of the relaxations $\mathcal{Z}^0, \ldots, \mathcal{Z}^M$, become exact if the radius ε of the Wasserstein ambiguity set $\mathcal{P}$ shrinks to zero with increasing sample size N.

Theorem 7 (Lift-and-project approximation quality)**.** *Suppose that* $N\varepsilon_N \to 0$ *as* $N \to \infty$*. Then, the optimal value of the distributionally robust two-stage problem* (3.2.1) *coincides with that of the convex hull reformulation* (3.3.3) *even if we approximate each* $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$*,* $i \in [N]$*, with* $\mathcal{Z}_i^0$ *in* (3.3.4a)*, i.e. the continuous relaxation.*

Proof. See proof in [2]. □

We emphasize that our method is not tied to any particular convexification technique. This feature is important because each technique has its advantages and disadvantages. For example, in the linear case (i.e., $\mathcal{K} = \mathbb{R}_+^{n'}$), it is known [111] that the approximations in order of decreasing tightness are those of [93], [92], [91], and [110]; however, this ranking is reversed when they are ordered based on increasing computational complexity. For its simplicity and tradeoff between tightness and tractability, we focus on the Lovász-Schrijver approximation [91] in the remainder of this section. We show how it can be used to obtain a single-stage approximation of the distributionally robust two-stage problem (3.2.1), and we provide practical guidelines for its efficient computation.

3.4.1 Lovász-Schrijver Approximation

The level-1 Lovász-Schrijver approximation $\mathcal{Z}^1$ is defined as a set-valued mapping, and the level-t approximation $\mathcal{Z}^t$ is defined as an iterated application of this mapping. For any $u \in \mathbb{R}_+$ and any conic representable set such as the continuous relaxation $\mathcal{Z}^0 = \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b} \in \mathcal{K}\}$, we denote by $\mathcal{Z}^0(u) = \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}u \in \mathcal{K}\}$ to be the homogenization of $\mathcal{Z}^0$ with respect to u. Next, we define the following lifted set:

$$\mathcal{L}(\mathcal{Z}^0) = \left\{ \begin{array}{lcll} & & \exists u^{j0}, u^{j1} \geq 0,\ u^{j0} + u^{j1} = 1, & j \in [M] \\ \boldsymbol{z} \in \mathbb{R}^n, & & \boldsymbol{z}^{j0} \in \mathcal{Z}^0(u^{j0}),\ \boldsymbol{z}^{j1} \in \mathcal{Z}^0(u^{j1}), & j \in [M] \\ \{\boldsymbol{z}^{j0}\}_{j\in[M]} \in \mathbb{R}^n, & : & \boldsymbol{z} = \boldsymbol{z}^{j0} + \boldsymbol{z}^{j1}, & j \in [M] \\ \{\boldsymbol{z}^{j1}\}_{j\in[M]} \in \mathbb{R}^n & & z_j^{j0} = 0,\ z_j^{j1} = u^{j1}, & j \in [M] \\ & & z_k^{j1} = z_j^{k1},\ \ k \in [M] : k > j, & j \in [M] \end{array} \right\}. \quad (3.4.2)$$

Consider now the following set-valued map, which is the projection of $\mathcal{L}(\mathcal{Z}^0)$ onto $\mathbb{R}^n$:

$$\mathcal{P}(\mathcal{Z}^0) = \left\{ \boldsymbol{z} \in \mathbb{R}^n : \exists \boldsymbol{z}^{j0}, \boldsymbol{z}^{j1},\ j \in [M] \text{ such that } \left(\boldsymbol{z}, \{\boldsymbol{z}^{j0}\}_{j\in[M]}, \{\boldsymbol{z}^{j1}\}_{j\in[M]}\right) \in \mathcal{L}(\mathcal{Z}^0) \right\}. \quad (3.4.3)$$

One can easily verify that $\mathcal{P}(\mathcal{Z}^0)$ is a convex relaxation of $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$. In fact, we have the following relationship [102, Theorem 1]:

$$\operatorname{cl}\operatorname{conv}(\mathcal{Z}) \subseteq \mathcal{P}(\mathcal{Z}^0) \subseteq \bigcap_{j\in[M]} \operatorname{cl}\operatorname{conv}(\{\boldsymbol{z} \in \mathcal{Z}^0 : z_j \in \{0,1\}\}) \subseteq \mathcal{Z}^0.$$

The set $\mathcal{P}(\mathcal{Z}^0)$ corresponds to the level-1 relaxations of the Lovász-Schrijver hierarchy. For any $t \geq 1$, the level-t relaxation is given by $\mathcal{Z}^t = \mathcal{P}(\mathcal{Z}^{t-1})$, and one can show that $\mathcal{Z}^M = \operatorname{cl}\operatorname{conv}(\mathcal{Z})$. This is known as the linear Lovász-Schrijver hierarchy.

For a given MICP representation of $\mathcal{Z}_i$ and any $t \geq 1$, we can use the level-t Lovász-Schrijver relaxation to approximate $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$ in (3.3.4a)–(3.3.4b). We can then dualize the inner maximization (3.3.4a) to obtain one global problem.

Remark 2 (Relationship to approximations obtained by relaxing the support)**.** *An alternative outer approximation of the distributionally robust two-stage problem* (3.2.1) *can be obtained by simply relaxing the zero-one constraints on the support in reformulation* (3.3.2)*; i.e., by replacing* $\Xi = \{\boldsymbol{\xi} \in \mathbb{Z}_+^M : \boldsymbol{E}\boldsymbol{\xi} \leq \boldsymbol{f}\}$ *in* (3.3.2) *with its continuous relaxation* $\{\boldsymbol{\xi} \in \mathbb{R}_+^M : \boldsymbol{E}\boldsymbol{\xi} \leq \boldsymbol{f}\}$*. The resulting approximation is intractable in general, unless the uncertainty* $\boldsymbol{\xi}$ *appears only in the objective of* $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ *(see discussion in Section 3.3). In the latter case, it can be easily seen that the resulting approximation coincides precisely with that obtained by replacing* $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$ *in* (3.3.5a)–(3.3.5b) *with the continuous relaxation* $\mathcal{Z}_i^0$*. Therefore, our lift-and-project approximation technique can be viewed as a generalization of this approach. Unlike the former, however, a crucial difference is that this approach provides no formal mechanism to improve the quality of the final approximation; we illustrate this empirically in Section 3.6.*

3.4.2 Numerical Considerations

We note that the level-t Lovász-Schrijver approximation becomes difficult to solve for even small values of t. In fact, we observe in our experiments that even the level-1 Lovász-Schrijver approximation is computationally challenging. We can reduce the size of the level-1 approximation by removing the last set of constraints in (3.4.2) $z_k^{j_1} = z_j^{k_1}$. This results in only a minor loss in approximation quality. The resulting relaxation is still equal to $\bigcap_{j \in [M]} \operatorname{cl}\operatorname{conv}(\{\boldsymbol{z} \in \mathcal{Z}_i^0 : z_j \in \{0,1\}\})$. In fact, given an optimal solution of the maximization over $\mathcal{Z}^1$, the relaxation is equal to $\bigcap_{j \in \mathcal{J}_i} \operatorname{cl}\operatorname{conv}(\{\boldsymbol{z} \in \mathcal{Z}_i^0 : z_j \in \{0,1\}\})$, where $\mathcal{J}_i \subseteq [M]$ is the index set of binary parameters whose optimal values are fractional. From Theorem 7, we expect $|\mathcal{J}_i|$ to be small for smaller values of θ, and thus the optimal value $\boldsymbol{\xi}$ to be close to being binary.

This motivates the following iterative heuristic to identify the index sets $\mathcal{J}_i$. Note that this procedure is independent of the MICP representation used for each $\mathcal{Z}_i$. We define $\tilde{\mathcal{Z}}_i^1$ to be the resulting approximation of $\bigcap_{j \in \mathcal{J}_i} \operatorname{cl}\operatorname{conv}(\{\boldsymbol{z} \in \mathcal{Z}_i^0 : z_j \in \{0,1\}\})$ in the following procedure.

1. Select tol $\in (0, 0.5)$, niter $\in \mathbb{Z}_+$. Set iter $\leftarrow 1$. For each $i \in [N]$, set $\tilde{\mathcal{Z}}_i^1 \leftarrow \mathcal{Z}_i^0$ and $\mathcal{J}_i \leftarrow \emptyset$.

2. For each $i \in [N]$, replace $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$ with its current approximation $\tilde{\mathcal{Z}}_i^1$ and dualize the corresponding problem (3.3.4a). Solve the resulting convex hull approximation (3.3.3).

3. For each $i \in [N]$, let $\bar{\boldsymbol{\xi}}^{[i]}$ be the optimal value of $\boldsymbol{\xi}$ in (3.3.4a), recovered as scaled dual multipliers. For each $j \in [M] \setminus \mathcal{J}_i$, if $\bar{\xi}_j^{[i]} \in [\texttt{tol}, 1 - \texttt{tol}]$, update $\mathcal{J}_i \leftarrow \mathcal{J}_i \cup \{j\}$ and $\tilde{\mathcal{Z}}_i^1$ as follows:

$$\tilde{\mathcal{Z}}_i^1 \leftarrow \left\{ \boldsymbol{z} \in \mathbb{R}^n : \begin{array}{ll} \exists u^{j0}, u^{j1} \geq 0,\ u^{j0} + u^{j1} = 1, & j \in \mathcal{J}_i \\ \exists \boldsymbol{z}^{j0} \in \mathcal{Z}_i^0(u^{j0}),\ \boldsymbol{z}^{j1} \in \mathcal{Z}_i^0(u^{j1}), & j \in \mathcal{J}_i \\ \boldsymbol{z} = \boldsymbol{z}^{j0} + \boldsymbol{z}^{j1}, & j \in \mathcal{J}_i \\ z_j^{j0} = 0,\ z_j^{j1} = u^{j1}, & j \in \mathcal{J}_i \end{array} \right\}.$$

4. If none of the index sets $\mathcal{J}_1, \ldots, \mathcal{J}_N$ were updated or if iter $\geq$ niter, stop. Otherwise, update iter $\leftarrow$ iter $+ 1$ and go to Step 2.

Note that the successive optimizations in Step 2 can benefit from an efficient initialization of their variables by using the optimal solution from the previous solve. Moreover, the size of these problems can be controlled by using smaller values of niter and larger values of tol, since they directly influence the size of $\mathcal{J}_i$ and $\tilde{\mathcal{Z}}_i^1$. In our implementation, we found that a setting of iterlim $= 5$ and tol $= 10^{-2}$ achieved a good tradeoff between approximation quality and computational effort.

3.5 Computing An Exact Penalty Parameter

We now present a procedure to check whether ρ^r, the penalty parameter resulting from solving a classical robust problem counterpart (see Section 3.3.2), is large enough for the distributionally robust problem for a radius θ, and a simple iterative procedure to update it if not optimal. Given a penalty parameter ρ, we assume the distributionally robust optimization problem where $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is replaced with $\mathcal{Q}^{\rho}(\boldsymbol{x}, \boldsymbol{\xi})$ is solved using any of the approximations developed in Section 3.4.

1. Start with $\hat{\rho} \leftarrow \rho^r$. Pick $a > 1$.

2. Solve

$$\min_{\boldsymbol{x} \in X, \alpha \geq 0} c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N} \sum_{i \in [N]} \max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}^{\hat{\rho}}(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \right\} \tag{3.5.1}$$

   using any of the lift-and-project approximations described in Section 3.4. Record solutions $(\hat{\boldsymbol{x}}, \hat{\alpha})$, and let $\hat{z}_i = \max_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}^{\hat{\rho}}(\hat{\boldsymbol{x}}, \boldsymbol{\xi}) - \hat{\alpha} \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \right\}$ for $i \in [N]$.

3. For each $i \in [N]$, solve

$$\begin{aligned} \max_{\boldsymbol{\xi}, \boldsymbol{\beta}, \rho} \quad & \boldsymbol{h}(\hat{\boldsymbol{x}})^\top \lambda + \mathbf{e}^\top \boldsymbol{\beta} - \mathbf{e}^\top \boldsymbol{\mu} - \hat{\alpha} \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \\ \text{s.t.} \quad & \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi} - \boldsymbol{W}_0^\top \boldsymbol{\lambda} \in \mathcal{Y}^*, \\ & \rho \mathbf{e} - 2\boldsymbol{\beta} + \boldsymbol{T}_0^\top \boldsymbol{\lambda} + \boldsymbol{\mu} \in \mathbb{R}^M_+, \\ & \boldsymbol{\xi} \in \Xi, \\ & \xi_j = 0 \Rightarrow \beta_j = 0, \qquad \forall j \in [M], \\ & \xi_j = 1 \Rightarrow \beta_j = \rho, \qquad \forall j \in [M], \\ & \rho, \boldsymbol{\lambda}, \boldsymbol{\mu} \geq 0. \end{aligned} \tag{3.5.2}$$

   Let $\hat{z}'_i$ be the optimal value for each $i \in [N]$.

4. If $\hat{z}_i \neq \hat{z}'_i$ for some i, set $\hat{\rho} \leftarrow a\hat{\rho}$ and go to step 2. Otherwise, exit.

In step 3, (3.5.2) corresponds to the inner maximization over Ξ in (3.5.1) where ρ is also a decision variable. Here, the second-stage problem $Q^{\rho}(\boldsymbol{x}, \boldsymbol{\xi})$ is dualized (recall $\mathcal{Y}^*$ is the dual cone of $\mathcal{Y}$), and the bilinear terms resulting from $\rho\boldsymbol{\xi}$ are handled via additional variables $\boldsymbol{\beta}$ and indicator constraints. Note that if $\hat{z}_i = \hat{z}'_i$ for all $i \in [N]$ in step 3, then $\hat{\rho}$ is large enough and we can exit. Otherwise, we pick $a\hat{\rho}$ as a new candidate, where $a > 1$. We obtain a penalty parameter $\hat{\rho}$ in $\mathcal{O}\left(\log_a\left(\frac{\hat{\rho}}{\rho^r}\right)\right)$ iterations. This procedure ensures that $\hat{\rho}$ is large enough for the convexified distributionally robust problem, and thus ensures the approximations are an upper-bound on the true optimal value of (3.2.1). This can similarly be applied for the case where we have indicator constraints in the second stage as in Corollary 2.

This procedure might not be practical, however, as it requires the solution of $2N$ MICPs at each iteration. To this end, we simply use ρ^r as a practical approximate penalty parameter in our experiments. As noted in Section 3.3.2, ρ^r can be efficiently computed by solving a deterministic problem (see Proposition 4 and preceding discussion). In our experiments, ρ^r is indeed an exact penalty parameter in all instances where we performed the above test, i.e. ρ^r is large enough such that the penalty reformulation is exact.

3.6 Computational Experiments

We illustrate the applicability of our method to operational problems in electric power systems in Section 3.6.1, and to design problems in multi-commodity flow networks in Section 3.6.2. Our goals are to: *(i)* study the lift-and-project approximations $\mathcal{Z}^0$ and $\mathcal{Z}^1$ in terms of their computational effort and ability to approximate $\operatorname{cl}\operatorname{conv}(\mathcal{Z})$; *(ii)* compare their out-of-sample performance with the standard sample average approximation and with classical two-stage robust optimization; and, *(iii)* elucidate the effect of two key parameters on the relative benefits of the two-stage distributionally robust problem (3.2.1) over these classical formulations: the "rareness" of network failures and the relative magnitude of "impact" when failures occur.

Our code was implemented in Julia 1.5.3, using JuMP 0.21.4. We used Mosek 9.2 for solving our lift-and-project approximations, and Gurobi 9.1.1 as the solver for the Benders and column-and-constraint generation schemes (which we compare in Sections 3.6.1 and 3.6.2 respectively), since the latter performed better than the former in solving the mixed-integer subproblems in those schemes; whereas Mosek performed better in solving the conic programming relaxations. All runs were conducted on an Intel Xeon 2.3 GHz computer, with a limit of four cores per run.

3.6.1 Optimal Power Flow

We use our method to address the security-constrained optimal power flow problem that is fundamental to the secure operation of electric power grids and solved every fifteen minutes or so by grid operators (e.g.,see [112, 113]). The goal is to determine voltages and generation levels of available generators so as to satisfy power demand in the network, while adhering to various physical and engineering constraints. For example, electric power between network nodes (also known as buses) can flow only along capacitated edges or transmission lines. As such, the latter are failure prone, and transmission line outages can lead to an unstable power network or even complete system failure, resulting in costly blackouts. However, such high-impact failure events are rare. For example, between the years 2000 and 2014, fewer than 1,500 power outages have occurred that affected 50,000 or more residents in the entire United States, which is fewer than 100 events per year [114]. This rarity complicates the accurate estimation of their underlying distribution.

Because electric power is governed by complex physical laws, optimal power flow is a highly nonlinear optimization problem. Nevertheless, the underlying physics can be approximated well by using second-order cone or semidefinite programming relaxations [115]. Although our method generalizes to any convex cone relaxation, we focus on the standard second-order cone relaxation [116], where X is second-order cone representable and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is the optimal value of a second-order cone program.

Our presentation of the first-stage model closely follows [116], whereas the second-stage model is inspired by [117]. Conceptually, the first-stage problem determines minimum cost power generation levels assuming no line outages. Upon line failure, the second-stage model seeks to adjust the power generation levels subject to physical constraints where failed lines cannot be used, with a goal of minimizing the total penalty cost of violating power balances. This model satisfies assumptions (A1), (A2), and (A3) and allows the use of the penalty reformulation (3.3.4b-ρ), which also has the advantage of using fewer variables and constraints compared with the linearized reformulation (see Section 3.3.3).

The operational state of transmission lines is modeled as a random binary vector $\boldsymbol{\xi}$ with support $\Xi = \{0,1\}^M$, where $\xi_l = 1$ indicates that line l has failed. In particular, since $\boldsymbol{\xi}$ represent on/off switches, we can use Corollary 2 to get not only a smaller MICP formulation but also tighter values of the penalty parameter $\rho = \rho^r$. The latter is computed by solving the classical robust counterpart as discussed in Section 3.3.2, which reduces to a deterministic problem (see Proposition 4); indeed, the second stage trivially attains its worst-case value when each component of $\boldsymbol{\xi}$ is one, that is, when all transmission lines fail.

Formulation

Let $\mathcal{G}, \mathcal{B}$, and $\mathcal{M}$ be the set of generators, buses, and transmission lines, respectively, and let $\mathcal{G}_i$ be the set of generators associated with bus $i \in \mathcal{B}$. We define $\delta(i) := \{j \in \mathcal{B} : (i,j) \in \mathcal{M} \text{ or } (j,i) \in \mathcal{M}\}$ to be the set of neighbors of bus $i \in \mathcal{B}$. Let p^g_k and q^g_k be the real and reactive power output of generator $k \in \mathcal{G}$, respectively, with lower and upper bounds denoted by $p^{\min}_k, p^{\max}_k$ and $q^{\min}_k, q_k^{max}$. We assume a linear cost c_k of power generation for generator $k \in \mathcal{G}$. Real load and reactive load at bus $i \in \mathcal{B}$ are denoted by p^d_i and q^d_i, respectively, and are known data. Let p^F_{ij} and q^F_{ij} be the real and reactive power flow on line (i,j), respectively, defined for $(i,j) \in \mathcal{M}$ and $(j,i) \in \mathcal{M}$, with line rating limit $f^{\max}_{ij}$ (note that $f^{\max}_{ij} = f^{\max}_{ji}$). Let $\boldsymbol{Y}$ be the $|B| \times |B|$ complex-valued nodal admittance matrix, whose components are $Y_{ij} = G_{ij} + \mathrm{i}B_{ij}$, where $\mathrm{i} = \sqrt{-1}$ is the unit imaginery number, and where G_{ij} and B_{ij} are the conductance and susceptance of line $(i,j) \in \mathcal{M}$, respectively (see [118] for details on computing $\boldsymbol{Y}$). We denote the real and imaginary parts of the complex voltage by e_i and f_i, respectively. As in [116], we define new variables such that $c_{ii} = e_i^2 + f_i^2$, $c_{ij} = e_i e_j + f_i f_j$ and $s_{ij} = e_i f_j - e_j f_i$. We define $\tilde{\boldsymbol{\xi}}$ to be a random binary vector with support $\Xi = \{0, 1\}^{|\mathcal{M}|}$, where $\tilde{\xi}_{ij} = 1$ if line $(i, j) \in \mathcal{M}$ fails and 0 otherwise. We have

$$\boldsymbol{x} = \left(\boldsymbol{p^g},\ \boldsymbol{q^g},\ \boldsymbol{p^F},\ \boldsymbol{q^F},\ \boldsymbol{c},\ \boldsymbol{s},\ \boldsymbol{\sigma^{p+}},\ \boldsymbol{\sigma^{p-}},\ \boldsymbol{\sigma^{q+}},\ \boldsymbol{\sigma^{q-}}\right)$$

as first-stage variables and

$$\boldsymbol{y} = \left(\tilde{\boldsymbol{\delta}},\ \tilde{\boldsymbol{p}}^{\boldsymbol{g}},\ \tilde{\boldsymbol{q}}^{\boldsymbol{g}},\ \tilde{\boldsymbol{p}}^{\boldsymbol{F}},\ \tilde{\boldsymbol{q}}^{\boldsymbol{F}},\ \tilde{\boldsymbol{c}},\ \tilde{\boldsymbol{s}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{p+}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{p-}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{q+}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{q-}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{pF}},\ \tilde{\boldsymbol{\sigma}}^{\boldsymbol{qF}}\right)$$

as second-stage variables. The two-stage model can be written as follows:

$$\min_{\boldsymbol{x}} \sum_{k \in \mathcal{G}} c_k p_k^g + \sum_{i \in \mathcal{B}} g_i \left(\sigma_i^{p+} + \sigma_i^{p-} + \sigma_i^{q+} + \sigma_i^{q-}\right) + \mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{p^g}, \tilde{\boldsymbol{\xi}})\right]$$

$$\text{s.t.} \quad \sum_{k \in \mathcal{G}_i} p_k^g - p_i^d + \sigma_i^{p+} - \sigma_i^{p-} = g_{ii} c_{ii} + \sum_{j \in \delta(i)} p_{ij}^F, \qquad \forall i \in \mathcal{B}, \qquad (3.6.1a)$$

$$\sum_{k \in \mathcal{G}_i} q_k^g - q_i^d + \sigma_i^{q+} - \sigma_i^{q-} = -b_{ii} c_{ii} + \sum_{j \in \delta(i)} q_{ij}^F, \qquad \forall i \in \mathcal{B}, \qquad (3.6.1b)$$

$$p_{ij}^F = -G_{ij} c_{ii} + G_{ij} c_{ij} + B_{ij} s_{ij}, \qquad \forall (i,j), (j,i) \in \mathcal{M}, \qquad (3.6.1c)$$

$$q_{ij}^F = B_{ij} c_{ii} - B_{ij} c_{ij} + G_{ij} s_{ij}, \qquad \forall (i,j), (j,i) \in \mathcal{M}, \qquad (3.6.1d)$$

$$c_{ij} = c_{ji},\ s_{ij} = -s_{ji}, \qquad \forall (i,j) \in \mathcal{M}, \qquad (3.6.1e)$$

$$c_{ij}^2 + s_{ij}^2 + \left(\frac{c_{ii} - c_{jj}}{2}\right)^2 \leq \left(\frac{c_{ii} + c_{jj}}{2}\right)^2, \qquad \forall (i,j) \in \mathcal{M}, \qquad (3.6.1f)$$

$$\underline{V}_i^2 \leq c_{ii} \leq \bar{V}_i^2, \qquad \forall i \in \mathcal{B}, \qquad (3.6.1g)$$

$$p_k^{\min} \le p_k^g \le p_k^{\max}, \qquad \forall k \in \mathcal{G}, \qquad (3.6.1h)$$

$$q_k^{\min} \le q_k^g \le q_k^{\max}, \qquad \forall k \in \mathcal{G}, \qquad (3.6.1i)$$

$$(p_{ij}^F)^2 + (q_{ij}^F)^2 \le (f_{ij}^{\max})^2, \qquad \forall (i,j),(j,i) \in \mathcal{M}, \qquad (3.6.1j)$$

$$\sigma_i^{p+}, \sigma_i^{p-}, \sigma_i^{q+}, \sigma_i^{q-} \ge 0, \qquad \forall i \in \mathcal{B}, \qquad (3.6.1k)$$

where $\mathcal{Q}(\boldsymbol{p^g}, \tilde{\boldsymbol{\xi}})$ is the optimal value of

$$\min_{\boldsymbol{y}} \sum_{i \in \mathcal{B}} g_i \left(\tilde{\sigma}_i^{p+} + \tilde{\sigma}_i^{p-} + \tilde{\sigma}_i^{q+} + \tilde{\sigma}_i^{q-} \right)$$

$$\text{s.t.} \quad \tilde{p}_k^g = p_k^g + \Delta_k \tilde{\delta} \qquad \forall k \in \mathcal{G}, \qquad (3.6.2a)$$

$$\sum_{k \in \mathcal{G}_i} \tilde{p}_k^g - p_i^d + \tilde{\sigma}_i^{p+} - \tilde{\sigma}_i^{p-} = g_{ii}\tilde{c}_{ii} + \sum_{j \in \delta(i)} \tilde{p}_{ij}^F, \qquad \forall i \in \mathcal{B}, \qquad (3.6.2b)$$

$$\sum_{k \in \mathcal{G}_i} \tilde{q}_k^g - q_i^d + \tilde{\sigma}_i^{q+} - \tilde{\sigma}_i^{q-} = -b_{ii}\tilde{c}_{ii} + \sum_{j \in \delta(i)} \tilde{q}_{ij}^F, \qquad \forall i \in \mathcal{B}, \qquad (3.6.2c)$$

$$\tilde{p}_{ij}^F = -G_{ij}\tilde{c}_{ii} + G_{ij}\tilde{c}_{ij} + B_{ij}\tilde{s}_{ij} + \tilde{\sigma}_{ij}^{pF}, \qquad \forall (i,j),(j,i) \in \mathcal{M}, \qquad (3.6.2d)$$

$$\tilde{q}_{ij}^F = B_{ij}\tilde{c}_{ii} - B_{ij}\tilde{c}_{ij} + G_{ij}\tilde{s}_{ij} + \tilde{\sigma}_{ij}^{qF}, \qquad \forall (i,j),(j,i) \in \mathcal{M}, \qquad (3.6.2e)$$

$$\tilde{c}_{ij} = \tilde{c}_{ji}, \; \tilde{s}_{ij} = -\tilde{s}_{ji}, \qquad \forall (i,j) \in \mathcal{M},$$

$$\tilde{c}_{ij}^2 + \tilde{s}_{ij}^2 + \left(\frac{\tilde{c}_{ii} - \tilde{c}_{jj}}{2} \right)^2 \le \left(\frac{\tilde{c}_{ii} + \tilde{c}_{jj}}{2} \right)^2, \qquad \forall (i,j) \in \mathcal{M},$$

$$\underline{V}_i^2 \le \tilde{c}_{ii} \le \bar{V}_i^2, \qquad \forall i \in \mathcal{B},$$

$$p_k^{\min} \le \tilde{p}_k^g \le p_k^{\max}, \qquad \forall k \in \mathcal{G},$$

$$q_k^{\min} \le \tilde{q}_k^g \le q_k^{\max}, \qquad \forall k \in \mathcal{G},$$

$$(\tilde{p}_{ij}^F)^2 + (\tilde{q}_{ij}^F)^2 \le (f_{ij}^{\max})^2, \qquad \forall (i,j),(j,i) \in \mathcal{M},$$

$$\tilde{\xi}_{ij} = 1 \implies \left[\tilde{p}_{ij}^F = 0, \; \tilde{q}_{ij}^F = 0 \right], \qquad \forall (i,j),(j,i) \in \mathcal{M}, \qquad (3.6.2f)$$

$$\tilde{\xi}_{ij} = 0 \implies \left[\tilde{\sigma}_{ij}^{pF} = 0, \; \tilde{\sigma}_{ij}^{qF} = 0 \right], \qquad \forall (i,j),(j,i) \in \mathcal{M}, \qquad (3.6.2g)$$

$$\tilde{\sigma}_i^{p+}, \tilde{\sigma}_i^{p-}, \tilde{\sigma}_i^{q+}, \tilde{\sigma}_i^{q-} \ge 0, \qquad \forall i \in \mathcal{B}.$$

Constraints (3.6.1a) and (3.6.1b) are the real and reactive power balance equations, respectively. Constraints (3.6.1c) and (3.6.1d) define the real and reactive power flow in both directions of all lines, respectively. Constraints (3.6.1e) and (3.6.1f) model the change of variables (see [116] for details), where the latter is the result of convexifying the original constraint $c_{ij}^2 + s_{ij}^2 = c_{ii}c_{jj}$. Constraints (3.6.1g), (3.6.1h) and (3.6.1i) enforce bounds on the voltage magnitude, and the real and reactive power generation, respectively. In the first stage, each generator $k \in \mathcal{G}$ has an associated generation cost c_k, and we penalize violating constraints (3.6.1a) and (3.6.1b) by g_i.

The second stage involves the same constraints with a few modifications. Namely, constraint (3.6.2a) adjusts the first-stage real power generation, where all generators are adjusted by a constant $\tilde{\delta}$, scaled by their predefined automatic generation control participation factor Δ_k, also known as the droop control policy. We set participation factors Δ_k to be proportional to the generation capacity for each generator $k \in \mathcal{G}$. Constraints (3.6.2f) ensure that no power can flow through lines under contingency ($\xi_{ij} = 1$). Note that if line (i, j) fails, then we must have $\tilde{p}_{ij}^F = \tilde{p}_{ji}^F = 0$ and $\tilde{q}_{ij}^F = \tilde{q}_{ji}^F = 0$, but variables $\tilde{c}_{ii}, \tilde{c}_{ij}$ and $\tilde{s}_{ij}$ should not be affected. Thus, unlike in the first stage, slacks $\tilde{\sigma}_{ij}^{pF}$ and $\tilde{\sigma}_{ij}^{qF}$ are added in constraints (3.6.2d) and (3.6.2e), respectively, so that (3.6.2d) and (3.6.2e) become redundant for (i, j) and (j, i) if line (i, j) has failed. These slacks are active only if $\xi_{ij} = 1$, enforced by constraints (3.6.2g). As in the first stage, the absolute values of the real and reactive power balance violation $\tilde{\sigma}_i^{p+} + \tilde{\sigma}_i^{p-}$ and $\tilde{\sigma}_i^{q+} + \tilde{\sigma}_i^{q-}$ for bus $i \in \mathcal{B}$, respectively, are penalized by g_i. Note that there is no cost on power generation in the second stage. We set the cost g_i of violating the balance equations to be $\phi \cdot \max_{k \in \mathcal{G}} c_k$.

Test Instances

We conduct our experiments on the standard IEEE 14-, 30- and 118-bus test cases from the PGLib-OPF library [119]. Note that the choise of ϕ depends on the economic cost of failure to meet power demand. Since loss of power and blackouts tend to be costly and the associated penalty costs much larger than the cost of generation, we set $\phi = 100$ in Sections 3.6.1 and 3.6.1 and analyze its sensitivity in Section 3.6.1.

We generate empirical data using a Bernoulli model. Specifically, we model each component of $\tilde{\boldsymbol{\xi}}$ as independent and identically distributed Bernoulli random variables with parameter ψM^{-1}, where M denotes the number of transmission lines. Note that this choice reflects the rare nature of line failures; in particular, it implies that only $\psi \cdot 100\%$ of training samples record a line failure. We set $\psi = 0.1$ in Sections 3.6.1 and 3.6.1 and analyze its impact in Section 3.6.1.

In all our experiments, for a fixed sample size N and radius ε, we report average results using 100 statistically independent sets of training samples, and we estimate the variance by reporting the standard deviation over these 100 runs. In Sections 3.6.1 and 3.6.1, the out-of-sample performances of a candidate solution are estimated by using 1,000 statistically independent sets of testing samples.

Approximation Quality and Computational Effort

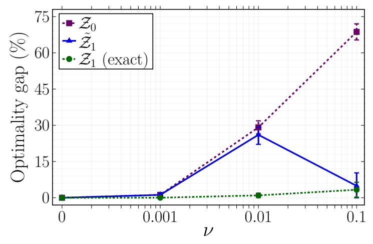
To study the quality of our lift-and-project approximations, we compute the following quantities for each sample size $N \in \{10, 100, 1000\}$ and radius $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$, where $\nu \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$: *(i)* the optimal value $v^\star$ of the convex hull reformulation (3.3.3) using the Benders scheme described in Appendix A.1, and *(ii)* the optimal values v^0, $\tilde{v}^1$ and v^1 of formulation (3.3.3) when the convex hulls $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$ in (3.3.4a)–(3.3.4b) are approximated by using the continuous relaxation $\mathcal{Z}^0$ and heuristically and exactly computed level-1 Lovász-Schrijver relaxations $\tilde{\mathcal{Z}}^1$ and $\mathcal{Z}^1$, respectively (see Section 3.4.2). The choice of the radius $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$ is motivated from Theorem 4, and we elaborate on it further in the next subsection.

Figure 3.1 reports the average (line plot) and standard deviation (error bar) of the optimality gaps, defined as $(v - v^\star)/v^\star \times 100\%$, where $v \in \{v^0, \tilde{v}^1, v^1\}$, based on 100 statistically independent sets of training samples. We make the following observations.

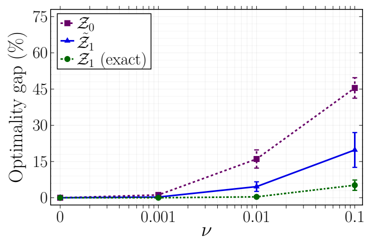
- The exact level-1 relaxation $\mathcal{Z}^1$ is near optimal, with optimality gaps never exceeding 10%, whereas the continuous relaxation $\mathcal{Z}^0$ is less accurate, especially for larger radii (*e.g.*, 50% gap for $\nu = 0.1$). The heuristically computed level-1 relaxation $\tilde{\mathcal{Z}}^1$ is also near optimal for small and large radii but performs relatively poorly for intermediate values of ν.

- For a fixed sample size N and decreasing radius ν, the optimality gaps of all approximations decrease to 0. For increasing ν, the gaps of the level-1 relaxations increase far less rapidly than that of the continuous relaxation.

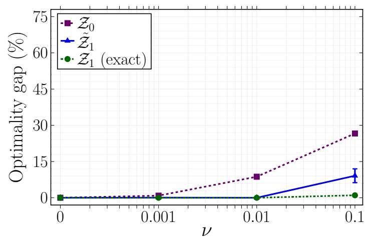
- For a fixed radius ν and increasing sample size N, the gaps of all approximations decrease.



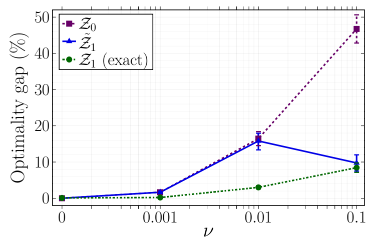
(a) 14-bus, $N = 10$



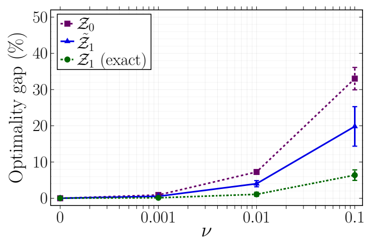
(b) 14-bus, $N = 100$



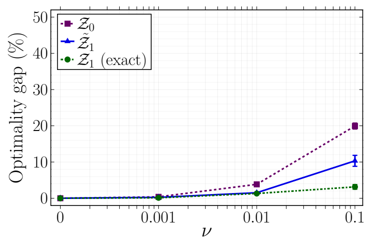
(c) 14-bus, $N = 1000$



(d) 30-bus, $N = 10$



(e) 30-bus, $N = 100$



(f) 30-bus, $N = 1000$

Figure 3.1: Optimality gaps using the continuous relaxation $\mathcal{Z}^0$ and the heuristically and exactly computed level-1 Lovász-Schrijver relaxations $\tilde{\mathcal{Z}}^1$ and $\mathcal{Z}^1$ as a function of ν and N, where $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$.

Figure 3.2 reports the average computation time to solve the various approximations and compares them with that of the Benders decomposition scheme. We observe the following.

- The continuous relaxation $\mathcal{Z}^0$ and heuristically computed level-1 relaxations $\tilde{\mathcal{Z}}^1$ have the smallest computation times, with the former being faster for larger values of N and ν and, in particular, for the larger 118-bus case where it is more than 10 times faster. When compared with the Benders scheme, the relative difference in their computation times is minor for small sample sizes N but increases significantly for large sample sizes. For $N = 1000$, both approximations run 10 times faster than the Benders scheme for the 14-bus case, while $\tilde{\mathcal{Z}}^1$ runs 4 times faster and $\mathcal{Z}^0$ runs almost 100 times faster for the 30-bus case.

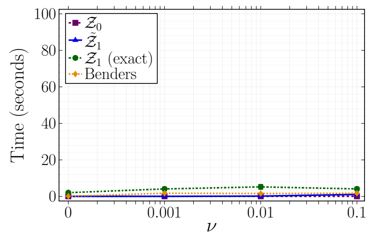
- The exact level-1 relaxation $\mathcal{Z}^1$ and the Benders scheme appear to be the most difficult to solve. Although not shown, for the 30-bus case the former took about 1, 2 and 10 minutes for $N = 10, 100$, and $1,000$, respectively, whereas for the larger 118-bus case neither scheme terminated within 10 minutes for $N = 10, 100$ or within 1 hour for $N = 1000$. Moreover, some of the MICP subproblems within the Benders scheme can cause slow convergence (e.g., due to search tree enumeration). This is evidenced by the fact that about 1% of the Benders runs did not terminate within 10 minutes even for the smaller 14-bus and 30-bus cases, respectively.

- In conjunction with Figure 3.2, the heuristically computed level-1 relaxation $\tilde{\mathcal{Z}}^1$ appears to offer the best tradeoff in terms of approximation quality and computational effort.

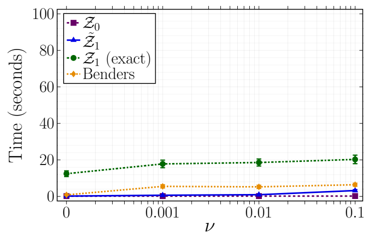
- The run times of all approximations, and in particular $\tilde{\mathcal{Z}}^1$, can be significantly improved by using an efficient initialization of their variables (see Section 3.4.2), which we did not implement.

Out-of-sample Performance and Finite Sample Guarantee

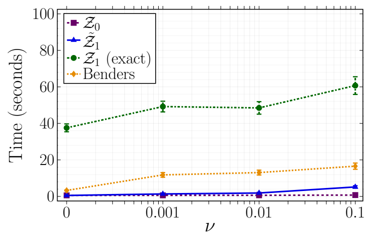
To understand the potential benefits of a distributionally robust approach, we evaluate its out-of-sample performance. For a given training data set of size N and a given choice of ν,



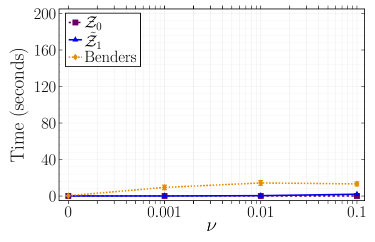
(a) 14-bus, $N = 10$



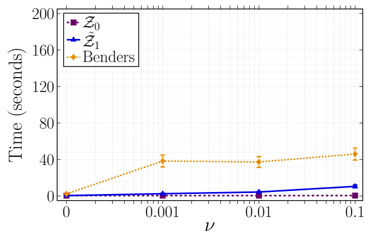
(b) 14-bus, $N = 100$



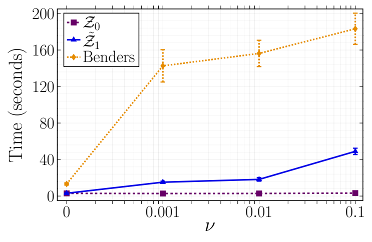
(c) 14-bus, $N = 1000$



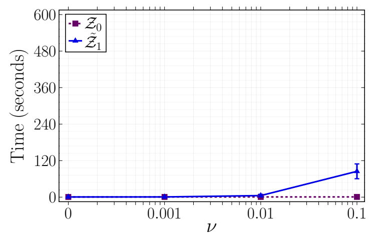
(d) 30-bus, $N = 10$



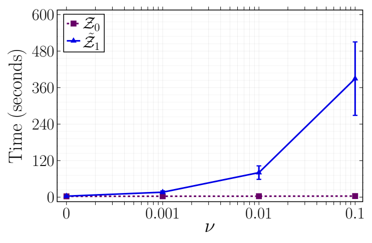
(e) 30-bus, $N = 100$



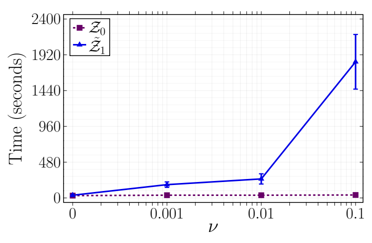
(f) 30-bus, $N = 1000$



(g) 118-bus, $N = 10$



(h) 118-bus, $N = 100$



(i) 118-bus, $N = 1000$

Figure 3.2: Computation times for solving formulation (3.3.3) using the continuous relaxation $\mathcal{Z}^0$, the heuristically computed and exact level-1 Lovász-Schrijver relaxations $\tilde{\mathcal{Z}}^1$ and $\mathcal{Z}^1$, and the Benders decomposition scheme, as a function of ν and N, where $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$.

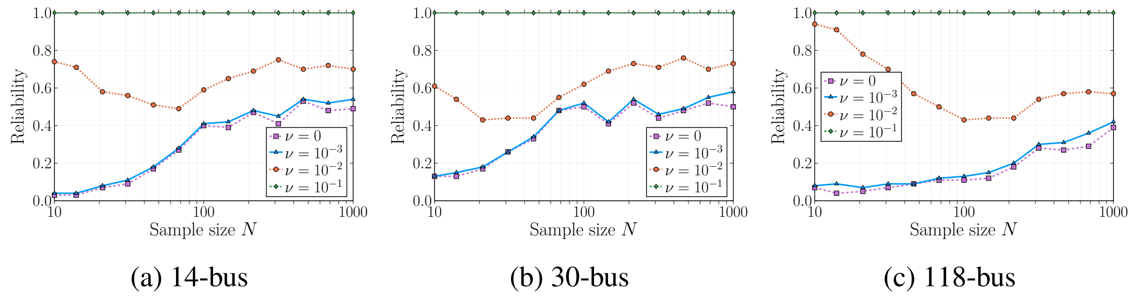
we obtain a candidate first-stage solution $\boldsymbol{x}^\nu$ by solving formulation (3.3.3) with the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ and Wasserstein radius $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$. We then estimate the out-of-sample performance of $\boldsymbol{x}^\nu$ by recording

$$z^\nu = \boldsymbol{c}(\boldsymbol{x}^\nu) + 1000^{-1}\sum_{i=1}^{1000} \mathcal{Q}(\boldsymbol{x}^\nu, \hat{\boldsymbol{\xi}}^i),$$

where $\hat{\boldsymbol{\xi}}^1, \ldots, \hat{\boldsymbol{\xi}}^{1000}$ are $1,000$ independently generated testing samples. This entire process is repeated 100 times for statistically independent sets of N training samples and 1,000 testing samples.

We first justify the choice of the radius $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$ as a function of the training sample size N. This dependence is motivated by inequality (3.2.4) in Theorem 4. However, the latter inequality can be loose, especially when accounting for problem-dependent constants such as the size and diameter of the support Ξ. Therefore, we empirically verify the finite sample guarantee of the first-stage solutions $\boldsymbol{x}^\nu$ under the tighter parameterization $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$ for several choices of the coefficient $\nu \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. Figure 3.3 reports the reliability of $\boldsymbol{x}^\nu$, which we define as the empirical probability (over the 100 sets of training samples) that the optimal value $\tilde{v}^1$ of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ is an upper bound on the out-of-sample cost z^ν.

Figure 3.3 shows that, for fixed values of N, the reliability of $\boldsymbol{x}^\nu$ increases with increasing values of ν, and this can be used to guide the choice of ν. For example, depending on their risk level, decision-makers can select the smallest value of ν with sufficiently high reliability. In particular, for training data sets with small sample size N, observe that $\nu = 10^{-2}$ yields an upper bound on the out-of-sample cost with probability more than 0.5, whereas $\nu = 10^{-1}$ yields an upper bound with probability 1.0. However, note that we cannot always access the true out-of-sample cost (and hence, the true reliability). Nevertheless, one could estimate the out-of-sample cost by using cross validation or the holdout method (e.g, see [18, 98]), and then select a value for the coefficient ν that offers a good trade-off between low out-of-sample cost and high reliability.



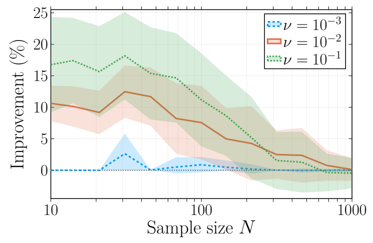
(a) 14-bus (b) 30-bus (c) 118-bus

Figure 3.3: Reliability of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$, as a function of training sample size N.

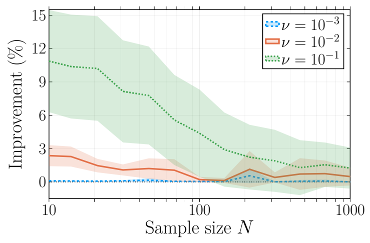
We now evaluate the benefits of our distributionally robust model over the sample average approximation, by computing the relative improvement in out-of-sample cost, which we define as $(z^0 - z^\nu)/z^0 \times 100\%$. Figure 3.4 reports the mean (solid line) and standard deviation (shaded ribbon) of the relative improvement over the 100 independent sets of training samples. We make the following observations from Figure 3.4.

- The distributionally robust model (3.2.1) consistently outperforms the sample average approximation, particularly for small sample sizes N. The magnitude of the relative improvement is instance dependent (roughly 15%, 10%, and 5% for the 14-, 30-, and 118-bus cases, respectively) but consistently decreases for large values of N as expected. The magnitude of the radius that leads to the best possible improvement also is instance dependent.

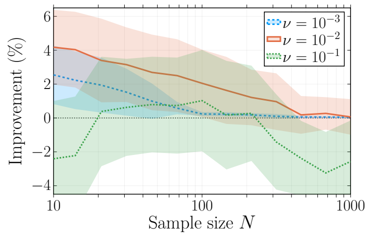
- The larger variances in improvement for smaller N and for larger instances can be partially explained by the combinatorial growth in the number of truly distinct training data sets of size N (i.e., those that lead to distinct first-stage solutions) that are possible under the rare event model of line outages. The large variances for $\nu = 10^{-1}$ can also be similarly explained by the larger number of truly distinct first-stage solutions that can result from slight variations in the training data set.



(a) 14-bus



(b) 30-bus



(c) 118-bus

Figure 3.4: Relative improvement in the out-of-sample performance of the distributionally robust two-stage model (3.2.1) when compared with the sample average approximation, as a function of sample size N.

We now compare the out-of-sample performance of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ corresponding to the best choice of ν (which is 10^{-1} for the 14- and 30-bus cases, and 10^{-2} for the 118-bus case) with *(i)* the continuous relaxation $\mathcal{Z}^0$ for the same ν, and *(ii)* a classical two-stage robust optimization model of the following form:

$$\min_{\boldsymbol{x} \in X} \; c(\boldsymbol{x}) + \max_{\boldsymbol{\xi} \in \Xi_K} \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}), \tag{3.6.3}$$

where $\Xi_K = \{\boldsymbol{\xi} \in \{0,1\}^M : \xi_1 + \ldots + \xi_M \leq K\}$ is the uncertainty set with K being the *budget* of uncertainty. In particular, for each instance, we let $K \in \{0, 1, 2, 5, 10\}$ and obtain optimal first-stage solutions $\boldsymbol{x}^K$ of problem (3.6.3) with the Benders decomposition scheme. As before, we estimate the out-of-sample performance of $\boldsymbol{x}^K$ as $z^K = \boldsymbol{c}(\boldsymbol{x}^K) + 1000^{-1} \sum_{i=1}^{1000} \mathcal{Q}(\boldsymbol{x}^K, \hat{\boldsymbol{\xi}}^i)$, where $\hat{\boldsymbol{\xi}}^1, \ldots, \hat{\boldsymbol{\xi}}^{1000}$ are the same $1,000$ testing samples used to estimate z^ν. We then record the best possible K yielding the lowest z^K, and compute the relative improvement in out-of-sample cost, which we define as $(z^K - z^\nu)/z^\nu \times 100\%$. Figure 3.5 reports the mean (solid line) and standard deviation (shaded ribbon) of the relative improvement over the 100 independent sets of training samples. We make the following observations from Figure 3.5.

- The distributionally robust model strongly outperforms its classical robust counterpart, across all instances, with relative improvements of 5%, 1% and 15% for the 14-, 30- and 118-bus cases, respectively. In contrast to Figure 3.4, the relative improvements increase with increasing values of N; this is expected since the classical robust model (3.6.3) ignores all sample data and therefore, it becomes overly conservative in the presence of a moderate amount of data. Thus, we observe that for small to moderate values of N, the distributionally robust model improves upon both the sample average approximation and classical robust optimization. Finally, although not shown, solving the classical robust model with the Benders scheme took longer than solving the level-1 relaxation, especially for the larger 118-bus case.

- The relative improvement of the level-1 relaxation $\tilde{\mathcal{Z}}^1$ over the continuous relaxation $\mathcal{Z}^0$ is smaller, but can be as high as 4% as seen in the 14-bus case. It should be noted that these quantities are necessarily upper bounded by the improvements over the sample average approximations reported in Figure 3.4, and can be significant for applications including optimal power flow, which are executed several times each day of the year. Finally, we note that the tradeoff between tighter in-sample optimality gaps (see Figure 3.1) and hence stronger finite sample reliability guarantees (see Figure 3.3) offered by the level-1 relaxation $\tilde{\mathcal{Z}}^1$, with the faster computation times for solving the continuous relaxation $\mathcal{Z}^0$ (see Figure 3.2), can guide the design of an algorithmic scheme.

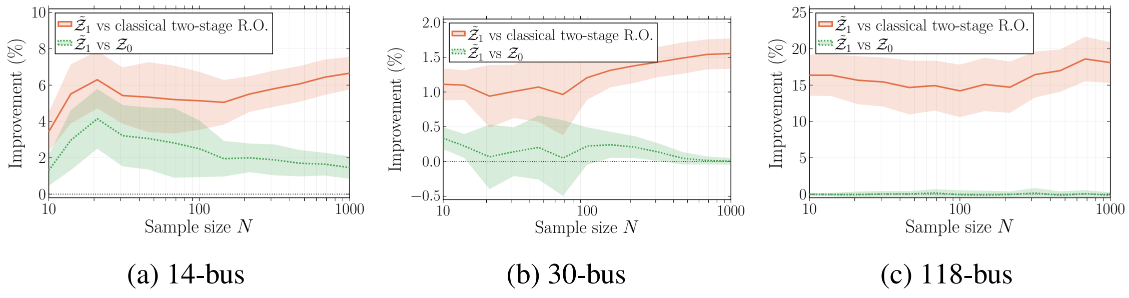


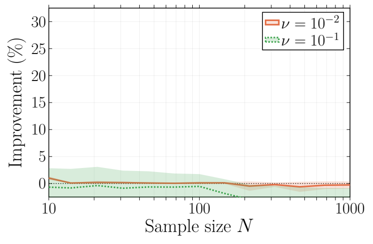
Figure 3.5: Relative improvement in the out-of-sample performance of the distributionally robust two-stage model (3.2.1) solved using the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$, when compared with the classical two-stage robust optimization model (3.6.3), and the continuous relaxation $\mathcal{Z}^0$.

Sensitivity Analysis

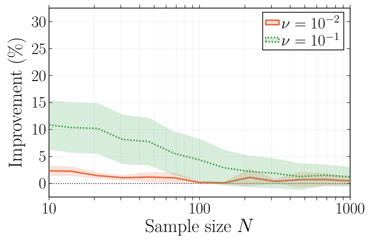
The instance-dependent behavior of the out-of-sample performance from the previous subsection suggests that it might also be influenced by other parameters. Here, we investigate the effect of the "rareness" ψ of transmission line failures and the relative magnitude ϕ of "impact" when failures occur. Recall from Section 3.6.1 that higher values of ψ increase the probability of line failures, whereas higher values of ϕ increase the penalty cost for failing to satisfy power demand due to transmission line failures. Figure 3.6 shows the relative improvement of the distributionally robust two-stage model (3.2.1) over the sample average approximation for various choices of ψ and ϕ. For brevity, we report results only for the 30-bus instance; the high-level insights do not change for other instances. We make the following observations from Figure 3.6.

- For fixed values of the line failure probability ψ, Figures 3.6a–3.6c show that as the impact due to failure ϕ increases, the relative benefits of a distributionally robust approach strongly increase. In other words, benefits increase with higher impacts of failures. Interestingly, Figure 3.6a also shows that if failures are rare but low impact, then ignoring them (as in the sample average approximation) may not incur high out-of-sample costs, even for small values of N.

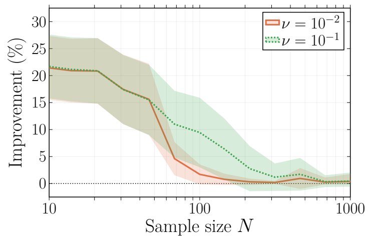
- For fixed values of the magnitude of impact ϕ, Figures 3.6d–3.6f show that as the probability of failures ψ increases, the relative benefits of a distributionally robust approach increases. However, observe that this does not necessarily imply that the relative benefits are small when line failure probabilities are small. Indeed, we observe that the relative benefits remain as high as 10% even when individual line failure probabilities are less than $0.05M^{-1}$.



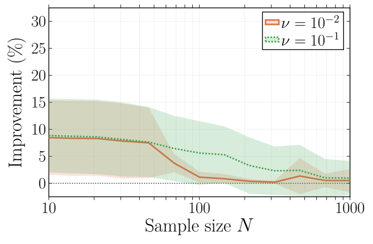
(a) $(\psi, \phi) = (0.1, 50)$



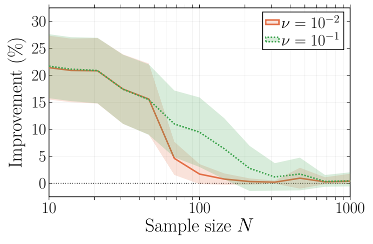
(b) $(\psi, \phi) = (0.1, 100)$



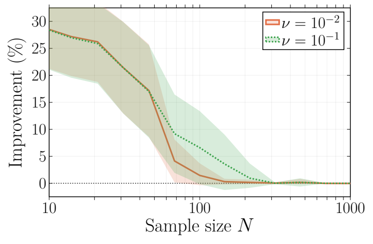
(c) $(\psi, \phi) = (0.1, 200)$



(d) $(\psi, \phi) = (0.05, 200)$



(e) $(\psi, \phi) = (0.1, 200)$



(f) $(\psi, \phi) = (0.2, 200)$

Figure 3.6: Relative improvement in the out-of-sample performance of the distributionally robust two-stage model (3.2.1) when compared with the sample average approximation, for various values of (ψ, ϕ).

3.6.2 Multi-commodity Network Design

We now consider multi-commodity network design problems that have applications in telecommunications, transportation, logistics and production planning, among others (e.g., see [120, 121]). In several of these applications, it is required to send flows to satisfy known demands between multiple origin-destination pairs or commodities. The goal is to minimize the total cost, which is the sum of fixed costs of installing arc capacities and variable costs of routing flows for each commodity. As such, failures of network elements can lead to a reduction in its available flow capacity and a subsequent failure to meet demands. This is particularly true in telecommunication networks where the loss of even a single (typically high-capacity) fiber-optic cable or router equipment can cause a substantial fraction of the overall flow (e.g., internet traffic) to be lost, leading to potentially huge economic impacts [122]. Fortunately, these networks are typically well-engineered and therefore, such high-impact failures are rare. However, the lack of data makes estimating the underlying failure probability difficult.

In the model we consider, the first-stage problem determines the arc capacities that can be used for routing flows. Upon failure, the second-stage model determines the routing of each commodity along the degraded network topology constrained by the first-stage arc capacities, and with the objective of minimizing the sum of variable routing costs and penalty costs for not satisfying demands.

For ease of exposition, we model only node failures, where $\boldsymbol{\xi}$ is supported on $\Xi = \{0,1\}^M$ and $\xi_i = 1$ indicates that node i has failed. Since $\boldsymbol{\xi}$ represent on/off switches, we can employ Corollary 2, and the penalty parameter $\rho = \rho^r$ can be computed as in the optimal power flow problem, where the classical robust counterpart again reduces to a deterministic problem, since the worst-case scenario occurs when each component fails.

Formulation

Our model is based on the fixed-charge multi-commodity network design problem which has been extensively studied in the literature, such as in [123] and [121]. We are given a directed network with nodes $\mathcal{V}$, arcs $\mathcal{A}$, and a set of commodities $\mathcal{K}$ with known demands d^k. For commodity $k \in \mathcal{K}$, let O_k be the origin node and D_k the destination node. Each arc (i, j) has an associated maximum capacity u_{ij}, per-unit cost of installing capacity f_{ij} and per-unit cost of flow c_{ij}. For each node $i \in \mathcal{V}$, we define the set of neighboring nodes incident to outgoing and incoming arcs as $\mathcal{V}^+(i) = \{j \mid (i, j) \in \mathcal{A}\}$ and $\mathcal{V}^-(i) = \{j \mid (j, i) \in \mathcal{A}\}$, respectively.

In the first stage, let x_{ij} be a variable between 0 and 1 which determines the fraction of capacity of arc (i, j) that can be used in the second stage. Note that this differs from a typical fixed-charge multi-commodity network design model, where x_{ij} is a binary variable determining whether arc (i, j) can be used. In the second stage, let y_{ij}^k be the amount of flow of commodity k on arc (i, j), and let σ_k be the unsatisfied demand of commodity k which we penalize by g_k. We consider node failures, and define $\boldsymbol{\xi}$ to be a binary vector where $\xi_i = 1$ indicates that node $i \in \mathcal{V}$ has failed. If a node fails, all arcs incident to it cannot be used. The model can be as written as follows:

$$\begin{aligned} \min_{x} \quad & \sum_{(i,j)\in\mathcal{A}} f_{ij}x_{ij} + \sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}} \left[\mathcal{Q}(\boldsymbol{x}, \tilde{\boldsymbol{\xi}}) \right] \\ \text{s.t.} \quad & x_{ij} \in [0, 1], \ \forall (i, j) \in \mathcal{A}. \end{aligned}$$

where $\mathcal{Q}(\boldsymbol{x}, \tilde{\boldsymbol{\xi}})$ is defined as the optimal objective value of the following linear program:

$$\min_{y,\sigma} \quad \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} c_{ij} y_{ij}^k + \sum_{k\in\mathcal{K}} g_k \sigma_k$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{V}^+(i)} y_{ij}^k - \sum_{j \in \mathcal{V}^-(i)} y_{ji}^k = \begin{cases} d^k - \sigma_k, & \text{if } i = O_k \\ -d^k + \sigma_k, & \text{if } i = D_k \\ 0, & \text{otherwise} \end{cases}, \ \forall k \in \mathcal{K}, \ \forall i \in \mathcal{V}, \tag{3.6.4a}$$

$$\sum_{k \in \mathcal{K}} y_{ij}^k \leq u_{ij} x_{ij}, \qquad \forall (i,j) \in \mathcal{A}, \tag{3.6.4b}$$

$$y_{ij}^k \leq d^k(1 - \tilde{\xi}_i), \qquad \forall (i,j) \in \mathcal{A}, \ \forall k \in \mathcal{K}, \tag{3.6.4c}$$

$$y_{ij}^k \leq d^k(1 - \tilde{\xi}_j), \qquad \forall (i,j) \in \mathcal{A}, \ \forall k \in \mathcal{K}, \tag{3.6.4d}$$

$$y_{ij}^k \geq 0 \qquad \forall (i,j) \in \mathcal{A}, \ \forall k \in \mathcal{K},$$

$$\sigma_k \geq 0 \qquad \forall k \in \mathcal{K}.$$

In the second stage, constraints (3.6.4a) are typical network flow constraints, with added non-negative variables σ_k which model unsatisfied demand. If we have a positive amount σ_k of unsatisfied demand for commodity k, then an amount of $d^k - \sigma_k$ would leave the origin node and enter the destination node. Note that variables σ_k ensure feasibility in the second stage for any $\boldsymbol{x}$ and $\boldsymbol{\xi}$. For each commodity k, unsatisfied demand σ_k is penalized by g_k. In our experiments, g_k is constant and set to $\phi \cdot \max_{(i,j) \in A} c_{ij}$, where $\phi = 1000$ is a pre-defined non-negative parameter. Constraint (3.6.4b) ensures the total flow on any arc (i, j) does not exceed the available capacity, which is determined by the first-stage decision x_{ij}. Finally, constraints (3.6.4c) and (3.6.4d) ensure there is no flow through arcs incident to a failed node.

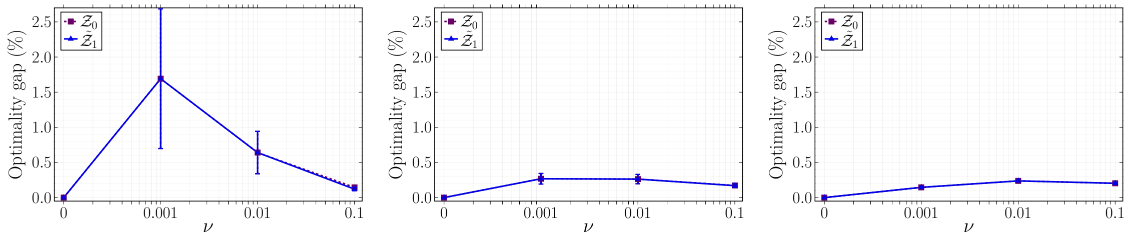
Test Instances

We conduct our experiments on the $(20, 230, 40, V, L)$ instance from the so-called Class I set of instances in [121]. As the name indicates, the instance has 20 nodes, 230 arcs, 40 commodities, and the letters V and L indicate that fixed-costs are relatively low compared to variable costs, and that the problem is loosely capacitated. We generate empirical data by modeling each component of $\tilde{\boldsymbol{\xi}}$ as independent and identically distributed Bernoulli random variables with parameter ψM^{-1}, where $\psi = 0.1$. As before, for a fixed sample size N and radius ε, we report average results using 100 statistically independent sets of training samples, and we estimate the variance by reporting the standard deviation over these 100 runs. The out-of-sample performances of candidate solutions are estimated by using 1,000 statistically independent sets of testing samples.

Approximation Quality and Computational Effort

Similar to Section 3.6.1, we compute the optimality gaps of the convex hull reformulation (3.3.3) when the convex hulls $\operatorname{cl}\operatorname{conv}(\mathcal{Z}_i)$ in (3.3.4a)–(3.3.4b) are approximated by using the continuous relaxation $\mathcal{Z}^0$ and heuristically computed level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$, for sample sizes $N \in \{10, 100, 1000\}$ and radii $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$, $\nu \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. In doing so, the true optimal value of each instance is computed by solving formulation (3.3.2) using the column-and-constraint generation scheme. The mixed-integer subproblems in this scheme are solved by dualizing the second-stage loss function as opposed to using its Karush-Kuhn-Tucker conditions, since it results in fewer additional variables and constraints, see [109]. Any bilinear expressions involving dual variables and uncertain parameters are reformulated using indicator constraints since the lack of *a priori* known upper bounds on the dual variables prohibits direct linearization using McCormick inequalities.

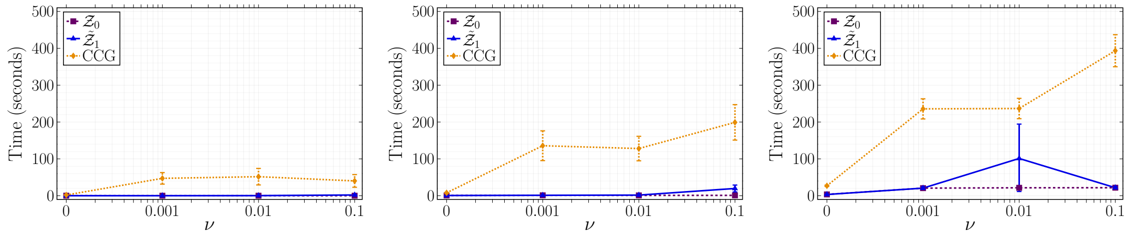
Figure 3.7 reports the average (line plot) and standard deviation (error bar) of the optimality gaps, whereas Figure 3.8 reports the corresponding computation times, based on 100 statistically independent sets of training samples. Figure 3.7 shows that both the continuous and level-1 relaxations provide very similar and near-optimal approximations with optimality gaps never exceeding 3% for $N = 10$ and 0.5% for $N \geq 100$. Interestingly, the optimal first-stage decisions are different, and this can be seen from their out-of-sample performance that we present in the next subsection.



(a) $N = 10$ (b) $N = 100$ (c) $N = 1000$

Figure 3.7: Optimality gaps using the continuous relaxation $\mathcal{Z}^0$ and the heuristically computed level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$, as a function of ν and N where $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$.

Figure 3.8 shows that both relaxations have small computation times (< 60 seconds on average) across all N and ν. When compared with the column-and-constraint generation scheme, the relative difference in their computation times is minor for small sample sizes N but increases significantly for large sample sizes, where the column-and-constraint generation method can be slower by more than a factor of 10. Similar to the Benders scheme, the mixed-integer subproblems in the column-and-constraint generation scheme can cause slow convergence, and roughly 3% of its runs did not terminate within 10 minutes.



(a) $N = 10$ (b) $N = 100$ (c) $N = 1000$

Figure 3.8: Computation times using the continuous $\mathcal{Z}^0$ and heuristically computed level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ for solving formulation (3.3.3), and using the column-and-constraint generation scheme for solving formulation (3.3.2), as a function of ν and N, where $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$

.

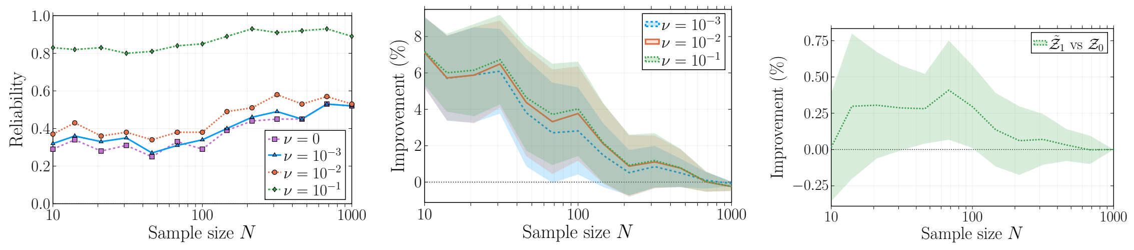
Out-of-sample Performance and Finite Sample Guarantee

Similar to Section 3.6.1, we estimate the out-of-sample performance of the first-stage solutions of the lift-and-project and sample average approximations for different sample sizes N and Wasserstein radii $\varepsilon = \nu\sqrt{N^{-1}\log(N+1)}$, where $\nu \in \{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. The results are summarized in Figure 3.9.

First, Figure 3.9a reports the reliability of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$, which is the empirical probability (over the 100 sets of training samples) that its optimal value is an upper bound on its out-of-sample cost. We observe that the reliability increases not only with increasing values of ν (for fixed values of N) but also with increasing values of N (for fixed values of ν). The choice $\nu = 10^{-1}$ is reliable with probability 0.8 for training data sets of small size N and this increases to > 0.9 for large values of N.

Figure 3.9b reports the relative improvement in out-of-sample cost of the distributionally robust model over the sample average approximation, over the 100 independent sets of training samples. As before, we find that the distributionally robust model consistently outperforms the sample average approximation, particularly for small sample sizes N, where the magnitude of the relative improvement can be as high as 7%, but decreases for large values of N.

Finally, Figure 3.9c reports the improvement in the out-of-sample performance of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ for the best choice of $\nu = 10^{-1}$, relative to the continuous relaxation $\mathcal{Z}^0$ for the same ν. The relative improvement of the level-1 relaxation $\tilde{\mathcal{Z}}^1$ over the continuous relaxation $\mathcal{Z}^0$ is small yet consistently non-negative. This is not unexpected since the latter already has tight in-sample optimality gaps, as can be seen from Figure 3.7. Nevertheless, we expect the relative improvements to be instance-dependent similar to optimal power flow, and even a few percentage points can result in long-term economic benefits.



(a) Reliability (b) Out-of-sample performance (c) Relative improvement

Figure 3.9: Reliability (left plot), and relative improvements in the out-of-sample performance of the level-1 Lovász-Schrijver relaxation $\tilde{\mathcal{Z}}^1$ when compared with the sample average approximation (middle plot), and the continuous relaxation $\mathcal{Z}^0$ (right plot), as a function of training sample size N.

3.7 Conclusions

In this chapter, we address applications affected by rare high impact zero-one uncertainties common in network optimization problems. Due to the rarity of failure events, historical data is often not rich enough to reliably estimate failure probabilities, making traditional methods potentially lead to poor out-of-sample performance. To this end, we tackle a two-stage distributionally robust model under the Wasserstein ambiguity set with zero-one uncertainties, and provide means to efficiently solve it. By leveraging ideas from penalty methods and lift-and-project techniques, we provide a simple, tractable and tight approximation that can be iteratively improved. We perform extensive computational experiments on two important applications, notably the optimal power flow problem with random transmission line failures and a multi-commodity network design problem with random node failures. We find that the method can strongly outperform classical sample average and robust optimization approaches, especially when failures are rare but can lead to high costs associated with loss of electric power or commodity flows.

CHAPTER 4
BINARY DISTRIBUTIONALLY ROBUST OPTIMIZATION UNDER THE WASSERSTEIN SET: A DISASTER RELIEF APPLICATION

4.1 Introduction

In this chapter, we turn our attention to two-stage distributionally robust optimization (DRO) under the Wasserstein set where binary variables are present in both the first and second stage. More specifically, we are motivated by a two-stage network design problem, where the first stage is a facility location problem, and the second stage is a fixed-charge transportation problem. Such models naturally arise in disaster relief operations, where the first stage consists of opening facilities, and pre-allocating resources such as medical kits or food supplies before observing a disaster, and the second stage consists of allocating resources from facilities to affected areas after a disaster. Natural disaster events occur relatively rarely, and the scarcity of data makes it difficult to reliably estimate the underlying probability distribution of the uncertainty [36, 37], especially in developing countries where infrastructure to collect data is limited. The great majority of the literature has been focused on solving either a stochastic program or robust optimization model for the problem of prepositining and routing resources to affected areas after a disaster. Surveys of recent work on disaster relief operations can be found in [124] and [125], where the latter is focused on two-stage stochastic programming solutions. A case study on hurricane threats on the coastal states of the United States along the Gulf of Mexico and the Atlantic Ocean is presented and solved in [126]. The same case study has been used in [36, 37, 127]. In [37], a two-stage robust model is solved. It is only very recently that attention has been drawn to using a distributionally robust optimization paradigm. In [36], and in [127], the authors solve a two-stage DRO model under a moment based ambiguity set, and a two-stage DRO model under a box and polyderal ambiguity set.

In this work, due to the limited data available in natural disaster management and the difficulty in estimating the underlying distribution, we are again focused on the Wasserstein ambiguity set to leverage its finite sample and theoretical out-of-sample performance guarantees [18]. As mentioned, DRO permits us to be risk averse towards the empirical distribution obtained from historical data. Recall that in two-stage DRO (TSDRO), we seek to minimize the first-stage cost and the worst-case expected value of the second-stage cost with respect to probability distributions belonging to an ambiguity set. We re-write the definition of TSDRO and the Wasserstein ambiguity set for convenience. In this chapter, we assume the support set Ξ is a finite, yet large, set of scenarios indexed by $s \in \mathcal{S}$. Given N samples $\{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}\}_{n\in\mathcal{N}}$ indexed by $\mathcal{N} \subseteq \mathcal{S}$, we define the empirical distribution as $\hat{\mathbb{P}}_n = \frac{1}{N}\sum_{n\in\mathcal{N}} \delta_{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}}$, where $\delta_{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}}$ is the dirac distribution assigning unit mass to $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$, i.e. each sample is assigned equal probability. The TSDRO problem can be written as

$$\min_{\boldsymbol{x}\in X} \quad \boldsymbol{c}^\top \boldsymbol{x} + \max_{\mathbb{P}\in\mathcal{B}(\mathbb{P}_N,\theta)} \mathbb{E}_{\mathbb{P}}\left[\mathcal{Q}(\boldsymbol{x},\boldsymbol{\xi})\right] \tag{DRO}$$

where $\mathbb{B}_W(\mathbb{P}_N,\theta)$ is the Wasserstein ball of radius $\theta \geq 0$ centered at the empirical distribution $\mathbb{P}_N$. The Wasserstein ball is defined as

$$\mathcal{B}_W(\mathbb{P}_N,\theta) = \{\mathbb{P} \in \mathcal{M}(\Xi) : \mathrm{d}_W(\mathbb{P},\mathbb{P}_N) \leq \theta\}$$

where $\mathrm{d}_W(\mathbb{P},\mathbb{P}_N)$ is the Wasserstein distance between $\mathbb{P}$ and $\mathbb{P}_N$ defined in (W) in Section 1.3.1.

Relatively limited work explore the case where integer variables are present in the second stage for DRO under the Wasserstein set. In [108], a Benders decomposition approach is used to solve a two-stage distributionally robust unit-commitment problem with binary variables over a Wasserstein ball. Kibaek Kim extends the classical dual decomposition method to DRO models under the Wasserstein set with mixed-integer variables to solve the

Lagrangian dual in [128]. In [26], an extension of the integer L-shaped method is proposed to solve TSDRO models with binary first-stage decisions and mixed-binary second-stage decisions under the Wasserstein set. In [30], the authors develop a cutting plane procedure to solve two-stage mixed-integer conic programs under convex ambiguity sets. In two-stage robust optimization, the K-adaptability algorithm was developed to handle second-stage binary variables in [129] and second-stage mixed-integer variables in [130]. The K-adaptability algorithm was extended to two-stage distributionally robust optimization with moment-based ambiguity sets in [131].

The remaining of the chapter is organized as follows. We first summarize our contributions in Section 4.1.1 and define notation in Section 4.1.2. In Section 4.2, we formally define our two-stage model and support set Ξ. In Section 4.3, we study the structure of our model and support set, and present a column-and-constraint generation (CCG) algorithm and the algorithm used to generate new scenarios. In Section 4.4, we discuss the numerical implications of our method, and how to tackle computational challenges, leading to an improved CCG algorithm. Finally, we present our computational experiments in Section 4.5.

4.1.1 Contributions

We summarize our contributions as follows:

· To the best of our knowledge, this is the first work to consider a Wasserstein ball in a two-stage DRO formulation with binary variables in the second stage for disaster relief operations, providing greater modeling power and handling a richer ambiguity set. By using a Wasserstein ambiguity set, we do not need to make any assumptions on the underlying distribution of the random parameters or need to estimate moments.

· By constructing a support set and defining an underlying distance metric of the support that is tailored to disaster management, we develop an efficient column-and-constraint generation algorithm, where we leverage the structure of our support set and second-stage value function.

· We show the conditions under which the second-stage value function is concave with respect to a subset of the uncertainty, leading to an efficient line search scheme to generate scenarios, and how these results extend to the case of a fixed-charge network flow second-stage problem.

· We perform extensive computational experiments, showing the computational efficiency of our method on synthetic instances with very large sets of scenarios, and illustrate our methods on the case study of hurricane threats on the Gulf of Mexico states, analyzing the solutions obtained from DRO and SAA.

Although our methods and results are not specific to hurricane disasters, we present our work in the context of hurricane threats.

4.1.2 Notation

Vectors are printed in bold to differentiate them from scalars. Given a finite support set of scenarios Ξ with an associated index set $\mathcal{S}$, we equivalently refer to both $\boldsymbol{\xi} \in \Xi$ and $s \in \mathcal{S}$ as scenarios. Similarly, for a set of samples of $\{\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}\}_{n\in\mathcal{N}} \subseteq \Xi$ indexed by $\mathcal{N} \subseteq \mathcal{S}$, we refer to $n \in \mathcal{N}$ and $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$ as samples. To differentiate between a random vector $\boldsymbol{\xi}$ and historical data, we denote samples by a hat as in $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$. For a support set Ξ, we define $\mathcal{M}(\Xi)$ to be the set of all probability distributions supported on Ξ and $\mathrm{d}(\cdot,\cdot)$ to be a valid underlying metric of Ξ.

4.2 Formulation

4.2.1 Two-stage Distributionally Robust Model

Given a network with nodes $\mathcal{V}$, let $I \subseteq \mathcal{V}$ be the set of possible facility locations, let $J \subseteq \mathcal{V}$ be the set of possible demand nodes, and let K be the set of commodities. In the first stage, binary variable o_i is 1 if facility $i \in I$ is opened at a cost of h_i, and continous variables z_{ik} represent the amount of commodity $k \in K$ pre-allocated to facility i at a unit cost of g_k. Each facility $i \in I$ has a known capacity of C_i, and a unit of commodity $k \in K$ requires a capacity of v_k.

In scenario $s \in \mathcal{S}$ of the second stage, let w^s_{ij} be the fixed-charge cost incurred if link (i, j) is used, let c^s_{kij} be the per-unit cost of transportation of commodity k on (i, j), and let d^s_{kj} be the demand of commodity k at node $j \in J$, for any facility $i \in I$ and demand node $j \in J$. We define the random vector $\boldsymbol{\xi^s}$ to be the concatenation of vectors $\boldsymbol{w^s}$, $\boldsymbol{c^s}$ and $\boldsymbol{d^s}$. We have binary variables σ_{ij} determining whether facility $i \in I$ is serving demand node $j \in J$ and non-negative variables t^k_{ij} which represent the amount of commodity k transported from facility i to demand node j. Any unmet demand is stored in $\boldsymbol{u}$ and is penalized by U. This ensures that the second-stage problem is feasible for any first-stage decision and any realization of the uncertainty. Moreover, we assume we know upper bounds M_{ij} on the amount that can be transported from facility i to demand node j. For example, M_{ij} could be the population at demand node $j \in J$.

Finally, we include a feasible region B in the first stage which can incorporate additional constraints such as available budget, or the maximum number of facilities that can be opened; in the second stage, C can include additional valid inequalities. B and C can be chosen to be empty. The TSDRO model is defined as

$$\begin{aligned}
\min_{\boldsymbol{o}, \boldsymbol{z}} \quad & \sum_{i \in I} \left[h_i o_i + \sum_{k \in K} g_k z_{ik} \right] + \max_{p \in \mathcal{B}_W(\mathbb{P}_N, \theta)} \sum_{s \in \mathcal{S}} p_s \mathcal{Q}(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{\xi^s}) \\
\text{s.t.} \quad & \sum_{k \in K} v_k z_{ik} \le C_i o_i, \forall i \in I, && \text{(4.2.1a)} \\
& z_{ik} \ge 0, \qquad \forall i \in I, \forall k \in K, \\
& o_i \in \{0, 1\}, \qquad \forall i \in I, \\
& (\boldsymbol{o}, \boldsymbol{z}) \in B
\end{aligned}$$

where the second-stage cost $\mathcal{Q}(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{\xi^s})$ is

$$
\begin{aligned}
\min_{\boldsymbol{\sigma}, \boldsymbol{t}, \boldsymbol{u}} \quad & \sum_{i \in I} \sum_{j \in J} \left[w^s_{ij} \sigma_{ij} + \sum_{k \in K} c^s_{kij} t^k_{ij} \right] + U \sum_{j \in J} \sum_{k \in K} u^k_j \\
\text{s.t.} \quad & \sum_{j \in J} t^k_{ij} \le z_{ik}, \qquad \forall i \in I,\ \forall k \in K, && \text{(4.2.2a)} \\
& \sum_{i \in I} t^k_{ij} + u^k_j \ge d^s_{kj}, \forall j \in J,\ \forall k \in K, && \text{(4.2.2b)} \\
& t^k_{ij} \le M_{ij} \sigma_{ij}, \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K, && \text{(4.2.2c)} \\
& t^k_{ij} \ge 0, \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K, \\
& \sigma_{ij} \in \{0, 1\}, \qquad \forall i \in I,\ \forall j \in J, \\
& u^k_j \ge 0, \qquad \forall j \in J,\ \forall k \in K, \\
& (\boldsymbol{\sigma}, \boldsymbol{t}, \boldsymbol{u}) \in C.
\end{aligned}
$$

If B includes a budget constraint, then $B = \{(\boldsymbol{o}, \boldsymbol{z}) : \sum_{i \in I} \left[o_i x_i + \sum_{k \in K} g_k z_{ik} \right] \le b\}$, where b is a pre-defined budget in the first stage. The set B could also include a limit on the number of facilties that can be built in the first stage. We include valid inequalities $\sigma_{ij} \le o_i$ in C for all $i \in I$ and $j \in J$.

Constraint (4.2.1a) ensures we do not exceed the capacity of each facility i. Constraint (4.2.2a) ensures we can only transport what is available from facility i, constraint (4.2.2b) ensures demand satisfaction with additional variables u^k_j to keep track of unmet demand, and constraint (4.2.2c) ensures we only use open links (i, j). We note that the second-stage problem is always feasible for all first-stage decisions $(\boldsymbol{o}, \boldsymbol{z})$ and random vectors $\boldsymbol{\xi}$.

Throughout the rest of the chapter, we assume uncertainty is only present in the constraints, such that $\boldsymbol{w}^s = \boldsymbol{w}$ and $\boldsymbol{c}^s = \boldsymbol{c}$ for all $s \in \mathcal{S}$.

4.2.2 Support Set

To construct the finite support set Ξ, we need to define a set of demand scenarios. Since the objective of DRO is to protect the decision-maker against potentially unobserved events, constructing a set of scenarios using historical data is undesirable, especially when data is limited. On the other hand, constructing an overly complicated support set could lead to unrealistic disaster scenarios. To this end, we define a set of scenarios such that they are descriptive of the disaster, and lead to a manageable support set. Constructing such a support set has modeling and algorithmic benefits as we will show.

Each scenario is defined by components which determine which nodes are affected and have a positive demand, and others which determine the intensity of the disaster. More specifically, we redefine our random vector $\boldsymbol{\xi}$ to be comprised of four components: the landfall of the disaster, the radius of impact, the path of the disaster, and the fraction of the population that is affected at the landfall node. Here, the first three components determine which nodes are affected by the disaster, and the last component determines the demand and thus the intensity at the landfall node. In any scenario, we assume that there can only be one landfall. Conditioned on the demand at the landfall node, we assume that the demands at the remaining affected nodes are deterministic, where the fraction of the population affected decreases the farther the node is from the landfall.

Let L be the set of landfalls, R be the set of radii of impact, A be the set of paths represented by angles, and let F be the set of possible fractions that can affect a population (i.e. demand). A scenario is constructed in a hierarchical fashion, where we first observe a two-dimensional vector $\boldsymbol{\xi_\ell}$ representing the longitude and latitude of the landfall node, followed by a radius of impact $\xi_r \in R$ around the landfall. Although $\boldsymbol{\xi_\ell}$ is a vector of the landfall nodes's coordinates, we shall mostly refer to $\boldsymbol{\xi_\ell}$ as a node in $\mathcal{V}$ for simplicity. The angle $\xi_a \in A$ determines the portion of nodes within the radius ξ_r which are affected by the disaster. ξ_a can, for example, represent the slice of the circle that is affected. Finally, we observe the fraction of the population affected $\xi_f \in F$ at the landfall node, which in turn determines the demand at the other affected nodes. More specifically, let $J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a) = \{\boldsymbol{\xi_\ell}, j_1, j_2, ...\} \subseteq J$ be the set of affected nodes as determined by the landfall $\boldsymbol{\xi_\ell}$, the radius of impact ξ_r and angle ξ_a, and assume nodes are in increasing order of distance from the landfall. Note that the landfall $\boldsymbol{\xi_\ell}$ is included in the set. We define functions $\Pi_j^k : L \times R \times A \times F \to [0, 1]$ which map the fraction of the population affected at the landfall node ξ_f to the fraction of the population needing commodity $k \in K$ at nodes $j \in J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$, such that $\Pi_{\boldsymbol{\xi_\ell}}^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) \geq \Pi_{j_1}^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) \geq \Pi_{j_2}^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) \geq \dots$ for all k. We allow the functions Π_j^k to be constant for all or any subset of the first three components. Note that conditioned on ξ_f, the demands at the affected nodes that are not the landfall are deterministic, and how they are determined is a modeling question. We give more specific details in Section 4.3.3.

We then have $\boldsymbol{\xi} = (\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f)$ as a random vector. Given a scenario $\boldsymbol{\xi}$, we implicitly have an associated demand vector $\boldsymbol{d}$ that is fed to model (4.2.2), and thus, only $\boldsymbol{d}$ appears in the constraints. We give an example of two scenarios with the same landfall and radius of impact but with different angles in Figure 4.1. This network was used in a case study of hurricane threats in the Gulf of Mexico states in [126] and is the case study on which we illustrate our method in Section 4.5.3.

In TSDRO under the Wasserstein ambiguity set, the model hedges against scenarios by transporting weights from the scenarios in the sample set to other scenarios, possibly not in the sample set (i.e. unobserved scenarios) at a cost proportional to the distance between them defined by $\mathrm{d}(\cdot, \cdot)$, subject to a budget θ, the radius of the Wasserstein ball (see discussion in Section 1.3.1). Our support set permits us to better measure dissimilarities between two scenarios by defining an underlying metric $\mathrm{d}(\cdot, \cdot)$ of Ξ that is tailored to the application. We define the distance between two scenarios $\boldsymbol{\xi^1}$ and $\boldsymbol{\xi^2}$ as the sum of squared deviations between the components of $\boldsymbol{\xi}$:

$$\mathrm{d}(\boldsymbol{\xi^1}, \boldsymbol{\xi^2}) = ||\boldsymbol{\xi_\ell^1} - \boldsymbol{\xi_\ell^2}||_2^2 + (\xi_r^1 - \xi_r^2)^2 + (\xi_a^1 - \xi_a^2)^2 + (\xi_f^1 - \xi_f^2)^2 \tag{4.2.3}$$

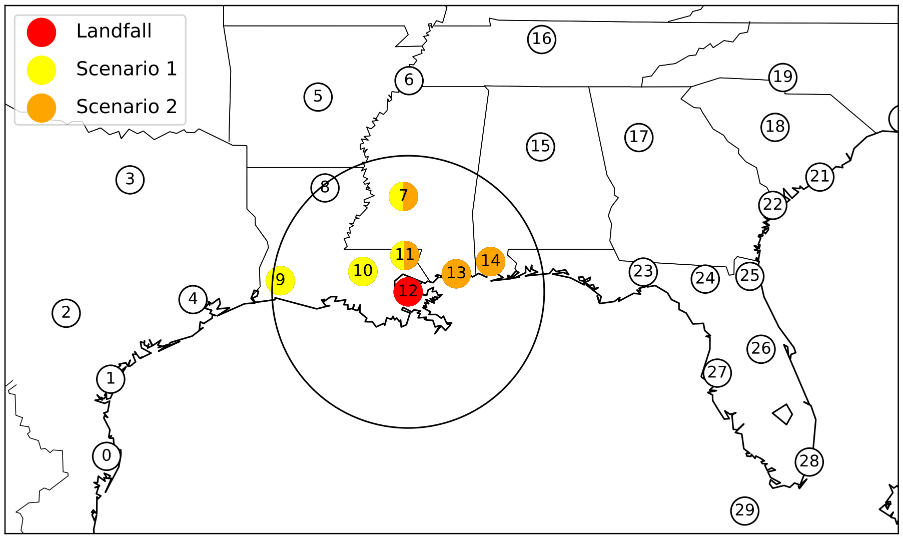


Figure 4.1: Example of two different angles ξ_a, where the landfall is New Orleans (node 12) and the radius is 320 km. Nodes filled with both colors are nodes affected in both scenarios.

Recall $\boldsymbol{\xi}_{\ell}$ is a two-dimensional vector representation of the coordinates, while the other components are scalars.

Consider the case where the support set is a set of demand vectors. Such random vectors can do a poor job at measuring dissimilarity between scenarios. As an example, consider the following hypothetical three scenarios in the network of Figure 4.1, where in the first scenario $\boldsymbol{\xi}^{\mathbf{1}}$, node 29 has a demand of 1000, in the second scenario $\boldsymbol{\xi}^{\mathbf{2}}$ node 28 has a demand of 1000, and node 12 has a demand of 1000 in the third scenario $\boldsymbol{\xi}^{\mathbf{3}}$, while all other nodes have a demand of 0. In this case, the distance between $\boldsymbol{\xi}^{\mathbf{1}}$ and $\boldsymbol{\xi}^{\mathbf{2}}$ is equal to the distance between $\boldsymbol{\xi}^{\mathbf{1}}$ and $\boldsymbol{\xi}^{\mathbf{3}}$, even though it is clear that we should have $\mathrm{d}(\boldsymbol{\xi}^{\mathbf{1}}, \boldsymbol{\xi}^{\mathbf{3}}) > \mathrm{d}(\boldsymbol{\xi}^{\mathbf{1}}, \boldsymbol{\xi}^{\mathbf{2}})$. This is because nodes 28 and 29 are close to each other while node 12 is farther away. By using the metric defined in (4.2.3), the model considers disaster scenarios hitting landfalls that are close to each other or having similar characteristics (radius, path and intensity) to be similar scenarios.

4.3 Column-and-Constraint Generation Algorithm

In this section, we present our column-and-constraint generation algorithm (CCG). We first present a known extensive reformulation of (4.2.1), then describe how we leverage the structure of our support set and second-stage value function.

4.3.1 Extensive Reformulation

To simplify notation, we concatenate all first-stage variables under the vector $\boldsymbol{x}$, and all second-stage variables under the vector $\boldsymbol{y}$. Let X be the feasible region of the first stage and let $Y(\boldsymbol{x}, \boldsymbol{\xi})$ be the feasible region of the second stage given $\boldsymbol{x}$ and random vector $\boldsymbol{\xi}$. More specifically, we work with (DRO), where $Q(\boldsymbol{x}, \boldsymbol{\xi}) = \min_{\boldsymbol{y}} \left\{\boldsymbol{q}^\top \boldsymbol{y} : \boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi})\right\}$.

From Theorem 1, we have that (4.2.1) is equivalent to

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + \theta\alpha + \frac{1}{N}\sum_{n\in\mathcal{N}} \gamma_n \\ \text{s.t.} \quad & \gamma_n \geq \boldsymbol{q}^\top \boldsymbol{y}^{\boldsymbol{s}} - \alpha\, \mathrm{d}(\boldsymbol{\xi}^{\boldsymbol{s}}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}), \; s \in \mathcal{S}, n \in \mathcal{N}, \\ & \boldsymbol{y}^{\boldsymbol{s}} \in Y(\boldsymbol{x}, \boldsymbol{\xi}^{\boldsymbol{s}}), \qquad s \in \mathcal{S}, \\ & \boldsymbol{x} \in X, \\ & \alpha \geq 0. \end{aligned} \tag{ER}$$

Due to the hierarchical nature of the scenarios, there is an obvious ordering of the scenarios with respect to the second-stage cost. For example, for the same landfall and direction, increasing the radius and/or intensity at the landfall leads to a greater second-stage cost. We can reduce the number of scenarios to consider in the model and get an equivalent problem as follows. First, we define the following set for each sample:

$$\bar{\Xi}_n = \{\boldsymbol{\xi} \in \Xi : \; (\xi_f < \hat{\xi}_f,\; \xi_r < \hat{\xi}_r) \text{ or } (\xi_f = \hat{\xi}_f,\; \xi_r < \hat{\xi}_r) \text{ or } (\xi_f < \hat{\xi}_f,\; \xi_r = \hat{\xi}_r).\}$$

We then define sets of scenarios Ξ_n for each sample $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}} = (\hat{\boldsymbol{\xi}}_{\boldsymbol{\ell}}^{\boldsymbol{n}}, \hat{\xi}_r^n, \hat{\xi}_a^n, \hat{\xi}_f^n)$ as:

$$\Xi_n = \Xi \setminus \bar{\Xi}_n.$$

The sets Ξ_n exclude scenarios where both intensity and radius are strictly smaller than the sample in question, and scenarios where one of the two is strictly smaller and the other is the same. For each Ξ_n, we define an associated index set $\mathcal{S}_n$ and define the set of all scenarios as $\mathcal{S}_{\mathcal{N}} = \bigcup_{n\in\mathcal{N}} \mathcal{S}_n$.

Theorem 8. (ER) *is equivalent to*

$$\begin{aligned}
\min \quad & \boldsymbol{c}^\top \boldsymbol{x} + \theta\alpha + \frac{1}{N}\sum_{n\in\mathcal{N}} \gamma_n \\
\textit{s.t.} \quad & \gamma_n \geq \boldsymbol{q}^\top \boldsymbol{y}^{\boldsymbol{s}} - \alpha\, \mathrm{d}(\boldsymbol{\xi}^{\boldsymbol{s}}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}), \quad && s \in \mathcal{S}_n, n \in \mathcal{N}, \quad && \textit{(c1)}, \\
& \boldsymbol{y}^{\boldsymbol{s}} \in Y(\boldsymbol{x}, \boldsymbol{\xi}^{\boldsymbol{s}}), && s \in \mathcal{S}_{\mathcal{N}}, && \textit{(c2)}, \\
& \boldsymbol{x} \in X, \\
& \alpha \geq 0.
\end{aligned} \tag{ER'}$$

The sets Ξ_n exclude dominated scenarios which need not be included in the model. For a sample $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$, each scenario $\boldsymbol{\xi} \in \bar{\Xi}_n$ has a corresponding scenario $\boldsymbol{\xi'} \in \Xi_n$ which dominates $\boldsymbol{\xi}$, where $\mathrm{d}(\hat{\boldsymbol{\xi}}, \boldsymbol{\xi'}) < \mathrm{d}(\hat{\boldsymbol{\xi}}, \boldsymbol{\xi})$ and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi'}) \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$. In other words, moving weight to $\boldsymbol{\xi'}$ would lead to a greater cost in the inner maximization of (4.2.1) while using less of the available budget θ in the Wasserstein ambiguity set. Note that the reduction in the number of scenarios included in the model directly depends on the sample set. We prove Theorem 8.

Proof. Recall from the proof of Theorem 1, we can write the inner maximization as

$$\begin{aligned}
\max_{\boldsymbol{q}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N} q_s^n \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi^s}) \\
\text{s.t.} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} \frac{1}{N}\,\mathrm{d}(\boldsymbol{\xi^s}, \boldsymbol{\hat{\xi}^n}) q_s^n \leq \theta, \\
& \sum_{s\in\mathcal{S}} q_s^n = 1 \qquad\qquad \forall n \in \mathcal{N}, \\
& q_s^n \geq 0, \qquad\qquad \forall s \in \mathcal{S},\ \forall n \in \mathcal{N}.
\end{aligned} \tag{4.3.1}$$

where q_s^n is the conditional probability of scenario $\boldsymbol{\xi^s}$ given we have observed $\boldsymbol{\hat{\xi}^n}$, the sampled scenario.

In (4.3.1), we are moving weight q_s^n from a sample $\boldsymbol{\hat{\xi}^n}$ to a scenario $\boldsymbol{\xi^s}$ for all s and n, with the objective of maximizing the expected cost of the second stage. Consider a sample $\boldsymbol{\hat{\xi}^n}$ and a scenario $\boldsymbol{\xi} \in \bar{\Xi}_n$. We show that there exists a scenario $\boldsymbol{\xi'}$ which dominates $\boldsymbol{\xi}$, in the sense that $\mathrm{d}(\boldsymbol{\hat{\xi}}, \boldsymbol{\xi'}) < \mathrm{d}(\boldsymbol{\hat{\xi}}, \boldsymbol{\xi})$ and $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi'}) \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$. Let $\boldsymbol{\xi'} = (\boldsymbol{\xi_\ell}, \hat{\xi}_r^n, \xi_a, \hat{\xi}_f^n)$. Then we have $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi'}) \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ for all $\boldsymbol{x} \in X$, since $\xi_f \leq \hat{\xi}_f^n$ and $\xi_r \leq \hat{\xi}_r^n$, where at least one of the two inequalities holds strictly, and $\boldsymbol{\xi_\ell} = \boldsymbol{\xi'_\ell}$ and $\xi_a = \xi'_a$. Moreover, we have $\mathrm{d}(\boldsymbol{\hat{\xi}^n}, \boldsymbol{\xi'}) < \mathrm{d}(\boldsymbol{\hat{\xi}^n}, \boldsymbol{\xi})$ since:

$$\begin{aligned}
\mathrm{d}(\boldsymbol{\hat{\xi}^n}, \boldsymbol{\xi'}) &= ||\boldsymbol{\hat{\xi}^n_\ell} - \boldsymbol{\xi_\ell}||_2^2 + (\xi_a - \xi_a)^2 \\
\mathrm{d}(\boldsymbol{\hat{\xi}^n}, \boldsymbol{\xi}) &= ||\boldsymbol{\hat{\xi}^n_\ell} - \boldsymbol{\xi_\ell}||_2^2 + (\hat{\xi}_r^n - \xi_r)^2 + (\hat{\xi}_a^n - \xi_a)^2 + (\hat{\xi}_f^n - \xi_f)^2
\end{aligned}$$

where one of $(\hat{\xi}_r^n - \xi_r)^2$ or $(\hat{\xi}_f^n - \xi_f)^2$ must be non-zero because $\boldsymbol{\xi} \in \bar{\Xi}_n$. □

4.3.2 Column & Constraint Generation

Since the number of scenarios $|\mathcal{S}_\mathcal{N}|$ can be very large, solving (ER') is computationally infeasible. As in typical CCG algorithms, we start with a subset of scenarios $\hat{\mathcal{S}}_n \subseteq \mathcal{S}_n$ for each sample n, solve a restricted model which we denote as (RER) and record optimal solutions $\hat{\boldsymbol{x}}, \hat{\boldsymbol{\gamma}}$ and $\hat{\alpha}$. For each sample $n \in \mathcal{N}$, we then search for a new scenario $s \in \mathcal{S}_n \backslash \hat{\mathcal{S}}_n$ such that $\hat{\gamma}_n < \mathcal{Q}(\hat{\boldsymbol{x}}, \boldsymbol{\xi}^{\boldsymbol{s}}) - \hat{\alpha}\, \mathrm{d}(\boldsymbol{\xi}^{\boldsymbol{s}}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}})$, where $\mathcal{Q}(\hat{\boldsymbol{x}}, \boldsymbol{\xi}^{\boldsymbol{s}})$ is obtained by solving the second-stage problem given first-stage decision $\hat{\boldsymbol{x}}$ and random vector $\boldsymbol{\xi}^{\boldsymbol{s}}$. Note that we can generate multiple scenarios per iteration. We repeat the process until we can no longer find such a scenario for any sample. We refer to the process of generating new scenarios as the separation problem. With a slight abuse of language, for some sample n, we say that a scenario s violates (c1) if constraint (c1) is violated at scenario s.

We summarize a vanilla CCG algorithm in Algorithm 2. The function `SEPARATION(`$\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}, \hat{\boldsymbol{x}}, \hat{\gamma}_n, \hat{\alpha}$`)` returns a set $V_n \in \mathcal{S}_n \setminus \hat{\mathcal{S}}_n$ of new scenarios which violate $\hat{\gamma}_n \geq Q(\hat{\boldsymbol{x}}, \boldsymbol{\xi}^{s}) - \hat{\alpha}\, \mathrm{d}(\boldsymbol{\xi}^{s}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}})$ for a sample n and optimal solutions $\hat{\boldsymbol{x}}, \hat{\boldsymbol{\gamma}}$ and $\hat{\alpha}$ of (RER), or an empty set if no such scenario is found.

Algorithm 2 CCG

```
1: Initialize $\hat{\mathcal{S}}_n$ for each $n \in \mathcal{N}$
2: while $\cup_{n\in\mathcal{N}} V_n \neq \emptyset$ do
3:     Solve (RER) and record optimal solutions $(\hat{\boldsymbol{x}}, \hat{\gamma}_n, \hat{\alpha})$.
4:     for each $n \in \mathcal{N}$ do
5:         $V_n \leftarrow$ SEPARATION($\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{\gamma}}, \hat{\alpha}$)
6:         $\hat{\mathcal{S}}_n \leftarrow \hat{\mathcal{S}}_n \cup V_n$
7:     end for
8: end while
```

4.3.3 Separation Problem

There are two main difficulties which arise in Algorithm 2. The first, more significant one, is in the separation step, where one might have to enumerate all scenarios in $\mathcal{S}_n \setminus \hat{\mathcal{S}}_n$, which is a very large set. Recall that for each scenario, we must solve a fixed-charge transportation problem (4.2.2). While today's commercial solvers can typically efficiently solve such models, the difficulty lies in repeatedly solving (4.2.2) a large number of times, and having to do it after each solve of (RER), the restricted model of (ER'). The second main difficulty is in solving (RER) after having generated many scenarios. After each iteration, the number of scenarios generated, if not controlled, can be large. Consider the case where each sample generates one new scenario. For each scenario s generated for a sample n, we add the associated constraint (c1), a new set of variables y^s, and a set of second-stage constraints (c2). Depending on the number of samples, this can lead to a large and difficult to solve master problem.

Given optimal solutions $\hat{\boldsymbol{x}}$ and $\hat{\alpha}$ of (RER), the separation problem consists of solving the following problem for each sample n:

$$\begin{aligned} \max_{\boldsymbol{\xi}} \quad & z_n(\boldsymbol{\xi}) := Q(\hat{\boldsymbol{x}}, \boldsymbol{\xi}) - \hat{\alpha}\, \mathrm{d}(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}) \\ \text{s.t.} \quad & \boldsymbol{\xi} \in \Xi \end{aligned} \tag{SP_n}$$

Let $\boldsymbol{\xi}^{\boldsymbol{s}*}$ be the optimal solution and let z_n^* be the optimal value of (SP_n). If $z_n^* > \hat{\gamma}_n$, then we generate new scenario s^*; otherwise, we conclude none of the scenarios violate (c1) for sample n.

One simple greedy approach is to enumerate the scenarios, and stop at the first one which violates constraint (c1). We repeat this for each sample n. Such a separation method generally leads to quickly generating new scenarios, especially in the early stages of the algorithm, but typically leads to a greater number of outer iterations (i.e. master solves). Moreover, as the algorithm progresses, we enumerate more and more scenarios before finding one which violates (c1), and in the worst case, could end up enumerating all scenarios for some of the samples. Alternatively, we can solve (SP_n) by enumerating all scenarios and pick the scenario with the greatest objective $z_n(\boldsymbol{\xi})$.

We instead leverage the structure of our support set Ξ and the optimal value of the second stage $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ to solve (SP_n). Recall that $J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ is the set of affected nodes determined by the landfall $\boldsymbol{\xi_\ell}$, radius of impact ξ_r and angle ξ_a, including the landfall $\boldsymbol{\xi_\ell}$. We make the following two assumptions:

Assumption 1. *The functions $\Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) : L \times R \times A \times F \to [0, 1]$ determining the fraction of the population needing commodity k at nodes $j \in J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ are concave in*

ξ_f for each landfall $\boldsymbol{\xi_\ell} \in L$, radius of impact $\xi_r \in R$, and angle $\xi_a \in A$.

Assumption 2. *A facility at node $i \in I$ has a limit k_i on the number of nodes it can serve, but is able to satisfy any amount of demand at nodes it is serving.*

Assumption 1 ensures we have well-behaved mappings Π_j^k. An example of a function satisfying Assumption 1 is the piece-wise linear function $\Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) = \min\{a\xi_f, \bar{\xi}_{fj}\}$ which sets upper bounds $\bar{\xi}_{fj}$ on each demand node j, where $a \leq 1$ is a pre-determined parameter. The upper bounds might depend on $\boldsymbol{\xi_\ell}$, ξ_r, ξ_a and $k \in K$, but we drop this dependence in $\bar{\xi}_{fj}$ to simplify notation. Since the intensity of the disaster decreases as we move farther from $\boldsymbol{\xi_\ell}$, upper bounds $\bar{\xi}_{fj}$ also decrease the farther j is from $\boldsymbol{\xi_\ell}$.

Under Assumption 2, we no longer pre-allocate resources in the first stage and thus, variables z_{ik} are removed from the model. The upper bounds k_i can either be a pre-defined parameter, or a first-stage decision replacing z_{ik}. Variables k_i can be interpreted as the size of the facility built at node $i \in I$. In the case where k_i is a pre-determined parameter, (4.2.1) becomes:

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & \sum_{i \in I} h_i o_i + \max_{p \in \mathcal{B}_W(\mathbb{P}_N, \theta)} \sum_{s \in \mathcal{S}} p_s Q(\boldsymbol{o}, \boldsymbol{\xi^s}) \\
\text{s.t.} \quad & o_i \in \{0,1\}, \forall i \in I, \\
& \boldsymbol{o} \in B
\end{aligned} \tag{4.3.2}$$

where the second-stage cost $Q(\boldsymbol{o}, \boldsymbol{\xi^s})$ is

$$\begin{aligned}
\min_{\boldsymbol{\sigma}, \boldsymbol{t}, \boldsymbol{u}} \quad & \sum_{i \in I} \sum_{j \in J} \left[w_{ij}\sigma_{ij} + \sum_{k \in K} c_{kij} t_{ij}^k \right] + U \sum_{j \in J} \sum_{k \in K} u_j^k \\
\text{s.t.} \quad & \sum_{j \in J} \sigma_{ij} \leq k_i, \qquad \forall i \in I, \\
& \sum_{i \in I} t_{ij}^k + u_j^k \geq d_{kj}^s, \forall j \in J, \forall k \in K, \\
& t_{ij}^k \leq M_{ij}\sigma_{ij}, \qquad \forall i \in I, \forall j \in J, \forall k \in K, \\
& \sigma_{ij} \leq o_i \qquad \forall i \in I, \forall j \in J,
\end{aligned} \tag{4.3.3a}$$

$$
\begin{aligned}
& t_{ij}^k \geq 0, && \forall i \in I,\ \forall j \in J,\ \forall k \in K, \\
& \sigma_{ij} \in \{0,1\}, && \forall i \in I,\ \forall j \in J, \\
& u_j^k \geq 0, && \forall j \in J,\ \forall k \in K, \\
& (\boldsymbol{\sigma}, \boldsymbol{t}, \boldsymbol{u}) \in C.
\end{aligned}
$$

Constraint (4.3.3a) ensures each facility $i \in I$ only serves at most k_i demand nodes $j \in J$. Note that if a demand node cannot be served, the unsatisfied demand is still penalized by U.

Assumptions 1 and 2 are key to exploiting the structure of the second-stage optimal cost $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ and lead to the following results.

Theorem 9. *Under assumptions 1 and 2, for any first-stage decision vector $\boldsymbol{x}$, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ is concave with respect to ξ_f for a fixed landfall $\boldsymbol{\xi_\ell}$, radius of impact ξ_r and angle ξ_a.*

Corollary 3. *Under assumptions 1 and 2, for any first-stage decision vector $\boldsymbol{x}$ and $\alpha \geq 0$, $z_n(\boldsymbol{\xi})$ is concave with respect to ξ_f for a fixed landfall $\boldsymbol{\xi_\ell}$, radius of impact ξ_r and angle ξ_a.*

Proof. We prove both Theorem 9 and Corollary 3. First note that each demand node is served by at most one facility. Given a set of open facilities, let $a = \{i_1, i_2, \dots\}$ be a set of assignments from facilities to nodes with positive demand, where i_j is the facility assigned to node j, and let A be the set of all possible assignments between facilities and demand nodes satisfying the cardinality constraints (4.3.3a). Let $\hat{J}$ be the set of demand nodes in $J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ that are not served by a facility in a set of assignments a. Let P_j be the population at node j. Define $f(\xi_f)$ to be the objective of the separation problem $z_n(\xi)$ as a function of ξ_f, where $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a are fixed.

First note that as we increase ξ_f for a fixed set of assignments $a \in A$, the fixed-charge cost component remains constant at $w_{i_j j}$, and the transportation cost is equal to $c_{k i_j j} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j$ for each served node j and commodity $k \in K$. For nodes that are not served, the cost is $U \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j$. The objective value of the second stage is then

$$\sum_{j \in J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a) \setminus \hat{J}} \left[w_{ijj} + \sum_{k \in K} c_{kijj} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \right] + U \sum_{j \in \hat{J}} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j. \quad (4.3.4)$$

Note that (4.3.4) is concave with respect to ξ_f by Assumption 1. We then have

$$\begin{aligned} f(\xi_f) &= Q(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha \, \mathrm{d}(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}) \\ &= \min_{a \in A} \Bigg\{ \sum_{j \in J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a) \setminus \hat{J}} \left[w_{ijj} + \sum_{k \in K} c_{kijj} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \right] \\ &\qquad\qquad + U \sum_{j \in \hat{J}} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \Bigg\} - \alpha \, \mathrm{d}(\boldsymbol{\xi}, \hat{\xi}^n), \end{aligned}$$

where the first term is the minimum of concave functions and is therefore concave with respect to ξ_f, and $-\alpha \, \mathrm{d}(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}})$ is concave since $\alpha \geq 0$. We conclude that $f(\xi_f)$ is concave.

□

Theorem 9 and its corollary motivate performing a line search on ξ_f. Since F is a discrete set, we perform a Fibonacci search on F, which is the discrete version of a golden-section line search. More specifically, we enumerate and fix $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a, and solve $\max_{\xi_f \in F} f(\xi_f)$ using a Fibonacci search. At each step of the Fibonacci search, we compute $f(\xi_f)$ by solving the second-stage problem $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$. This separation scheme, which we refer to as `SEPARATION-FIB`$\Big(\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}, \hat{\boldsymbol{x}}, \hat{\gamma}_n, \hat{\alpha}\Big)$, is described in Algorithm 3. The Fibonacci search `FIBONACCI`$\Big(F, \boldsymbol{\xi_\ell}, \xi_r, \xi_a, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}\Big)$ takes as arguments the set of fractions F, the values of the fixed components of $\boldsymbol{\xi}$ and the sample $\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}$. It is presented in Algorithm 4.

In Algorithm 3, we enumerate components $(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ in $L \times R \times A$. In the inner loop, we apply Theorem 8 and only consider components ξ_f such that $\boldsymbol{\xi} \in \Xi_n$, and exclude scenarios which are already included in (RER) to get a subset $\hat{F} \subseteq F$. If $\hat{F}$ is not empty, we call the Fibonacci search function in line 12 to get the scenario ξ^0 which maximizes $f(\xi_f)$. We keep track of the greatest value of z and the maximizing scenario by z^* and

$\boldsymbol{\xi}^*$, respectively. These are updated in lines 14-17. `SEPARATION-FIB` either returns the index associated with $\boldsymbol{\xi}^*$ if that scenario violates (c1), or returns an empty set otherwise.

Details on the Fibonacci search can be found in [132]. We detail our implementation of the Fibonacci search for self-containment. In Algorithm 4, we define Ω to be the array of scenarios in increasing order of $\xi_f \in F$ and an array Θ of Fibonacci numbers up to the q^{th} Fibonacci number, such that $q \geq |\Omega| - 1$. Our sequence of Fibonacci numbers starts with 0, 1 and 1. Let $m = |\Theta|$ be the size of this array. We index the i^{th} element of Ω and Θ by Ω_i and Θ_i, and start the indexing at 0. In lines 4-8, we handle the corner cases where there are only one or two scenarios in Ω, in which case it is a simple check. Throughout the search, we keep track of three variables: ℓ, i_a and u, where $\ell \leq i_a \leq u$. These are indices of Ω which represent intervals where the maximum might be. The three variables are Fibonacci numbers and the idea is to maintain a constant ratio of the interval sizes, such that $\frac{u-i_a}{i_a-\ell}$ is close to the golden-ratio. In order to search the intervals, we define a fourth index i_b between i_a and u. We then compare $z_n(\Omega_{i_a})$ to $z_n(\Omega_{i_b})$. If $z_n(\Omega_{i_a}) > z_n(\Omega_{i_b})$, then we know the maximum is somewhere between ℓ and i_b, otherwise, it is between i_a and u. We update the indices accordingly in lines 13-21. In order to maintain the same interval ratios as the algorithm progresses, i_b is also chosen to be a Fibonacci number where $i_b = \ell + u - i_a$. We repeat this process until $i_a = i_b$. Note that this will find the maximum if it is not at one of the initial endpoints (i.e. 0 or Θ_{m-1}). If either $\ell = 0$ or $u = \Theta_{m-1}$ after exiting the while-loop, we must do a manual check against i_a. This is done in lines 23-29.

Remark 3. *`SEPARATION-FIB` terminates in $\mathcal{O}(\log |F|)$ steps. Thus, to solve (SP_n), the second-stage problem is solved $\mathcal{O}(|L| \times |R| \times |A| \times \log |F|)$ times, as opposed to $\mathcal{O}(|L| \times |R| \times |A| \times |F|)$ times if enumerating all scenarios.*

Since F is essentially the demand component of our random vector, we expect it to have the highest cardinality compared to L, R, and A. In fact, since two-stage DRO under the Wasserstein set aims to protect the decision-maker against unobserved scenarios, a finer discretization of F might be desired. By using `SEPARATION-FIB`, the separation step

Algorithm 3 SEPARATION-FIB$\Big(\hat{\xi}^n, \hat{\boldsymbol{x}}, \hat{\gamma}_n, \hat{\alpha}\Big)$

```
1: Input: sample $\hat{\xi}^n$ and optimal solutions of (RER) $\hat{\boldsymbol{x}}$, $\hat{\gamma}_n$ and $\hat{\alpha}$
2: $V \leftarrow \emptyset$
3: $z^* \leftarrow -\infty$
4: for each $(\boldsymbol{\xi_\ell}, \xi_r, \xi_a) \in L \times R \times A$ do
5:     $\hat{F} \leftarrow \emptyset$
6:     for each $\xi_f \in F$ do
7:         $\boldsymbol{\xi} \leftarrow (\boldsymbol{\xi_\ell}, \xi_r, \xi_r, \xi_f)$
8:         if $\xi \in \Xi_n$ & $\xi \notin \hat{S}_n$ then
9:             $\hat{F} \leftarrow \hat{F} \cup \{\xi_f\}$
10:        end if
11:    end for
12:    if $\hat{F} \neq \emptyset$ then $\boldsymbol{\xi}^{\mathbf{0}} \leftarrow$ FIBONACCI$\Big(\hat{F}, \boldsymbol{\xi_\ell}, \xi_r, \xi_a, \hat{\xi}^n\Big)$        ▷ Solve $\max_{\xi_f \in \hat{F}} f(\xi_f)$
13:    // Update maximizing $\xi^*$ and optimal value $z^*$
14:    if $z_n(\boldsymbol{\xi}^{\mathbf{0}}) > z*$ then
15:        $\boldsymbol{\xi}^* \leftarrow \boldsymbol{\xi}^{\mathbf{0}}$
16:        $z^* \leftarrow z_n(\boldsymbol{\xi}^{\mathbf{0}})$
17:    end if
18: end for
19: Let $s^*$ be the index associated with $\boldsymbol{\xi}^*$
20: if $\hat{\gamma}_n < z^*$ then
21:     return $\{s^*\}$
22: else
23:     return $\emptyset$
24: end if
```

scales logarithmically in the size of F.

We note that if we wish to solve the original model (4.2.1), in which case Assumption 2 does not hold, SEPARATION-FIB is not guaranteed to return an optimal solution to (SP_n). However, we observe in our experiments that $z_n(\boldsymbol{\xi})$ is still concave with respect to ξ_f for a fixed landfall, radius and angle in the vast majority of cases, and in the few cases where it is not concave, the solution is optimal or close to optimal. Thus, we can perform SEPARATION-FIB as an efficient heuristic. To ensure an exact method, however, we would have to perform a full enumeration to solve (SP_n) once no scenarios can be generated. See Section 4.5 for more details.

Algorithm 4 FIBONACCI$\left(F, \boldsymbol{\xi_\ell}, \xi_r, \xi_a, \hat{\xi}^n\right)$

```
    **Input** set $F$ in increasing order, $\boldsymbol{\xi_\ell}$, $\xi_r$, $\xi_a$ and sample $\hat{\xi}^n$

 1: $\Omega \leftarrow [(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f)]_{\xi_f \in F}$
 2: $\Theta$: array of Fibonacci numbers of size $m$ in increasing order

 3: // Corner cases
 4: if $|\Omega| = 1$ then
 5:     return $\boldsymbol{Omega_0}$
 6: else if $|\Omega| = 2$ then
 7:     return $\boldsymbol{\xi}^* := \arg\max_{\boldsymbol{\xi}} \{z_n(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \{\boldsymbol{\Omega_0}, \boldsymbol{\Omega_1}\}\}$
 8: end if

 9: // Initialize indices $\ell$, $u$, $i_a$ and $i_b$
10: $(\ell, i_a, u) \leftarrow (0, \Theta_{m-3}, \Theta_{m-1})$
11: $i_b \leftarrow \ell + u - i_a$
12: while $i_a \neq i_b$ do
13:     if $i_b > |\Omega| - 1$ or $z_n(\boldsymbol{\Omega_{i_a}}) > z_n(\boldsymbol{\Omega_{i_b}})$ then
14:         $u \leftarrow i_b$
15:         $i_b \leftarrow i_a$
16:         $i_a \leftarrow \ell + u - i_b$
17:     else
18:         $\ell \leftarrow i_a$
19:         $i_a \leftarrow i_b$
20:         $i_b \leftarrow \ell + u - i_a$
21:     end if
22: end while
23: if $\ell = 0$ then
24:     $\boldsymbol{\xi}^* \leftarrow \arg\max_{\boldsymbol{\xi}} \{z_n(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \{\boldsymbol{\Omega_{i_a}}, \boldsymbol{\Omega_\ell}\}\}$
25: else if $u = \Theta_{m-1}$ then
26:     $\boldsymbol{\xi}^* \leftarrow \arg\max_{\boldsymbol{\xi}} \{z_n(\boldsymbol{\xi}) : \boldsymbol{\xi} \in \{\boldsymbol{\Omega_{i_a}}, \boldsymbol{\Omega_u}\}\}$
27: else
28:     $\boldsymbol{\xi}^* \leftarrow \boldsymbol{\Omega_{i_a}}$
29: end if
30: return $\boldsymbol{\xi}^*$
```

4.3.4 Fixed-Charge Network Flow

We show that our results extend to the case where the second-stage problem is an uncapacitated fixed-charge network flow problem. Given a graph $G = (\mathcal{V}, \mathcal{A})$, we define all nodes incident to outgoing and incoming arcs as $\delta^+(e_1) = \{e_2 : (e_1, e_2) \in \mathcal{A}\}$ and

$\delta^-(e_2) = \{e_1 : (e_1, e_2) \in \mathcal{A}\}$, respectively. In line with Assumption 2, we assume each facility sends an uncapacitated vehicle to serve demand nodes, and can serve up to k_i demand nodes. Variable σ_{ij} is 1 if facility i is serving demand node j. We introduce a new variable $\boldsymbol{\beta}$ where $\beta_{e_1e_2}$ is 1 if arc (e_1, e_2) is used. Variable $t^{ik}_{e_1e_2}$ represents the amount of commodity k transported through arc (e_1, e_2) by facility i's vehicle. Finally, let M be a sufficiently large number. We write the second-stage problem as:

$$\min_{\boldsymbol{\sigma}, \boldsymbol{\beta}, \boldsymbol{t}, \boldsymbol{u}} \quad \sum_{(e_1,e_2)\in\mathcal{A}} \left[w_{e_1e_2}\beta_{e_1e_2} + \sum_{i\in I}\sum_{k\in K} c_{ke_1e_2} t^{ik}_{e_1e_2} \right] + U \sum_{j\in J}\sum_{k\in K} u^k_j \tag{4.3.5a}$$

$$\text{s.t.} \quad \sum_{j\in J} \sigma_{ij} \le k_i, \qquad \forall i \in I, \tag{4.3.5b}$$

$$\sigma_{ij} \ge \frac{1}{M} \sum_{e_1\in\delta^-(j)} t^{ik}_{e_1j}, \qquad \forall i \in I,\ \forall j \in J, \tag{4.3.5c}$$

$$\sum_{e_2\in\delta^+(i)} t^{ik}_{ie_2} - \sum_{e_1\in\delta^-(i)} t^{ik}_{e_1i} = \sum_{j\in J} d^s_{kj}\sigma_{ij} \quad \forall i \in I,\ \forall k \in K, \tag{4.3.5d}$$

$$\sum_{e_2\in\delta^+(j)} t^{ik}_{je_2} - \sum_{e_1\in\delta^-(j)} t^{ik}_{e_1j} = -d^s_{kj}\sigma_{ij} \quad \forall i \in I,\ \forall j \in J,\ \forall k \in K, \tag{4.3.5e}$$

$$u^k_j \ge d^s_{kj} - M \sum_{i\in I} \sigma_{ij} \qquad \forall j \in J,\ \forall k \in K, \tag{4.3.5f}$$

$$t^{ik}_{e_1e_2} \le M\beta_{e_1e_2}, \qquad \forall (e_1, e_2) \in \mathcal{A},\ \forall i \in I,\ \forall k \in K, \tag{4.3.5g}$$

$$\sigma_{ij} \le o_i \qquad \forall i \in I,\ \forall j \in J, \tag{4.3.5h}$$

$$t^k_{ij} \ge 0, \qquad \forall i \in I,\ \forall j \in J,\ \forall k \in K,$$

$$\sigma_{ij} \in \{0, 1\}, \qquad \forall i \in I,\ \forall j \in J,$$

$$u^k_j \ge 0, \qquad \forall j \in J,\ \forall k \in K,$$

$$(\boldsymbol{\sigma}, \boldsymbol{t}, \boldsymbol{u}) \in C.$$

Constraints (4.3.5b) ensure a facility i serves at most k_i demand nodes as in (4.3.2); constraints (4.3.5c) set σ_{ij} to 1 if facility i is serving node j; constraints (4.3.5d) are network flow constraints at facility nodes, ensuring d_{kj} units of commodiy k leaves the facility for each demand node j it is serving; constraints (4.3.5e) ensure facility i's vehicle delivers d_{kj} units of commodiy k to node j if i is serving j; constraint (4.3.5f) ensures $u_j^k = d_{kj}^s$ if demand node $j \in J$ is not served by any facility $i \in I$, i.e. $\sum_{i \in I} \sigma_{ij} = 0$; constraints (4.3.5g) ensure $\beta_{e_1 e_2} = 1$ if $t_{e_1 e_2}^{ik} > 0$ for any commodity k and facility i; finally constraints (4.3.5h) ensure facility i can serve demand node j only if facility i is open.

We have the following similar result.

Theorem 10. *Under Assumption 1, if the second-stage problem corresponds to* (4.3.5)*, then for fixed first-stage decision vector $\boldsymbol{x}$ and $\alpha \geq 0$, $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ and $z_n(\boldsymbol{\xi})$ are concave with respect to ξ_f for a fixed landfall $\boldsymbol{\xi_\ell}$, radius of impact ξ_r, and angle ξ_a.*

Proof. The proof is very similar to the proof of Theorem 9 and its corollary. Let $\hat{I}$ be the set of open facilities. For an open facility $i \in \hat{I}$, let $J_i \subseteq J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ be the set of demand nodes it serves, and let $\rho_i = \{\rho_{ij}\}_{j \in J_i}$ be a set of paths from facility i to nodes $j \in J_i$, where ρ_{ij} is a set of edges from facility i to demand node j. Thus, $\{\rho_i\}_{i \in \hat{I}}$ corresponds to a solution to the second-stage problem, representing paths from each open facility $i \in \hat{I}$ to the demand nodes they each serve. Let ρ be the set of all possible $\{\rho_i\}_{i \in \hat{I}}$ satisfying constraint (4.3.5b). Recall that $\hat{J}$ is the set of demand nodes in $J(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$ that are not served by any facility, and $f(\xi_f)$ corresponds to $z_n(\boldsymbol{\xi})$ as a function of ξ_f, where $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a are fixed.

For a fixed set of paths for each facility $i \in \hat{I}$, i.e. for a fixed $\{\rho_i\}_{i \in \hat{I}}$, the second-stage cost is:

$$\sum_{i \in \hat{I}} \sum_{j \in J_i} \sum_{e \in \rho_{ij}} \left[w_{e_1 e_2} + \sum_{k \in K} c_{k e_1 e_2} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \right] + U \sum_{j \in \hat{J}} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j, \quad (4.3.6)$$

which is concave with respect to ξ_f. We then have

$$\begin{aligned} f(\xi_f) &= Q(\boldsymbol{x}, \boldsymbol{\xi}) - \alpha\, \mathrm{d}(\boldsymbol{\xi}, \boldsymbol{\hat{\xi}^n}) \\ &= \min_{\{\rho_i\}_{i\in\hat{I}} \in \rho} \Bigg\{ \sum_{i\in\hat{I}} \sum_{j\in J_i} \sum_{e\in\rho_{ij}} \Bigg[w_{e_1 e_2} + \sum_{k\in K} c_{k e_1 e_2} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \Bigg] \\ &\qquad\qquad + U \sum_{j\in\hat{J}} \Pi_j^k(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) P_j \Bigg\} - \alpha\, \mathrm{d}(\xi, \hat{\xi}^n), \end{aligned}$$

□

and the result follows.

4.4 Numerical Considerations

In this section, we discuss the numerical implications of our method, and how it can be improved. We first present numerical improvements in solving (RER), then discuss implementation details in the separation step which can significantly improve computational performance. Finally, we conclude with our final column-and-constraint generation algorithm incorporating these changes which we name CCG-FIB.

4.4.1 Solving (RER)

We have so far only discussed how to efficiently solve the separation problem, but have not tackled the restricted model (RER), which might become difficult to solve as the algorithm progresses and as we generate many scenarios, especially for larger numbers of samples. First, to get a tighter formulation, we include valid inequalities $y_{ij} \leq o_i$ in C in the second-stage model (4.2.2) as mentioned in Section 4.2. Moreover, we set upper bounds M_{ij} in constraint (4.2.2c) to be the highest demand that can possibly occur in any scenario at demand node j, i.e. $M_{ij} = \max_{\xi_f \in F} \xi_f P_j$ for all i, where P_j is the population at node j.

To control the size of (RER), we avoid generating too many scenarios in an iteration by limiting the number of scenarios that can be generated. The limit can be dynamically adjusted as the algorithm progresses. There are many variations one can think of, such as generating multiple scenarios for one sample until we hit the limit, or setting a limit for each sample in addition to a limit across all samples. For simplicity, we ensure only one scenario is generated per sample, and only impose a limit on the total number of scenarios generated across all samples, which we keep constant throughout the algorithm. Finally, we observe in our experiments that the solver spends the vast majority of the time proving optimality, and seems to quickly find good, feasible integer solutions. To this end, we also dynamically adjust the MIP gap tolerance when solving (RER), where we start with a loose tolerance and tighten it as the algorithm progresses. This avoids the occasional long stalls that happen in our experiments.

4.4.2 Separation Problem Implementation

In the separation problem, when searching for a scenario which violates constraint (c1) for some sample n, we solve the second-stage problem to get the optimal value $\mathcal{Q}(\hat{\boldsymbol{x}}, \boldsymbol{\xi})$ for a first-stage vector $\hat{\boldsymbol{x}}$ and candidate scenario $\boldsymbol{\xi}$. Once the second stage is solved once at the candidate scenario $\boldsymbol{\xi}$, we do not need to re-compute $\mathcal{Q}(\hat{\boldsymbol{x}}, \boldsymbol{\xi})$ during the search for other samples. Thus, if we solve the separation problem by enumerating all scenarios at the first sample, we can perform a check for scenarios which violate (c1) via a quick enumeration for the remaining samples. We are still enumerating all scenarios but only perform simple operations for each scenario instead of solving the second-stage MIP, which can be done very fast. This leads to the question of how beneficial it is to use Fibonacci search in the separation step. For one fixed sample, using `SEPARATION-FIB` does indeed lead to far fewer scenarios enumerated. However, it is possible that for the next sample, we cannot always take advantage of the fact that the second-stage problem has been solved during the search at the previous sample. In other words, depending on the scenarios visited during the Fibonacci search for one sample and fixed components $\boldsymbol{\xi}_{\boldsymbol{\ell}}$, ξ_r and ξ_a, we might still have to solve the second-stage model for unvisited scenarios during the search at another sample at the same fixed components. As the number of samples increases, it becomes more likely that this event occurs, especially for larger sets F. We observe in our experiments that although performing SEPARATION-FIB for each sample $n \in \mathcal{N}$ can never perform worse than a full enumeration for each sample, the computational benefits can be further improved.

We empirically observe that for fixed $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a, the maxima of $f(\xi_f)$ can be close to each other across different samples, if not the same in many instances. This motivates the following procedure: we perform SEPARATION-FIB for the first sample in the list, then simply consider scenarios for which the second model has been solved for the remaining samples, where we return the scenario for which $z_n(\boldsymbol{\xi})$ is greatest. In other words, we are assuming that for the same fixed components $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a, the maximum always occurs at the same ξ_f across the samples. In the next iteration, we call SEPARATION-FIB for the second sample, and so on. This heuristic to solving (SP_n) leads to significant computational benefits as we see in Section 4.5.2. Note that once no scenarios can be generated, we still have to perform a final check where we perform SEPARATION-FIB for each sample to ensure an exact method.

4.4.3 CCG-FIB Algorithm

We formalize and present the final column-and-constraint generation algorithm CCG-FIB in Algorithm 5. In this context, the set of samples $\mathcal{N}$ is interpreted as an array which can be indexed. We refer to the scenarios at which the second-stage model has been solved at the current first-stage decision $\hat{\boldsymbol{x}}$ as $\hat{\Xi}$, and the process of enumerating them as LAZY-ENUMERATION$\left(\hat{\Xi}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n}}, \hat{\gamma}_n, \hat{\alpha}\right)$. In general, this function can return a set of scenarios $V_n \subseteq \hat{\Xi}$ violating (c1), but we limit it to returning at most one sceanrio whose objective value $z_n(\boldsymbol{\xi})$ is the greatest. Recall that for a sample n, LAZY-ENUMERATION only considers scenarios that are in $\hat{\Xi}$ and that are not in the current model (i.e. scenarios in $S_n \setminus \hat{S}_n$). We keep track of a boolean final_check, which is initialized to False. When

False, we perform SEPARATION-FIB for one sample, and LAZY-ENUMERATION for the remaining samples. Thus, during this phase, we break out of the while-loop (line 9) at the end of the first iteration and jump to line 13, where we enumerate scenarios $\hat{\Xi}$, i.e. scenarios for which the second-stage problem has been solved at the current iteration. We then increment the sample index i in line 17, where the index loops back to 0 after reaching the end of the array. This ensures we rotate through the samples for which we fully solve $\max_{\boldsymbol{\xi}} z_n(\boldsymbol{\xi})$. Once no scenarios are generated, final_check becomes True (line 22). During this phase, we perform SEPARATION-FIB for each sample, and reset the sample index i to 0 in line 19. If no scenarios are generated, we reset final_check to False (line 24) and the algorithm exits the outer while-loop to terminate.

Finally, we note that we have omitted the changes to solving (RER) in Algorithm 5 for ease of exposition, specifically the limit on the number of scenarios generated and the adjustement of the MIP tolerance of (RER). By definition of SEPARATION-FIB and LAZY-ENUMERATION, we have that $|V_n| = 1$, i.e. we generate at most one scenario per sample, and given a limit on the total number of scenarios that can be generated, we either exit the inner while-loop (5-11) or the for-loop in lines 13-16 once the number of generated scenarios hits the limit. Moreover, given an array of MIP tolerances, we decrease the tolerance when solving (RER) once no scenarios can be generated during the phase where final_check is false. In other words, instead of setting final_check to True in line 22, we decrease the MIP tolerance and restart the process. Once we get to the lowest (tightest) desired tolerance, we proceed as normal where we enter the final check phase once no scenarios can be generated during the first phase.

4.5 Computational Experiments

We perform extensive computational experiments to illustrate our method. We perform a series of experiments on randomly generated networks to test the computational benefits of our proposed method and perform experiments on a case study of hurricane threats on

Algorithm 5 CCG-FIB

```
1: Initialize $\hat{\mathcal{S}}_n$ for each $n \in \mathcal{N}$
2: $i \leftarrow 0$
3: while $\cup_{n\in\mathcal{N}} V_n \neq \emptyset$ or not final_check do
4:     Solve (RER) and record optimal solutions $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\gamma}}, \hat{\alpha})$
5:     while $i \leq |\mathcal{N}| - 1$ do
6:         $n \leftarrow \mathcal{N}_i$
7:         $V_n \leftarrow$ SEPARATION-FIB($\hat{\boldsymbol{\xi}}^{\boldsymbol{n}}, \hat{\boldsymbol{x}}, \hat{\gamma}_n, \hat{\alpha}$)
8:         $\hat{\mathcal{S}}_n \leftarrow \hat{\mathcal{S}}_n \cup V_n$
9:         if not final_check then break
10:        $i \leftarrow i + 1$
11:    end while
12:    if not final_check then
13:        for each $n' \in \mathcal{N},\ n' \neq n$ do
14:            $V_{n'} \leftarrow$ LAZY-ENUMERATION $\left(\hat{\Xi}, \hat{\boldsymbol{\xi}}^{\boldsymbol{n'}}, \hat{\gamma}_{n'}, \hat{\alpha}\right)$
15:            $\hat{\mathcal{S}}_{n'} \leftarrow \hat{\mathcal{S}}_{n'} \cup V_{n'}$
16:        end for
17:        $i \leftarrow i + 1 \pmod{|\mathcal{N}|}$
18:    else
19:        $i \leftarrow 0$
20:    end if
21:    if $\cup_{n\in\mathcal{N}} V_n = \emptyset$ & not final_check then        ▷ Start final check
22:        final_check ← True
23:    else if $\cup_{n\in\mathcal{N}} V_n = \emptyset$ & final_check then        ▷ Algorithm terminates
24:        final_check ← False
25:    end if
26: end while
```

the coastal states of the United States along the Gulf of Mexico and the Atlantic Ocean presented in [126] and further studied in [36], analyzing the solutions obtained by the DRO model under the Wasserstein radius and by SAA. All experiments were performed on compute nodes running Intel Xeon Gold 6226 “Cascade Lake” at 2.7 Ghz. We limit our resources to 8 GB of RAM and 5 processors. We use GUROBI 9.1.0 as the solver for all models.

We first give a brief description of the data and other details that are common to the synthetic instances and the case study. We then discuss how we construct the synthetic instances and results, before tackling the case study.

Table 4.1: Per unit purchase cost, volume, and transportation cost per unit distance of water, food and medical kits.

Commodity	Unit Purchase Cost ($)	Volume (ft^3)	Transportation Cost ($)
Water (1000 galons)	647.7	144.6	0.3
Food (1000 kits)	5420	83.33	0.04
Medicine Kit (serves 4)	140	1.16	5.8e-4

4.5.1 Data

Commodity data

We consider only one commodity for simplicity, which can be interpreted as a bundle containing a medical kit, food and water. To calculate the per-unit cost of the bundle, we take the sum of the cost of each commodity adjusted to the needs of one person. More specifically, as in [126], we assume the average person needs three meals and two snacks per day, and one gallon per day. We further assume that a disaster lasts five days. Data related to water, food, and medical kits from [126] is given in Table 4.1. Thus, 1000 gallons of water costing \$647.7 to purchase and \$0.3 per unit of distance to transport translates to a cost of approximately \$3.24 to purchase and \$0.0015 to transport per person. Similarly, meal kits cost \$135.5 to purchase and \$0.001 per unit of distance to transport per person, and a medical kit costs \$35 to purchase and \$$1.45$e-$4$ to transport. We then have that a bundle costs about \$173.74 to purchase and \$0.0026 per unit of distance to transport.

Facility data

In both the synthetic instances and the case study, we assume we can open one type of facility of capacity 408,200 ft^3 at a cost of \$188,400 as in [36] (equivalent to the medium facility in [126]). Following a similar process as with the costs, a bundle occupies about 3.1 ft^3. Thus, we assume the facility has a capacity of about 131,837, and the bundle occupies a space of one unit.

Support Set

The construction of our support set is heavily influenced by the Hurricane Database (HURDAT) provided by the National Oceanic and Atmospheric Administration (NOAA) [133], which logs the paths and intensities of more than 150 years of hurricanes, as well as by the constructed scenarios in [126]. A visualization tool of the HURDAT data is available at https://coast.noaa.gov/hurricanes. Note that although the network of nodes is randomly generated in the synthetic instances, we assume the shores are to the east and south, representing the Atlantic Ocean and the Gulf of Mexico, respectively, so that the following applies to both the synthetic instances and the case study.

We assume there can only be one landfall node $\boldsymbol{\xi_\ell}$ per scenario, and that landfalls are close to shore. The landfall determines the remaining affected nodes depending on the radius of impact and angle (or path) of the hurricane. We define the set of radii as $R = \{\xi_r = 100i, i = 0, 1, \ldots, 5\}$ in kilometers and the set of angles as $A = \{0, \frac{-\pi}{4}, \frac{-\pi}{2}, \frac{\pi}{4}, \frac{\pi}{2}\}$ in radians. Note that depending on the network, the set of nodes hit by a disaster can be the same for two different radii, leading to the same scenario. We exclude redundant scenarios from our support.

We observe in the HURDAT data that hurricanes hitting the coastal states of the United States along the Gulf of Mexico and the Atlantic Ocean tend to take a curved path, generally starting from the south-east and curving towards the north-east and decreasing in intensity as the hurricane advances in-land. Thus, given a landfall and radius of impact, we assume only nodes within the northern half-circle are affected, where the half-circle is rotated by the angle ξ_a. For example, if $\xi = 0$, then the base of the half-circle is horizontal, and nodes that are north of the base and within the half-circle are affected. A positive angle rotates the base of the half-circle counter-clockwise and a negative angle rotates it clockwise. In the example given in Figure 4.1, scenario 1 is obtained with $\xi_a = \frac{\pi}{4}$ and scenario 2 with $\xi_a = \frac{-\pi}{4}$. Moreover, we choose functions $\Pi_j(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f) = \min\{\xi_f, \bar{\xi}_{fj}\}$ for all j, $\boldsymbol{\xi_\ell}$, ξ_r and ξ_a. Upper bounds $\bar{\xi}_{fj}$ (and thus the functions Π_j) are independent of ξ_r and only depend on $\boldsymbol{\xi_\ell}$ and ξ_a. Given a landfall and angle, we determine the upper bound $\bar{\xi}_{fj}$ as follows. We look at all the nodes that can be affected at the maximum radius. If node j is the i^{th} furthest affected node from the landfall, then $\bar{\xi}_{fj}$ is set to the i^{th} largest value in F, or to the smallest value in F if $i > |F|$. This is to also satisfy the fact that the intensity of the hurricane decreases as it travels in-land (i.e. the fraction of the population affected decreases as we move farther away from the landfall) and to satisfy Assumption 1.

Although we do not know the true distribution in practice, we construct a probability distribution associated with each scenario to simulate and test our methods. We use this distribution to get the sample set $\mathcal{N}$ and simulate the out-of-sample performance. It is unlikely that the components of our random vector are truly independent, but we assume independence for simplicity as in [36]. As such, the probability of a scenario $(\boldsymbol{\xi_\ell}, \xi_r, \xi_a, \xi_f)$ is the product of the probabilities of each component. For the radii, we assume a skew-normal distribution with the median of R, standard deviation and -1 as parameter values for the location, scale and shape, respectively. This is a normal distribution that is slightly left skewed. For the angles, we assume $\mathbb{P}\left[\xi_a = 0\right] = \frac{4}{15}$, $\mathbb{P}\left[\xi_a = \frac{-\pi}{4}\right] = \frac{5}{15}$, $\mathbb{P}\left[\xi_a = \frac{-\pi}{2}\right] = \frac{3}{15}$, $\mathbb{P}\left[\xi_a = \frac{\pi}{4}\right] = \frac{2}{15}$, $\mathbb{P}\left[\xi_a = \frac{\pi}{2}\right] = \frac{1}{15}$. The idea is that it is more likely for a hurricane to be pathing east (negative angles have a higher probability) and more likely to be pathing north than at a very steep angle ($\mathbb{P}\left[\xi_a = 0\right] > \mathbb{P}\left[\xi_a = \frac{-\pi}{2}\right]$). We note that these probabilities were obtained in an ad-hoc manner for illustration purposes based on observations of the HURDAT data.

Finally, the probability distribution of ξ_f is determined based on the number of hurricanes by category that have hit the coastal states between 1851 and 2004. Hurricanes are classified in catagories from 1 to 5, with a higher catagory indicating a more major hurricane. These catagories are based on the SAFFIR-SIMPSON scale which is a rating determined by the hurricane's wind speeds. For example, Hurricane Katrina was a catagory 5 hurricane which hit the New Orleans area in 2005. According to [126] and [134], the number of hurricanes that have hit the coastal states between 1851 and 2004 at each catagory from 1 to 5 is 113, 74, 76, 18 and 3, respectively. Given a set F of possible values for ξ_f, we order F in increasing order, assign equal intervals of F to each catagory and assign the same weight to each value in the interval based on the number of historical hurricanes at each catagory. For example, if $F = \{0.1, 0.2, 0.3, \ldots, 1\}$, then $\xi_f \in \{0.1, \ldots, 0.2\}$ would be considered category 1 hurricanes and each value in the interval is assigned a weight of 113; $\xi_f \in [0.21, 0.4]$ would be category 2 hurricanes and each value is assigned a weight of 74, and so on. We then normalize so that the probabilities sum to one.

We give details on the probability distribution of $\boldsymbol{\xi_\ell}$, the landfall node, in the following sections, as it is different for the synthetic instances and the case study.

4.5.2 Synthetic Instances

Instance Generation

To create the synthetic instances, we randomly generate a set of nodes on a grid by uniformly picking x and y coordinates between 0 and 20. To avoid generating nodes that are too close to each other, we set a minimum threshold such that any two nodes are at least a Euclidean distance of 2 away from each other. We assume the coast is on the right and bottom sides of the grid as in the case study, and assume nodes within a threshold distance of the coast can be a potential landfall node. Figure 4.2 is an example of a randomly generated network, where yellow nodes are potential landfall nodes, blue nodes are potential facility nodes, and orange nodes are both. We then generate random populations for each node, where we sample from a uniform distribution on the interval $[1\text{e-}5, 2\text{e-}6]$.

Recall that we use a cost of \$173.74 to purchase and \$0.0026 per unit of distance to transport a unit of commodity. Thus, we set the per-unit cost of transportation from facility i to demand node j to be $c_{ij} = 0.0026\,\mathrm{d}_{ij}$, where d_{ij} is the ℓ_2-norm between node i and j in our grid. Note that we scale the distances up so that the order of magnitude of the distances is similar to that of the case study. We do this so that the balance between the first- and second-stage costs is reasonable and comparable to the case study. We set the fixed-charge

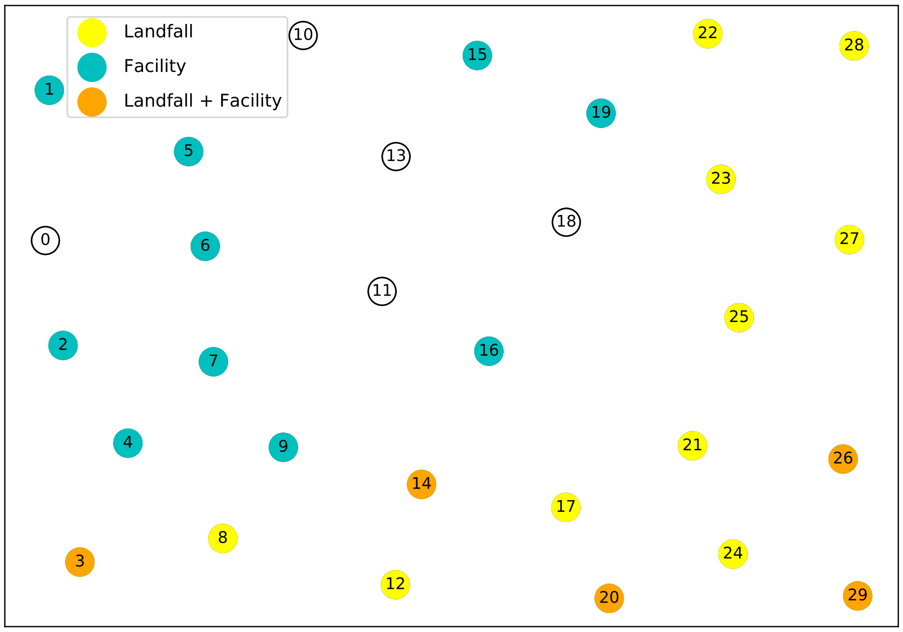


Figure 4.2: Landfall nodes and facilities of a randomly generated network of 30 nodes.

cost to be a multiple of the cost of transportation, so that $w_{ij} = \phi c_{ij}$. In our experiments, we set $\phi = 5000$ and the penalty for unmet demand $U = 10$.

Finally, we construct a probability distribution for the landfall nodes that is inversely proportial to the sum of distances to the closest eastern and southern points, such that the closer a node is to the coast, the higher the likelihood of it being a landfall.

Experiment Setup

We perform four sets of experiments for different combinations of sample set sizes N and radii θ of the Wasserstein set, where we run tests for a sample size of 10 and 50, and a radius of 1e-2 and 1e-3 for the Wasserstein ball. For each set of experiment, we test how our methods perform as the discretization of F becomes finer, where we pick 15, 30, 50 and 75 equi-distant values between 0.001 and 0.3.

In these sets of experiments, we assume Assumptions 1 and 2 hold, and are thus solving (4.3.2). In the second stage, we fix the limit on the number of demand nodes that a facility can serve k_i to 3 for all facilities i. We keep the Gurobi parameters to their default values with the exception of the relative MIPGap and Threads parameter. We start with a loose tolerance of 0.12 for the relative MIP gap to solve (RER) and adjust it to 0.01 as detailed in Section 4.4.3. We set the limit on the number of scenarios generated per iteration to 5 when $N = 10$ and to 10 when $N = 50$. To reduce the computational resources used, we limit the number of threads that Gurobi can use to 1 when the tolerance is 0.12, and increase it to 5 when the tolerance is 0.01. Finally, we set a time limit of 8 hours for all experiments.

We randomly generate networks of 30 nodes, where we randomly pick a set I of 15 potential facility locations; every node in the network is a potential demand node. We compare our method to two classical CCG algorithms, where the separation step is a simple enumeration process. In one case we stop at the first scenario violating constraint (c1), and in the other we perform a full enumeration to pick the scenario $\boldsymbol{\xi}$ which maximizes $z_n(\boldsymbol{\xi})$ for sample $n \in \mathcal{N}$. We refer to the former as CCG-ENUM-0 and the latter as CCG-ENUM-1. Note that both algorithms are the same as CCG-FIB but only differ in the separation function called in line 7 of CCG-FIB. Thus, for both CCG-ENUM-0 and CCG-ENUM-1, after performing an enumeration of scenarios for sample n where we are solving the second-stage problem, we perform `LAZY-ENUMERATION` for samples $n' \neq n$. Moreover, we are still applying Theorem 8 but do not leverage the structure of the second-stage value function.

We also define CCG-FIB-0, which is similar to CCG-FIB but with a slight modification to the call to `SEPARATION-FIB`. Instead of performing Fibonacci search for each combination of components $(\boldsymbol{\xi_\ell}, \xi_r, \xi_a)$, we exit the outer-loop as soon as the Fibonacci search returns a scenario which violates (c1). Note that we can go a step further and terminate the separation step as soon as a scenario which violates (c1) is found during the Fibonacci search, but our experiments showed it is always better to finish the Fibonacci search before terminating the separation step, as it is relatively cheap to finish running Fibonacci search and it might lead to a scenario $\boldsymbol{\xi}$ with a greater objective value $z_n(\boldsymbol{\xi})$. The idea behind CCG-FIB-0 is to get the best of both CCG-ENUM-0 and CCG-FIB, where the separation

Table 4.2: Average number of scenarios across randomly generated networks for fixed size of the set F.

Size of F	Avg. Number of Scenarios
15	3070
30	6468
50	10020
75	14715

problem is solved fast in the earlier stages of the algorithm, permitting to quickly generate scenarios, before naturally transitioning to CCG-FIB as the algorithm progresses, avoiding enumerating close to all scenarios in the later stages as would happen in CCG-ENUM-0.

The performance of our methods depend on the sample set and the order of the samples in the set. Thus, for each sample set size and discretization of F, we run 10 experiments where we generate a new random network and a new random set of samples. We provide the average number of scenarios across generated networks for each discretization of F after removing redundant scenarios in Table 4.2.

For all experiments and computed metrics, we only consider runs where all methods terminated within the time limit. For $N = 50$ and $\theta = 0.001$, one run did not terminate for $|F| = 50$ and two runs did not terminate for $|F| = 75$ for all four methods. These were the same three runs for all methods. This is due to the solver stalling when solving (RER) in certain iterations. Such stalls can be circumvented by further adjusting the MIP tolerances or increasing the number of threads available to Gurobi. For $N = 50$ and $\theta = 0.01$, CCG-ENUM-0 did not terminate in one of the runs.

Results

In Figure 4.3, we plot the average runtimes over the 10 runs as a function of the size of F for each sample set size N and radius of the Wasserstein ball θ. The upper and lower limits of the ribbon are one standard deviation from the average. In general, we observe that the runtimes are faster for the smaller radius. For $N = 10$, CCG-FIB significantly outperforms CCG-ENUM-0 and CCG-ENUM-1, with CCG-FIB-0 being a close second. We note, however, that CCG-FIB-0 has a greater standard deviation of runtimes than CCG-FIB, suggesting the latter is a more stable method. For $\theta = 0.001$, CCG-FIB is about 39% to 40% faster on average than both CCG-ENUM-0 and CCG-ENUM-1 when $|F| = 15$, and is about 69% faster on average when $|F| = 75$. Similarly, for $\theta = 0.01$, CCG-FIB is about 27% faster than CCG-ENUM-1 and 52% faster than CCG-ENUM-0 on average when $|F| = 15$, and is about 66% faster than CCG-ENUM-1 and 65% faster on average than CCG-ENUM-0 when $|F| = 75$. For $N = 50$, the gap between the methods is not as pronounced, but CCG-FIB still outperforms the other three. The two best performing methods are CCG-FIB and CCG-ENUM-1. For $|F| = 75$, on average, CCG-FIB is about 11% faster than CCG-ENUM-1 when $\theta = 0.001$, and about 21% faster when $\theta = 0.01$. In general, we observe that the benefit of using CCG-FIB increases as the size of $|F|$ increases, which is expected as the number of scenarios enumerated is logarithmic in the size of F for CCG-FIB but linear for CCG-ENUM-0 and CCG-ENUM-1.

As we increase the size of the sample set, solving (RER) becomes harder, and a larger portion of time is spent solving (RER) and smaller portion solving the separation step. This explains why the difference between CCG-FIB and CCG-ENUM-1 is not as large when $N = 50$, and why CCG-ENUM-0 and CCG-FIB-0 start to fall behind. For the latter, the separation step is solved fast (at least in the earlier stages), but (RER) is solved much more often, increasing runtime.

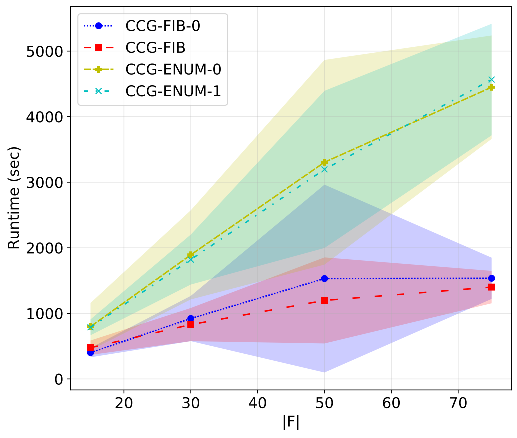
To better understand the behavior of the four methods, we record the number of second-stage problems solved during the separation step and the time spent in the separation step, where we aggregate the values across all iterations and the 10 runs to get a distribution for both metrics. We fix $\theta = 0.01$, and plot the cumulative distribution (CDF) of the time spent in the separation step in Figure 4.4 and of the number of second-stage solves in Figure 4.5.

When comparing CCG-FIB to CCG-ENUM-1, the number of second-stage solves in the separation problem is significantly less for the former. We see that in over 80% of the iterations, the number of second-stage problems solved is over 50% less in CCG-FIB than in CCG-ENUM-1 when $|F| = 15$. When $|F| = 75$, the gap is much more significant, with CCG-FIB consistently solving fewer than 2500 second-stage problems in over 80% of the iterations, compared to CCG-ENUM-1 ranging between 7500 and over 15000. The increase in the number of second-stage solves during CCG-FIB occurs in the final phase (i.e. when `final_check` is set to `True`, and `SEPARATION-FIB` is applied to every sample). For $|F| = 75$, we see that the number of second-stage solves in CCG-FIB gradually increases from about 2500 to 10000 for $N = 10$, and from about 2500 to 14000 for $N = 50$. As discussed in Section 4.4.2, while `SEPARATION-FIB` does lead to significantly fewer second-stage solves for one sample, we might end up visiting new scenarios for which we have to compute the second-stage problem if applied to all samples. Our experiments confirm our intuition that for larger sample sets, this is more likely to happen. Indeed, the maximum number of second-stage solves in CCG-FIB jumps from about 10000 to 14000 when increasing the sample size from 10 to 50.

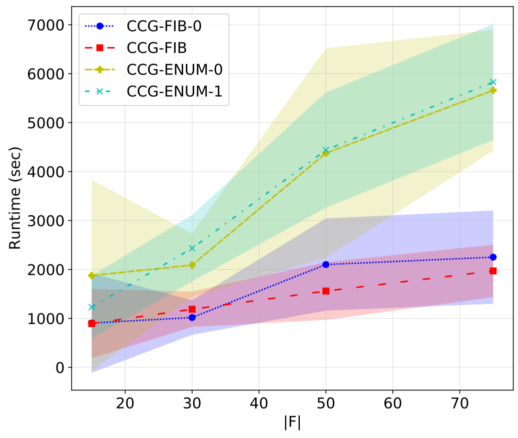
As expected, the CDF for both metrics are very similar. We note that the CDFs for CCG-FIB-0 are always upper-bounded by the CDFs for CCG-FIB, where they start very close to CCG-ENUM-0 and get close to CCG-FIB. This is in line with our idea behind CCG-FIB-0. However, as seen in Figure 4.3, the increase in the number of times (RER) is solved does not justify it. This analysis also permits us to see how long CCG-FIB spends in the final phase (i.e. `final_check` = `True`). Figures 4.5 and 4.4 show that fewer than 20% of iterations are in the final phase when looking at all iterations across the 10 runs.

4.5.3 Case Study: Hurricane Threats in the Gulf of Mexico States

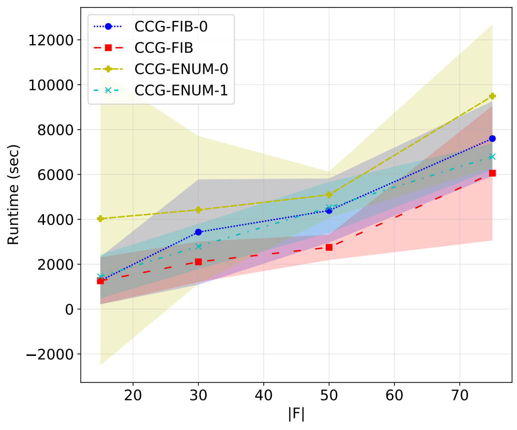
We now turn our focus to the case study of hurricanes threats on the coastal states of the United States first presented in [126]. We plot the network and the potential landfall nodes in Figure 4.6. A list of the cities corresponding to each node is given in Table B.1. We



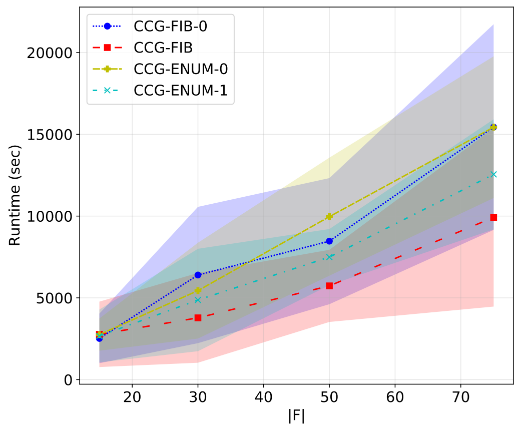
(a) $N = 10, \theta = 0.001$



(b) $N = 10, \theta = 0.01$



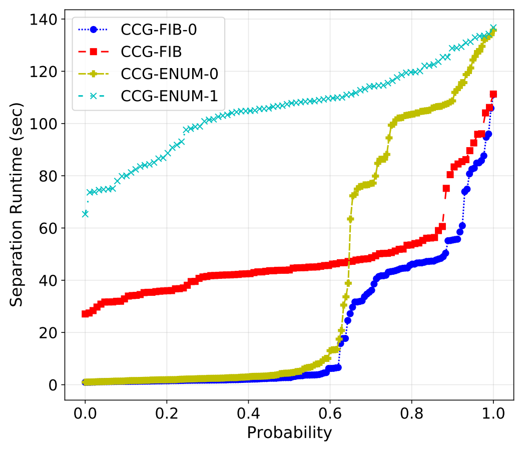
(c) $N = 50, \theta = 0.001$



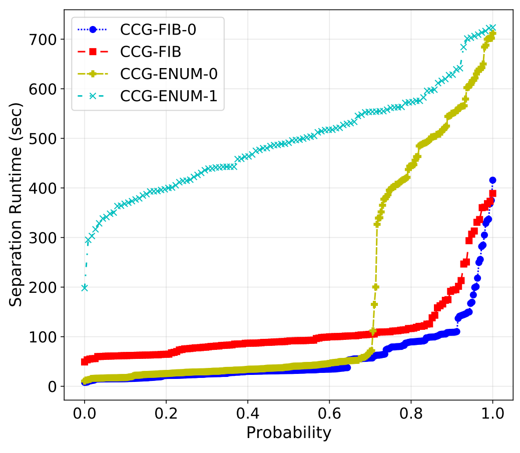
(d) $N = 50, \theta = 0.01$

Figure 4.3: Average runtimes of CCG-FIB, CCG-FIB-0, CCG-ENUM-0, and CCG-ENUM-1, for various sample set sizes N and Wasserstein radii θ over 10 runs, with ribbons spanning two standard deviations.

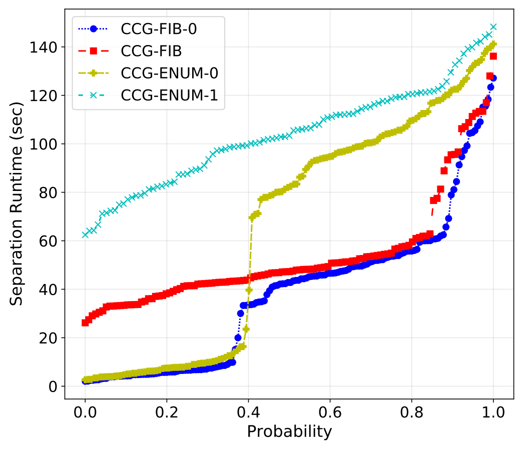
assume all nodes are potential facility locations and potential demand nodes. In this section, we analyze the solutions obtained from DRO for various radii levels, and from SAA (when $\theta = 0$). We solve problem (4.2.1), in which case Assumption 2 no longer holds. Recall that in this version of the model, we are opening facilities and pre-allocating resources in the first stage. We use CCG-FIB to solve all models, where we set the MIP tolerance to 0.12 and adjust it to 1e-3 as described in Section 4.4.3. As mentioned in section 4.3.3, once CCG-FIB terminates, we run one iteration of CCG-ENUM-1 to ensure an exact method. In almost all runs, CCG-ENUM-1 terminates in one iteration and does not generate additional scenarios, suggesting CCG-FIB has a strong computational performance as a heuristic even



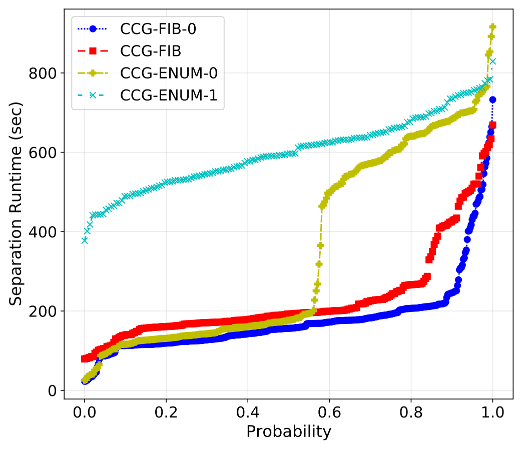
(a) $N = 10$, $|F| = 15$



(b) $N = 10$, $|F| = 75$



(c) $N = 50$, $|F| = 15$

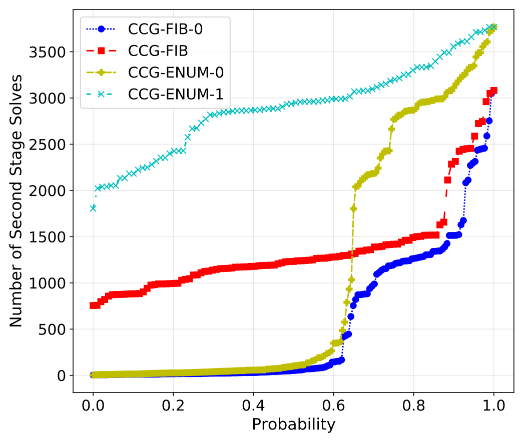


(d) $N = 50$, $|F| = 75$

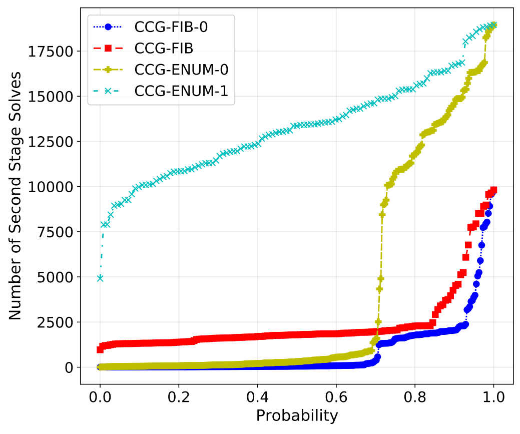
Figure 4.4: Cumulative distribution of runtimes in the separation step of each method, for sample set sizes 15 and 50, and for two discretizations of F, where $|F| = 15$ and $|F| = 75$.

if Assumption 2 does not hold.

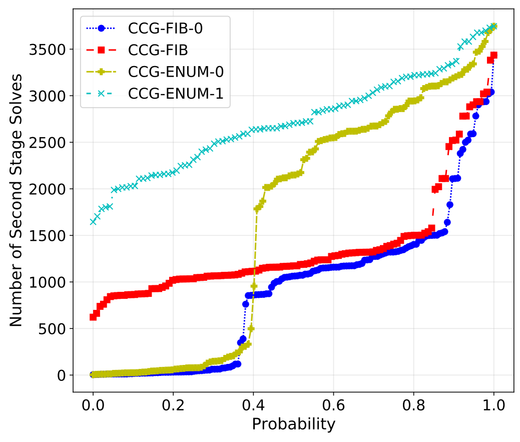
As in the synthetic instances, we use the data and support set described in Sections 4.5.1, 4.5.1 and 4.5.1, and fix $|F| = 50$ where we pick 50 equi-distant values between 0.001 and 0.3. We assume a similar cost structure for transportation costs, where the per-unit cost of transportation from facility i to demand node j is $c_{ij} = 0.0026\,\mathrm{d}_{ij}$, where d_{ij} is the haversine distance between nodes i and j. We set the fixed-charge cost to be $w_{ij} = \phi c_{ij}$ where $\phi = 5000$. Moreover, we construct a probability distribution for the landfall nodes by using the frequency of hurricanes by region recorded in [126] and [134].



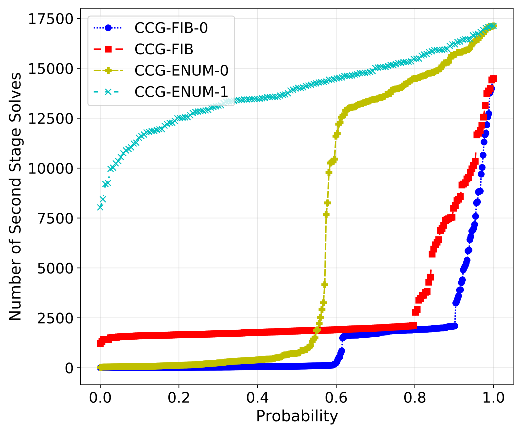
(a) $N = 10, |F| = 15$



(b) $N = 10, |F| = 75$



(c) $N = 50, |F| = 15$



(d) $N = 50, |F| = 75$

Figure 4.5: Cumulative distribution of number of second-stage solves during the separation step, for sample set sizes 15 and 50, and for two discretizations of F, where $|F| = 15$ and $|F| = 75$.

Results

We test the out-of-sample performance of our DRO model for various Wasserstein radii and penalty parameters U. As the radius of the Wasserstein ball increases, we are more risk averse, leading to higher costs in the first stage as we increase the number of facilities open and the amount of resources pre-allocated with the hope of satisfying more demand in more scenarios. As such, there is a natural trade-off between the first-stage costs and the penalty incurred for unsatisfied demand U. We analyze the out-of-sample performance of DRO for $U \in \{5, 10, 100\}$. We run tests for $N \in \{10, 50\}$, and for radii $\theta \in \{$1e-5, 5e-5,

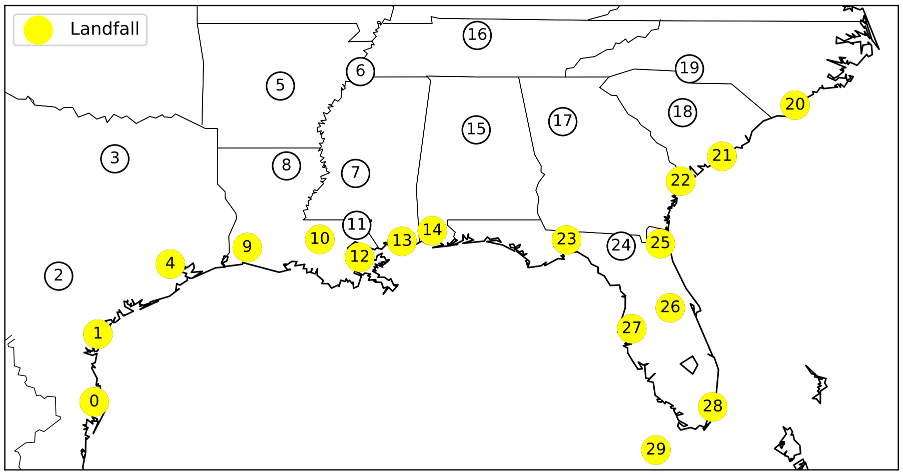


Figure 4.6: Map of the network with potential landfall nodes colored in yellow.

1e-4, 5e-4, 1e-3, 5e-3, 0.01, 0.05, 0.1}.

Note on radius selection. Picking a good radius for the Wasserstein ambiguity set is not straight forward and is the subject of research in the literature. Although there exists some theoretical radii which give probabilistic guarantees on the out-of-sample performance (see [18] and Section 3.2.1 in Chapter 3), such radii tend to be too large in practice and depend on constants which might be difficult to compute. In practice, the radius selection can be done in an ad-hoc manner through trial and error, or through k-fold cross-validation [18]. Given that we are dealing with very small sets of samples, cross-validation might be ill-suited. To this end, we choose a wide range of radii spanning a few orders of magnitudes for simplicity to illustrate our DRO model.

For each set of experiment, we generate N samples using the constructed probability distribution, solve the DRO model (4.2.1) with a Wasserstein radius of θ, and record optimal solutions $(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}})$. We then solve $Q(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{\xi})$ for all $\boldsymbol{\xi} \in \mathcal{S}$ and compute

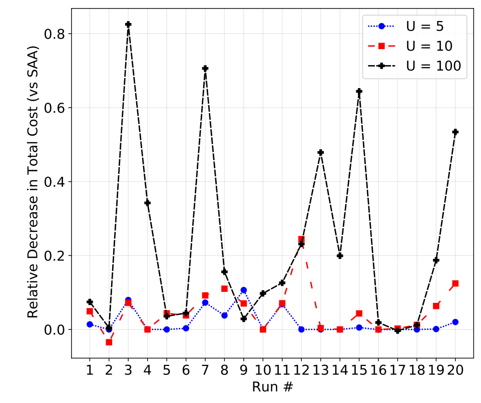
$$\mathbb{E}\left[Q(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{\xi})\right] = \sum_{s \in S} p_s Q(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{\xi}^{\boldsymbol{s}}),$$

where $\{p_s\}_{s \in S}$ is the true, constructed, underlying distribution. We repeat this process 20 times for different sets of samples. We define the out-of-sample cost of run $i \in \{1, \ldots, 20\}$

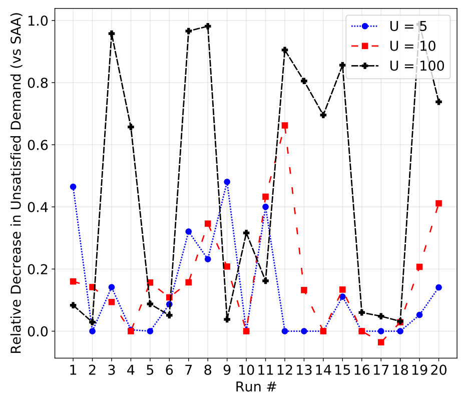
for radius θ as $v_\theta^i = \sum_{i \in I} \left[h_i o_i^\theta + g z_i^\theta\right] + \mathbb{E}\left[Q(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{\xi})\right]$, where v_0^i corresponds to the out-of-sample cost of SAA. For each run i, we record the non-zero Wasserstein radius which leads to the lowest out-of-sample cost as $\theta^* \neq 0$. In Figures 4.7a and 4.7c, we plot the improvement of DRO over SAA in the total out-of-sample cost at θ^*, i.e. we plot $1 - \frac{v_{\theta^*}^i}{v_0^i}$ for all values of U. In Figures 4.7b and 4.7d, we plot the relative decrease in the average unsatisfied demand when using DRO over SAA. The average unmet demand for a fixed first-stage solution is defined as $\sum_{s \in \mathcal{S}} p_s \boldsymbol{u}^{s*}$, where $\boldsymbol{u}^{s*}$ is the unmet demand resulting from solving $Q(\boldsymbol{o}^{\boldsymbol{\theta}}, \boldsymbol{z}^{\boldsymbol{\theta}}, \boldsymbol{\xi}^{\boldsymbol{s}})$.

For $N = 10$, the relative improvement in total cost increases as U increases. For $U = 5$, the improvement over SAA is close to 0 for many runs, with a few runs sitting between about 1.4% and 8% and achieves up to 10% decrease in total cost. The average improvement over the runs is about 2%. For $U = 10$, DRO performs worse than SAA in run 2 with a 3.5% increase in total out-of-sample cost, but outperforms SAA otherwise, with decreases in out-of-sample cost of up to about 24%, with many runs achieving over 4% decrease. The average over the improvements is about 5%. Finally, as we increase U to 100, the decrease in the out-of-sample cost becomes significant, achieving over 80% decrease in one of the runs and an average decrease of 23% over all runs. We observe that even when there is no or very little improvement in the out-of-sample cost, DRO consistently satisfies more demand than SAA for all U. For example, for $U = 10$, DRO satisfies up to 66% more demand, with an average of over 16%.

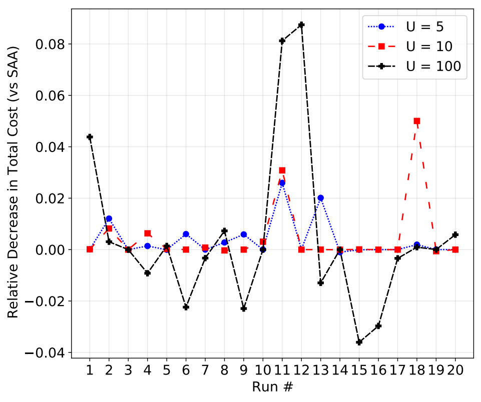
The results tell a different story for $N = 50$. DRO achieves a decrease in out-of-sample cost of about 3% and 5% in two runs when $U = 10$ and up to 2.5% when $U = 5$, but the change is close to 0% in most runs. For $U = 100$, we observe that DRO achieves similar or lower costs than SAA, with up to 8% decrease, but achieves a higher cost in a few runs, up to a 3.6% in total cost. Increasing the penalty parameter U has two potential outcomes: 1) If SAA has a high rate of unmet demand, then DRO has more potential to lead to a decrease in total out-of-sample cost, where a higher value of U means the decrease in the



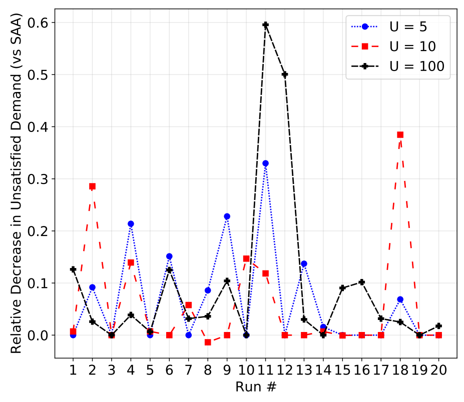
(a) Total Cost Improvement for $N = 10$



(b) Decrease in Unsatisfied Demand for $N = 10$



(c) Total Cost Improvement for $N = 50$



(d) Decrease in Unsatisfied Demand for $N = 50$

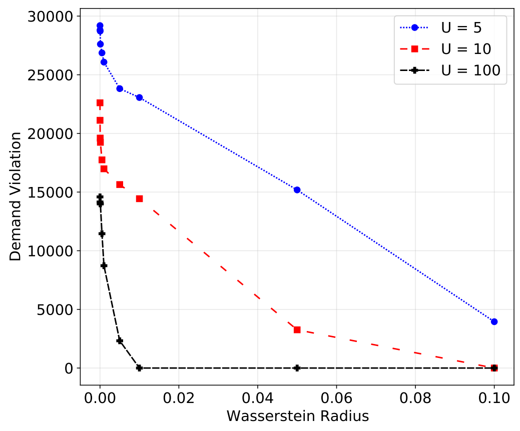
Figure 4.7: Relative improvement of DRO over SAA in total out-of-sample cost and unsatisfied demand at θ^*.

second-stage costs outweights the increase in the first stage; 2) If the sample set contains one or a few scenarios with a high demand, a very high penalty forces SAA to open many facilities and pre-allocate a very large quantity of resources at each facility, leading to an already high cost solution and low unsatisfied demand on average across all scenarios. In the second case, there is less opportunity for DRO to decrease the second-stage cost and make up for any increase in the first-stage cost. We note that the second case is more likely to happen for larger N while the first case is more likely for smaller N as seen in the set of experiments for $N = 10$.

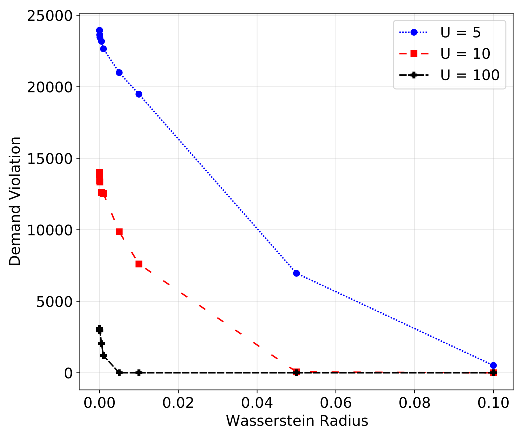
For $N = 50$ and $U = 100$, for run 6, the SAA solution opens 16 facilities and fills almost all of them to full capacity, and the average unsatisfied demand is about 91 units. At $\theta =$ 1e-5, DRO pre-allocates an additional 46,882 units across facilities with remaining capacity, and at $\theta =$ 5e-5, DRO opens a new facility and pre-allocates an additional 73,672 units compared to $\theta =$ 1e-5 (or about 120,555 more units compared to SAA). This leads to a significant increase in the first-stage cost, with little benefits (on average) in the second stage. For $\theta =$ 1e-5 and $\theta =$ 5e-5, the average unmet demand is close to 80 and 64, respectively. We observe that for a high value of U, increasing the Wasserstein radius leads to more drastic changes in the solution. For runs 15 and 16, SAA open 5 facilities filling them to almost full capacity. Increasing the radius to $\theta =$ 1e-5 leads to an additional opened facility, and all six facilities at almost full capacity. This suggests that for high values of U, smaller incrememnts in the Wasserstein radius could yield more useful solutions. Indeed, in Figure 4.8, we can see how the average unmet demand, averaged across the 20 runs, quickly reaches 0 for higher values of U as we increase θ. Regardless, it seems solutions are more unstable as evidenced by the runs where SAA opens over 10 facilities and allocating resources to almost full capacity everywhere. One can also circumvent such solutions by applying a budget on the number of facilities or first-stage cost in the set B in the first stage (see model 4.2.1).

Although the improvement of DRO over SAA in total out-of-sample cost is not significant for $N = 50$, we see that DRO satisfies significantly more demand in the second stage across all runs and all values of U as seen in Figure 4.7d, similar to the sets of experiments for $N = 10$. DRO satisfies up to 30% more demand when $U = 5$, up to 38% when $U = 10$, and up to 60% when $U = 100$.

Finally, we report the first-stage solutions obtained for one run when $N = 10$ and $N = 50$ in tables 4.3 and 4.4, respectively, to illustrate how solutions might evolve as the radius of the Wasserstein ball increases. We can see how the model opens more facilities and pre-allocate more resources as the radius increases, as expected. For $N = 10$, the



(a) $N = 10$



(b) $N = 50$

Figure 4.8: Average unsatisfied demand as a function of the radius of the Wasserstein ball θ for $N = 10$ and $N = 50$, averaged across all runs.

model initially only opens one facility in Biloxi at $\theta = 0$ and one in Mobile at $\theta = 1\text{e-}5$, hedging against disasters in the Louisianna area, before opening a facility in Jacksonville when $\theta = 5\text{e-}5$ and Savannah when $\theta = 5\text{e-}4$. For $N = 50$, the model covers similar areas for all θ, but mostly just increases the amount pre-allocated at the facilities and opens new facilities in the same area.

We note that in practice, θ^* is of course unknown. However, it is generally the case in our experiments that smaller changes in the radius and in the solution of SAA lead to better out-of-sample cost. As we increase the radius, once there is a drastic jump in the first-stage cost caused by a large amount of additional resources pre-allocated or additional facilities opened, the out-of-sample cost tends to get worse (depending on U), as the first-stage costs incurred outweigh the decrease in the second-stage costs. The radius of the Wasserstein ball provides a convenient lever of risk aversness to decision-makers to control the trade-off between first-stage costs and potential second-stage costs.

4.6 Conclusions

The lack of available data in natural disaster management applications can potentially lead to optimistic solutions and poor out-of-sample performance when using classical stochas-

Table 4.3: First-stage solutions of run 20, showing the amount of pre-allocated resources at each open facility (facilities with 0 pre-allocated resources are closed), where $N = 10$ and $U = 10$ for various Wasserstein radii θ.

Facility \ θ	0	1e-05	5e-05	1e-4	5e-4	1e-3	5e-3	0.01
4	0	0	0	0	0	0	131837	131837
10	0	0	0	0	0	0	0	131837
12	0	0	91926	91926	91926	91926	131837	0
13	91926	0	0	0	0	0	0	0
14	0	131837	0	0	0	0	0	0
22	0	0	0	0	63543	0	0	0
25	0	0	63543	63543	0	63543	125521	125499

Table 4.4: First-stage solutions of run 20, showing the amount of pre-allocated resources at each open facility (facilities with 0 pre-allocated resources are closed), where $N = 50$ and $U = 10$ for various Wasserstein radii θ.

Facility \ θ	0	1e-05	5e-05	1e-4	5e-4	1e-3	5e-3	0.01
10	0	0	0	0	0	0	100942	125923
11	69438	89299	89299	0	71327	71327	0	0
13	0	0	0	90491	0	0	0	0
23	0	0	0	0	127448	127448	47420	0
24	0	0	0	0	0	0	94128	78615
25	91345	92536	92536	91345	0	0	0	88031

tic programming methods such as the Sample Average Approximation. In this chapter, we tackle a two-stage distributionally robust optimization model under the Wasserstein ambiguity set, where the first stage is a facility location problem, where we decide facility locations and how much resources to pre-allocate, and where the second stage is a fixed-charge transportation problem. We develop an efficient column-and-constraint generation algorithm, where we leverage the structure of our support set and second-stage value function to efficiently solve the separation problem. Specifically, we show conditions under which the second-stage value function is concave with respect to the demand component of our random vector and show how our results extend to the case where the second stage is a fixed-charge network flow problem. We also provide guidance on numerical improvements which can significantly impact computational efficiency. Finally, we conduct extensive computational experiments on synthetic instances, showing strong computational performance compared to (mostly) classical column-and-constraint generation algorithms where we are able to solve instances with a very large set of scenarios, and analyze the solutions obtained from our DRO model on a case study of hurricane threats on the coastal states of the United States. We show that our DRO model achieves lower out-of-sample costs than SAA in many instances, and consistently satisfies significantly more demand after a disaster occurs, even when the out-of-sample costs are comparable.

CHAPTER 5
CONCLUSIONS

It is becoming increasingly important to understand and leverage available data in decision-making problems. With the ubiquity of uncertainty, assuming a deterministic model is likely to lead to arbitrarily poor solutions. Although stochastic optimization comes with a computational cost, the ability to model uncertainty in the decision-making process has proven invaluable in applications across a wide variety of fields, such as in logistics and supply chain, healthcare, finance, or power systems.

The second chapter of this thesis studies a closely related method to the Benders decomposition algorithm often used in two-stage stochastic proramming. More specifically, we address privacy concerns and decentralized storage of data in Dantzig-Wolfe decomposition, i.e. the dual of Benders. We present a consensus-based solution based on applying ADMM to the dual of the master problem at each iteration, thus preserving privacy of information and leading to a fully distributed method. We show that we can easily recover high-quality primal solutions, and provide bounds on the optimality gap and feasibility violation. We illustrate our methods on synthetic instances and cutting stock problems from the literature. Although our goal is to address privacy concerns and decentralized storage of data, our experiments show that the computational price to handle such issues is negligible.

The greater portion of this thesis tackles two-stage stochastic programming in the presence of limited data, where we switch to a distributionally robust optimization (DRO) paradigm under the Wasserstein ambiguity set. We first handle two-stage DRO with zero-one uncertainties where both stages are allowed to be convex conic programs. We are interested in applications where zero-one random variables represent failures in the system which are rare but have a high impact. We provide reformulations based on bilinear programming and penalty methods, which we then approximate using lift-and-project techniques, and provide means to efficiently solve the resulting approximations. We demonstrate the effectiveness of our method on the optimal power flow problem and a multi-commodity network design problem, where we compare the out-of-sample performance of our approximations of the two-stage DRO to SAA and robust optimization, and perform sensitivity analysis on the rarity of failure occurences and the impact of the failures, where we vary the probability of failure and the penalty of violating constraints in the second stage (i.e. unsatisfied demand), respectively.

In Chapter 4, we study a two-stage DRO model under the Wasserstein set where binary variables are present in both stages. The presence of binary variables in the second stage significantly complicates the solution of the model, where traditional techniques can only be applied as a relaxation. We are specifically focused on a disaster relief application, where the first stage is a facility location problem, opening facilities and pre-allocating resources before observing a disaster, and the second stage is a fixed-charge transportation problem, routing the resources to affected areas after observing the disaster. We first construct a finite support set that is tailored to disaster management applications, leading to a result that the optimal cost of the second-stage problem is concave with respect to a subset of the uncertainty, namely the demand component which tends to have the highest cardinality. We then develop a column-and-constraint generation algorithm, where we are able to efficiently generate new scenarios via a line search scheme. We show our results extend to the case where the second stage is a fixed-charge network flow problem. We show that our proposed method provides a significant computational speedup, and illustrate the benefits of considering a DRO model under the Wasserstein set over SAA.

A drawback of using the Wasserstein ambiguity set is the selection of the radius of the ball, θ. While our selection was somewhat guided by theory in Chapter 3, it is generally not straightforward to determine what constitutes a good radius in practice. While cross-validation might be a good solution as described in chapters 3 and 4, it is unclear how well it would perform for small sample sizes, which is the original motivation of using

DRO. In Chapter 4, we observe, however, that one can examine the resulting solutions for various radii levels to determine a solution which makes sense for the application. Indeed, using DRO under the Wasserstein ambiguity set provides decision-makers with a control over risk-aversness, permitting them to compare various solutions. This is often desired in practice.

While radius selection, finite sample guarantees, and methodologies to efficiently solve distributionally robust problems are the subject of many papers in the literature, a relatively less explored area is the effect of the underlying distance metric of the support set on the out-of-sample performance when using a Wasserstein set. While we make note of a potential benefit from a modeling perspective in Chapter 4, the benefit on the out-of-sample performance of using a distance metric that is tailored to the application is unclear. For example, there exist more sophisticated distances defined on graphs which can potentially be used as the underlying metric of the support set.

Finally, we note that many two-stage network optimization problems exhibit a special structure with respect to the first-stage decision or random vector. For example, under certain conditions, the fixed-charge transportation problem from Chapter 4 is supermodular with respect to the binary decision vector of opening facilities. Submodularity and supermodularity can potentially lead to algorithmic improvements in solving the master problem, or the development of a cutting plane procedure using submodular cuts, for example.

Appendices

APPENDIX A
DATA-DRIVEN TWO-STAGE CONIC OPTIMIZATION WITH RARE HIGH-IMPACT ZERO-ONE UNCERTAINTIES

A.1 Benders Decomposition

The central idea in Benders decomposition is to solve the convex hull reformulation (3.3.3) by iteratively refining an inner approximation of the value function $Z_i(\boldsymbol{x}, \alpha)$, for each $i \in [N]$. We generically write the latter as $Z_i(\boldsymbol{x}, \alpha) = \max_{\boldsymbol{z} \in \mathcal{Z}_i} \boldsymbol{\gamma}(\boldsymbol{x}, \alpha)^\top \boldsymbol{z}$. Recall that the latter optimization problem can be formulated as an MICP, in one of two equivalent forms, by using the linearized reformulation (see Section 3.3.1) or the penalty reformulation (see Section 3.3.2). To present the Benders decomposition algorithm, we re-write (3.3.3) as a semi-infinite program:

$$\begin{aligned} \min_{\boldsymbol{x} \in X,\, \alpha \geq 0} \quad & c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i=1}^{N} \sigma_i \\ \text{s.t.} \quad & \sigma_i \geq \boldsymbol{\gamma}(\boldsymbol{x}, \alpha)^\top \boldsymbol{z}, \quad \forall \boldsymbol{z} \in \mathcal{Z}_i, i \in [N]. \end{aligned}$$

Observe that, in both cases where an MICP representation is possible, $\boldsymbol{\gamma}(\boldsymbol{x}, \alpha)$ is componentwise convex and $\mathcal{Z}_i \subseteq \mathbb{R}^n_+$; therefore, each of the semi-infinite constraints in the above problem defines a convex feasible region (in $\boldsymbol{x}$, α and $\boldsymbol{\sigma}$). We present the algorithm next.

1. Initialize $\hat{\mathcal{Z}}_i = \emptyset$, for each $i \in [N]$.

2. Solve the master problem:

$$\begin{aligned} \min_{\boldsymbol{x} \in X,\, \alpha \geq 0,\, \boldsymbol{\sigma} \in \mathbb{R}^N} \quad & c(\boldsymbol{x}) + \alpha\theta + \frac{1}{N}\sum_{i=1}^{N} \sigma_i \\ \text{s.t.} \quad & \sigma_i \geq \max\left\{ \mathcal{Q}(\boldsymbol{x}, \hat{\boldsymbol{\xi}}^i), \boldsymbol{\gamma}(\boldsymbol{x}, \alpha)^\top \boldsymbol{z} \right\}, \quad \forall \boldsymbol{z} \in \hat{\mathcal{Z}}_i, i \in [N]. \end{aligned} \tag{A.1.1}$$

Let $(\boldsymbol{x}^\star, \alpha^\star, \boldsymbol{\sigma}^\star)$ denote an optimal solution.

3. For each $i \in [N]$, solve:

$$\underset{\boldsymbol{z} \in \mathcal{Z}_i}{\text{maximize}} \; \boldsymbol{\gamma}(\boldsymbol{x}^\star, \alpha^\star)^\top \boldsymbol{z} \tag{A.1.2}$$

   Let $\boldsymbol{z}^{\star,i}$ denote an optimal solution.

4. For each $i \in [N]$, if $\boldsymbol{\gamma}(\boldsymbol{x}^\star, \alpha^\star)^\top \boldsymbol{z}^{\star,i} > \sigma_i^\star$, add $\boldsymbol{z}^{\star,i}$ to $\hat{\mathcal{Z}}_i$.

   If $\hat{\mathcal{Z}}_i$ was not updated for any $i \in [N]$, stop. Otherwise, go to Step 2.

Note that the subproblem (A.1.2) can be solved as an MICP by using either the McCormick linearization or the penalty-based formulation from Section 3.3.

A.2 Proofs

Proof. Proof of Theorem 5. Under the stated assumptions of (A1) complete and (A2) sufficiently expensive recourse, strong duality holds between $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ and its dual $\mathcal{Q}_d(\boldsymbol{x}, \boldsymbol{\xi})$. Along with the fact that $d(\boldsymbol{\xi}, \hat{\boldsymbol{\xi}}^{(i)}) = \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^{(i)} \right\|$ is induced by a norm, the result from Lemma 4 allows us to equivalently reformulate the distributionally robust two-stage problem (3.2.1) in the form (3.3.3), where

$$Z_i(\boldsymbol{x}, \alpha) = \sup_{\boldsymbol{\xi} \in \Xi} \left\{ \mathcal{Q}_d(\boldsymbol{x}, \alpha) - \alpha \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \right\}.$$

By substituting the expression for $\mathcal{Q}_d(\boldsymbol{x}, \boldsymbol{\xi})$ from (3.3.1) and introducing the epigraphical variable τ, we obtain

$$Z_i(\boldsymbol{x}, \alpha) = \max_{\boldsymbol{\xi} \in \Xi, \boldsymbol{\lambda} \in \mathbb{R}^L_+, \tau \in \mathbb{R}_+} \left\{ [\boldsymbol{T}(\boldsymbol{x})\boldsymbol{\xi} + \boldsymbol{h}(\boldsymbol{x})]^\top \boldsymbol{\lambda} - \alpha\tau : \begin{array}{l} \left\| \boldsymbol{\xi} - \hat{\boldsymbol{\xi}}^i \right\| \le \tau, \\ \boldsymbol{q}(\boldsymbol{\xi}) - \boldsymbol{W}(\boldsymbol{\xi})^\top \boldsymbol{\lambda} \in \mathcal{Y}^* \end{array} \right\}.$$

Next, we *(i)* use the affinity of $\boldsymbol{q}$ and $\boldsymbol{W}$: $\boldsymbol{q}(\boldsymbol{\xi}) = \boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi}$ and $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}_0 + \sum_{j \in [M]} \xi_j \boldsymbol{W}_j$, *(ii)* linearize the products $\boldsymbol{\lambda}\boldsymbol{\xi}^\top$ by setting them equal to (the new variable) $\boldsymbol{\Lambda}$, and *(iii)* use the definition of the norm cone $\mathcal{C}^{M+1} = \left\{(\boldsymbol{\xi}, \tau) \in \mathbb{R}^M \times \mathbb{R} : \|\boldsymbol{\xi}\| \leq \tau\right\}$ to obtain

$$Z_i(\boldsymbol{x}, \alpha) = \max_{(\boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\Lambda}, \tau) \in \mathcal{Z}_i} \left\{\langle \boldsymbol{T}(\boldsymbol{x}), \boldsymbol{\Lambda} \rangle + \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\lambda} - \alpha\tau\right\},$$

where $\mathcal{Z}_i$ is defined in (3.3.4b). The objective function of this maximization problem is linear in its decision variables. Therefore, we can equivalently replace the feasible region with the closure of its convex hull to obtain the stated reformulation. □

Proof. Proof of Proposition 4. Observe that $\mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi}^r) \geq \mathcal{Q}(\boldsymbol{x}, \boldsymbol{\xi})$ for all $\boldsymbol{\xi} \in \Xi$ and all $\boldsymbol{x} \in \mathcal{X}$. Indeed, the objective function of the problem on the left-hand side is greater than that on the right-hand side: $(\boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi}^r)^\top \boldsymbol{y} \geq (\boldsymbol{q}_0 + \boldsymbol{Q}\boldsymbol{\xi})^\top \boldsymbol{y}$ for all $\boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}_+^{N_2}$. Also, the feasible region of the problem on the left-hand side is a superset of the one on the right: $\boldsymbol{W}_0\boldsymbol{y} \geq \boldsymbol{T}_0\boldsymbol{\xi}^r + \boldsymbol{h}(\boldsymbol{x}) \geq \boldsymbol{T}_0\boldsymbol{\xi} + \boldsymbol{h}(\boldsymbol{x})$. Therefore, $\boldsymbol{\xi}^r$ is a worst-case realization of the parameters independent of the first-stage decision $\boldsymbol{x} \in \mathcal{X}$. □

APPENDIX B
BINARY DISTRIBUTIONALLY ROBUST OPTIMIZATION UNDER THE WASSERSTEIN SET: A DISASTER RELIEF APPLICATION

Table B.1: List of cities corresponding to nodes in Figure 4.6 (from [126]).

0	Brownsville, TX	15	Birmingham
1	Corpus Christi, TX	16	Nashville
2	San Antonio, TX	17	Atlanta, GA
3	Dallas Ft. Worth, TX	18	Columbia, SC
4	Houston, TX	19	Charlotte, NC
5	Little Rock, AR	20	Wilmington, NC
6	Memphis, TN	21	Charleston, SC
7	Jackson, MS	22	Savannah, GA
8	Monroe, LA	23	Tallahassee, FL
9	Lake Charles, LA	24	Int. I10 & I75, FL
10	Baton Rouge, LA	25	Jacksonville, FL
11	Int. I10 & I55, LA	26	Orlando, FL
12	New Orleans, LA	27	Tampa, FL
13	Biloxi, MS	28	Miami, FL
14	Mobile, AL	29	Key West, FL

REFERENCES

[1] M. E. Tonbari and S. Ahmed, "Consensus-based dantzig-wolfe decomposition", *arXiv preprint arXiv:1905.03309*, 2021.

[2] A. Subramanyam, M. E. Tonbari, and K. Kim, "Data-driven two-stage conic optimization with rare high-impact zero-one uncertainties", *arXiv preprint arXiv:2001.04934*, 2020.

[3] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems", *Numerische mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[4] C. C. Carøe and R. Schultz, "Dual decomposition in stochastic integer programming", *Operations Research Letters*, vol. 24, no. 1-2, pp. 37–45, 1999.

[5] R. T. Rockafellar and R. J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty", *Mathematics of operations research*, vol. 16, no. 1, pp. 119–147, 1991.

[6] G. B. Dantzig and P. Wolfe, "The decomposition algorithm for linear programs", *Econometrica: Journal of the Econometric Society*, pp. 767–778, 1961.

[7] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs", *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[8] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro, "The sample average approximation method applied to stochastic routing problems: A computational study", *Computational Optimization and Applications*, vol. 24, no. 2-3, pp. 289–333, 2003.

[9] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello, "The sample average approximation method for stochastic discrete optimization", *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 479–502, 2002.

[10] D. Bertsimas, V. Gupta, and N. Kallus, "Robust sample average approximation", *Mathematical Programming*, vol. 171, no. 1, pp. 217–282, 2018.

[11] A. Ben-Tal and A. Nemirovski, "Robust optimization–methodology and applications", *Mathematical programming*, vol. 92, no. 3, pp. 453–480, 2002.

[12] A. Shapiro, "Tutorial on risk neutral, distributionally robust and risk averse multistage stochastic programming", *Available at Optimization Online*, 2018.

[13] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review", *arXiv preprint arXiv:1908.05659*, 2019.

[14] Z. Chen, M. Sim, and H. Xu, "Distributionally robust optimization with infinitely constrained ambiguity sets", *Operations Research*, vol. 67, no. 5, pp. 1328–1344, 2019.

[15] W. Wiesemann, D. Kuhn, and M. Sim, "Distributionally robust convex optimization", *Operations Research*, vol. 62, no. 6, pp. 1358–1376, 2014.

[16] X. Yu and S. Shen, "Multistage distributionally robust mixed-integer programming with decision-dependent moment-based ambiguity sets", *Mathematical Programming*, pp. 1–40, 2020.

[17] E. Delage and Y. Ye, "Distributionally robust optimization under moment uncertainty with application to data-driven problems", *Operations research*, vol. 58, no. 3, pp. 595–612, 2010.

[18] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations", *Mathematical Programming*, vol. 171, no. 1-2, pp. 115–166, 2018.

[19] A. Ben-Tal, D. Den Hertog, A. De Waegenaere, B. Melenberg, and G. Rennen, "Robust solutions of optimization problems affected by uncertain probabilities", *Management Science*, vol. 59, no. 2, pp. 341–357, 2013.

[20] D. Love and G. Bayraksan, "Phi-divergence constrained ambiguous stochastic programs for data-driven optimization", *Technical report, Department of Integrated Systems Engineering, The Ohio State University, Columbus, Ohio*, 2015.

[21] G. C. Calafiore, "Ambiguous risk measures and optimal robust portfolios", *SIAM Journal on Optimization*, vol. 18, no. 3, pp. 853–877, 2007.

[22] B. Kocuk, "Conic reformulations for kullback-leibler divergence constrained distributionally robust optimization and applications", *arXiv preprint arXiv:2007.05966*, 2020.

[23] R. Gao and A. J. Kleywegt, "Distributionally robust stochastic optimization with wasserstein distance", *arXiv preprint arXiv:1604.02199*, 2016.

[24] D. Kuhn, P. M. Esfahani, V. A. Nguyen, and S. Shafieezadeh-Abadeh, "Wasserstein distributionally robust optimization: Theory and applications in machine learning", in *Operations Research & Management Science in the Age of Analytics*, INFORMS, 2019, pp. 130–166.

[25] D. Wozabal, “A framework for optimization under ambiguity”, *Annals of Operations Research*, vol. 193, no. 1, pp. 21–47, 2012.

[26] M. Bansal, K.-L. Huang, and S. Mehrotra, “Decomposition algorithms for two-stage distributionally robust mixed binary programs”, *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2360–2383, 2018.

[27] D. Bertsimas, V. Gupta, and N. Kallus, “Robust sample average approximation”, *Mathematical Programming*, vol. 171, no. 1-2, pp. 217–282, 2018.

[28] J. Blanchet, Y. Kang, and K. Murthy, “Robust Wasserstein profile inference and applications to machine learning”, *Journal of Applied Probability*, vol. 56, no. 3, pp. 830–857, 2019.

[29] W. Xie, “Tractable reformulations of two-stage distributionally robust linear programs over the type-∞ wasserstein ball”, *Operations Research Letters*, vol. 48, no. 4, pp. 513–523, 2020.

[30] F. Luo and S. Mehrotra, “A decomposition method for distributionally-robust two-stage stochastic mixed-integer conic programs”, *Mathematical Programming*, pp. 1–45, 2021.

[31] A. Ben-Tal, O. El Housni, and V. Goyal, “A tractable approach for designing piecewise affine policies in two-stage adjustable robust optimization”, *Mathematical Programming*, vol. 182, no. 1, pp. 57–102, 2020.

[32] A. Georghiou, D. Kuhn, and W. Wiesemann, “The decision rule approach to optimization under uncertainty: Methodology and applications”, *Computational Management Science*, 2018.

[33] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, “Adjustable robust solutions of uncertain linear programs”, *Mathematical Programming*, vol. 99, no. 2, pp. 351–376, 2004.

[34] D. Bertsimas and V. Goyal, “On the power and limitations of affine policies in two-stage adaptive optimization”, *Mathematical programming*, vol. 134, no. 2, pp. 491–531, 2012.

[35] D. Bertsimas, S. Shtern, and B. Sturt, “Two-stage sample robust optimization”, *Operations Research*, 2021.

[36] K. S. Shehadeh and E. L. Tucker, “A distributionally robust optimization approach for location and inventory prepositioning of disaster relief supplies”, *arXiv preprint arXiv:2012.05387*, 2020.

[37] G. A. Velasquez, M. E. Mayorga, and O. Y. Özaltın, “Prepositioning disaster relief supplies using robust optimization”, *IISE Transactions*, vol. 52, no. 10, pp. 1122–1140, 2020.

[38] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997.

[39] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006, vol. 5.

[40] M. E. Lübbecke and J. Desrosiers, “Selected topics in column generation”, *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[41] G. L. Nemhauser and L. Wolsey, *Integer Programming and Combinatorial Optimization*. Wiley, 1988.

[42] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers”, *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[43] Y. Hong, J. Vaidya, and H. Lu, “Secure and efficient distributed linear programming”, *Journal of Computer Security*, vol. 20, no. 5, pp. 583–634, 2012.

[44] S. Zeadally, A.-S. K. Pathan, C. Alcaraz, and M. Badra, “Towards privacy protection in smart grid”, *Wireless personal communications*, vol. 73, no. 1, pp. 23–50, 2013.

[45] F. Belletti, C. Le Floch, S. Moura, and A. M. Bayen, “Privacy-preserving dual splitting distributed optimization with application to load flattening in california”, in *2015 54th IEEE Conference on Decision and Control (CDC)*, IEEE, 2015, pp. 3355–3360.

[46] M. DeCamp, N. J. Farber, A. M. Torke, M. George, Z. Berger, C. C. Keirns, and L. C. Kaldjian, “Ethical challenges for accountable care organizations: A structured review”, *Journal of general internal medicine*, vol. 29, no. 10, pp. 1392–1399, 2014.

[47] J. M. McWilliams, L. A. Hatfield, M. E. Chernew, B. E. Landon, and A. L. Schwartz, “Early performance of accountable care organizations in medicare”, *New England Journal of Medicine*, vol. 374, no. 24, pp. 2357–2366, 2016.

[48] J. Gondzio, P. Gonzalez-Brevis, and P. Munari, “New developments in the primal-dual column generation technique”, *European Journal of Operational Research*, vol. 224, no. 1, pp. 41–51, 2012.

[49] J. Gondzio and R. Sarkissian, “Column generation with a primal-dual method”, *Logilab Technical Report 96.6, University of Geneva*, 1996.

[50] A. Nedić and A. Ozdaglar, “Approximate primal solutions and rate analysis for dual subgradient methods”, *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[51] E. Gustavsson, M. Patriksson, and A.-B. Strömberg, “Primal convergence from dual subgradient methods for convex optimization”, *Mathematical Programming*, vol. 150, no. 2, pp. 365–390, 2015.

[52] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck, “Comparison of bundle and classical column generation”, *Mathematical Programming*, vol. 113, no. 2, pp. 299–344, 2006.

[53] J. Gondzio, P. Gonzalez-Brevis, and P. Munari, “Large-scale optimization with the primal-dual column generation method”, *Mathematical Programming Computation*, vol. 8, no. 1, pp. 47–82, 2016.

[54] M. Lubin, K. Martin, C. G. Petra, and B. Sandıkçı, “On parallelizing dual decomposition in stochastic integer programming”, *Operations Research Letters*, vol. 41, no. 3, pp. 252–258, 2013.

[55] M. Bürger, G. Notarstefano, and F. Allgöwer, “A polyhedral approximation framework for convex and robust distributed optimization”, *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 384–395, 2013.

[56] O. L. Mangasarian, “Privacy-preserving linear programming”, *Optimization Letters*, vol. 5, no. 1, pp. 165–172, 2011.

[57] W. Li, H. Li, and C. Deng, “Privacy-preserving horizontally partitioned linear programs with inequality constraints”, *Optimization Letters*, vol. 7, no. 1, pp. 137–144, 2013.

[58] Q. Li, R. Heusdens, and M. G. Christensen, “Privacy-preserving distributed optimization via subspace perturbation: A general framework”, *arXiv preprint arXiv:2004.13999*, 2020.

[59] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization”, *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[60] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks”, *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[61] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, “Distributed optimization with local domains: Applications in mpc and network flows”, *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2015.

[62] C. Xi, Q. Wu, and U. A. Khan, “On the distributed optimization over directed networks”, *Neurocomputing*, vol. 267, pp. 508–515, 2017.

[63] A. Nedic and A. Ozdaglar, “Cooperative distributed multi-agent optimization”, in *Convex optimization in signal processing and communications*, D. P. Palomar and Y. C. Eldar, Eds., Cambridge university press, 2010.

[64] M. Hong and Z.-Q. Luo, “On the linear convergence of the alternating direction method of multipliers”, *Mathematical Programming*, vol. 162, no. 1-2, pp. 165–199, 2017.

[65] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, “A general analysis of the convergence of ADMM”, in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15, Lille, France, 2015, pp. 343–352.

[66] X. Cai, D. Han, and X. Yuan, “On the convergence of the direct extension of ADMM for three-block separable convex minimization models with one strongly convex function”, *Computational Optimization and Applications*, vol. 66, no. 1, pp. 39–73, 2017.

[67] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA: SIAM, 2017.

[68] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, “Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems”, *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2015.

[69] B. Wohlberg, “Admm penalty parameter selection by residual balancing”, *arXiv preprint arXiv:1704.06209*, 2017.

[70] C. Chen, B. He, Y. Ye, and X. Yuan, “The direct extension of admm for multi-block convex minimization problems is not necessarily convergent”, *Mathematical Programming*, vol. 155, no. 1-2, pp. 57–79, 2016.

[71] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, “Parallel multi-block admm with o (1/k) convergence”, *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[72] A. Goncalves, X. Liu, and A. Banerjee, “Two-block vs. multi-block admm: An empirical evaluation of convergence”, *arXiv preprint arXiv:1907.04524*, 2019.

[73] Y. Lou, L. Yu, S. Wang, and P. Yi, “Privacy preservation in distributed subgradient optimization algorithms”, *IEEE transactions on cybernetics*, vol. 48, no. 7, pp. 2154–2165, 2017.

[74] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi, “Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2012.

[75] A. Frangioni, “About lagrangian methods in integer optimization”, *Annals of Operations Research*, vol. 139, no. 1, pp. 163–193, 2005.

[76] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo, “Parallel distributed computing using python”, *Advances in Water Resources*, vol. 34, no. 9, pp. 1124–1139, 2011.

[77] L. Dalcın, R. Paz, M. Storti, and J. D’Elıa, “MPI for python: Performance improvements and MPI-2 extensions”, *Journal of Parallel and Distributed Computing*, vol. 68, no. 5, pp. 655–662, 2008.

[78] L. Dalcín, R. Paz, and M. Storti, “MPI for python”, *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1108–1115, 2005.

[79] G. Belov and G. Scheithauer, “A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths”, *European Journal of Operational Research*, vol. 141, no. 2, pp. 274–294, 2002.

[80] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, and Y. Shao, “Alternating criteria search: A parallel large neighborhood search algorithm for mixed integer programs”, *Computational Optimization and Applications*, vol. 69, no. 1, pp. 1–24, 2018.

[81] A. Bitlislioğlu, I. Pejcic, and C. Jones, “Interior point decomposition for multi-agent optimization”, *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 233–238, 2017.

[82] D. Bienstock and A. Verma, “The $N - k$ problem in power grids: New models, formulations, and numerical experiments”, *SIAM Journal on Optimization*, vol. 20, no. 5, pp. 2352–2380, 2010.

[83] North American Electric Reliability Corporation, “Transmission System Planning Performance Requirements”, *TPL-001-4*, 2017.

[84] P. Praks, V. Kopustinskas, and M. Masera, “Monte-carlo-based reliability and vulnerability assessment of a natural gas transmission system due to random network component failures”, *Sustainable and Resilient Infrastructure*, vol. 2, no. 3, pp. 97–107, 2017.

[85] B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang, “Optical network design and restoration”, *Bell Labs Technical Journal*, vol. 4, no. 1, pp. 58–84, 1999.

[86] B. Berche, C. Von Ferber, T. Holovatch, and Y. Holovatch, “Resilience of public transport networks against attacks”, *The European Physical Journal B*, vol. 71, no. 1, pp. 125–137, 2009.

[87] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[88] J. Barrera, T. Homem-de-Mello, E. Moreno, B. K. Pagnoncelli, and G. Canessa, “Chance-constrained problems and rare events: An importance sampling approach”, *Mathematical Programming*, vol. 157, no. 1, pp. 153–189, 2016.

[89] A. Budhiraja, S. Lu, Y. Yu, and Q. Tran-Dinh, “Minimization of a class of rare event probabilities and buffered probabilities of exceedance”, *Annals of Operations Research*, vol. 302, no. 1, pp. 49–83, 2021.

[90] K. Postek, D. den Hertog, and B. Melenberg, “Computationally tractable counterparts of distributionally robust constraints on risk measures”, *SIAM Review*, vol. 58, no. 4, pp. 603–650, 2016.

[91] L. Lovász and A. Schrijver, “Cones of matrices and set-functions and 0–1 optimization”, *SIAM Journal on Optimization*, vol. 1, no. 2, pp. 166–190, 1991.

[92] H. D. Sherali and W. P. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems”, *SIAM Journal on Discrete Mathematics*, vol. 3, no. 3, pp. 411–430, 1990.

[93] J. B. Lasserre, “Global optimization with polynomials and the problem of moments”, *SIAM Journal on optimization*, vol. 11, no. 3, pp. 796–817, 2001.

[94] G. A. Hanasusanto and D. Kuhn, “Conic programming reformulations of two-stage distributionally robust linear programs over Wasserstein balls”, *Operations Research*, vol. 66, no. 3, pp. 849–869, 2018.

[95] G. Xu and S. Burer, “A copositive approach for two-stage adjustable robust optimization with uncertain right-hand sides”, *Computational Optimization and Applications*, vol. 70, no. 1, pp. 33–59, 2018.

[96] A. Ardestani-Jaafari and E. Delage, “Linearized robust counterparts of two-stage robust optimization problems with applications in operations management”, *Available at Optimization Online*, 2017.

[97] A. Mittal, C. Gokalp, and G. A. Hanasusanto, “Robust quadratic programming with mixed-integer uncertainty”, *INFORMS Journal on Computing*, vol. 32, no. 2, pp. 201–218, 2020.

[98] R. Jiang, M. Ryu, and G. Xu, “Data-driven distributionally robust appointment scheduling over wasserstein balls”, *arXiv preprint arXiv:1907.03219*, 2019.

[99] S. Burer and H. Dong, “Representing quadratically constrained quadratic programs as generalized copositive programs”, *Operations Research Letters*, vol. 40, no. 3, pp. 203–206, 2012.

[100] G. Eichfelder and J. Jahn, “Set-semidefinite optimization”, *Journal of Convex Analysis*, vol. 15, no. 4, pp. 767–801, 2008.

[101] R. A. Stubbs and S. Mehrotra, “A branch-and-cut method for 0-1 mixed convex programming”, *Mathematical Programming*, vol. 86, pp. 515–532, 1999.

[102] M. Çezik and G. Iyengar, “Cuts for mixed 0-1 conic programming”, *Mathematical Programming*, vol. 104, no. 1, pp. 179–202, 2005.

[103] J. Povh and F. Rendl, “Copositive and semidefinite relaxations of the quadratic assignment problem”, *Discrete Optimization*, vol. 6, no. 3, pp. 231–241, 2009.

[104] S. Burer and A. N. Letchford, “Non-convex mixed-integer nonlinear programming: A survey”, *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

[105] V. Gabrel, M. Lacroix, C. Murat, and N. Remli, “Robust location transportation problems under uncertain demands”, *Discrete Applied Mathematics*, vol. 164, pp. 100–111, 2014.

[106] F. Glover, “Improved linear integer programming formulations of nonlinear integer problems”, *Management Science*, vol. 22, no. 4, pp. 455–460, 1975.

[107] A. Thiele, T. Terry, and M. Epelman, “Robust linear optimization with recourse”, Lehigh University, Bethlehem, PA, Tech. Rep., 2009.

[108] C. Zhao, “Data-driven risk-averse stochastic program and renewable energy integration”, PhD thesis, University of Florida, 2014.

[109] B. Zeng and L. Zhao, “Solving two-stage robust optimization problems using a column-and-constraint generation method”, *Operations Research Letters*, vol. 41, no. 5, pp. 457–461, 2013.

[110] E. Balas, S. Ceria, and G. Cornuéjols, “A lift-and-project cutting plane algorithm for mixed 0–1 programs”, *Mathematical Programming*, vol. 58, no. 1-3, pp. 295–324, 1993.

[111] M. Laurent, “A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0–1 programming”, *Mathematics of Operations Research*, vol. 28, no. 3, pp. 470–496, 2003.

[112] O. Alsac and B. Stott, “Optimal load flow with steady-state security”, *IEEE Transactions on Power Apparatus and Systems*, pp. 745–751, 1974.

[113] N. Chiang and A. Grothey, “Solving security constrained optimal power flow problems by a structure exploiting interior point method”, *Optimization and Engineering*, vol. 16, no. 1, pp. 49–71, 2015.

[114] U.S. Energy Information Administration, “Annual Summaries”, *Electric Disturbance Events (OE-417)*, 2017, See also http://insideenergy.org/2014/08/18/data-explore-15-years-of-power-outages/.

[115] S. H. Low, “Convex relaxation of optimal power flow—part i: Formulations and equivalence”, *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 15–27, 2014.

[116] B. Kocuk, S. S. Dey, and X. A. Sun, “Strong socp relaxations for the optimal power flow problem”, *Operations Research*, vol. 64, no. 6, pp. 1177–1196, 2016.

[117] U.S. Department of Energy Advanced Research Projects Agency-Energy, “SCOPF Problem Formulation: Challenge 1”, *Grid Optimization Competition*, 2019.

[118] R. D. Zimmerman and C. E. Murillo-Sánchez, “Matpower 6.0 user’s manual”, *PSERC: Tempe, AZ, USA*, 2016.

[119] S. Babaeinejadsarookolaee, A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang, *et al.*, “The power grid library for benchmarking ac optimal power flow algorithms”, *arXiv preprint arXiv:1908.02788*, 2019.

[120] M. Minoux, “Networks synthesis and optimum network design problems: Models, solution methods and applications”, *Networks*, vol. 19, no. 3, pp. 313–360, 1989.

[121] T. G. Crainic, A. Frangioni, and B. Gendron, “Bundle-based relaxation methods for multicommodity capacitated fixed charge network design”, *Discrete Applied Mathematics*, vol. 112, no. 1, pp. 73–99, 2001.

[122] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network", *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 749–762, 2008.

[123] N. Boland, M. Fischetti, M. Monaci, and M. Savelsbergh, "Proximity benders: A decomposition heuristic for stochastic programs", *Journal of Heuristics*, vol. 22, no. 2, pp. 181–198, 2016.

[124] M. Sabbaghtorkan, R. Batta, and Q. He, "Prepositioning of assets and supplies in disaster operations management: Review and research gap identification", *European Journal of Operational Research*, vol. 284, no. 1, pp. 1–19, 2020.

[125] E. Grass and K. Fischer, "Two-stage stochastic programming in disaster management: A literature survey", *Surveys in Operations Research and Management Science*, vol. 21, no. 2, pp. 85–100, 2016.

[126] C. G. Rawls and M. A. Turnquist, "Pre-positioning of emergency supplies for disaster response", *Transportation research part B: Methodological*, vol. 44, no. 4, pp. 521–534, 2010.

[127] W. Wang, K. Yang, L. Yang, and Z. Gao, "Two-stage distributionally robust programming based on worst-case mean-cvar criterion and application to disaster relief management", *Transportation Research Part E: Logistics and Transportation Review*, vol. 149, p. 102 332, 2021.

[128] K. Kim, "Dual decomposition of two-stage distributionally robust mixed-integer programming under the wasserstein ambiguity set", *Preprint manuscript*, 2020.

[129] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann, "K-adaptability in two-stage robust binary programming", *Operations Research*, vol. 63, no. 4, pp. 877–891, 2015.

[130] A. Subramanyam, C. E. Gounaris, and W. Wiesemann, "K-adaptability in two-stage mixed-integer robust optimization", *Mathematical Programming Computation*, vol. 12, no. 2, pp. 193–224, 2020.

[131] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann, "K-adaptability in two-stage distributionally robust binary programming", *Operations Research Letters*, vol. 44, no. 1, pp. 6–11, 2016.

[132] D. E. Ferguson, "Fibonaccian searching", *Communications of the ACM*, vol. 3, no. 12, p. 648, 1960.

[133] C. W. Landsea and J. L. Franklin, “Atlantic hurricane database uncertainty and presentation of a new database format”, *Monthly Weather Review*, vol. 141, no. 10, pp. 3576–3592, 2013.

[134] E. S. Blake, E. N. Rappaport, J. D. Jarrell, and C. Landsea, “The deadliest, costliest, and most intense united states tropical cyclones from 1851 to 2004 (and other frequently requested hurricane facts)”, *NOAA Technical Memorandum NWS TPC-4*, 2007.