ABSTRACT

Title of Dissertation: WORKING IN REVERSE: ADVANCING INVERSE OPTIMIZATION IN THE FIELDS OF EQUILIBRIUM AND INFRASTRUCTURE MODELING
Stephanie Ann Allen
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Steven A. Gabriel
Department of Mechanical Engineering

Professor John P. Dickerson
Department of Computer Science

Transportation and infrastructure modeling allows us to pursue societal aims such as improved disaster management, traffic flow, and water allocation. Equilibrium programming enables us to represent the entities involved in these applications such that we can learn more about their dynamics. These entities include transportation users and market players. However, determining the parameters in these models can be a difficult task because the entities involved in these equilibrium processes may not be able to articulate or to disclose the parameterizations that motivate them. The field of inverse optimization (IO) offers a potential solution to this problem by taking observed equilibria to these systems and using them to parameterize equilibrium models. In this dissertation, we explore the use of inverse optimization to parameterize multiple new or understudied subclasses of equilibrium problems as well as expand inverse optimization's application to new infrastructure domains. In the first project of our dissertation, our contribution

to the literature is to propose that IO can be used to parameterize cost functions in multi-stage stochastic programs for disaster management and can be used in disaster support systems. We demonstrate in most of our experiments that using IO to obtain the hidden cost parameters for travel on a road network changes the protection decisions made on that road network when compared to utilizing the mean of the parameter range for the hidden parameters (also referred to as "uniform cost"). The protection decisions made under the IO cost parameterizations versus the true cost parameterizations are similar for most of the experiments, thus lending credibility to the IO parameterizations. In the second project of our dissertation, we extend a well-known framework in the IO community to the case of jointly convex generalized Nash equilibrium problems (GNEPs). We demonstrate the utility of this framework in a multi-player transportation game in which we vary the number of players, the capacity level, and the network topology in the experiments as well as run experiments assuming the same costs among players and different costs among players. Our promising results provide evidence that our work could be used to regulate traffic flow toward aims such as reduction of emissions. In the final project of our dissertation, we explore the general parameterization of the constant vector in linear complementarity problems (LCPs), which are mathematical expressions that can represent optimization, game theory, and market models [75]. Unlike the limited previous work on inverse optimization for LCPs, we characterize theoretical considerations regarding the inverse optimization problem for LCPs, prove that a previously proposed IO solution model can be dramatically simplified, and handle the case of multiple solution data points for the IO LCP problem. Additionally, we use our knowledge regarding LCPs and IO in a water market allocation case study, which is an application not previously explored in the IO literature, and we find that charging an additional tax on the upstream players enables the market to reach a system optimal. In sum,

this dissertation contributes to the inverse optimization literature by expanding its reach in the equilibrium problem domain and by reaching new infrastructure applications.

WORKING IN REVERSE: ADVANCING INVERSE OPTIMIZATION IN THE FIELDS OF EQUILIBRIUM AND INFRASTRUCTURE MODELING

by

Stephanie Ann Allen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Steven A. Gabriel, Chair/Co-Advisor
Professor John P. Dickerson, Chair/Co-Advisor
Professor Maria K. Cameron
Professor Kimia Ghobadi
Professor Subramanian Raghavan

Foreword

In accordance with the University of Maryland, College Park Graduate School Doctor of Philosophy Degree Policies, Stephanie Allen states in this forward that she has made substantial contributions to all of the coauthored work included in this dissertation. The letter at the beginning of this dissertation also speaks to this. The signed copy of this letter has been sent to and approved by the UMD Graduate School.

Acknowledgments

When I started this journey, I knew that it would be arduous, infuriating, and one of the most emotionally trying experiences of my life. And, I was right, but it ended up being both much more difficult than I could have imagined yet also vastly more fulfilling than I could have dreamed. I found confidence in who I am and what I want out of my life during the last five and a half years, and I honestly can say that I am at peace with myself as I finish this journey. I could not have navigated or traversed this long path though without many people, and I am grateful for the opportunity to thank them here.

To my family: My mother, in particular, helped me survive the pandemic, through many phone calls and virtual art nights. You have always been my biggest cheerleader, and I am truly grateful to have you in my life. My sisters have kept me constantly amused and inspired to continue achieving. And, to my father, thank you for cooking for me during my visits home and for quietly supporting and believing in me.

To my advisors: Steve, you have been with me from the beginning. I am grateful for the researcher you have inspired me to be, and thank you tremendously for giving me the space and grace I needed during some very difficult moments during the process. John, I know we met later in my Ph.D. years, but thank you for teaching me the importance of communicating ideas clearly and continuing to inspire me to do research for the social good. And thank you to you both for helping me to cite a bit less compulsively!

To my collaborators: Thank you to Dr. Daria Terekhov, who has pushed me to be a better researcher and who took the time and interest to work with someone completely outside of her university. Nathan Boyd, thank you for being a great friend with whom to get stuck in London and for being a truly wonderful collaborator.

To my professors: Thank you to Dr. Jacob Bedrossian and Dr. Maria Cameron for providing a rigorous first year series in scientific computation. I'm constantly drawing upon the skills you instilled within me, and I'm grateful for the roles you have both played in my graduate journey. Thank you to Dr. Tom Goldstein and Dr. Howard Elman for informative classes on numerical optimization and numerical linear algebra.

To my friends: Where to begin! I am blessed to have a number of wonderful friends. Luke and Lora, you both provided a refuge when I needed it, and I will never forget your kindness and your grace. Shin, thank you for being an example of perseverance in my life and for introducing me to wonderful food. Sai, you are one of the most generous people I have ever met; thank you for being a kindred spirit with whom I can agree on so many issues. Ishfaaq, Jerry, Noah, and Shashank, thank you for providing so much fun, laughter, and joy during the pandemic over the course of our outdoor get-togethers. Zack, Ben, Vaughn, Gabriel, Al, and the rest of my board game group, thank you for helping me get through the pandemic and bringing more fun into my life on most Friday nights.

Meredith and Paige, thank you for picking up the phone whenever I call to run something by you and for being constants in my life. You are some of my oldest and dearest friends, and I would not be where I am without your support. Anna, thank you for your kindness and your dedication to make the world a better place. Aidan, thank you for visiting me at some key moments in my grad school life and providing a sounding board during those moments.

Tony, I know our journey has just begun, but I am grateful we found each other toward the end of my dissertation. It has been a beautiful experience falling for someone as I close this chapter of my life.

To the UMD Mathematics Department: Thank you to both Jessica and Cristina for helping me navigate the bureacracy of graduate school and for your kindness in doing so.

To Board and Brew: I've spent many hours in your coffee shop working through assignments, working on research, and finding a refuge during difficult moments. Thank you to all the wonderful staff members at this coffee shop for making me feel at home and welcome in this space.

Dedication: And, finally, I dedicate this dissertation to all of those facing barriers in the fight to finish their dissertation journeys and to all of those individuals who are supporting them in those processes. I cannot express enough the magnitude of my gratitude to those who supported me in my struggles as a woman in STEM in the Ph.D. process. I aim to pay this forward.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BPR	Bureau of Public Roads
CI	Confidence Intervals
CSM	Cost-Sharing Market
DSS	Disaster Support System
FEMA	Federal Emergency Management Agency
GAMS	General Algebraic Modeling System
GCM	General Commodity Market
GIS	Geographic Information System
GNEP	Generalized Nash Equilibrium Problem
IO	Inverse Optimization
KKT	Karush–Kuhn–Tucker
LCP	Linear Complementarity Problem
MATLAB	Matrix Laboratory
MCP	Mixed Complementarity Problem
MIP	Mixed-Integer Program
MPEC	Mathematical Program with Equilibrium Constraints
N & D	Nguyen & Dupuis
OD	Origin-Destination
PH	Progressive Hedging
QVI	Quasi-variational Inequality
RHS	Right Hand Side
SNPP	Stochastic Network Protection Problem
VI	Variational Inequality

The Graduate School, University of Maryland

2123 Lee Building, 7809 Regents Drive

College Park, MD 20742

Dear Dean Fetter,

We provide this letter at the beginning of Stephanie Allen's dissertation to confirm that she has made a substantial contribution to the coauthored work contained in this dissertation. This dissertation is composed of three projects.

Project 1 is entitled "A Hybrid Inverse Optimization-Stochastic Programming Framework for Network Protection," and it was authored by Stephanie Allen, Daria Terekhov, and Steven Gabriel. For project 1, Stephanie Allen did the vast majority of the work on this project, from framework conception to code writing to paper writing. She received some advice from Daria Terekhov on implementation of the train-test part of the experiments, but she had brought the idea of these types of experiments herself. Daria Terekhov and Steven Gabriel provided editorial feedback.

Project 2 is entitled "Using Inverse Optimization to Learn Cost Functions in Generalized Nash Games," and it was authored by Stephanie Allen, Steven Gabriel, and John Dickerson. Stephanie Allen did the vast majority of the work on this project, including extending the inverse optimization framework discussed to jointly convex generalized Nash equilibrium problems, forming the transportation game, writing the code, and writing the paper. Steven Gabriel worked out one of the proofs and one of the counter examples in the Appendix, and he talked about a paper in one of his classes that Allen found very useful in extending the IO framework to jointly convex generalized Nash equilibrium problems. Steven Gabriel and John Dickerson provided

editorial comments on the paper.

Finally, project 3 is entitled "Inverse Optimization for Parameterization of Linear Complementarity Problems and for Incentive Design in Markets," and it was authored by Stephanie Allen, Steven Gabriel, and Nathan Boyd. Stephanie Allen did the vast majority of the work on this project, including the new theory and proofs, the majority of the coding for the case study, and the writing of the paper. Steven Gabriel provided some suggestions for additional avenues to explore for the theory as well as provided much editorial feedback. Nathan Boyd helped Stephanie Allen with understanding the application behind the case study as well as interpreting results obtained from solving the linear complementarity problems involved. Nathan Boyd also provided editorial feedback on the case study section.

We provide the signatures of the co-advisors of Stephanie Allen as well as the signature of her Graduate Program Director.

Sincerely,

Stephanie Allen

Dr. Steven A. Gabriel, Co-Advisor

Dr. John P. Dickerson, Co-Advisor

Dr. Howard Elman, AMSC Graduate Director

Chapter 1: Introduction

Transportation and infrastructure modeling allows us to pursue societal aims such as improved disaster management, traffic flow, and water allocation. Equilibrium programming enables us to represent the entities involved in these applications such that we can learn more about their dynamics. These entities include transportation users and market players. However, determining the parameters in these models can be a difficult task because the entities involved in these equilibrium processes may not be able to articulate or to disclose the parameterizations that motivate them. The field of inverse optimization (IO) offers a potential solution to this problem.

1.1 Background: Equilibrium Problems

Before discussing inverse optimization, we provide some background on equilibrium problems in general. First, we describe the concept of a Nash equilibrium, which applies to games in which the players have some kind of "market power" or, in other words, the ability to affect each other and the outcome of the game [75]. Indeed, with entities and companies in our world gaining more market power over time, Nash games are becoming more and more relevant. For convex K_i and convex $f_i(\mathbf{x}_i)$, we can write player i's problem as [75]:

$$\min_{\mathbf{x}_i \in K_i} f_i(\mathbf{x}_i, \mathbf{x}_{-i}) \tag{1.1.1}$$

According to Gabriel et al. [75], a Nash equilibrium $\mathbf{x}^*$ "has the property that, for every i, $\mathbf{x}_i^*$ solves (1.1.1) given $\mathbf{x}_{-i} = \mathbf{x}_{-i}^*$." Essentially, this means that x_i^* is a Nash equilibrium if it is the optimal solution to player i's optimization problem assuming that the other players in the game do not change their decision variables. We can find an Nash equilibrium for the above problem using a relationship called a variational inequality in which we aim to find an $\mathbf{x}^*$ such that [75]:

$$G(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0, \quad \forall \mathbf{x} \in K = \prod_i K_i \tag{1.1.2a}$$

$$G(\mathbf{x}^*) = \begin{bmatrix} \nabla_{\mathbf{x}_1} f_1(\mathbf{x}) \\ \vdots \\ \nabla_{\mathbf{x}_I} f_I(\mathbf{x}) \end{bmatrix} \tag{1.1.2b}$$

Thus, there is an equivalence between a solution to this variational inequality (VI) and a Nash equilibrium point for the multi-player game [75]. The variational inequality can also represent other equilibrium phenomenon such as the traffic equilibrium principle by Wardrop [190]. Smith [167] prove that an adjusted version of this variational inequality problem encodes the properties of the Wardrop equilibrium principle, with the G function representing the cost of flow on either paths or arcs.

We can then transform the VI to what is known as a mixed complementarity problem. Given that G and g are convex and continuously differentiable, h is affine, and a constraint qualification holds, both the following VI and MCP-VI produce the same solution sets [75, 93].

VI(K, G): Find $\mathbf{x}^* \in K$ such that:

$$G(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in K = \{\mathbf{x} \in \mathbb{R}^n | g(\mathbf{x}) \leq 0,\ h(\mathbf{x}) = 0\} \tag{1.1.3}$$

MCP-VI: Find $\mathbf{x}^*$, $v^* \in \mathbb{R}^m$, and $\beta^* \in \mathbb{R}^p$ such that:

$$G(\mathbf{x}^*) + \nabla g(\mathbf{x}^*)^T v^* + \nabla h(\mathbf{x}^*)^T \beta^* = 0, \quad x \text{ is free.} \tag{1.1.4a}$$

$$0 \geq g(\mathbf{x}^*) \perp v^* \geq 0 \tag{1.1.4b}$$

$$h(\mathbf{x}^*) = 0,\ \beta^* \text{ is free.} \tag{1.1.4c}$$

The value of being able to transform the VI into a mixed complementarity problem (MCP) is that it allows us to stack the KKT conditions of Nash problems into one MCP and use the PATH solver [54] in GAMS to solve the problem [75]. Sometimes, we are able to simplify the MCP into what is known as a linear complementarity problem (LCP) which can be defined for nonnegative variables $\mathbf{z} \in \mathbb{R}^n_{\geq 0}$ and parameters $M \in \mathbb{R}^{n \times n}$ and $q \in \mathbb{R}^n$ as follows [48]:

$$0 \leq \mathbf{z} \perp M\mathbf{z} + q \geq 0 \tag{1.1.5}$$

The $\perp$ indicates the inner product between the two vectors above. We further explore the parameterization of this LCP in Chapter 4.

1.2 Background: Inverse Optimization

Inverse optimization allows a user to parameterize particular functions in optimization and equilibrium problems using solutions to these problems [3, 21, 205]. This means we can take observed data regarding phenomena in the world and use this data to find the parameters that define the motivating functions at the heart of the models that represent the phenomena. Mathematically, we represent this as follows. For optimization problems, we have the classic problem of, given some function $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ and set $\mathcal{S}$, find $\mathbf{x}$ that solves the following problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1.2.1a}$$

$$\mathbf{x} \in \mathcal{S} \tag{1.2.1b}$$

Conversely, in part of the inverse optimization literature, we take a solution $\mathbf{x}$ [3, 37] or a set of solutions $\mathcal{X}$ [13, 112] and use them to find the parameters of the function $f(\mathbf{x})$. Another part of the inverse optimization literature uses the solutions to find the parameters for the constraints defining $\mathcal{S}$ [36, 82, 161, 174]. What's more, additional literature has explored explicitly dealing with when the solution(s) provided are noisy [8, 132], with the statistical properties of the inverse optimization problem [8], and with robust versions of this problem [81]. For equilibrium problems, we use the variational inequality (VI) formulation from Bertsimas et al. [21] to describe inverse optimization for these types of problems. In a VI problem, we find a $\hat{\mathbf{x}} \in \mathbb{R}^n$ such that the following relationship is satisfied for function $F(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and set $\mathcal{F} \subseteq \mathbb{R}^n$:

$$F(\hat{\mathbf{x}})^T(\mathbf{x} - \hat{\mathbf{x}}) \geq 0, \ \forall \mathbf{x} \in \mathcal{F} \tag{1.2.2}$$

We know from Gabriel et al. [75] that the VI generalizes many game theory and market problems which are themselves equilibrium problems and generalizes their mathematical structures including linear complementarity problems (LCPs) and mixed complementarity problems (MCPs). As Bertsimas et al. [21] demonstrate, one part of the inverse optimization literature aims to parameterize the $F(\mathbf{x})$ function using solution(s) to this problem [4, 21, 154]. Other work focuses upon parameterizing both $F(\mathbf{x})$ and $\mathcal{F}$ [103]. See [38] for a recent review of the inverse optimization literature. Our work falls into the equilibrium category of inverse optimization.

1.3 Summary of the Dissertation

In this dissertation, we explore the use of inverse optimization to parameterize multiple new or understudied subclasses of equilibrium problems as well as expand inverse optimization's application to new infrastructure domains. Indeed, we pursue a combination of extending methods in the literature after being inspired by applications and applying methods in the literature to new applications. Using inverse optimization to parameterize new or understudied subclasses of equilibrium problems is important because it allows us to parameterize the entities involved in these systems to better understand their motivations. Introducing inverse optimization to new applications extends the field's use and involved more communities in its development. This dissertation is composed of three projects in total, which comprise the next three chapters.

In the first project of our dissertation [5], our contribution to the literature is to propose that IO can be used to parameterize cost functions in multi-stage stochastic programs for disaster

management and can be used in disaster support systems. Specifically, the multi-stage stochastic program that we parameterize is a challenging problem because it is a stochastic optimization problem with a traffic equilibrium problem embedded within it, forming a mathematical problem with equilibrium constraints (MPEC) problem. MPECs are mathematical programming structures that embed equilibrium problems and/or optimization problems within their constraints [75]. We use inverse optimization on a separate traffic equilibrium problem to obtain parameters for the travel cost functions inherent in the objective function and equilibrium constraints of this MPEC. Through computational experimentation, we demonstrate in most of our experiments that using IO to obtain the hidden cost parameters for travel on a road network changes the protection decisions made on that road network when compared to utilizing the mean of the parameter range for the hidden parameters (also referred to as "uniform cost"). The protection decisions made under the IO cost parameterizations versus the true cost parameterizations are similar for most of the experiments, thus lending credibility to the IO parameterizations.

In the second project of our dissertation [4], we extend a well-known framework in the IO community to the case of jointly convex generalized Nash equilibrium problems (GNEPs). In particular, we take the inverse optimization techniques of Keshavarz et al. [112] and Ratliff et al. [154] and integrate an additional term to account for the jointly convex constraints, justifying and proving this addition using GNEP and VI theory. We demonstrate the utility of this framework in a multi-player transportation game in which we vary the number of players, the capacity level, and the network topology in the experiments as well as run experiments assuming the same costs among players and different costs among players. Our promising results provide evidence that our work could be used to regulate traffic flow toward aims such as reduction of emissions.

In the final project of our dissertation, we explore the general parameterization of the

constant vector in linear complementarity problems (LCPs), which are mathematical expressions that can represent optimization, game theory, and market models [75]. Unlike the limited previous work on inverse optimization for LCPs, we characterize theoretical considerations regarding the inverse optimization problem for LCPs, prove that a previously proposed IO solution model can be dramatically simplified, and handle the case of multiple solution data points for the IO LCP problem. What's more, we use our knowledge regarding LCPs and IO in a water market allocation case study, which is an application not previously explored in the IO literature, and we find that charging an additional tax on the upstream players enables the market to reach the system optimal. In sum, this dissertation contributes to the inverse optimization literature by expanding its reach in the equilibrium problem domain and by addressing new infrastructure applications.

These three projects together draw upon similar themes and have some overlapping characteristics, which we demonstrate in the following Venn Diagram:

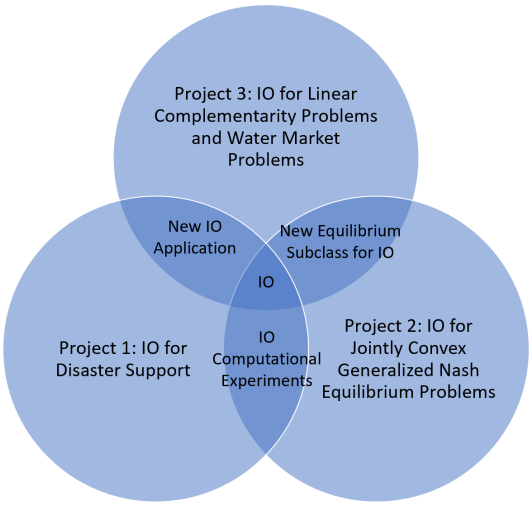


Figure 1.1: Commonalities Between the Three Dissertation Projects

As can be seen and as has been demonstrated throughout this Introduction, all three projects share the common theme of inverse optimization. They also all deal with some kind of infrastructure

application, with project 1 dealing with road network protection, project 2 employing a traffic game among multiple players, and project 3 exploring a water market application. Projects 1 and 2 have the commonality of embarking on intensive computational experiments to validate the approaches. Projects 1 and 3 both apply inverse optimization to a new application area, with project 1 demonstrating inverse optimization for disaster relief and project 3 employing inverse optimization in the context of water markets. Projects 2 and 3 extend inverse optimization techniques and/or theory to new or understudied sub-classes of equilibrium problems, with project 2 addressing jointly convex generalized Nash equilibrium problems (GNEPs) and project 3 exploring linear complementarity problems (LCPs).

Indeed, we can see the progression and maturation process of the research through these three projects as well as coverage of many topics and concepts in optimization. We start at project 1 with a new application idea and a difficult computational task in the form of solving a stochastic MPEC, touching on areas including stochastic programming and handling complementarity constraints embedded in optimization problems. We advance through project 2 in which we take a theoretical step in our research and advance inverse optimization in a more general sense to a new sub-class of equilibrium problems, stretching our understanding of building blocks of equilibrium models including game theory, variational inequalities, and mixed complementarity problems. This project also involved a large computational study. Indeed, in both of the first two projects, we learned strategies and techniques for approaching large computational studies. Finally, in the culmination of our research, we embark upon a much more difficult theoretical task: the parameterization of linear complementarity problems which encompass optimization, game theory, and market models. In approaching this task, we consider the multiplicity of solutions to these problems and broader inverse optimization considerations such as the implications of moving

between the original problem and the inverse optimization problem. What's more, in project 3, we approach a new application and, in that case study, we demonstrate the power of inverse optimization to impact policy directly through using system optimal solutions to parameterize markets and, thus, encourage the addition of new taxes or subsidies. Indeed, we provide a framework through which to parameterize LCPs, touching on such topics as complementarity cones and matrix analysis, and a demonstration of the capabilities of inverse optimization to influence markets and policy in profound ways.

Therefore, to conclude, we showcase in this dissertation that the worlds of theory and application must be in conversation with each other in order to advance our understanding of both, thus underscoring the importance of this broad field we call Applied Mathematics.

1.4 Papers and Presentations

Before we progress to the rest of the dissertation, we outline the papers and presentations that have been produced as a result of this dissertation.

- **Project 1:** [5]
  - Papers: Allen, Stephanie, Daria Terekhov, and Steven A. Gabriel. "A Hybrid Inverse Optimization-Stochastic Programming Framework for Network Protection." *arXiv preprint arXiv:2110.00488* (2021). (Under review by a journal)
  - Presentations: INFORMS 2020 & Trans-Atlantic Infraday Conference 2021
- **Project 2:** [4]
  - Papers: Allen, Stephanie, Steven A. Gabriel, and John P. Dickerson. "Using inverse

optimization to learn cost functions in generalized Nash games." *Computers & Operations Research* (2022): 105721.

  - Presentations: INFORMS 2021, ECOM 2021, & NTNU Winter School 2022

- **Project 3:**

  - Papers: Allen, Stephanie, Steven A. Gabriel, & Nathan T. Boyd, "Inverse Optimization for Parameterization of Linear Complementarity Problems and for Incentive Design in Markets," Available on arXiv soon.

Chapter 2: A Hybrid Inverse Optimization-Stochastic Programming Framework for Network Protection

Note: The contents of this chapter come mostly from a paper we have published on arXiv, referenced here: [5]. There are certain additions that were made in response to the first round of reviews from an academic journal. We have received the second round of reviews which will not greatly change the contents of the chapter, thus we will address these comments post-dissertation. We have presented this work at INFORMS 2020 and at the Trans-Atlantic Infraday Conference in 2021. We thank our co-authors on this chapter: Daria Terekhov and Steven Gabriel. Stephanie Allen did the vast majority of the work on this project, from framework conception to code writing to paper writing. She received some advice from Daria Terekhov on implementation of the train-test part of the experiments, but she had brought the idea of these types of experiments herself. Daria Terekhov and Steven Gabriel provided editorial feedback.

2.1 Introduction

Given the threat of natural disasters, it is imperative that communities and nations prepare in order to mitigate the consequences. According to NOAA [139], in the year 2020, there were 22 "weather and climate disasters" that cost 1 billion or more US dollars in the United States, and 262 people died in these disasters. Governments and planning agencies often have little

foresight of the type of disaster that might strike, meaning they must be prepared for many different potential events. To be effective, governments must make strategic decisions before and immediately after crises such that financial and human costs are minimized. Indeed, in a report by Federal Emergency Management Agency (FEMA) entitled "National Response Framework" [68], officials state there are five mission areas to this response framework which are prevention, protection, mitigation, response, and recovery, thus indicating that the US government actively intervenes at the various decision points in the disaster management cycle. We note that we use the words "disaster" and "crisis" interchangeably in this chapter.

As a response to these events, researchers have developed disaster support systems (DSS), which include systems that evacuate people out of tunnels [6], deliver supplies to affected individuals [73], decide where to place supplies in anticipation of a disaster [157], and extinguish wildfires [201]]. Wallace and De Balogh [187] define DSSes as systems which contain "a data bank, a data analysis capability, normative models, and technology for display and interactive use of the data and models." We focus on the data analysis and normative model elements of these requirements which, respectively, mathematically examine and present information for decision makers and help make decisions. One type of normative model used in the disaster management community which we employ in this chapter is a multi-stage stochastic program. This type of model is proposed as a way to make decisions regarding protecting networks against disasters or bringing supplies to communities after disasters. The 2013 US National Infrastructure Protection Plan [52] proposes seven principles for organizations involved in critical infrastructure, one of which states: "Risk should be identified and managed in a coordinated and comprehensive way across the critical infrastructure community to enable the effective allocation of security and resilience resources." This principle supports the work of the aforementioned multi-stage

programs for disaster relief. Indeed, in the report, the need for incorporating "security and resilience into the design and operation of assets, systems, and networks" is stated as part of the way to reduce vulnerabilities in infrastructure, which requires accurate representations of these structures. The misspecification of the parameters defining these structures can severely limit the usefulness of the multi-stage programs for disaster relief. In particular, in this chapter, we focus on transportation cost parameters, whose misspecification could lead to incorrect protection decisions, such as allocating too few or too many resources to parts of road networks affected by landslides and flash floods. These incorrect protection decisions could happen if, for instance, cost parameters are assigned such that more traffic is assumed to be using a road than actual data shows, which in turn leads to more protective measures being allocated to this road rather than to the road that actually sees more traffic. We propose using inverse optimization to recover these cost parameters.

Inverse optimization (IO) allows a user to parameterize particular functions in optimization and equilibrium problems using solutions to these problems [3, 21, 205]. This means we can take observed data regarding phenomena in the world and use this data to find the parameters that define the motivating functions at the heart of the models that represent the phenomena. For this chapter, we focus on parameterizing the cost functions of a traffic equilibrium model, which means we take data regarding traffic flow and find the values of parameters of the cost functions that induced that flow.

Inverse optimization has two main advantages over other parameter estimation approaches: (a) a user is able to employ a model of an entire system such as an optimization or equilibrium problem with constraints [3, 21] when performing parameter estimation which explicitly allows us to incorporate knowledge of the constrained model that generated the data into the estimation

method and (b) we can more intuitively understand what is going on with regard to this method than "black box" methods such as deep learning (see [84] for more information on deep learning). There are techniques such as structural estimation that can be compared to inverse optimization but, as [21] outline in Appendix 2 of their paper, there are several advantages of an inverse optimization formulation over structural estimation, which include (a) allowing for approximate equilibria and (b) formulating a less complicated problem (because there are no bilinear terms; see [21] for more details).

2.1.1 Contribution

To our knowledge, neither the DSS literature nor the multi-stage stochastic program for disaster relief literature have explored inverse optimization as a tool for estimating model parameters. We propose inverse optimization as a new approach for data analysis and demonstrate its ability to recover similar protection decisions as the originally parameterized stochastic network protection model. We also demonstrate that accurate knowledge regarding the cost functions matters because it can change protection decisions when compared to the assumption of uniform cost for most of our experiments. If our protection decisions differ when we have more informed knowledge of hidden traffic parameters versus when we have less information, this is important. Indeed, having more accurate information enables us to protect the arcs that have the greatest impact on traffic flow. This chapter's goal is to demonstrate the joint inverse optimization and stochastic programming framework on smaller networks and simplified situations, leaving more realistic networks and situations to future work.

The rest of the chapter is organized as follows. Section 2.2 investigates the literature

related to our problem. Section 2.3 provides appropriate mathematical background. Section 2.4 explains the experimental structure, and Section 2.5 explains the results. Section 2.6 discusses our conclusions and ideas for future work.

2.2 Literature Review

The literature in this section demonstrates that multi-stage disaster relief models and DSS models have not used inverse optimization previously.

2.2.1 Inverse Optimization for Transportation Problems

Although inverse optimization has not been previously utilized in disaster relief, it has been used to parameterize cost functions in the transportation literature, which is relevant to our framework because we use the techniques from this literature to parameterize the cost functions in the stochastic network protection problem. Within the transportation literature, Thai et al. [177] use a mathematical program with equilibrium constraints to minimize the difference between the simulated solutions and optimal solutions to the traffic equilibrium problem as a way of recovering the specified cost function parameters. Thai et al. [176] use a combination of methods by [21] and [40] to create a multi-objective program that minimizes the duality gap for the variational inequality and the difference between the optimal and observed solutions. Bertsimas et al. [21] use their inverse variational inequality problem along with kernel methods to estimate the cost functions. Zhang et al. [206] and Zhang et al. [207] follow [21]'s methodology, with [206] involving different categories of vehicles and [207] emphasizing recovering both cost function and origin-destination matrices from real-world traffic data. Chow et al. [43] use

techniques from [3] but augment them to handle the nonlinear nature of their problem. Finally, Allen et al. [4] extend [154]'s parameterization framework for multi-player Nash problems to the case of jointly convex generalized Nash equilibrium problems and demonstrate this framework by parameterizing a transportation game.

2.2.2 Multi-Stage Disaster Relief Models

There is a substantial literature on multi-stage stochastic programs for disaster relief and protection; see [85]. Methods for estimating cost functions in road networks include fuzzy numbers, Euclidean distances, road distance data, the Bureau of Public Roads (BPR) function (see Section 2.3.1.1), and stochastic programming. None of them use inverse optimization to estimate costs, which is what we propose in this chapter.

Zheng et al. [208] estimate cost parameters for moving supplies after natural disasters using fuzzy numbers for the time it takes to traverse between the supply and demand nodes. Barbarosoglu and Arda [16] use road information and Euclidean distances between points for their cost parameters in their stochastic model pertaining to distributing supplies after natural disasters. Chu et al. [44] also calculate the travel cost for several routes/paths of origin-destination pairs to capture the idea that one or more routes could fail in a disaster in their stochastic network protection model. Travel cost is measured by a variable which takes on real numbers between 0 and 1 and which measures how close to the shortest path the demand for an OD pair is allowed to take through the network. Noyan et al. [144] and Doyen et al. [56] use a mixture of road data along with scenario dependent costs to form their cost functions, while Mohammadi et al. [133] use exclusively scenario dependent costs.

Fan and Liu [65] employ the BPR function for their stochastic network protection problem for arc costs, but no stochastic parameters are involved. For the rest of the BPR literature, either the capacity is impacted by the protection decisions and/or some of the parameters in the BPR function are stochastic [7, 66, 67, 124, 125]. For these multi-stage stochastic programs, inverse optimization methods would have captured a set of parameters that led to given flow patterns, which could have been used to augment the existing cost function approaches.

2.2.3 Disaster Support Systems

We focus on reviewing DSSes that have a data analysis step in their processes. First, there are disaster support system papers that determine important quantities and parameters via simulation and/or physical models of the situation [6, 50, 59, 73, 118, 162, 178, 182, 201, 202]. Second, DSS papers can also determine parameters via data processing as in [72, 101, 204], or they can utilize machine learning to determine modeling structures as in [1]. Other DSS papers use geographic information system (GIS) techniques to estimate parameters [45, 157]. Horita et al. [100] combine GIS and sensor information to estimate parameters. In addition, some papers propose data fusion techniques such as ensemble Kalman filters [148] and gradient based methods [111]. Inverse optimization allows a user to propose a model of the system and parameterize the model using data/simulated solutions[1] and optimality conditions [3, 21, 37, 205], which can augment the information gained from data and, thus, could be useful for DSSes. However, as can be seen from this review, DSSes have not used inverse optimization for data analysis.

[1] See [171] for an example using proposed solutions.

2.3 Hybrid Framework

In the next few sections, we will explain our data analysis component (which presents information for decision makers) along with our normative model (which helps make decisions). The data analysis component employs Bertsimas et al.'s [21] work on parameterizing cost functions for the traffic equilibrium problem, which uses traffic data to find the parameters for the cost functions involved in this model. We utilize the techniques from Bertsimas et al. [21] on inverse optimization because Bertsimas et al. [21] handles equilibrium problems, which are an important component of the normative model. The normative model comes from Fan and Liu [65] who suggest a two-stage network protection problem with equilibrium constraints to make protection decisions for road networks. We propose pairing the two components together in the following sequence of steps, with θ representing the collection of parameters to be estimated by the inverse optimization model:

1. Input data $\hat{\mathbf{x}}^j,\ j = 1...J$ into inverse optimization model (2.3.4) and obtain θ
2. Form stochastic network protection problem (SNPP) (2.3.8) with θ
3. Solve the SNPP (2.3.8) and obtain protection decisions $\mathbf{u}$.

2.3.1 Data Analysis Component: Inverse Optimization

Bertsimas et al. [21] utilize variational inequalities (VI) to represent optimization and equilibrium problems. They assume that the following extended variational inequality describes the ϵ equilibrium of a system, with $F : \mathbb{R}^n \to \mathbb{R}^n$, $\mathcal{F} \subset \mathbb{R}^n$, $\mathbf{x} \in \mathcal{F}$, and $\epsilon \in \mathbb{R}_+$ [21]:

$$F(\mathbf{x}^*)^T(\mathbf{x} - \mathbf{x}^*) \geq -\epsilon, \ \forall \mathbf{x} \in \mathcal{F}. \tag{2.3.1}$$

When $\epsilon = 0$, we recover the classical VI that was identified in Chapter 1. In the case of our traffic application, we assume that F is the cost function, representing the time per vehicle along each arc in the set $\mathcal{A}$ of arcs [124, 125]. Therefore, expression (2.3.1) states that $\mathbf{x}^*$ solves the VI if the inner product between F at $\mathbf{x}^*$ and the difference between any point in $\mathcal{F}$ and $\mathbf{x}^*$ is greater than a small, negative number.

To discuss the traffic equilibrium problem, we need to define the F function and $\mathcal{F}$ set which requires introducing some new notation. Bertsimas et al. [21] describe the Wardrop traffic equilibrium with nodes $\mathcal{N}$ and arcs $\mathcal{A}$ as having a node-arc incidence matrix $N \in \{-1, 0, 1\}^{|\mathcal{N}| \times |\mathcal{A}|}$ [129], vectors $\mathbf{d}^w \in \mathbb{R}^{|\mathcal{N}|}$ which contain the origin destination locations represented by the set $\mathcal{W}$ with a negative entry for the origin and a positive entry for the destination [129], and a feasible set $\mathcal{F}$. Bertsimas et al. [21] define the $\mathcal{F}$ set as:[2]

$$\mathcal{F} = \left\{ \mathbf{x} : \exists \mathbf{x}^w \in \mathbb{R}_+^{|\mathcal{A}|} \ s.t. \ \mathbf{x} = \sum_{w \in \mathcal{W}} \mathbf{x}^w, \ N\mathbf{x}^w = \mathbf{d}^w \ \forall w \in \mathcal{W} \right\} \tag{2.3.2}$$

in which $\mathbf{x}^w \in \mathbb{R}_+^{|\mathcal{A}|}$ represents the flow between origin and destination w and $\mathbf{x} \in \mathbb{R}_+^{|\mathcal{A}|}$ represents the composite flow vector. The corresponding F function for the variational inequality is defined as $\mathbf{c}(\mathbf{x})$ such that $c_a : \mathbb{R}_+^{|\mathcal{A}|} \to \mathbb{R}_+$ for arc a.

The multipliers associated with the constraints in the $\mathcal{F}$ set should be non-negative because they represent the time it takes to travel from the associated node to the destination w [65]. Therefore, we need to turn the equalities in the $\mathcal{F}$ set into inequalities [14, 15]. Respecting

[2] We define the $\mathcal{F}$ set for the inverse optimization model differently; see Appendix A.1.2.

the definition of N and $\mathbf{d}^w$, our traffic equilibrium problem in complementarity form is then [14, 15, 21, 65, 75, 165]:[3]

$$0 \leq \mathbf{c}(\mathbf{x}^w) + N^T \mathbf{y}^w \perp \mathbf{x}^w \geq 0, \ \forall w \in \mathcal{W} \tag{2.3.3a}$$

$$0 \leq \mathbf{d}^w - N\mathbf{x}^w \perp \mathbf{y}^w \geq 0, \ \forall w \in \mathcal{W} \tag{2.3.3b}$$

We know we can undergo this transformation from the variational inequality to the complementarity form due to the discussion in Chapter 1 about this transformation. We can show that $\mathbf{d}^w - N\mathbf{x}^w = 0$ when there is a solution for (2.3.3) and when we assume that the $\mathbf{c}(\mathbf{x})$ function is greater than 0 for all $\mathbf{x} \geq 0$ in a proof which is adapted from Ban [15]. See Appendix A.1.1. We use (2.3.3) to generate data for the inverse optimization part of the framework, and the data generation process can be found in Section 2.4.1.

For the inverse optimization model in Bertsimas et al. [21], we can then form an optimization model including each data point $\hat{\mathbf{x}}^j$, $j = 1, ..., J$ (with J representing the total number of data points utilized) representing the flow on the network such that:

- There is one OD pair for each instance $\hat{\mathbf{x}}^j$.
- There is the same node-arc incidence matrix N for each $\hat{\mathbf{x}}^j$.

The inverse optimization model for J data points (corresponding to each of the $\hat{\mathbf{x}}^j$ flow patterns), parameters $\theta \in \Theta$ with Θ as a convex subset of $\mathbb{R}^Z$ (Z representing a number of parameters),

[3] We keep the row in N that contains the destination, which is different from [15] and [14].

$\mathbf{y}^j \in \mathbb{R}^{|\mathcal{N}|}$, and $\epsilon \in \mathbb{R}^J$ is:

$$\min_{\theta \in \Theta, \mathbf{y}, \epsilon} ||\epsilon||_2^2 \tag{2.3.4a}$$

$$-(N)^T \mathbf{y}^j \leq \mathbf{c}(\hat{\mathbf{x}}^j; \theta), \ j = 1, ..., J, \tag{2.3.4b}$$

$$\mathbf{y}^j \geq 0, \ j = 1, ..., J, \tag{2.3.4c}$$

$$\mathbf{c}(\hat{\mathbf{x}}^j; \theta)^T \hat{\mathbf{x}}^j + (\mathbf{d}^j)^T \mathbf{y}^j \leq \epsilon^j, \ j = 1, ..., J, \tag{2.3.4d}$$

The derivation of this mathematical program can be found in Appendix A.1.2. In this section of the Appendix, we explain the Berstimas et al. [21] method in much more detail. We note that the $\mathbf{y}^j$ dual variables are important because they correspond, as in the complementarity problem above, to the travel distance to destination j [65]. We solve this mathematical program using the `ipopt` solver [185], which, for convex problems such as the one above, does solve the problem to global optimality [186]. We set the `tol` parameter to $1e-12$. There can be multiple forms for the vector-valued arc cost function $\mathbf{c}(\mathbf{x}; \theta)$, which the next subsection will cover.

Before moving to the next section, it is important to address one of the questions that has been asked of inverse optimization: how is this different from linear regression? Linear regression in its unconstrained form as presented in [84] would not be able to express the constraint relationships in (2.3.4b)-(2.3.4d). In addition, in order to attempt to model our problem with linear regression, we would need target total cost values for origin-destination pairs on the road network, which

is something we do not have for our problem. We assume that we only have the values of the decision variables. We need the dual problem to establish the right optimization relationships to find our parameter values [21]. Even in linear regression's constrained form (as expressed by these papers: [80, 122, 169]), the constraints are on the parameters themselves, which Θ in (2.3.4a) can already express. Consequently, since even constrained linear regression cannot accommodate the relationships expressed in (2.3.4b)-(2.3.4d), then we need a more complicated model to parameterize our cost functions. The equations in (2.3.4b)-(2.3.4d) represent the satisfaction of dual feasibility and strong duality for the linear program produced by the variational inequality created by the traffic equilibrium problem. Indeed, one of the most important aspects of the model (2.3.4) is the set of dual variables $\mathbf{y}^j$ which must satisfy two sets of inequality relationships (2.3.4b) and (2.3.4d). These dual variables represent the cost of traveling on the network from the various nodes to the destination node. What's more, the inequalities in (2.3.4b)-(2.3.4d) encode the variational inequality conditions seen in (2.3.1) and, thus, the equilibrium relationship needed for the problem.

2.3.1.1 Different Types of Cost Functions

We propose two different formulations for the vector valued function $\mathbf{c}(\mathbf{x})$ which represents the time per vehicle along each arc in the set of arcs $\mathcal{A}$ [124, 125]. Note that θ will represent the collection of all parameters for a given function.

- **Linear Cost:** We assume that $c_a(\mathbf{x}) = \phi_a \mathbf{x}_a + \beta_a,\ \phi_a \in \mathbb{R}_+,\ \beta_a \in \mathbb{R}_+$, such that

$$\mathbf{c}(\mathbf{x}) = \begin{bmatrix} \phi_1 & & \\ & \ddots & \\ & & \phi_{|\mathcal{A}|} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{|\mathcal{A}|} \end{bmatrix} \tag{2.3.5}$$

  This function has been used for representing travel times such as in [166]. Siri et al. [166] label the β_a as the free flow travel times i.e., travel times without any interaction with other travelers [21, 43, 177, 206, 207]. ϕ_a is the factor of additional time of having one more unit of flow on the arc. In this chapter, we assume that the free flow travel times are given, and our goal is to estimate ϕ_a for all $a \in \mathcal{A}$ (see Section 2.4.5).

- **Bureau of Public Roads Function:** The Bureau of Public Roads function (BPR) [27, 180] is a common function utilized by transportation researchers when modeling flow along arcs in a network [21, 176, 206, 207]. From [165], the BPR function for arc a is:

$$c_a(\mathbf{x}_a) = t_a^0 \left(1 + \alpha_a \left(\frac{\mathbf{x}_a}{c_a'}\right)^{\beta}\right). \tag{2.3.6}$$

  The t_a^0 is the free-flow travel time, c_a' is the "practical capacity" which we just take as the normal capacity, and α_a & β are parameters which, following [165], are commonly assumed to be 0.15 and 4 respectively, regardless of the arc. In contrast, in this chapter, we will assume that the α_a parameter is different for each arc and that it is the quantity we estimate with inverse optimization (see Section 2.4.5). We linearize the BPR function using standard techniques [127, 197].

In our experiments, we compare the protection decisions made under the costs imputed using IO with the protection decisions made when a user has the original parameterization from (2.3.3) and with the protection decisions when a user assumes uniform cost, meaning average ϕ for the linear cost function and 0.15 for the α in the BPR cost function. Section 2.4.2 will explain this further. We choose these two cost functions because (a) the linear cost function is easy to use, and (b) the BPR function is standard in the literature.

2.3.2 Normative Model: Two-Stage Stochastic Model

For the stochastic network protection model portion of the framework, we implement Fan and Liu's [65] two-stage network protection model with complementarity constraints with a few changes in the capacity function, the conservation of flow constraints, and the objective function. We adopt much of the notation from [65] and extended definitions for these terms can be found in Appendix A.2.1:

- $\mathcal{A}$: the set of network arcs, and m as the number of arcs.
- $\mathcal{N}$: the set of network nodes, and n as the number of nodes.
- K: the number of destinations of flow in the network.
- $\mathcal{S}$: the scenario set.
- $x_a^{k,s}$: the flow on arc a that is destined for the kth destination in scenario s. The vector $\mathbf{x}^{k,s} \in \mathbb{R}^m$ denotes the flow on all arcs. (units=thousands of vehicles)
- f_a^s: the total flow on arc a in scenario s, and $\mathbf{f}^s$ as the vector containing all of the arcs. (units=thousands of vehicles)

- u_a: the decision variable controlling resources used to protect an arc a against a crisis. (units=proportion of necessary resources needed to fully insure the arc)

- W: the node-link adjacency matrix.

- $\mathbf{q}^k \in \mathbb{R}^n$: designates the amount of flow originating at each node that is headed to destination k. (units = thousands of vehicles)

- $h_a^s(u_a)$: the capacity of an arc a given first stage decision u_a under scenario s:

$$h_a^s(u_a) = \begin{cases} \text{cap}_a \text{ if } a \notin \bar{\mathcal{A}} \\ \text{cap}_a - m_a^s(1-u_a) \text{ if } a \in \bar{\mathcal{A}} \end{cases} \tag{2.3.7}$$

  with cap_a representing capacity of the arc without it being affected by a disaster, m_a^s representing the amount of damage done to arc a in scenario s if not protected, and $\bar{\mathcal{A}}$ represents the set of arc vulnerable to the disaster. Note that m_a^s could be 0 in certain scenarios. (units = thousands of vehicles)

- $t_a(\mathbf{f}^s)$ represents the time per vehicle along arc a [124, 125] as a function of the flows $\mathbf{f}^s$ in scenario s. We explore multiple different forms for t_a, described in Section 2.3.1.1.

- $\lambda_i^{k,s}$ as the minimum travel time from node i to node k in scenario s [65]. (units=travel time)

- $\mathbf{d}^{k,s}$ as the vector of extra variables that acts as a buffer for any flow that cannot be properly apportioned. (units=thousands of vehicles).

- p^s as the probability of each scenario s.

Fan and Liu's [65] model with a modification to the complementarity constraints based on work by [15] and [14] is thus:

$$\min \sum_{s \in S} p^s Q^s(\mathbf{u}, \mathbf{f}^s) \tag{2.3.8a}$$

$$\text{s.t. } \mathbf{u} \in \mathcal{D} \tag{2.3.8b}$$

$$f_a^s = \sum_{k=1}^{K} x_a^{k,s} \leq h_a^s(u_a),\ a \in \mathcal{A},\ s \in \mathcal{S} \tag{2.3.8c}$$

$$0 \leq x_{ij}^{k,s} \perp \left(t_a(\mathbf{f}^s) + \lambda_j^{k,s} - \lambda_i^{k,s}\right) \geq 0,\ \forall (i,j) \in \mathcal{A},\ \forall k = 1...K,\ \forall s \in \mathcal{S} \tag{2.3.8d}$$

$$0 \leq \mathbf{q}^k + \mathbf{d}^{k,s} - W\mathbf{x}^{k,s} \perp \lambda^{k,s} \geq 0,\ \forall k = 1...K,\ \forall s \in \mathcal{S} \tag{2.3.8e}$$

The $Q^s(\mathbf{u}, \mathbf{f}^s)$ function (2.3.8a) in the objective function has the following form:

$$Q^s(\mathbf{u}, \mathbf{f}^s) = \langle \psi, \mathbf{u} \rangle + \gamma \langle \mathbf{f}^s, \mathbf{t}(\mathbf{f}^s) \rangle + 10000 \sum_{k=1}^{K} ||\mathbf{d}^{k,s}||_2^2$$

The first term denotes the total cost of protection (with ψ as the dollar amount it costs to protect each arc fully); this term differs from the [65] paper which instead uses the cost of repair. The second term computes the total travel time for all of the flow on each arc, sums these amounts, and then multiplies the sum by γ, which transforms travel time to financial units [65], keeping in the same units as the first term. Note, $\mathbf{t}(\mathbf{f}^s)$ corresponds to the $\mathbf{c}(\mathbf{x})$ function from Section

2.3.1. The third term makes it extremely costly for the model to use any of the buffer that the $\mathbf{d}^{k,s}$ vectors provide for the conservation of flow (2.3.8e) constraints. Constraints (2.3.8b) represent the budgetary and technological restrictions that [65] define. We specified this further as just budgetary constraints of the form:

$$\sum_{a \in \mathcal{A}} u_a \leq I \tag{2.3.9}$$

with I representing the number of arcs we can afford to fully protect. However, because u_a are continuous variables, we can protect more than I number of arcs partially because we are treating u_a as proportions. The capacity constraints (2.3.8c) have an s dependence for the h_a^s functions because the $m_a^s \; \forall a \in \bar{\mathcal{A}}$ are scenario-dependent. The constraints in (2.3.8d) encapsulate the idea that there should be no flow on the arc a on its way to destination k in scenario s unless that arc is part of the minimal travel time route to destination k. The complementarity constraints in (2.3.8e) include the conservation of flow constraints that ensure flow begins and ends at the appropriate places in the network. The $\mathbf{d}^{k,s}$ vectors are buffers in case some of this flow does not fulfill the conservation of flow constraints; Fan and Liu [65] define them as variables to ensure a feasible solution.

We specify some of the parameters for the model that will not change over the course of the chapter:

- The cap_a value is set to 8 for all arcs a.
- The m_a^s value (amount of damage) is set to 8.
- In the objective function, we set $\gamma = 1$ (following [65]) and set the ψ vector to 1 because we do not want cost to be prohibitive.

2.3.2.1 Big M Method for Complementarity Constraints and Progressive Hedging Algorithm for Solving Stochastic Network Protection Problem

Fan and Liu [65] note in their paper that the stochastic network protection problem is difficult to solve because of (1) the complementarity constraints and (2) the stochastic elements. In order to handle the complementarity constraints, we use the disjunctive constraint/big M method approach [74, 95]. As an example, we take the complementarity condition from (2.3.8d) and produce a series of constraints:

$$x_{ij}^{k,s} \geq 0 \tag{2.3.10a}$$

$$t_a(\mathbf{f}^s) + \lambda_j^{k,s} - \lambda_i^{k,s} \geq 0 \tag{2.3.10b}$$

$$x_{ij}^{k,s} \leq M_{ij}^{k,s}(b_{ij}^{k,s}) \tag{2.3.10c}$$

$$\left(t_a(\mathbf{f}^s) + \lambda_j^{k,s} - \lambda_i^{k,s}\right) \leq M_{ij}^{k,s}(1 - b_{ij}^{k,s}) \tag{2.3.10d}$$

The $b_{ij}^{k,s}$ is a binary variable, and $M_{ij}^{k,s}$ is a sufficiently large number, which forces at least one of the two terms in (2.3.10c) or (2.3.10d) to be 0. We repeat the same procedure for the complementarity constraints in (2.3.8e). See Appendix A.2.2 for information on calculating the $M_{ij}^{k,s}$ values. We note that this addition of binary variables increases our computational costs because, now, we are dealing with a mixed-integer problem (MIP). To handle the stochasticity of this problem, we follow [65] by employing the progressive hedging (PH) algorithm. Proposed by

Rockafellar and Wets [156], the PH algorithm at its most basic level solves scenario subproblems created by the random variable(s) involved in the original problem using an approach in which there is a penalty term that encourages first-stage variables to tend toward the "aggregate" solution [156] that is computed after each iteration of the algorithm. See the following references for more information about using the algorithm, about setting its parameters, and about the fact that the PH algorithm serves as a heuristic when handling MIP problems: [32, 49, 65, 83, 86, 106, 119, 136, 149, 160, 183, 191]. We use the implementation of the PH algorithm found in the `pysp` extension [192] of the `pyomo` package [95, 96] in Python. We use `gurobi` [90] for the mixed-integer quadratic programming sub-problems arising as part of the PH algorithm.

We note that the binary variables created by the big M method cause the SNPP to become a mixed integer quadratic program in the case of the linear cost function. In the case of the BPR cost function, we need to integrate a non-convex constraint into the mixed integer quadratic program. We handle this non-convexity by turning on `gurobi`'s non-convex flag. This non-convexity means that we could be obtaining local solutions for the BPR function version of the SNPP.

Overall, with regard to the size of our problem, one of the computational bottlenecks is the presence of the complementarity constraints in (2.3.8d) and (2.3.8e). They produce binary variables due to the big M method, which means as the network becomes larger with more arcs, as the number of scenarios increases, and/or as the number of origin-destination pairs K increases (the beginning and ending points of flow on the network), more binary variables are produced and, thus, the computational complexity increases. Thus, the size of the problem is dependent upon the number of arcs, scenarios, and OD pairs. For the BPR cost function, there is the added dimension of the number of breakpoints in the linearization of the function, which adds additional

binary variables outside of the complementarity constraints.

2.4 Experimental Design

In this section, we define our experimental setup and the metrics by which we will evaluate the experiments. Most of our experimental results demonstrate that inverse optimization enables users to recover comparable protection decisions as the original cost protection decisions, and there is a difference between protection decisions made under uniform cost parameters and the original or IO parameterizations.

2.4.1 Data Generation

We generate data (observations of flow $\hat{\mathbf{x}}^j$ for all $j = 1, ..., J$), as discussed in Section 2.1, using the forward problem in the form of the complementarity model (2.3.3); we solve (2.3.3) using PATH [54, 70] in GAMS. The set Λ represents the origin-destination pairs utilized for each run of the complementarity model. For this chapter, Λ is the set of all different origin-destination (OD) pairs for each network for the data generation process, one pair for each run of the complementarity model. In more complicated versions, Λ would consist of multiple different OD pairs per run of the complementarity model. The algorithm below illustrates generating the $\hat{\mathbf{x}}^j$ data for $j = 1, ..., J$ given a set of configurations Λ with $|\Lambda| = J$.

Algorithm 1: Generating the Data

Data: The set Λ of configurations

for j = *1:*$|\Lambda|$ **do**

Build the traffic equilibrium model (2.3.3) with the jth configuration of OD pair(s)

Solve the traffic equilibrium model (2.3.3) with PATH in GAMS

Store optimal $\hat{\mathbf{x}}^j$

end

In part A of the experiments, the generated data $\hat{\mathbf{x}}^j, j = 1...J$ is used as input into the inverse optimization model to determine the parameters θ for the cost function. In part B of the experiments, a subset of the generated data $\hat{\mathbf{x}}^j, j = 1...J$, is used as input into the inverse optimization model to determine the parameters θ for the cost function. Namely, for each of the OD pairs, we leave out the data corresponding to that OD pair to obtain parameters θ that correspond to that OD pair. This parallels the training-testing framework used in machine learning [84]. For both experimental setups, we then carry through with the rest of the hybrid framework described at the beginning of Section 2.3 to obtain the protection decisions. See Section 2.4.3 for more details. We note that the data produced by the methods outlined in this section is non-noisy data and, thus, represents perfect equilibrium solutions to the traffic equilibrium model (2.3.3). This in part causes the forthcoming defined "flow error" to be low for our experiments.

2.4.2 Metrics

In order to evaluate our hybrid framework, we must solve the stochastic network protection problem three times for each set of generated data because we must compare the protection decisions under the original cost parameters, the inverse cost parameters, and the assumption of

uniform cost parameters:

- We define the *IO information protection decisions*, denoted $\mathbf{u}$, as the protection decisions the two-stage model would make based on the cost vector obtained using the inverse optimization algorithm. This cost vector is defined as θ.

- We define the *original information protection decisions*, denoted $\hat{\mathbf{u}}$, as the protection decisions that the two-stage model would make if it were directly given the original cost structure that was used in (2.3.3) to generate the data $\hat{\mathbf{x}}^j, j = 1...J$. The goal of the framework is for the inverse optimization algorithm to be able to provide a cost estimate that will result in the same/comparable protection decisions as under the original cost structure. We use the Original-IO metric below to evaluate the similarity between the protection decisions: the closer the Original-IO metric is to 0, the better IO is at recovering parameters leading to the original (assumed correct) decisions. The cost vector corresponding to the original cost structure is $\hat{\theta}$.

- We define the *uniform information protection decisions*, denoted $\bar{\mathbf{u}}$, as those decisions that the two-stage model would make if it were given a uniform cost structure for the network. If the original information decisions differ significantly from the uniform information decisions, then this provides evidence that knowing the cost structure of the network is important. The cost vector corresponding to the uniform costs is $\bar{\theta}$.

Our performance metrics capture the difference between the protection decisions made under different costs:

- **Original-IO (O-IO)**: $||\hat{\mathbf{u}} - \mathbf{u}||_2$

- **Uniform-IO (U-IO)**: $||\bar{\mathbf{u}} - \mathbf{u}||_2$
- **Uniform-Original (U-O)**: $||\bar{\mathbf{u}} - \hat{\mathbf{u}}||_2$.

One other metric used in part B of the experimental results is known as flow error in which we take the difference between the flow produced by the original parameters and the flow produced by the inverse optimization parameters. We take the 2-norm between these two flow vectors. Mathematically, this can be expressed as:

$$\text{Flow Error: } ||\hat{\mathbf{x}} - \mathbf{x}||_2 \tag{2.4.1}$$

2.4.3 Procedures for Experiments A and B

In part A of the experimental results, the full framework is as follows:

Algorithm 2: Part A of the Experiments

Data: Network Structure, Type of Cost Function

for *i=1:10* **do**

Generate the data $\hat{\mathbf{x}}^j, j = 1...J$ according to Algorithm 1

Input data $\hat{\mathbf{x}}^j,\ j = 1...J$ into inverse optimization model (2.3.4) and obtain θ

$\mathbf{u} = \text{SNPP}(\theta)$

$\hat{\mathbf{u}} = \text{SNPP}(\hat{\theta})$

$\bar{\mathbf{u}} = \text{SNPP}(\bar{\theta})$

Calculate the performance metrics according to Section 2.4.2

end

In part B of the experimental results, the full framework is as follows:

Algorithm 3: Part B of the Experiments

Data: Network Structure, Type of Cost Function, Number of OD pairs

for *i=1:10* **do**

Generate the data $\hat{\mathbf{x}}^j, j = 1...J$ according to Algorithm 1

for *k=1:number_of_OD_pairs* **do**

Input data $\hat{\mathbf{x}}^j,\ j = 1...J$ with the $\hat{\mathbf{x}}^j$ corresponding to the kth OD pair removed into inverse optimization model (2.3.4) and obtain θ

Calculate the flow error by generating the flow for the kth OD pair under the θ and compare this flow with $\hat{\mathbf{x}}^k$

end

for $k \in$ *subset_of_OD_pairs* **do**

$\mathbf{u} = \text{SNPP}(\theta)$

$\hat{\mathbf{u}} = \text{SNPP}(\hat{\theta})$

$\bar{\mathbf{u}} = \text{SNPP}(\bar{\theta})$

Calculate the performance metrics according to Section 2.4.2

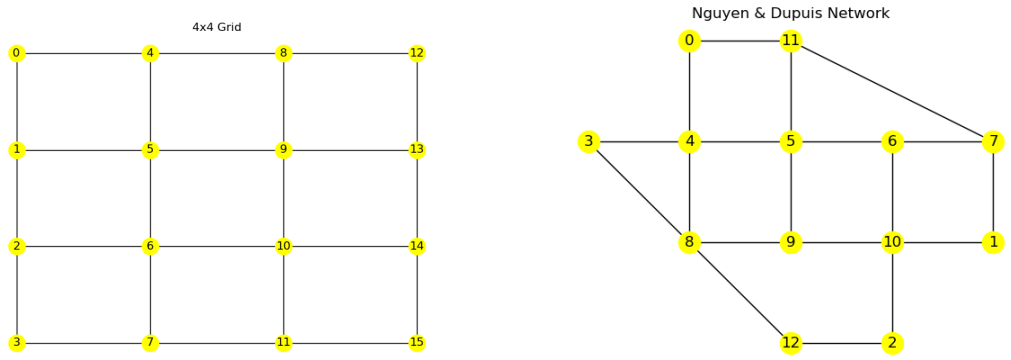
end

end

In part A, there are 10 trials for each network structure and type of cost function; in part B, within each trial, there are separate sub-trials for each OD pair in which (a) the flow error metric is calculated for the kth OD pair and (b) the performance metrics from Section 2.4.2 are calculated for a subset of OD pairs. We choose to take 10% of the OD pairs for computational reasons. The flow error metric corresponds to a type of test error for the IO algorithm because we leave out $\hat{\mathbf{x}}^k$ when calculating θ for that OD pair and, then, we take the 2-norm between the flow obtained from Algorithm 1 and the flow obtained for the kth OD pair under θ [4, 84].

2.4.4 Networks and Scenarios

We consider two networks on which to test our hybrid framework: a 4x4 grid in Figure 3.2a and the Nguyen & Dupuis network [142] in Figure 2.1b, both of which we make bidirectional.



(a) 4x4 Directed Grid Network, 16 Nodes & 48 Arcs (b) Nguyen & Dupuis Network, 13 Nodes & 38 Arcs

Figure 2.1: Illustrative Road Networks Used for Experiments

Experiment I: In the first experiment, the linear cost function along with the 4x4 grid are utilized. The linear cost function is $\phi_a x_a + \beta_a$ for each arc a. The scenarios are chosen such that every other pair of arcs are vulnerable to complete destruction. This can be seen in Figure 2.2 through the placement of the triangles with lightening bolts, indicating the arc pairs at risk. Each arc pair is given a 1/12 chance of failing.

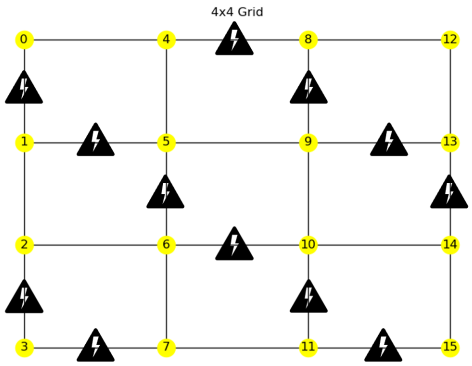


Figure 2.2: Scenarios for Experiments I & II

Furthermore, for the experiments involving the linear cost function, only the ϕ_a parameters are estimated. β_a, the free flow travel time for arc a, is assumed to be known in the majority of the papers cited in the literature review on estimating cost functions using inverse optimization [21, 43, 176, 177, 206, 207]. Consequently, for the $\hat{\theta}$, $\bar{\theta}$, and θ stochastic protection models involving linear cost, the β_a terms are the same across all of them. For the $\hat{\theta}$ protection model, the ϕ_a terms are the original cost values generated by the `unifrnd` MATLAB function, as further discussed in Section 2.4.5. For the θ protection model, the ϕ_a terms come from the inverse optimization model (2.3.4). Finally, for the $\bar{\theta}$ protection model, the $\phi_a = 6$ for all a, as indicated in Table 2.1 in Section 2.4.5.

Experiment II: In Experiment II, the inverse optimization algorithm computes the α_a parameters for each arc a for the BPR cost function. Wong and Wong [199] support having different α_a parameters across the network because they vary the α based on the structure of the network involved. Lu et al. [126] create their BPR function such that the α parameter value differs for each type of vehicle in their simulation, thus again showing that the α parameter can be different than the standard uniform 0.15 noted in Section 2.3.1.1. The IO model is given the randomly

chosen t_a^0 parameters and the capacity levels (all set to 8), and it is asked to estimate the α_a values. The uniform parameter value that is chosen for the BPR experiments is 0.15 because that is the traditionally chosen parameter value [165]. The scenario set up is the same as in Experiment I (see Figure 2.2).

Experiment III: Experiment III uses the linear cost function for the Nguyen & Dupuis network. Figure 2.3 illustrates the arcs that have a chance of failing, which are again chosen such that every other pair of arcs are vulnerable to complete destruction. There are 9 pairs of arcs indicated, which means each pair has a 1/9 chance of completely failing.

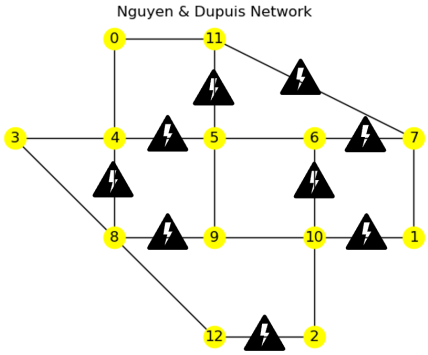


Figure 2.3: Scenarios for Experiments III & IV

The notes about the linear cost function discussed in Experiment I hold for this experiment; only the network has changed along with the $\mathbf{q}^k$ for part A of the SNPP solves. See the beginning of Section 2.4.5 for the note about $\mathbf{q}^k$ for part A of the experiments.

Experiment IV: Experiment IV employs the BPR cost function along with the Nguyen & Dupuis network. The scenario pattern is the same as in Experiment III (see Figure 2.3), and the description of Experiment II for the BPR function holds for this experiment as well.

Some more of the details for Experiments I-IV can be found in Table 2.1, with the following explanation.

- Network: The network used
- Cost Function: The transportation function used, as defined in Section 3.1.1.
- Parameters: Distributions from which the original cost function coefficients are drawn. We choose the ends of the uniform distributions for the ϕ_a, β_a, and t_a^0 parameters to be 2 and 10 so as to produce a fairly varied range of numbers but not too large of a range. We choose the range for the uniform distribution for α_a to be between 0.1 and 0.2 because typically α_a is set to 0.15 [165], so we wanted to generate parameter values that were not too large or too different from this value. The capacity c'_a values set to 8 are due to the fact that we are sending 8 units of flow between nodes in the networks.
- # Scenarios: References the number of arc pairs that are vulnerable to destruction. We obtain these numbers because we are choosing every other pair of arcs to be candidates for destruction, and each of these pairs is assigned an equal chance of failing. Therefore, we arrive at the number of scenarios listed for each experiment.
- ρ Used: Indicates a parameter value in the PH Algorithm. We obtain this value after many runs of the PH Algorithm with various values of ρ.
- Number of Cores: Refers to the number of computer cores utilized for the experiments.

Experiment #	I	II	III	IV
Network	4x4 Grid	4x4 Grid	N & D	N & D
Cost Function	Linear	BPR	Linear	BPR
Parameters	$\phi_a \sim U[2, 10]$ $\beta_a \sim U[2, 10]$	$\alpha_a \sim U[0.1, 0.2]$ $t_a^0 \sim U[2, 10]$ $c'_a = 8$	$\phi_a \sim U[2, 10]$ $\beta_a \sim U[2, 10]$	$\alpha_a \sim U[0.1, 0.2]$ $t_a^0 \sim U[2, 10]$ $c'_a = 8$
# Scenarios	12	12	9	9
ρ **Used**	5	5	5	5
Number of Cores	8	8	8	8

Table 2.1: Experiment Descriptions

To summarize all of the experiments, we have the following experiment matrix. To distinguish between experiments A and B, we indicate the number of origin-destination pairs in the SNPP solves via k. For part A experiments, there are two origin-destination pairs for the SNPP solves ($k = 2$) and, for part B experiments, there is one origin-destination pair for each of the SNPP solves ($k = 1$).

	A	B
I	Experiment IA: Linear, 4x4, & $k = 2$	Experiment IB: Linear, 4x4, & $k = 1$
II	Experiment IIA: BPR, 4x4, & $k = 2$	Experiment IIB: BPR, 4x4, & $k = 1$
III	Experiment IIIA: Linear, N & D, & $k = 2$	Experiment IIIB: Linear, N & D, & $k = 1$
IV	Experiment IVA: BPR, N & D, & $k = 2$	Experiment IVB: BPR, N & D, & $k = 1$

Table 2.2: Experiment Matrix

2.4.5 Details of the Experiments

In this section, we elaborate upon some of the details of the experiments. For each trial of each experiment, Algorithm 1 is used to generate $\hat{\mathbf{x}}^j$ for $j = 1...J$. Next, for each trial of each experiment, the hybrid framework is used to estimate a set of parameters for the current cost function and to find the protection decisions under that parameterization given a set budget.

In the experiments, three components are varied: the type of graph (4x4 grid vs. Nguyen & Dupuis (N & D)), the type of cost function (linear vs. BPR functions), and the origin-destination specification (see Table 2.2 as reference). For all of the experiments, the following remain the same:

- The MATLAB built-in function `unifrnd` is used to create the random original costs for the data generation part of each trial of each experiment.
- There are 10 trials for each experiment, each with a different original (and random) cost

parameterization.

- For part A of the experiments, the k for $\mathbf{q}^k$ is equal to 2. For the 4x4 Grid, eight units of flow go from node 0 to node 15 and from 15 to node 0 and, for the Nguyen & Dupuis network, eight units of flow go from node 0 to node 2 and from node 2 to node 0. See Figure 2.1 for the node references. For part B of the experiments, the k for each run of the SNPP is equal to 1, and the end points correspond to the chosen kth OD pair in the subset of OD pairs chosen in Algorithm 3. These subsets are chosen randomly.

- The set of scenarios correspond to each pair of arcs indicated in Figures 2.2 and 2.3. Each pair of arcs has an equal chance of failing.

- The budget for constraint (2.3.9) is set to 6. It could be modified in future work.

- For each run of the stochastic network protection problem (SNPP) in part A, $\epsilon = 0.01$ or $\epsilon = 0.001$ for the $g^{(k)}$ PH error metric [191]; we utilize the default error metric outlined in the `pysp` documentation in [94]. The maximum number of iterations is 300 for part A. Therefore, the progressive hedging algorithm stops if it reaches the tolerance or if it reaches the maximum number of iterations, whichever occurs first. Two runs of each experiment are done, one under $\epsilon = 0.01$ and one under $\epsilon = 0.001$. For part B, the adjustments are that the stochastic solves are undertaken with an $\epsilon = 0.001$ and an iteration maximum of 150. The parameters were changed between the two parts because we theorized that part B would be more computationally intensive.

2.5 Experimental Results

2.5.1 Experiment IA-IVA

The results of the experiments are evaluated with respect to two hypotheses regarding confidence intervals [188] of the means and to a direct comparison of the means of the O-IO, U-IO, and U-O metrics defined in Section 2.4.2.[4] See Appendix A.3.2 for information on the medians and minimum/maximums of the data and see Appendix A.3.3 for the run times of the experiments.

The first hypothesis states the O-IO confidence intervals for the experiments will include 0 because the inverse optimization framework can recover comparable protection decisions to the $\hat{\theta}$ protection decisions. Looking at Tables 2.3-2.4, all of the confidence intervals include 0. Therefore, there is evidence in favor of the hypothesis.

The second hypothesis states that the U-IO and U-O metric confidence intervals will not include 0 because having either cost parameters that are learned from IO or the original parameters makes a difference in protection decisions when compared to the case of uniform cost parameters. Tables 2.3-2.4 indicate that Experiments IA-IIIA present evidence in favor of the hypothesis, but Experiment IVA falls short since the confidence intervals for U-IO and U-O both include 0.

Experiments IA-IIIA demonstrate that using $\bar{\theta}$ (uniform cost) leads to different protection decisions than the $\hat{\theta}$ or θ costs. Comparing the means as a percentage of the total budget in Tables 2.3 and 2.4, we see that the O-IO metric as a percentage of the budget is small, while the U-IO and the U-O metrics as a percentage of the budget are many times greater. The small values of the

[4] See Appendix A.3.1 for the flow error metrics for the IO α values because, as can be seen from Figure 2.5, the IO α values are different from the original α values.

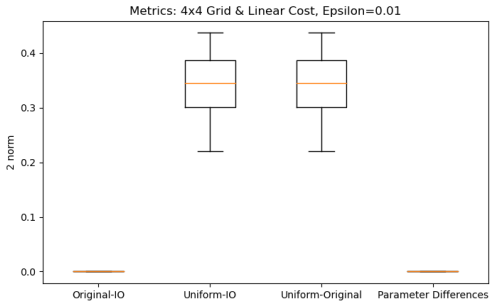
O-IO metric as a percentage of the budget indicate that IO can be used to recover parameters for the SNPP model, while the comparatively large values of the U-IO and U-O metrics indicate that the uniform protection decisions are quite different from the IO and original protection decisions, confirming the value of IO in recovering the network parameters. Boxplots in Figures 2.4 and 2.5 (along with the boxplots for Experiment IIIA in Appendix A.3.2) tell the same story.

	Experiment IA		**Experiment IIA**		**Experiment IIIA**		**Experiment IVA**	
	Mean	CI	Mean	CI	Mean	CI	Mean	CI
O-IO	0.0001 0.0%	(-0.0001, 0.0002)	0.0015 0.03%	(-0.0007, 0.0038)	0.0374 0.62%	(-0.0144, 0.0892)	0.0041 0.07%	(-0.0017, 0.0099)
U-IO	0.3449 5.75%	(0.2795, 0.4103)	0.0564 0.94%	(0.0115, 0.1013)	0.2617 4.36%	(0.2019, 0.3215)	0.0261 0.43%	(-0.028, 0.0801)
U-O	0.3449 5.75%	(0.2795, 0.4103)	0.0559 0.93%	(0.0108, 0.1009)	0.2573 4.29%	(0.2011, 0.3135)	0.0264 0.44%	(-0.0285, 0.0812)

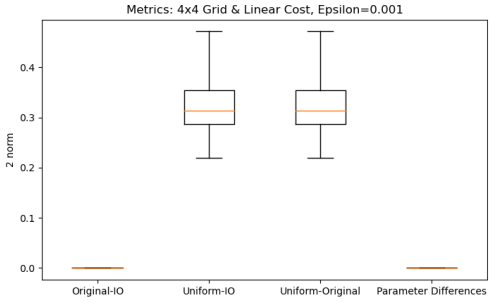
Table 2.3: Means, Means as Percentage of $I = 6$ Budget, and 99% Confidence Intervals (CI) for Experiments IA-IVA, $\epsilon = 0.01$

	Experiment IA		**Experiment IIA**		**Experiment IIIA**		**Experiment IVA**	
	Mean	CI	Mean	CI	Mean	CI	Mean	CI
O-IO	0.0001 0.0%	(-0.0001, 0.0002)	0.0013 0.02%	(-0.0002, 0.0027)	0.0251 0.42%	(-0.017, 0.0673)	0.0117 0.19%	(-0.0168, 0.0402)
U-IO	0.3304 5.51%	(0.2545, 0.4063)	0.0638 1.06%	(0.01, 0.1176)	0.2521 4.2%	(0.1873, 0.3168)	0.0197 0.33%	(-0.016, 0.0554)
U-O	0.3304 5.51%	(0.2545, 0.4063)	0.0634 1.06%	(0.0095, 0.1173)	0.2504 4.17%	(0.1807, 0.3202)	0.0257 0.43%	(-0.0158, 0.0673)

Table 2.4: Means, Means as Percentage of $I = 6$ Budget, and 99% Confidence Intervals (CI) for Experiments IA-IVA, $\epsilon = 0.001$

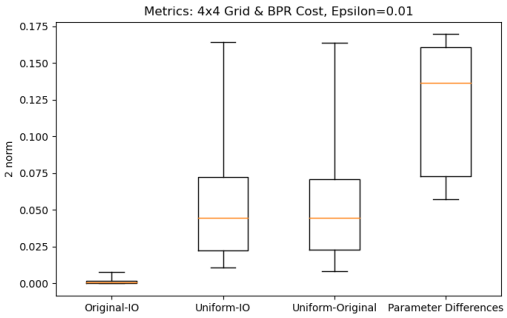


(a) Experiment IA Results for $\epsilon = 0.01$

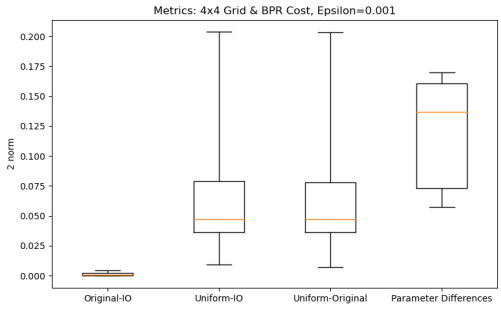


(b) Experiment IA Results for $\epsilon = 0.001$

Figure 2.4: Experiment IA Results: 4x4 Grid Network with Linear Cost. The parameter differences refer to the ϕ differences.



(a) Experiment IIA Results for $\epsilon = 0.01$



(b) Experiment IIA Results for $\epsilon = 0.001$

Figure 2.5: Experiment IIA Results: 4x4 Grid with BPR Function. Parameter differences here refers to the α differences.

2.5.2 Experiments IB-IVB

From Algorithm 3, we obtain results for Experiments IB-IVB. We calculate the metrics described in Section 2.4.2 for 10% of the OD pairs, but we calculate the flow error between the original and inverse optimization parameterizations for all of the origin-destination pairs. There are 240 OD pairs for the grid network and 156 OD pairs for the N & D network.

In Figures 2.6-2.9, we plot histograms of the decision difference metrics between $\mathbf{u}, \hat{\mathbf{u}}, \bar{\mathbf{u}}$

outlined in the first part of Section 2.4.2 for Experiments IB-IVB. We see that, for the linear function for both networks, there is a clear difference between the IO-Original metric and IO-Uniform & Original-Uniform metrics, with the IO-Original & IO-Uniform histograms showing that there is a large difference between the IO/original and uniform protection decisions. For the BPR grid Experiment IIB, there are still some OD pairs that result in protection decisions that are different between the uniform parameterization and the IO/original parameterizations. For the BPR N&D network Experiment IVB, there are really no differences between the three histograms, thus this experiment does not showcase the value of inverse optimization for making protection decisions. We do note that there is more variability in the hidden parameters for the linear cost function than for the BPR cost function, which certainly contributed to the differing IO-Uniform and Original-Uniform results between the two functions on the two graphs.

Finally, in Figures 2.10-2.11, we have the flow errors for all the OD pairs for each of the four experiments. We see that there is low flow error for all of the experiments except Experiment IVB which is BPR N & D. Therefore, we note that, as in experiment IVA, experiment IVB does not perform in the way we were expecting, which may mean not all cost function and network pairings are suitable to inverse optimization. However, most of the experiments (Experiments IB-IIIB) do support the idea that inverse optimization is recovering parameterizations that lead to similar flow patterns as the original parameterizations. We note that this low flow error is in part due to the way in which we designed the train-test experiments, with only leaving one observation out of each fold.

In these results, we demonstrate that, in most of the experiments, (a) inverse optimization can recover cost parameters that produce the same flow values as the original parameters, (b) the protection decisions are very similar between the original and inverse optimization costs, and (c)

for at least a portion of the origin-destination pairs, there were differences between the uniform cost and original & inverse optimization cost protection decisions.

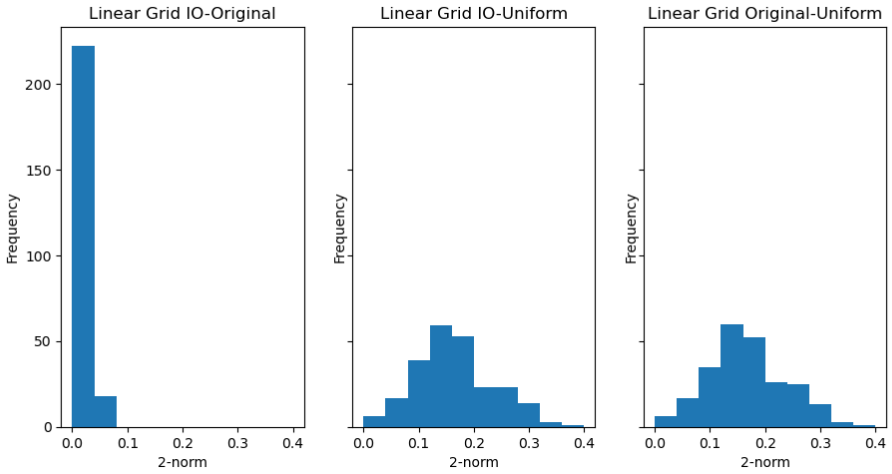


Figure 2.6: Experiment IB All OD Pairs

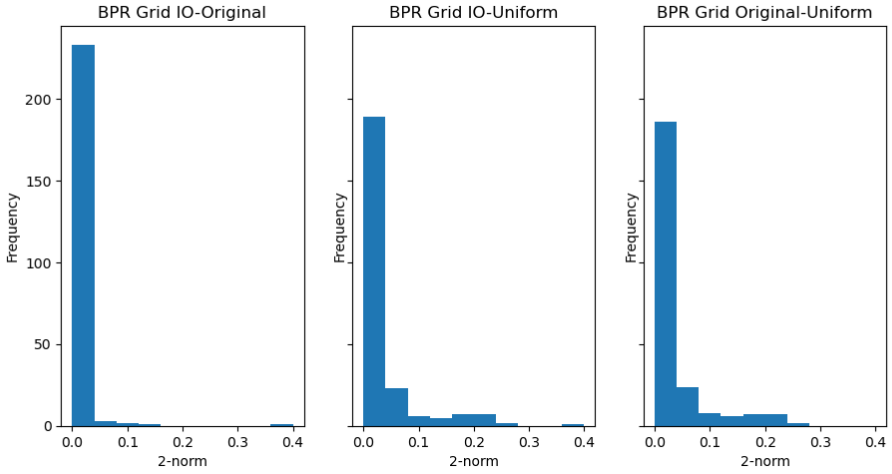


Figure 2.7: Experiment IIB All OD Pairs

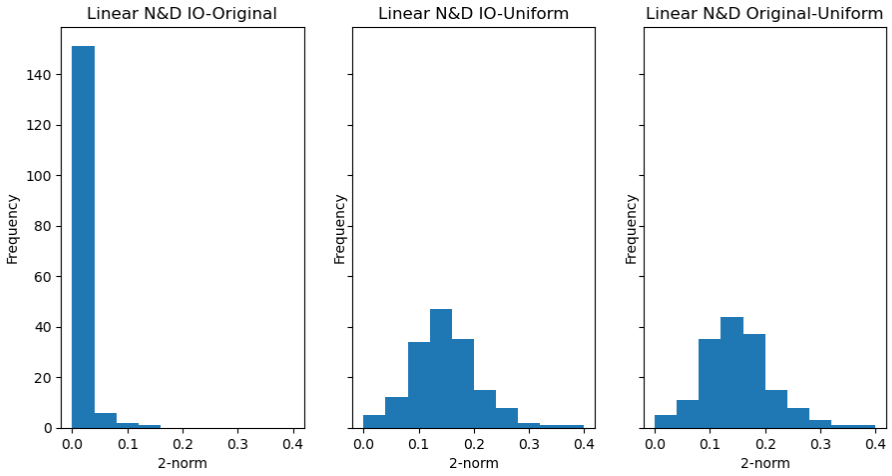


Figure 2.8: Experiment IIIB All OD Pairs

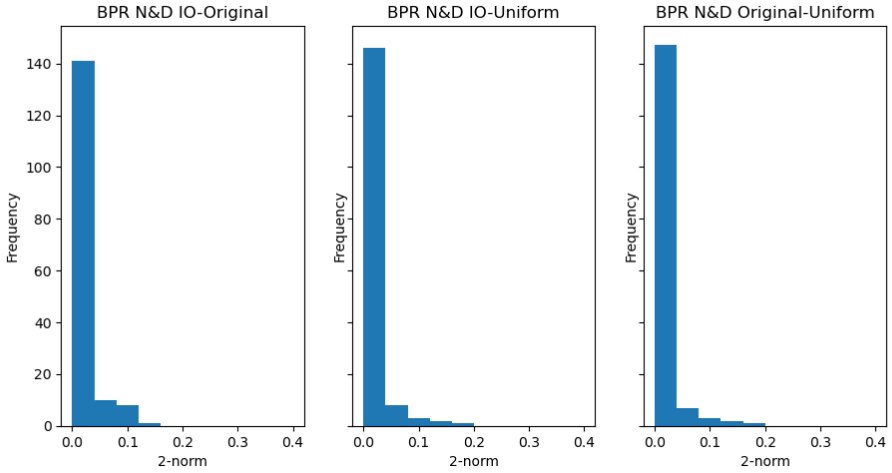


Figure 2.9: Experiment IVB All OD Pairs

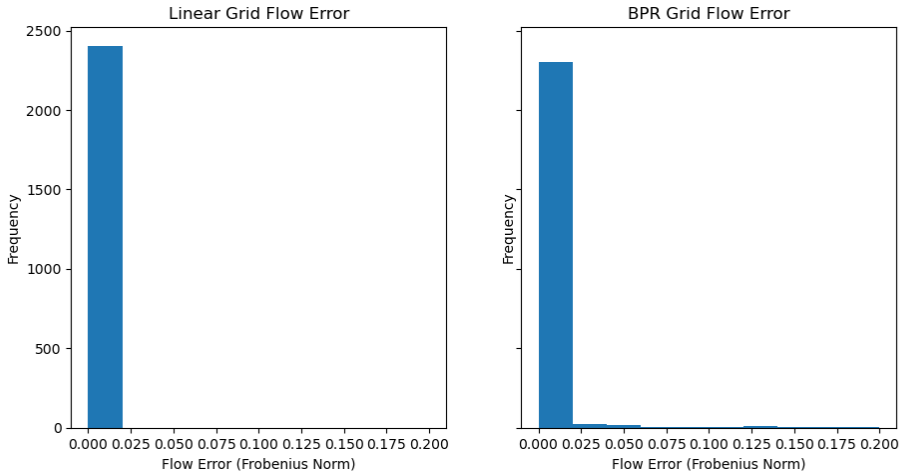


Figure 2.10: Flow Error for Linear and BPR on 4x4 Directed Grid Network: Note a few of the BPR flow error values fell outside of the 0-0.2 flow error range.

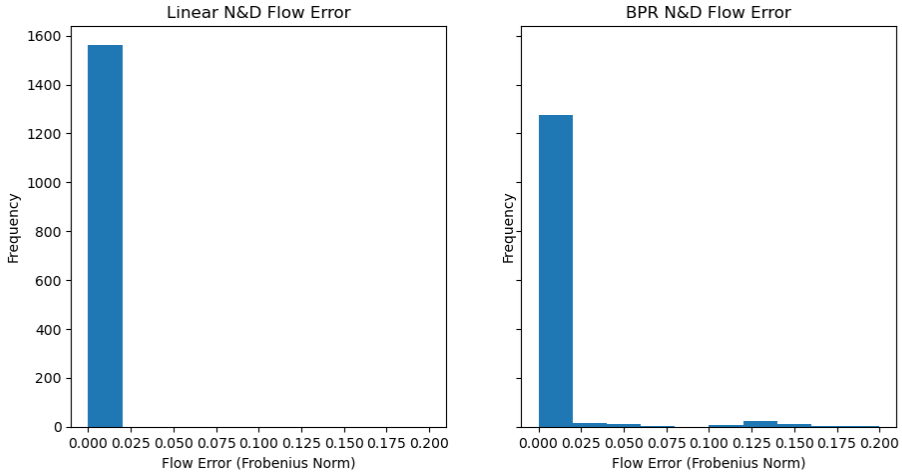


Figure 2.11: Flow Error for Linear and BPR on N & D Directed Grid Network: Note some of the BPR flow error values fell outside of the 0-0.2 flow error range.

2.6 Conclusions & Future Work

In this chapter, we have demonstrated that inverse optimization can be used as a tool to make better protection decisions in multi-stage stochastic programs for disaster relief. Using

two different networks and two different cost functions, we demonstrate that IO can be used to recover network parameters that produce similar protection decisions as the original parameters that were used to generate the data. For most of the experiments, we also demonstrate that the protection decisions are different when we have either cost parameters learned from IO or the original cost parameters compared to the protection decisions that would have been made under the assumption of uniform cost. These results suggest that inverse optimization can be used as a data analysis approach in a DSS and as a way to estimate cost parameters in multi-stage stochastic programs for disaster management. Indeed, inverse optimization is valuable precisely because it can estimate the hidden parameters in the functions that drive optimization-based systems. Being able to propose an optimization or equilibrium model of a system and, then, being able to find the parameters for the functions that drive those models are important parameter estimation capabilities. Inverse optimization is a parameter estimation tool with these capabilities and, as opposed to a method such as deep learning, we can actually understand the method in terms of optimality conditions and do not lose important properties such as convexity (for more information on deep learning, see [84]).

This chapter shows a proof of concept of the use of inverse optimization for DSS, and we leave intensive realistic studies as future work. This future work would require algorithmic innovations because the current model has to contend with issues of complementarity, stochasticity, and functional approximation (in the case of the BPR function). However, the results from this chapter demonstrate that inverse optimization is a valuable tool to be used in parameter estimation for multi-stage programs for disaster relief and decision support systems. This means that (a) we can trust the inverse optimization parameterizations to credibly predict the flow values on the networks and (b) there is a difference between using knowledge of the hidden parameters

versus not using this knowledge in protection decision making. Therefore, decision makers with an optimization or equilibrium model can use inverse optimization to estimate their model parameters. Accurate parameters will lead to accurate protection decisions, which is fundamental to disaster preparedness because, if incorrect decisions are made, important arcs in the network might not be protected adequately in the event of a natural disaster.

With regard to future work, estimating costs such that they are a function of the disaster would be something worth pursuing; indeed, it may be possible to incorporate risk metrics such as those proposed by [31] and [87]. In addition, expanding the experiments such that there is interaction between OD pairs in both the data set for the IO mathematical program and in the flow patterns for the SNPP would enrich the analysis. Furthermore, obtaining real data on scenarios and on traffic patterns would allow us to take these simulated results and apply them to the real world. Indeed, we would want to expand this work to a larger case study with this real data, which would require more algorithmic innovations as we increase the size and intricacy of the problem.

2.7 Chapter Acknowledgements

Allen was partially funded by a Graduate Fellowship in STEM Diversity while completing this research (formerly known as a National Physical Science Consortium Fellowship), which is funded by the Department of Defense. Allen was also supported by a Flagship and Dean's Fellowships from the University of Maryland, College Park and has worked for Johns Hopkins University Applied Physics Lab in the summers. Terekhov and Gabriel have no funding sources or conflicts of interest to report.

We would like to thank Dr. David Woodruff of University of California, Davis for answering questions regarding the `pysp` package.

Chapter 3: Using Inverse Optimization to Learn Cost Functions in Generalized Nash Games

Note: Most of the contents of this chapter come from our article published in *Computers & Operations Research*: [4]. We have made some additions to this research since the article was published, specifically the experiments marked as Part B in the experiments section. We have presented this research at ECOM 2021, at INFORMS 2021, and the NTNU Winter School in 2022. We thank our co-authors on this chapter: Steven Gabriel and John Dickerson. Stephanie Allen did the vast majority of the work on this project, including extending the inverse optimization framework [112, 154] discussed to jointly convex generalized Nash equilibrium problems, forming the transportation game, writing the code, and writing the paper. Steven Gabriel worked out one of the proofs and one of the counter examples in the Appendix, and he talked about a paper in one of his classes that Allen found very useful in extending the IO framework to jointly convex GNEPs. Steven Gabriel and John Dickerson provided editorial comments on the paper.

3.1 Introduction

In traditional optimization problems, a model with exogenous data is given, and the goal is to find a feasible solution that maximizes or minimizes a given objective function. In *inverse optimization* (IO) problems, a particular solution or set of solutions is instead given as input,

and the goal is to find the parameters of the *optimization problem* that resulted in that observed solution or set of solutions [3, 21, 36, 37, 82, 174, 205]. Applications of inverse optimization are myriad, including those in medical [34], energy [161], and transportation areas [207].

As a more specific example of inverse optimization, we may take as input observations of strategic behavior from rational, utility-maximizing players acting in equilibrium in a non-cooperative game and then infer the utility functions being optimized by those players' behaviors, as was done by [154, 155, 193, 194]. More specifically, Ratliff et al. [154] observe the outputs of a multi-player Nash game in which the players can affect each other's objective functions in their individualized optimization problems and, then, utilize those outputs to parameterize the players' objective functions. In the present chapter, we extend [154]'s framework to problems with "coupled" constraints as referenced by [158], [92], and [62] which involve all other players' variables in the constraints of the feasible region. This means we are parameterizing the objective functions of players in a game in which the players can affect not only each other's objective functions but also each other's feasible regions. These games with "coupled" constraints are known as jointly convex *generalized Nash equilibrium problems (GNEP)* [62, 63, 75, 92, 158]. There are important models in economics, communications, and energy that are GNEPs (see Facchinei and Kanzow [63] and Gabriel et al. [75]), so dealing with the jointly convex case of GNEPs is an important step in parameterizing the broader class of GNEPs. As our application in the chapter, we focus on the context of a multi-player generalized Nash transportation game with a "coupled" constraint in which players travel across a road network with the goal of minimizing travel time, and we utilize their observed behavior to parameterize their underlying cost functions in the game (see Figure 3.1). In the case of this transportation problem, the "coupled" constraint is a capacity constraint that requires the flow across all players on the various arcs to stay below

a certain threshold.

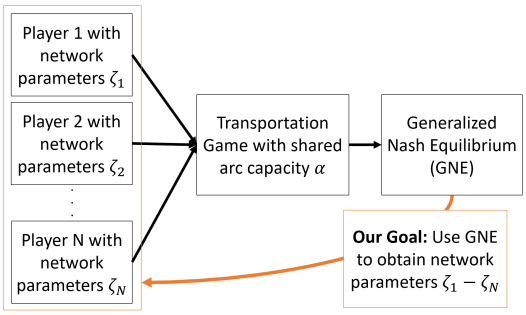


Figure 3.1: Framework for the Transportation Game

We formulate this transportation problem on small but illustrative networks, showing via simulated experiments with multiple players that we can recover objective function parameters for the players using observed player behavior in the GNEP with a "coupled" constraint such that the same flow patterns hold under the recovered parameterizations as under the original cost parameterizations. We also provide some supporting theoretical results regarding a special case of our transportation problem and the existence of unique solutions to this transportation problem. Although we utilize the transportation application in the chapter, the extension of Ratliff et al.'s [154] framework outlined in Section 3.3.3 can be utilized for other jointly convex GNEPs, as will be explained in that section.

3.1.1 Related Work

Work due to Keshavarz et al. [112] on parameterizing the objective function of a convex program has been influential in the inverse optimization community and has been discussed and referenced in multiple other inverse optimization papers such as [8, 21, 61, 154]. Ratliff et al. [154] apply the ideas of that paper to the context of game theory by making the argument that the KKT conditions [26] and other added constraints parallel different properties of Nash

equilibria, as further illustrated by [158] and [153]. Ratliff et al. [154] use data to parameterize the objective functions of players in an energy game. This combination of game theory with models/algorithms to learn parameterizations is not new; indeed, there is a subcommunity of the economics and computer science community that seeks to estimate the parameters of the players in their games [140, 151, 154, 155, 193, 194]. Notably, [140] reconstructs the incentives of players using data by assuming no-regret learning on the part of the players, which allows for more realistic modeling of players than the stronger Nash assumption of perfect knowledge. The authors of [193, 194] also encode the requirement of low regret into their inverse optimization model of multi-player interaction in a correlated equilibrium game while maximizing entropy of the mixed-strategy actions of the players. The authors of [151] point to the importance of having robust methods when estimating players' parameters because our rationality assumptions can be problematic, and our models of the players might be incorrect. Other literature directly from the inverse optimization community does deal with game theory and multi-player models. Nguyen [141] proposes an inverse optimization strategy for peer-to-peer energy systems in which the movement of energy from one entity to another is modeled, but the players are not modeled independently, thus making it different from our model. Chen et al. [39] augment Bertsimas et al.'s [21] inverse optimization framework to obtain parameters for a player's "rivals"; the authors seem to be dealing with a "coupled" constraint in their IO model, but they are focused upon one application whereas we present a way of handling this type of constraint more generally. Fernández-Blanco et al. [69] use a bilevel approach to inverse optimization and model electricity demand for a "pool" of consumers, thus not following our approach of a game theoretic model. Risanger et al. [155] do model an energy equilibrium problem and use inverse optimization techniques from [112], but they do not include a "coupled" constraint. Finally, there is literature

on finding cost functions for differential games, but those models differ significantly from the models we envision; interested readers can see references such as [11, 107, 135].

With regard to transportation problems, our example used in the present work, the methods of inverse optimization and game theory have both been applied before. The inverse optimization literature on transportation problems has focused on parameterizing cost functions [57, 108, 143, 200, 206], finding capacity values on a network [43], and also on finding the "weights" for components of objective functions in transportation-themed models [41, 42]. In reference to game theory and transportation, [60] reviews a number of game theory models that deal with such transportation phenomenon as "speed choice" and "merging," and [203] identify types of games that appear in transportation research which include generalized Nash equilibrium games. In a recent work, [168] solve a multi-player version of the bipartite graph transportation problem as a GNEP, thus demonstrating that generalized Nash problems are a difficult problem type that is still being explored. The merging of game theory/equilibrium problems, the estimation of parameters for these problems, and transportation has been seen in a number of papers [21, 22, 43, 114, 117, 175, 176, 177, 206, 207]. In particular, [117, 175], and [22] focus on and/or cover the combination of routing games in transportation with estimating parameters, a category into which our chapter more specifically fits. Other researchers have pursued similar work to ours in the inverse GNEP space [98], taking more of a pure mathematical approach. Still others have utilized a "coupled" capacity constraint in their models of player movement [200] in an effort to estimate "network dual prices" for the constraint.

For the papers that closely align with the current chapter, we contrast their methodology with ours. In [175], the researchers focus on "minimizing risk" with regard to estimating the cost functions in a routing game, and the reseachers do not deal with any "coupled" constraints.

By contrast, we focus on KKT/optimality conditions instead of statistical quantities and tackle a different kind of problem in the form of a generalized Nash equilibrium problem. In [22], the researchers approach estimating parameters in a routing problem through a "query" system in which they: (a) control part of the objective function to obtain a desired flow pattern and (b) control part of the flow to obtain a desired flow pattern. This contrasts with our approach since we recover a set of parameterizations instead of influencing the flow, and we again work with a generalized Nash game while these researchers do not. [117] most closely align with our work because the researchers' work can be directly applied to atomic routing games, as opposed to the previous two papers [22, 175] that focused upon non-atomic routing games in which individual players are not considered (see [159] for more details on atomic versus non-atomic routing games). While [117] devise a mathematical program to solve the problem of estimating parameters, similar to our approach, they do not deal with a "coupled" constraint in their approach. Hemmecke et al. [98] focus on a theoretical framework for finding cost functions in generalized Nash games, which contrasts with our approach that extends an applied framework using optimality conditions to find cost functions in generalized Nash games with "coupled" constraints. Finally, [200] formulate a set of linear programs based on work by [3] to handle their problem of finding "network dual prices" for their "coupled" capacity constraint and utilize an averaging algorithm to find these prices. Although [200] work with a "coupled" constraint, we work with and extend another model from [154] and [112] that has been influential in the literature (as seen in [8, 21, 61, 154]).

3.1.2 Our Contribution

We contribute to the literature by extending the parameterization framework of Ratliff et al. [154] to the context of jointly convex GNEP games, which allows us to model more sophisticated interactions among players in Nash games by considering the players' effects on each other in both their objective functions and constraint sets [63, 75]. We expand upon Ratliff et al. [154]'s extension of Keshavarz et al. [112]'s framework because it was relatively straightforward to expand. This differs from other inverse optimization literature that either does not include "coupled" constraint(s) [155, 193], does not model the problem as a multi-player game with individual player optimization problems [69, 141], or does include the former two elements but makes the inverse model too specific to their application [39] (and builds off the framework of Bertsimas et al. [21] as opposed to Keshavarz et al. [112]). What's more, while [98] discuss theoretical results with regard to inverse optimization and GNEPs, we pursue an applied framework to recover parameterizations for cost functions. For our example transportation game with a "coupled" capacity flow constraint for the arcs on the network, we differ from the transportation papers that combine equilibrium problems with inverse optimization, such as [21, 207], and the other papers listed in Section 3.1.1 involving estimating parameters, including the non-atomic routing game literature of [175] and [22], because our method models traffic at the level of the player and includes a "coupled" constraint. Our approach also differs from [117] who model traffic at the level of the player because of this "coupled" constraint. Finally, although one of [200]'s models includes a "coupled" constraint, we focus on extending the framework by [154] and [112] as opposed to their use of [3]'s framework. Our contribution to the literature is significant because we extend a framework that has been influential in the literature (as seen in

[8, 21, 61, 154]) to jointly convex generalized Nash equilibrium problems.

3.2 Preliminaries

As our application in the chapter, we examine the problem of combining N-players' shortest path problems to create a traffic game in which the N players affect each other's costs on the network via interaction terms in their objective functions as well as via a "coupled" capacity constraint in the feasible region. The problem parallels what is known as an atomic routing game [159] which features multiple players each with the ability to affect the outcome of the traffic game and which can be represented as a multi-user Nash equilibrium game [46, 146]. However, atomic routing games as presented in [159] do not have "coupled" constraints. We do note that this model was also inspired by [200] who formulate their problem as a set of *inverse* shortest path problems and who did include a "coupled" capacity constraint in one of their formulations.

For a network with n arcs and m nodes, let $x_i \in \mathbb{R}^n$ to be the flow decision variables for the network arcs for each player $i \in [N]$, which can be fractional. The positive diagonal matrix $\mathcal{C}_i \in \mathbb{R}^{n \times n}$ represents the coefficients for the interactions between player i's flow and the aggregate flow, capturing the dynamic costs associated with increasing flow from the players. The vector $\bar{c}_i \in \mathbb{R}^n$ represents the base cost for traveling along each link in the network; this is also called the free flow travel time by various papers [21, 43, 166, 177, 206, 207]. We also take $D \in \{-1, 0, 1\}^{m \times n}$ as the conservation-of-flow matrix which keeps track of the net flow at the nodes [129] and $\alpha \in \mathbb{R}^n$ as the collective capacity for the N players' flow on each arc. The vector $f_i \in \mathbb{R}^m$ for each player i indicates the starting or origin (in negative units of flow) and ending or destination (in positive units of flow) points in the network of the flow for each player $i \in [N]$.

Thus, these two points make up an origin-destination or OD pair. As an example for f_i, if player $i = 1$ begins at node 2 and ends at node 12 in a network with 16 nodes, then their f_1 vector would have a dimension of 16×1, and we would place a -1 at entry 2 and a 1 at entry 12. Given all of these variable and parameter definitions, the resulting optimization problem faced by player i is:

$$\min_{x_i} x_i^T \mathcal{C}_i \left(\sum_{j=1}^{N} x_j \right) + \bar{c}_i{}^T x_i \tag{3.2.1a}$$

$$Dx_i = f_i \text{ (dual variable: } v_i) \tag{3.2.1b}$$

$$x_i \geq 0 \text{ (dual variable: } u_i) \tag{3.2.1c}$$

$$\sum_{j=1}^{N} x_j \leq \alpha \text{ (dual variable: } \bar{u}) \tag{3.2.1d}$$

This is a N-player traffic game in which the N players affect each other's costs on the network via the objective function, which has an interaction term between player i's flow and all other players' flow and a term representing the base cost for traveling along each arc. Also, there is a shared constraint in the flows (3.2.1d). This constraint makes this problem a generalized Nash equilibrium problem [62, 63, 75, 92, 158]. We also have the conservation of flow constraints in (3.2.1b) for each player i and the non-negativity constraints in (3.2.1c) for each player i.

In this chapter, we will examine the case in which the $\mathcal{C}_i$ matrices and $\hat{c}_i$ vectors are equal across all N players and in which they are different across all N players. The first case represents the situation in which each player faces the same interaction costs and free flow travel times as all other players which is a common assumption in aggregate models such as [21, 176, 177]. The second case represents a view in which the players do not have a consensus view of the interaction

costs or the free flow travel times, as proposed by [41, 42, 200].

In order to obtain these matrix and vector parameters, we extend the parameterization formulation proposed by [154] to GNEPs with "coupled" constraints. [154] call their formulation a residual model after the similar [112] model. Our extended residual model is defined as follows for the N players [112, 115, 154]:

$$\min_{\mathcal{C}_i, \bar{c}_i, v_i^k, u_i^k, \bar{u}^k} \sum_k \left(\overbrace{\sum_{i=1}^{N} ||\mathcal{C}_i \left(2x_i^k + \sum_{j \neq i} x_j^k \right) + \bar{c}_i + D^T v_i^k - u_i^k + \bar{u}^k||_1}^{\text{stationarity}} + \overbrace{\sum_{i=1}^{N} ||(x_i^k)^T u_i^k||_1}^{\text{complementarity \#1}} + \overbrace{||(\alpha - \sum_j x_j^k)^T (\bar{u}^k)||_1}^{\text{complementarity \#2}} \right)$$

$$L_1 \leq diag(\mathcal{C}_i) \leq U_1 \; \forall i \tag{3.2.2a}$$

$$L_2 \leq \bar{c}_i \leq U_2 \; \forall i \tag{3.2.2b}$$

$$\bar{u}^k \geq 0 \; \forall k \; u_i^k \geq 0 \; \forall i, k \tag{3.2.2c}$$

At a high level, our objective function naturally decomposes into three parts (labeled above as "stationarity," "complementarity #1," and "complementarity #2"); we now describe the intuition behind each part.

- The **stationarity** component is a norm corresponding to the stationarity conditions of the optimization problem for each player i and each data piece k, with the data pieces representing the flow patterns we observe between the origin-destination pairs on the network. For this norm, the x_i^k are data of the flow patterns for each player i and each origin-destination pair k, and our variables are the diagonal of the $\mathcal{C}_i \; \forall i$, the vector $\bar{c}_i \; \forall i$, u_i^k as the

dual variables for constraint (3.2.1c) for all i, k, v_i^k as the dual variables for constraint (3.2.1b) for all i, k, and $\bar{u}^k$ as the dual variables for constraint (3.2.1d) for each data piece/origin-destination pair k. An origin-destination (OD) pair indicates the beginning (origin) and ending (destination) points of a player's path. We have one set of $\bar{u}^k$ dual variables for each k that are shared across the players, which will be further explained in Section 3.3.

- The **complementarity #1** norm refers to the complementarity conditions for the nonnegativity constraint across the data and the players
- The **complementarity #2** norm refers to the complementarity conditions that pair with the "coupled" constraint for each data piece.

In conjunction, these three components of the objective allow us to minimize the error across the relevant KKT conditions for all players and all data. The constraints on this objective include requiring that the diagonal of the $\mathcal{C}_i$ matrices remain between two bounds $L_1 \in \mathbb{R}^n$ and $U_1 \in \mathbb{R}^n$ as captured in (3.2.2a) and that the vectors $\bar{c}_i$ remain between two bounds $L_2 \in \mathbb{R}^n$ and $U_2 \in \mathbb{R}^n$ as captured in (3.2.2b). The use of upper and lower bounds for the $\mathcal{C}_i$ and $\bar{c}_i$ was inspired by papers such as [21, 37, 112, 115, 154] which discuss utilizing "prior information" [112] pertaining to the parameters to be estimated to aid the mathematical program in finding viable parameterizations. We remember that the diagonals of $\mathcal{C}_i$ represent the interaction costs for each player i and the vectors $\bar{c}_i$ represent the free flow travel cost for each player i so, by placing bounds on these variables, their estimated values will be on the same order of magnitude as the original values that generated the simulation data.[1] The final constraint (3.2.2c) on the objective requires that the

[1] We do want to ensure that a set of parameterizations results in a convex Nash game since we are dealing with a

dual variables $\bar{u}^k$ and u_i^k are non-negative because they correspond with inequality constraints.

One important point about this model is that it can be solved in polynomial time. From [26], we know we can convert the L-1 norm components into linear constraints, with positive variables that bound the constraints from above and below and that replace the L-1 terms in the objective function. This leads to a linear program, which we also know from [26] can be solved in polynomial time via the interior point method. Furthermore, the growth in the number of variables as a function of the inputs also grows as a polynomial function, the details of which can be seen in Appendix B.1. Overall, we summarize all of this in the following lemma:

Lemma 1. *Problem (3.2.2) can be solved in polynomial time, and the number of variables grows as a polynomial function of the inputs.*

Having established the complexity of our inverse optimization model, we present an example of our framework.

Example 3.2.1. *We randomly generate a parameterization for the arcs on a 4x4 grid which consists of 48 total arcs. There are 48 arcs on this grid because there are 4 sets of 3 edges moving down the grid and then the same pattern moving across the grid; therefore, we multiply* $(4)(3)$ *by 2 and, then, multiply it by 2 again because we assume two arcs going in opposite directions per edge. We generate this parameterization by uniformly choosing 48 numbers between 1 and 5 for the diagonal of* $\mathcal{C}_i$ *such that* $\mathcal{C}_1 = ... = \mathcal{C}_N = \mathcal{C}$ *and uniformly choosing 48 numbers between 2 and 10 for the vector* $\bar{c}_i$ *such that* $\bar{c}_1 = ... = \bar{c}_N = \bar{c}$*. We set the number of players as* $N = 10$ *and then generate the flows for these parameterizations using all 240 origin-destination (OD)*

set of minimization problems across players, and convex games result in Nash equilibrium solutions [146, 154, 158]. Due to the fact that our minimization problems for all of the players are linear in each player's variables, we do not require any extra constraints [146, 154, 158], but this might be required for other applications as demonstrated by [115].

pairs - a number obtained by calculating $\binom{16}{2}$, since there are 16 nodes in a 4x4 grid, and then multiplying by 2 to account for our assumption that the arcs from origin a to destination b might be different from the arc from origin b to destination a. For each OD pair, all of the players each start this one unit of flow at the origin and flow this one unit of flow to the destination, meaning that for $N = 10$ players there are 10 units of flow on the network. We then input the resulting flows under these origin-destination pairs into the residual model above (3.2.2) to obtain the parameterizations for $\mathcal{C}$ and $\bar{c}$. Thus, the "knowns" in the inverse optimization model are the conservation-of-flow matrix, the origin-destination pairs, and the capacity levels of the arcs, while the "unknowns" in the inverse optimization model are the $\mathcal{C}_i$ and $\bar{c}_i$ cost coefficients.

With regard to the results of example 3.2.1, the original parameterizations and the parameterizations obtained by the inverse optimization residual model do not align. Indeed, the Frobenius norm difference between the two $\mathcal{C}$ matrices is 2.4777, and the 2-norm difference between the two parameterizations for the $\bar{c}$ vector is 22.4281. However, if we instead use an error metric that compares the flows under the original parameterizations versus the flows under the IO parameterizations, we see promising results. Indeed, when we compare the flows among the 10 players for all 240 origin destination pairs across all 48 arcs on the network, we find that the difference between the flow patterns using a Frobenius norm metric is 8.1369e-06. We utilize the Frobenius norm because it allows us to calculate essentially a 2-norm between the two flow matrices, one resulting from the original costs and one resulting from the IO costs. Consequently, as indicated by the low Frobenius norm error, the IO solution leads to the same type of flow pattern observed under the original parameterization for the set of OD pair configurations considered, thus making the parameters obtained by the IO residual model a reasonable solution to the inverse optimization

problem.

Next, Section 3.3 explains the connection between the residual model [112, 154] and the generalized Nash equilibrium problem. Section 3.4 presents the experimental framework. Section 3.5 discusses our experimental results, and Section 3.6 concludes with a brief discussion and open problems.

3.3 Theoretical Background and Connections

In this section, we describe our simulation approach, form the residual model more abstractly, and then make the connection between the simulation conditions and the residual model. In Section 3.3.1, we discuss the connection between jointly convex GNEPs and mixed complementarity problems, with the MCPs forming the simulation approach. In Section 3.3.2, we point to the residual model utilized by [154]. Finally, in Section 3.3.3, we propose and prove a collorary stating that we can incorporate "coupled" constraints into the residual model to recover the parameterizations for the objective functions of the players in a jointly convex GNEP.

3.3.1 Theoretical Considerations for Generating the Simulation Data

Using the notation from [63], for each player v in a generalized Nash problem, the goal is to solve the following optimization problem, with N as the number of players, n_v as the number of variables for player v, $x^v \in \mathbb{R}^{n_v}$ as the variables for player v, $\mathbf{x}^{-v}$ as containing all other players' variables, $X_v(\mathbf{x}^{-v}) \subseteq \mathbb{R}^{n_v}$ as the "feasible set" parameterized by the other players' variables, $\theta : \mathbb{R}^{\bar{n}} \to \mathbb{R}^N$ with $\bar{n} = \sum_v n_v$, and $\theta_v : \mathbb{R}^{\bar{n}} \to \mathbb{R}$ (with the recognition that the last function has

$\mathbf{x}^{-v}$ as parameters in its functional form once we move to the GNEP problem)

$$\min_{x^v} \theta_v(x^v, \mathbf{x}^{-v}) \tag{3.3.1a}$$

$$x^v \in X_v(\mathbf{x}^{-v}) \tag{3.3.1b}$$

The authors of [63] and [75] convey that a solution to this problem is an $\mathbf{x}$ such that no player can minimize their objective further, fixing the other players' variables. We can form a problem to find this solution (known as a generalized Nash equilibrium) as long as we have the Convexity Assumption stated in [63]:

Definition 1 (Convexity Assumption [63]). *For every player v and every $\mathbf{x}^{-v}$, the objective function $\theta(\cdot, \mathbf{x}^{-v})$ is convex and the set $X_v(\mathbf{x}^{-v})$ is closed and convex.*

With this Convexity Assumption, the following theorem holds in which the solution of a quasi-variational inequality $\text{QVI}(\mathbf{X}(\mathbf{x}), \mathbf{F}(\mathbf{x}))$ problem is defined as $\mathbf{x}^* \in \mathbf{X}(\mathbf{x}^*) \subseteq \mathbb{R}^{\bar{n}}$ such that

$$\mathbf{F}(\mathbf{x}^*)^T(\mathbf{y} - \mathbf{x}^*) \geq 0 \; \forall \mathbf{y} \in \mathbf{X}(\mathbf{x}^*) \tag{3.3.2}$$

Theorem 1 (Theorem 3.3 of [63]). *Let a GNEP be given, satisfying the Convexity Assumption, and suppose further that the θ_v functions are C^1 for all v. Then a point $\mathbf{x}$ is a generalized Nash equilibrium if and only if it is a solution of the quasi-variational inequality QVI$(\mathbf{X}(\mathbf{x}), \mathbf{F}(\mathbf{x}))$, where*

$$\mathbf{X}(\mathbf{x}) = \prod_{v=1}^{N} X_v(\mathbf{x}^{-v}), \quad \mathbf{F}(\mathbf{x}) = (\nabla_{x^v} \theta_v(\mathbf{x}))_{v=1}^{N} \tag{3.3.3}$$

This QVI representation of the GNEP can be further simplified to a related variational inequality (VI) under certain conditions on the $X_v(\mathbf{x}^{-v})$. For a variational inequality, the goal is to find $\mathbf{x}^* \in \mathbf{X} \subseteq \mathbb{R}^{\bar{n}}$, with $\mathbf{X}$ not parameterized by the variables, such that

$$\mathbf{F}(\mathbf{x}^*)^T(\mathbf{y} - \mathbf{x}^*) \geq 0, \ \forall \mathbf{y} \in \mathbf{X} \tag{3.3.4}$$

We see that the $\mathbf{X}$ set is different between the QVI and the VI, with the QVI $\mathbf{X}$ set having dependence on the $\mathbf{x}^{-v}$ variables. One way to transform the QVI into a VI is via joint convexity of the GNEP, a property that applies if the following definition holds pertaining to the $X_v(\mathbf{x}^{-v})$ sets:

Definition 2 (Definition 3.6 of [63]). *Let a GNEP be given, satisfying the convexity assumption. We say that this GNEP is jointly convex if for some closed, convex* $\mathbf{X} \subseteq \mathbb{R}^{\bar{n}}$ *and all* $v = 1, ..., N$, *we have*

$$X_v(\mathbf{x}^{-v}) = \{x^v \in \mathbb{R}^{n_v} : (x^v, \mathbf{x}^{-v}) \in \mathbf{X}\} \tag{3.3.5}$$

This definition means that the x^v and $\mathbf{x}^{-v}$ are all part of the same set $\mathbf{X}$, that any "coupled" constraints include all of the variables from all of the players, and that these "coupled" constraints are convex in all of the players' variables [62, 63, 75, 92, 158]. If we assume that the multipliers are equal across the copies of the "coupled" constraints for each player, then we can combine the constraints from each set $X_v(\mathbf{x}^{-v})$ into one set $\mathbf{X}$, which also implies that we will have one copy of any "coupled" constraints that depend upon all the players' variables in the set $\mathbf{X}$ [62, 63, 75, 92, 158]. As a result, we can utilize the following theorem:

Theorem 2 (Theorem 3.9 of [63]). *Let a jointly convex GNEP be given with* C^1 *functions* θ_v.

Then every solution of the $VI(\mathbf{X}, \mathbf{F})$ (where $\mathbf{X}$ is the set in the definition of joint convexity and, as usual, $\mathbf{F}(\mathbf{x}) = (\nabla_{x^v}\theta_v(\mathbf{x}))_{v=1}^N$), is also a solution of the GNEP.

This theorem states that we can utilize the VI with the combined $\mathbf{X}$ set instead of the QVI with $\mathbf{X}(\mathbf{x})$ to obtain a solution to the original generalized Nash problem [62, 63, 64, 75, 92, 158]. The equality of multipliers assumption leads to less complicated models and methods; see [138] for an example of a method that does not utilize the equality of multipliers assumption. We know there exists a solution to this VI based upon the following theorem from [93] who cite [58, 97]:

Theorem 3 (Theorem 3.1 of [93])**.** *Let $\mathbf{X}$ be a nonempty, compact and convex subset of $\mathbb{R}^{\bar{n}}$ and let $\mathbf{F}$ be a continuous mapping from $\mathbf{X}$ into $\mathbb{R}^{\bar{n}}$. Then there exists a solution to the problem VI$(\mathbf{X}, \mathbf{F})$.*

Due to the fact that our $\mathbf{X}$ is nonempty, compact, and convex as a result of the linear constraints, the lower bound provided by the non-negativity constraints, and the upper bound provided by the capacity constraint and due to the fact that our function $\mathbf{F}$ is continuous, we know that a solution exists for our VI. See Appendix B.2 for uniqueness considerations for the VI. With the VI formulation, we can then use a well-known theorem from [64] to convert the VI into a mixed complementarity problem. Facchinei and Pang [64] define the $\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^{\hat{n}} : A\mathbf{x} \leq b, G\mathbf{x} = d\}$ such that $A \in \mathbb{R}^{\hat{m}\times\hat{n}}, G \in \mathbb{R}^{\hat{l}\times\hat{n}}, b \in \mathbb{R}^{\hat{m}}, d \in \mathbb{R}^{\hat{l}}$. They have the following proposition:

Proposition 1 (Proposition 1.2.1 from [64])**.** *Let $\mathbf{X}$ be defined as above. A vector $\mathbf{x}$ solves the VI$(\mathbf{X}, \mathbf{F})$ if and only if these exists $\lambda \in \mathbb{R}^{\hat{m}}$ and $\mu \in \mathbb{R}^{\hat{l}}$ such that*

$$0 = \mathbf{F}(\mathbf{x}) + G^T\mu + A^T\lambda \tag{3.3.6a}$$

$$0 = d - G\mathbf{x} \tag{3.3.6b}$$

$$0 \leq \lambda \perp b - A\mathbf{x} \geq 0 \tag{3.3.6c}$$

This proposition states that, if we find a solution to the KKT system, we will solve the VI defined by $\mathbf{X}$ and $\mathbf{F}$, so long as the $\mathbf{X}$ is a set of linear inequality and linear equality constraints. Therefore, for GNEP applications with this type of $\mathbf{X}$, we can generate simulation data to perform inverse optimization analysis by forming a mixed complementarity problem (MCP), as we do for the transportation example in Section 3.4.

3.3.2 General Residual Model [154]

Having explained that we use a mixed complementarity form of our generalized Nash game (3.2.1) to produce our simulation data by operating under the assumption of equal multipliers for "coupled" constraints [62, 63, 75, 92, 158], we can explain the inverse optimization model we use to extract parameters from this simulation data. We utilize the residual model showcased in [154], which is quite similar to the one presented in [112], with the difference being that [154] sum over players in a game. As a note to the reader, we will explain this model utilizing the notation from [154]'s more recent paper [115]. The model starts with the assumption that each player solves an optimization problem such as the following:

$$\min\{f_i(x_i, x_{-i})|x_i \in \mathcal{B}_i\}$$

with $f_i(x;\theta_i) = \langle \phi_i(x_i, x_{-i}), \zeta_i \rangle + \bar{f}_i(x)$ and $\mathcal{B}_i = \{x_i | h_{i,j}(x_i) \leq 0, j = 1, ..., l_i,\ g_{i,q}(x) = 0,\ q = 1, ..., ll_i\}$, in which i represents the player, l_i represents the number of inequality constraints for player i, ll_i represents the number of equality constraints for player i, ζ_i represents the linear parameterization of the objective function for player i, and $\bar{f}_i$ is the "known" term in the objective function [115]. In our game, each player also has one "coupled" constraint inequality in the set $\mathcal{B}_i$ through which the players can influence each others' feasible regions. Extending this model to accommodate this "coupled" constraint will be covered in Section 3.3.3.

If we take the KKT conditions of this problem [26], we obtain the following relevant "residuals" [112, 115, 154], in which D_i represents the derivative with respect to x_i:

$$r_{s,i}^{(k)}(\zeta_i, \mu_i, \nu_i) = D_i f_i \left(x_i^{(k)}, x_{-i}^{(k)} \right) + \sum_{j=1}^{l_i} \mu_i^j D_i h_{i,j} \left(x_i^{(k)} \right) + \sum_{q=1}^{ll_i} \nu_i^q D_i g_{i,q} \left(x_i^{(k)} \right) \tag{3.3.7a}$$

$$r_{c,i}^{j,(k)}(\mu_i) = \mu_i^j h_{i,j} \left(x_i^{(k)} \right),\ j \in \{1, ..., l_i\} \tag{3.3.7b}$$

with k representing the "kth observation" in the game, s representing the label for the stationarity residual, and c representing the label for the complementarity residual [115, 154]. We then have $\mathbf{x}^{(k)}$ data for each instance k of the game as a composite vector containing all the players' variable values, and we attempt to choose ζ, μ, ν such that we minimize the residuals presented above. This leads to the optimization problem proposed by [154] and [115] inspired by [112]:

$$\min_{\mu,\zeta,\nu} \sum_{i=1}^{N} \sum_{k=1}^{\eta_i} \chi_i \left(r_{s,i}^{(k)}(\zeta, \mu, \nu), r_{c,i}^{(k)}(\mu) \right) \tag{3.3.8a}$$

$$\zeta_i \in Z_i,\ \mu_i \geq 0,\ \forall i \in \{1, ..., N\} \tag{3.3.8b}$$

with N representing the total number of players and η_i representing the number of times a player i engages in the game which, in our application, means the number of origin-destination runs the player undertakes. We assume in our implementation that η_i is equal across all players, meaning that each player engages in each iteration of the game. It is also important to note that there are separate parameterizations for each player i in the form of ζ_i. Furthermore, these parameterizations each belong to their own convex set Z_i. Finally, we specify that χ_i is a "non-negative convex penalty function" [115, 154] which, according to [26], is applied to each set of residuals such that a sum of residuals is formed for each set of residuals. This objective for the optimization problem thus acts as a type of loss function for finding a parameterization for the problem [26, 38, 115, 154].

3.3.3 Generalized Nash Connection to Residual Model

With background on the conversion of the generalized Nash problem to a mixed complementarity form and the use of the residual model for inverse optimization, we presently discuss extending the residual inverse optimization model to accommodate jointly convex generalized Nash equilibrium problems. We note that the notation used in this subsection differs in some ways from the notation in Section 3.3.1, mainly in the definition of the $\mathbf{X}$ set. Before we present our corollary regarding incorporating "coupled" constraints into the inverse optimization model, we state a theorem from [62] in modified notation from [63]. This theorem requires the following specification of $\mathbf{X}$ for

the purposes of the theorem as:

$$\mathbf{X} = \{\mathbf{x} \in \mathbb{R}^{\bar{n}} : h_q(\mathbf{x}) \leq 0 \; q = 1, ..., m, \; g_j(\mathbf{x}) = 0 \; j = 1, ..., p\} \tag{3.3.9}$$

where m represents in this case the total set of all inequality constraints across all players and p represents the total set of equality constraints across all players. In the set $\mathbf{X}$, we incorporate all of the constraints for all of the players, including one copy of the "coupled" constraints. We make the assumption that we began with a jointly convex GNEP [62]. We could also write the subset of constraints for each player v as:

$$X_v(\mathbf{x}^{-v}) = \{x^v : h_q(x^v, \mathbf{x}^{-v}) \leq 0 \; q = 1, ..., m, \; g_j(x, \mathbf{x}^{-v}) = 0 \; j = 1, ..., p\} \tag{3.3.10}$$

For each of these subsets $X_v(\mathbf{x}^{-v})$, all of the constraints for all of the players still exist, but we are differentiating between player v's variables and the other players' variables more explicitly; each subset $X_v(\mathbf{x}^{-v})$ also has its own copy of the "coupled" constraints. The set $\mathbf{X}$ and the subsets $X_v(\mathbf{x}^{-v}) \; \forall v$ lead to two statements of the KKT conditions. First for the $X_v(\mathbf{x}^{-v})$ subsets, according to [62, 63, 64], if we assume that a constraint qualification holds for each player, a solution for each player v will correspond with a KKT point that satisfies the following KKT conditions for each individual player v:

$$\nabla_{x^v} \theta_v(x^v, \mathbf{x}^{-v}) + \sum_{q=1}^{m} \lambda_q^v \nabla_{x^v} h_q(x^v, \mathbf{x}^{-v}) + \sum_{j=1}^{p} \nu_j^v \nabla_{x^v} g_j(x^v, \mathbf{x}^{-v}) = 0 \tag{3.3.11a}$$

$$0 \leq \lambda_q^v \perp h_q(x^v, \mathbf{x}^{-v}) \leq 0, \; \forall q \tag{3.3.11b}$$

$$g_j(x^v, \mathbf{x}^{-v}) = 0,\ \nu_j^v \text{ free},\ \forall j \tag{3.3.11c}$$

Second for the $\mathbf{X}$ set, we know from [19, 62, 64] that, if a constraint qualification is satisfied, then the KKT conditions are satisfied by a solution to the VI (assuming the presence of relevant multipliers) with $\mathbf{F}$ defined as (3.3.3) and $\mathbf{X}$ defined as (3.3.9). We also know from [64] that, if h_q are convex and g_j are affine, then a solution to the KKT conditions is a solution to the VI with $\mathbf{F}$ defined as (3.3.3) and $\mathbf{X}$ defined as (3.3.9). Therefore, we have the resulting KKT conditions from the VI as:

$$\mathbf{F}(\mathbf{x}) + \sum_{q=1}^{m} \lambda_q \nabla_{\mathbf{x}} h_q(\mathbf{x}) + \sum_{j=1}^{p} \nu_j \nabla_{\mathbf{x}} g_j(\mathbf{x}) = 0 \tag{3.3.12a}$$

$$0 \leq \lambda_q \perp h_q(\mathbf{x}) \leq 0, \forall q \tag{3.3.12b}$$

$$g_j(\mathbf{x}) = 0,\ \nu_j \text{ free},\ \forall j \tag{3.3.12c}$$

We can now convey Theorem 4.8 from [63] who cite [62, 92] which states that we can move between the two forms of the KKT conditions (3.3.11) and (3.3.12) so long as we assume the multipliers λ^v, ν^v for (3.3.11) are equal across all players v. This statement of the theorem is slightly different from either [62]'s statement or [63]'s statement because we have added equality constraints and their associated multipliers into our formulation of these KKT conditions.

Theorem 4 (Theorem 4.8 [63])**.** *Consider the jointly convex GNEP with $h_q\ \forall q, g_j\ \forall j, \theta_v$ being C^1. Then the following statements hold:*

- *Let $\bar{\mathbf{x}}$ be a solution of the VI$(\mathbf{X}, \mathbf{F})$ (with $\mathbf{X}$ defined as (3.3.9), $\mathbf{F}$ defined as (3.3.3), and VI defined as (3.3.4)) such that the KKT conditions (3.3.12) hold with some multipliers $\bar{\lambda}$ and $\bar{\nu}$. Then $\bar{\mathbf{x}}$ is a solution of the GNEP, and the corresponding KKT conditions (3.3.11) are satisfied with $\lambda^1 = ... = \lambda^N = \bar{\lambda}$ and with $\nu^1 = \nu^2 = ... = \nu^N = \bar{\nu}$.*

- *Conversely, assume that $\bar{\mathbf{x}}$ is a solution of the GNEP such that the KKT conditions (3.3.11) are satisfied with $\lambda^1 = ... = \lambda^N$ and with $\nu^1 = \nu^2 = ... = \nu^N$. Then $(\bar{\mathbf{x}}, \bar{\lambda}, \bar{\nu})$ with $\bar{\lambda} = \lambda^1$ and with $\bar{\nu} = \nu_1$ is a KKT point of VI$(\mathbf{X}, \mathbf{F})$, and $\bar{\mathbf{x}}$ itself is a solution of VI$(\mathbf{X}, \mathbf{F})$.*

Proof. As a brief proof of this theorem (which is based off of, and is a relatively direct extension of, a related proof by [62]), for the first statement, we notice a correspondence between (3.3.11a) concatenated for all players v and (3.3.12a) as long as the multipliers are equal across players and are equal to the $\bar{\lambda}, \bar{\nu}$ multipliers. This correspondence continues for (3.3.11b) & (3.3.12b) and (3.3.11c) & (3.3.12c) so long as the multipliers are equal. Therefore, the $(\bar{\mathbf{x}}, \bar{\lambda}, \bar{\nu})$ from (3.3.12) is a KKT point for the conditions (3.3.11) for each player. According to [62], as long as the conditions (3.3.11) are sufficient for optimality for each player, which is true if we assume $\mathbf{F}$ is convex, h_q are convex, and g_j are affine [26], $\bar{\mathbf{x}}$ with multipliers $\bar{\lambda}, \bar{\nu}$ is a solution for each player's optimization problem, thus making it a GNEP solution.

For the second statement, we point to the fact that, now that we are given that the multipliers are equal across all of the players, then there is a correspondence between (3.3.11) (concatenated for all players v for the stationarity constraint) and (3.3.12) as long as $\bar{\lambda} = \lambda_1 = ... = \lambda_N$ and $\bar{\nu} = \nu_1 = ... = \nu_N$. Consequently, we have a KKT point for the (3.3.12) conditions that comes from the given solution to (3.3.11). We know from [64] that if there is a KKT point for the conditions (3.3.12) and we assume the constraints h_q are convex and g_j are affine, then the $\bar{\mathbf{x}}$

associated with the KKT point is a solution for the VI. □

Our technique, as discussed in Section 3.2, can be looked at as a corollary to this theorem, where we state that we can use the stacked KKT conditions with one copy of the "coupled" constraints to parameterize the cost functions of the players. This corollary is ours and encapsulates the methodological contribution of this paper.

Corollary 1. *With the assumption of equal multipliers for the "coupled" constraint(s) in a GNEP, the stationarity conditions and complementarity conditions of the VI KKT conditions (3.3.12) may be used to form a residual model of the form seen in [112, 115, 154], with χ as a "non-negative convex penality function" as in those papers and in [26] and with k representing the multiple data points $\mathbf{x}^k$:*

$$\min_{\zeta_i,\lambda_q,\nu_j} \sum_k \chi\left(\mathbf{F}(\boldsymbol{x}^k) + \sum_{q=1}^{m} \lambda_q \nabla_{\boldsymbol{x}} h_q(\boldsymbol{x}^k) + \sum_{j=1}^{p} \nu_j \nabla_{\mathbf{x}} g_j(\mathbf{x}^k)\right) + \sum_{k,q} \chi\left(\lambda_q h_q(\boldsymbol{x}^k)\right) \tag{3.3.13a}$$

$$\lambda_q \geq 0 \; \forall q, \; \zeta_i \in Z_i \; \forall i \tag{3.3.13b}$$

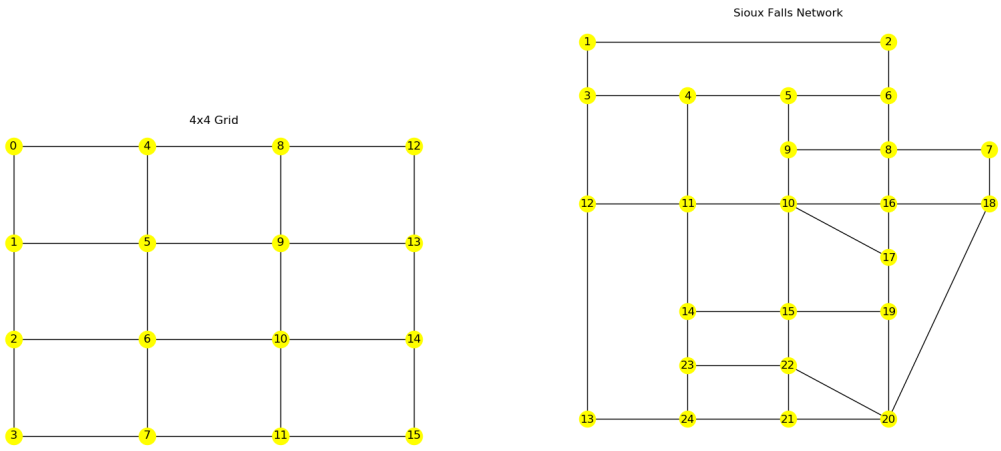
Proof. We know from Theorem 4 that there is a correspondence between the solutions to (3.3.11) for all players ν and the solution to (3.3.12) so long as the multipliers are equal across these two sets of KKT conditions. Therefore, we can take the stationarity condition and the complementarity conditions of (3.3.12) as the residuals for the GNEP problem and form the same kind of residual problem as (3.3.8). □

As a result of corollary 1, we can find parameterizations for players' cost functions in generalized Nash games with "coupled" constraints. Specifically, in our transportation problem, we can find

$\mathcal{C}_i$ and $\bar{c}_i$ for players $i \in [N]$. In other applications, we would replace the complementarity #2 term in (3.2.2) with the appropriate "coupled" constraint(s) and their associated multipliers.[2] Next, in Section 4, we explain our experimental framework, walking the reader through our simulation set-up and the specifications for our residual model for the transportation problem.

3.4 Experimental Framework

To test and explore our inverse optimization framework for jointly convex GNEPs, we use the transportation game discussed in Section 3.2 on various grid networks, which can be expressed as a certain number of nodes vertically and a certain number of nodes horizontally (2 nodes by 2 nodes, 3 nodes by 3 nodes, 4 nodes by 4 nodes, and 5 nodes by 5 nodes), and on the Sioux Falls network [120, 179]. Examples of these networks are visualized in Figures 3.2a and 3.2b, respectively.



(a) 4 nodes x 4 nodes Grid

(b) Sioux Falls Network [120, 179]

Figure 3.2: Networks utilized for testing inverse optimization framework

[2]The stationarity and complementarity #1 parts of (3.2.2) would also change for a different application.

To generate the solutions to use in our inverse optimization framework, we solve the following mixed complementarity problem [26, 63, 75]:

$$0 \leq x_i \perp \mathcal{C}_i \left(2x_i + \sum_{j \neq i} x_j \right) + \bar{c}_i + D^T v_i + \bar{u} \geq 0 \; \forall i = 1, ..., N \tag{3.4.1a}$$

$$0 \leq \alpha - \left(\sum_j x_j \right) \perp \bar{u} \geq 0 \tag{3.4.1b}$$

$$Dx_i = f_i, \; v_i \text{ free } \forall i = 1, ..., N \tag{3.4.1c}$$

with dual variables of $u_i \; \forall i, \bar{u}$, and $v_i \; \forall i$. We note that the system established in (3.4.1) will be solved repeatedly for different right-hand side f_i vectors to generate our simulation data. We utilize the implementation of PATH [54, 70] from the software GAMS [77, 78] for this problem.

As stated in the introduction, our goal is to solve the inverse optimization residual model for points generated from the GNEP, specifically with the purpose of finding the parameters $\mathcal{C}_i$ and $\bar{c}_i$ for each player i. In some of the experiments, these $\mathcal{C}_i$ and $\bar{c}_i$ will be assumed the same across all players and, in others, they will be assumed different across all players. For our experiments, we follow a general framework in which various elements can be modified. The general framework is as follows:

Algorithm 4: Basic Experimental Framework

Input: D conservation-of-flow matrix, n nodes, m arcs, α capacity level, N number of players, bounds L_1, L_2, U_1, U_2

Exhaustively generate $\mathcal{P}$, the set of all unique origin-destination pairs for a given network

for *each pair in* $\mathcal{P}$ **do**

Solve the mixed complementarity problem (3.4.1) form of GNEP (3.2.1) for RHS f_i $\forall i$ constructed from the current pair, with one unit of flow for each player

Store the N player flow vectors x_i $i = 1, ..., N$ (which are the data points)

end

Input the sets of data points into the IO mathematical program (3.2.2) to obtain estimates for $\mathcal{C}_i$ and $\bar{c}_i$ $\forall i$

We utilize all $\mathcal{P}$ origin-destination pairs for a given network because we want to gain a detailed picture of flow on that network. However, we recognize that there are many more configurations of possible flow from which we could have drawn which may lead to further study, including configurations that would involve the players starting and ending at different points.

3.5 Experimental Results

In this section, we present the numerical results of experiments with the proposed inverse optimization framework to parameterize the GNEP (3.2.1) using the two types of networks showcased in Section 3.4. To implement the experimental setup described in Section 3.4, we used a data generation and optimization pipeline including state of the art solvers from PATH [54, 70] in GAMS and Gurobi [90]. This included randomly generating from a uniform distribution the original costs for the $\mathcal{C}_i$ and $\bar{c}_i$ parameters in the simulation model outlined in (3.4.1). See

Appendix B.5 for more information about the experimental setup.

The boxplots[3] in this section denote two different measures of error. First, there is the value of the objective function of the residual model, denoting how closely the chosen parameterizations cause the input data to satisfy the KKT conditions [112, 115, 154]. Second, there is the total flow error metric, which measures the difference between the flows under both the original parameterization and the IO residual model parameterization using the same set of OD pairs $\mathcal{P}$. The Frobenius-norm metric is utilized, which acts as a vector 2-norm for matrices. This is useful compared to other matrix norms because it calculates a total difference between the two matrices, not a maximal difference along rows or columns or an abstract eigenvalue metric as in the case of the more traditional matrix norms [195]. This leads to the following metric.

Definition 3 (Flow error)**.** *The **flow error** is calculated by first taking the squared difference between the flow under the original parameterization and the flow under the IO parameterization for all origin-destination pairings, all players, and all arcs. Then, a sum is taken across these squared differences and, finally, the square root of this summation is taken. This error represents the Frobenius norm between the two sets of flows.*

This error is also normalized by the total number of arc flows, which is calculated by multiplying the number of origin-destination pairings, the number of players, and the number of arcs. It forms the following normalized metric.

Definition 4 (Normalized flow error)**.** *The **normalized flow error** is calculated by dividing the flow error by a factor created by multiplying the number of origin-destination pairings, the number of players, and the number of arcs. It represents a per unit level of error.*

[3]For all of the box plots, the "whiskers" are placed at quartile 1 - 1.5 (quartile 3 - quartile 1) and at quartile 3 + 1.5 (quartile 3 - quartile 1), and the "dots" are outliers [105].

Note that data are collected regarding the differences between the original parameterizations and the IO parameterizations, and it is found that there are differences between the two. However, the important metrics are the objective function value and the flow error metrics because the first measures how well the IO parameterization fits the model and the model data and because the second showcases whether or not the IO parameterization leads to the same flow patterns observed for the OD pairs under the original parameterization. Note also that the objective function values for the experiments can be found in Appendix B.7 and timing information for the experiments can be found in Appendix B.8. Details regarding the Experiments can be found in Table 3.1.

Finally, we do have two sets of results to discuss in this section. In the first set of results, we do not employ a train-test methodology but, in the second set of results, we do employ a train-test methodology. This train-test methodology uses standard five fold cross-validation in which the data set is split into five non-intersecting sets [84]. In each of the five iterations of the cross-validation procedure, we use four out of the five sets to find the parameters for the transportation game using inverse optimization and, then, we "test" on the remaining fifth of the data by finding the flows on the OD pairs corresponding to those data pieces and compare the resulting flows against this data. We define the following sets and functions in order to modify Goodfellow et al.'s [84] k-fold cross validation algorithm to our application:

- $\mathcal{G}$ - the graph we are studying
- $\mathcal{D}^{\mathcal{G}}$ - the data set of flows corresponding to each of the OD pairs for the given graph
- $\mathcal{D}_i^{\mathcal{G}}$ - the ith segment of the data set when splitting the data into k disjoint subsets
- IO$(\cdot)$ - the inverse optimization framework applied to the data

#	Network	Costs	Players (N)	# Trials	$\mathcal{C}$ Bounds	$\bar{\mathbf{c}}$ Bounds	α
1	2x2-5x5	Same	2, 5, 10	10	[1,5]	[5,20]	$0.5(N)$, N
2	Sioux Falls	Same	2, 5, 10	10	[1,5]	[5,20]	$0.5(N)$, N
3	2x2-5x5	Different	2, 5, 10	10	[1,5]	[5,20]	$0.5(N)$, N
4	Sioux Falls	Different	2, 5, 10	10	[1,5]	[5,20]	$0.5(N)$, N

Table 3.1: Experimental Details. Here, # refers to the experiment group number, Network refers to the graph upon which the experiment was run, $\mathcal{C}$ Bounds indicating the range for randomization and the bounds used in the IO mathematical program for the diagonal of the $\mathcal{C}$, $\bar{c}$ Bounds indicates the same for the $\bar{c}$ parameters, and α refers to that parameter value. These experiments are repeated for both the part A and part B results.

- θ - parameterization resulting from IO$(\cdot)$
- TG$(\theta, \mathcal{D}_i^{\mathcal{G}})$ - traffic game given some parameterization θ and the OD pairs corresponding with the $D_i^{\mathcal{G}}$ data set
- $\mathcal{E}_i$ - error between the resulting flows under θ parameterization and the original flows $D_i^{\mathcal{G}}$ for segment i

Algorithm 5: k-Fold Cross Validation Framework from Goodfellow et al. [84] (modified to this application)

Input: $\mathcal{G}$, $\mathcal{D}^{\mathcal{G}}$, IO$(\cdot)$, k folds

Split $\mathcal{D}^{\mathcal{G}}$ into k mutually exclusive subsets $\mathcal{D}_i^{\mathcal{G}}$, whose union is $\mathcal{D}^{\mathcal{G}}$

for $i = 1 : k$ **do**

$\theta = \text{IO}(\mathcal{D}^{\mathcal{G}} \backslash \mathcal{D}_i^{\mathcal{G}})$

$\mathcal{E}_i = ||\text{TG}(\theta, \mathcal{D}_i^{\mathcal{G}}) - \mathcal{D}_i^{\mathcal{G}}||_2$

end

We label the non train-test results as "part A" and the train-test results as "part B." We also note that we only report flow error for the part B experiments, so the boxplot discussions above only apply to the part A experiments. Part B uses histograms

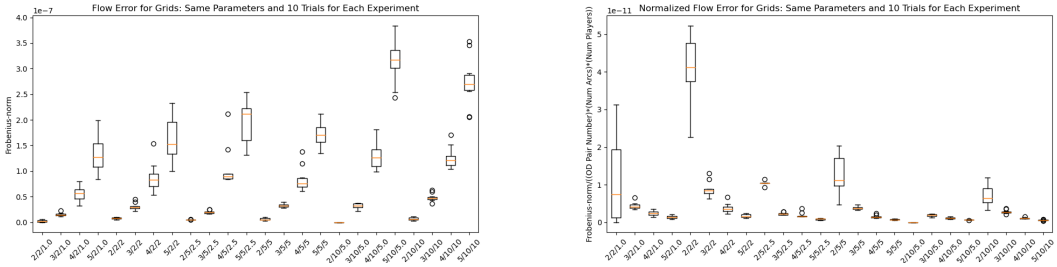
3.5.1 Broader Observations about the Results: Part A

Overall, the results from the four experimental groups (with two in the same cost randomized costs category and two in the different cost randomized costs category) are encouraging. The maximum objective function values for all the experiments across the groups are on the order of 1e-6, and the maximum flow errors (see Definition 3) for all of the experiments across the groups are on the order of 1e-6 or 1e-7. Therefore, these error metrics indicate that viable parameterizations are being recovered for the OD pair sets under which we are testing the framework. Indeed, the fact that these errors do not greatly differ between same and different costs is encouraging because it indicates that the approach can work to recover people's different perceptions of road networks [21, 41, 42, 176, 177, 200]. It should be noted that, for different costs, the experiments with $\alpha = 5$ and $N = 10$ for the 5x5 grid and Sioux Falls did not solve completely, with the 5x5 grid experiment only able to complete 6 iterations and the Sioux Falls experiment unable to complete any at the time of this posting. While Lemma 1 shows that (3.2.2) is solvable in polynomial time, any solver will naturally encounter scalability problems as the problem size increases. However, the reality that the framework did work for most of the experiments under restrictive α values of $(0.5)N$ validates the extension of the framework, because some of the OD pairs involved start, end, or both nodes where only two arcs were coming out of or into the node, which meant the capacity constraints were guaranteed to be tight.

3.5.2 Same Randomized Costs: Part A

For experimental groups 1 and 2, there is the assumption of "same randomized costs" across the players, meaning $\mathcal{C}_i$ and $\bar{c}_i$ are equal across all players i such that $\mathcal{C}_1 = \mathcal{C}_2 = ... = \mathcal{C}_N = \mathcal{C}$

and $\bar{c}_1 = \bar{c}_2 = ... = \bar{c}_N = \bar{c}$. Experiment group 1 (Grid) iterates over generated 2x2, 3x3, 4x4, and 5x5 grids, each with 10 trials of randomly chosen parameters, which the inverse optimization residual model attempts to estimate. For each grid, three different player numbers ($N = 2, 5, 10$) and two different α values, one set to half the number of players for each N and one set to exactly the number of players for each N, are considered. The graphs for this experiment group display many box plots, and the system utilized for labeling the box plots is Grid Size/Number of Players/Alpha Value. Each set of eight moving from left to right indicate the same number of players. With regard to flow error, Figure 3.3a demonstrates that, for all of the subsets of players and α values, as grid size increases, the total flow error also increases. It is important to note that everything on this graph is still on the order of 1e-7, but this trend is also evident. It likely results from the fact that, as grid size increases, the number of arcs and number of OD pairs also increases and, since this measure is calculated across all OD pairs, all players, and all arcs, then even if consistent error were assumed across all arcs, the norm would have to increase. Indeed, upon examining the accompanying Figure 3.3b, it is apparent that the normalized flow error, calculated as described in Definition 4, decreases as grid size increase across all of the subsets of players and α values.

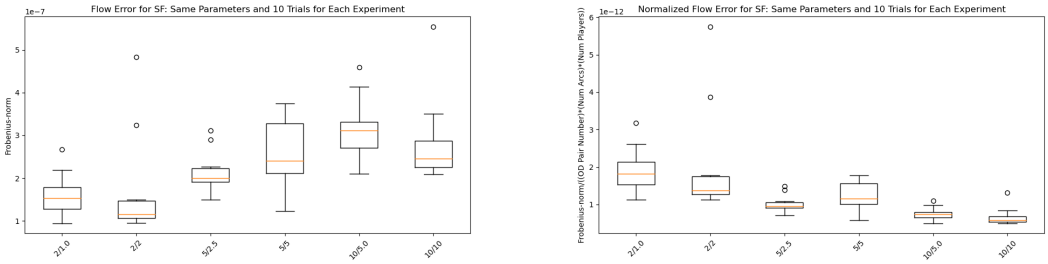


(a) Flow Error for Experiment Group 1

(b) Normalized Flow Error for Experiment Group 1

Figure 3.3: Flow Error Metrics for Experiment Group 1: The labeling at the bottom of the graphs indicates attributes of the boxplot, specifically the Grid Size/Number of Players/Alpha Value

Experiment group 2 (Sioux Falls) again consists of 10 trials for each of $N = 2, 5, 10$ players and the two $\alpha = (0.5)N, N$ values. The labeling for the graphics is set to Number of Players/Alpha Value. In examining flow errors, Figure 3.4a showcases that the flow errors do not appear to follow a set pattern when moving between the different numbers of players, yet they are all still small and on the order of 1e-7 or lower. However, in Figure 3.4b, the median flow error decreases as number of players increases, even on this very small scale (1e-12).



(a) Flow Error for Experiment Group 2

(b) Normalized Flow Error for Experiment Group 2

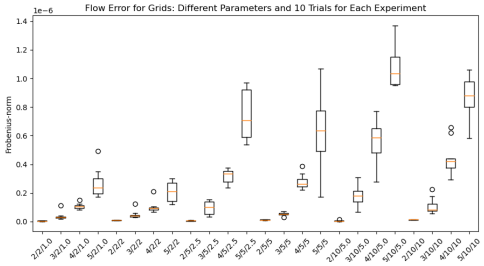
Figure 3.4: Flow Error Metrics for Experiment Group 2: The labeling at the bottom of the graphs indicates attributes of the boxplot, specifically the Number of Players/Alpha Value

3.5.3 Different Randomized Costs: Part A

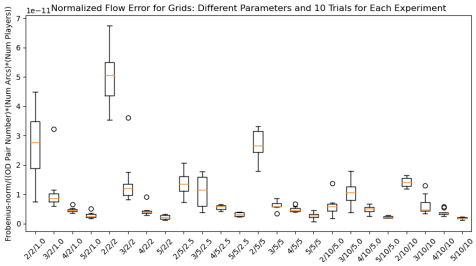
For experimental groups 3 and 4, there is the assumption "different randomized costs" across the players, meaning that a new C_i and a new $\bar{c}_i$ are drawn for each player i. Experiment group 3 is a mirror image of experiment group 1, except that all of the players do not have the same costs. One issue with this experiment group was that not all of the trials finished for the case of the 5x5 grid, 10 players, and $\alpha = 5$; only 6 of the trials finished in under 24 hours,[4] so they are the ones included in the box plots. Similar to the flow error for the grids under the same costs (Figure 3.3a), Figure 3.5a demonstrates that the median flow error increases as the grid size

[4]This means about 24 hours were given for each trial before the trial was stopped, with the exception of one that was stopped by the computer before convergence.

increases for all the subsets of player number and α value. Figure 3.5b illustrates much of the same decrease in median normalized flow as grid size increases (among the subsets) as in Figure 3.3b for same costs across all players.



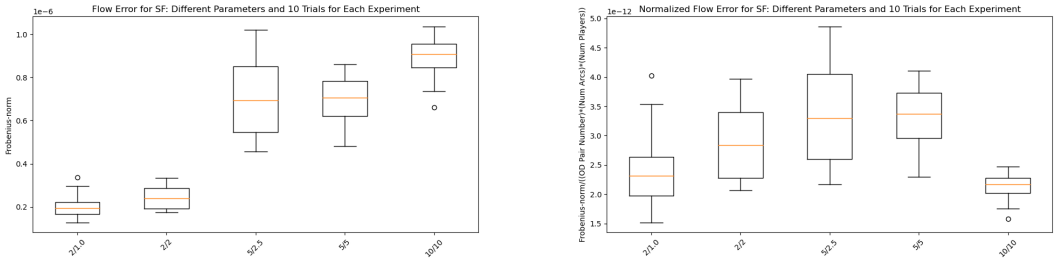
(a) Flow Error for Experiment Group 3



(b) Normalized Flow Error for Experiment Group 3

Figure 3.5: Flow Error Metrics for Experiment Group 3: The labeling at the bottom of the graphs indicates attributes of the boxplot, specifically the Number of Players/Alpha Value. Note: Only 6 trials are included for 5/10/5.0, see note in the text.

Experiment group 4 (Sioux Falls) again consists of 10 trials for each of $N = 2, 5, 10$ players and the two $\alpha = (0.5)N, N$ values. The labeling for the graphics is Number of Players/Alpha Value. It should be noted that the trials did not finish at the time of this posting for the $N = 10$ and $\alpha = 5$ experiment, so those boxplots are not included in Figure 3.6. As concerns flow error, Figure 3.6a shows increasing median flow error as player number increases, which was not as visible in Figure 3.4a for experiment group 2. However, unlike Experiment group 2's (Sioux Falls, but in the same cost setting) Figure 3.4b, Figure 3.6b does not demonstrate the same consistent increase in median normalized flow error, with normalized median flow decreasing with $N = 10$ and $\alpha = 10$.



(a) Flow Error for Experiment Group 4

(b) Normalized Flow Error for Experiment Group 4

Figure 3.6: Flow Error Metrics for Experiment Group 4: The labeling at the bottom of the graphs indicates attributes of the boxplot, specifically the Number of Players/Alpha Value. Note: The trials did not finish in time for $N = 10$ and $\alpha = 5$, hence that boxplot is not included.

3.5.4 Broad Conclusions: Part B

For this part of the experiments, we present a representative sample of the results, namely the results for the 2×2 Grid, 4×4 Grid, and the Sioux Falls graph.[5] See Appendix for the additional graphs. For the broad conclusions, we find that the Sioux Falls experiments overall produce better results than the various Grid experiments for both the same and different costs experiments, despite the Sioux Falls network being arguably more complicated than the other two Grids. As expected for all of the networks, the different costs flow errors are greater than the same costs flow errors.

3.5.5 Same Randomized Costs: Part B

For presenting our results in the next two subsections, we use histograms to denote the flow errors. There are 50 flow errors per histogram, since there are 10 random cost parameterizations to recover and 5 folds of the train-test set-up. The histograms are positioned according to the

[5]We note that not all of the experiments finished for the 5×5 Grid for different costs, in particular the $p = 10$ experiments. Not all of the experiments finished for different costs for the Sioux Falls graph, namely $p = 10$ results.

player number p on the vertical axis and the α value on the horizontal axis. This convention carries through to the different costs experiment.

The results overall are encouraging. The Sioux Falls results in Figure 3.8 in particular demonstrate that the parameterizations recovered result in flow errors that are quite small. The oddities are some of the results for $\alpha = N$ in the 2×2 Grid and for $p = 2$ in the 4×4 Grid. These errors are a bit larger than we expected, especially considering the results we received back from Sioux Falls. Further work would need to explore if there is a graph structure element to why certain graphs produce better errors than others.

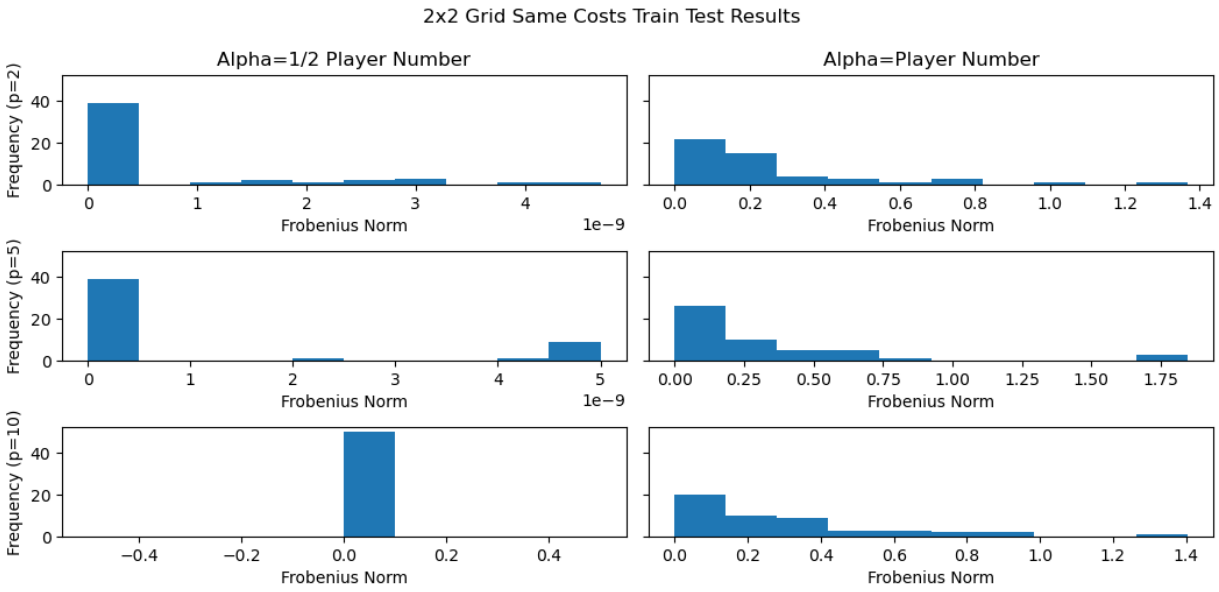


Figure 3.7: 2×2 Grid Train Test Same Costs

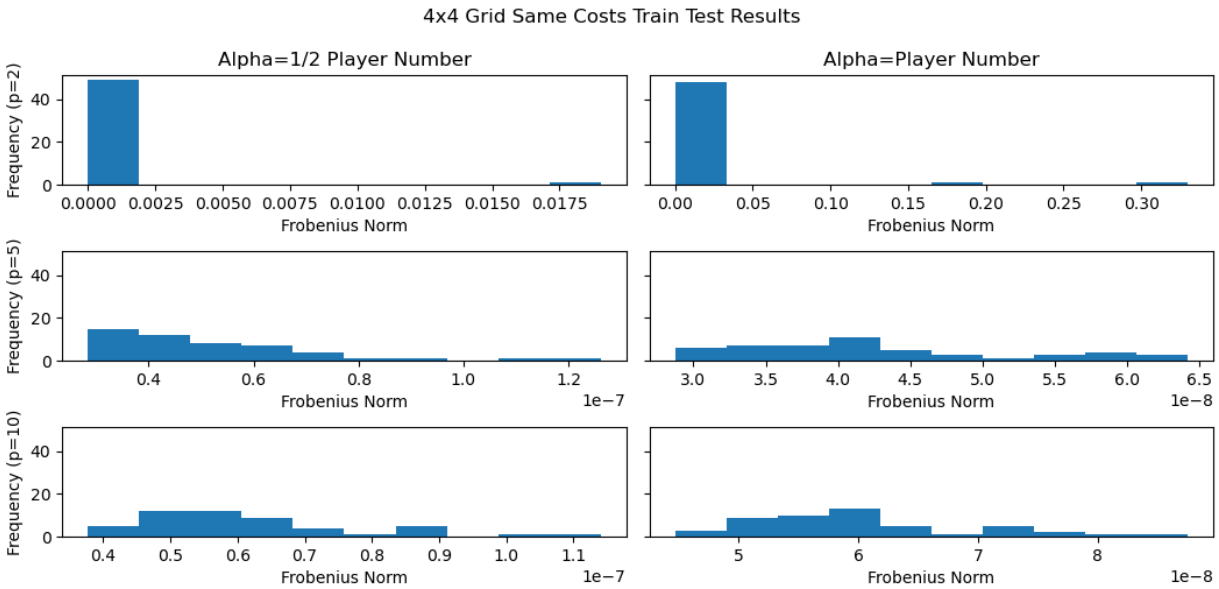


Figure 3.8: 4×4 Grid Train Test Same Costs

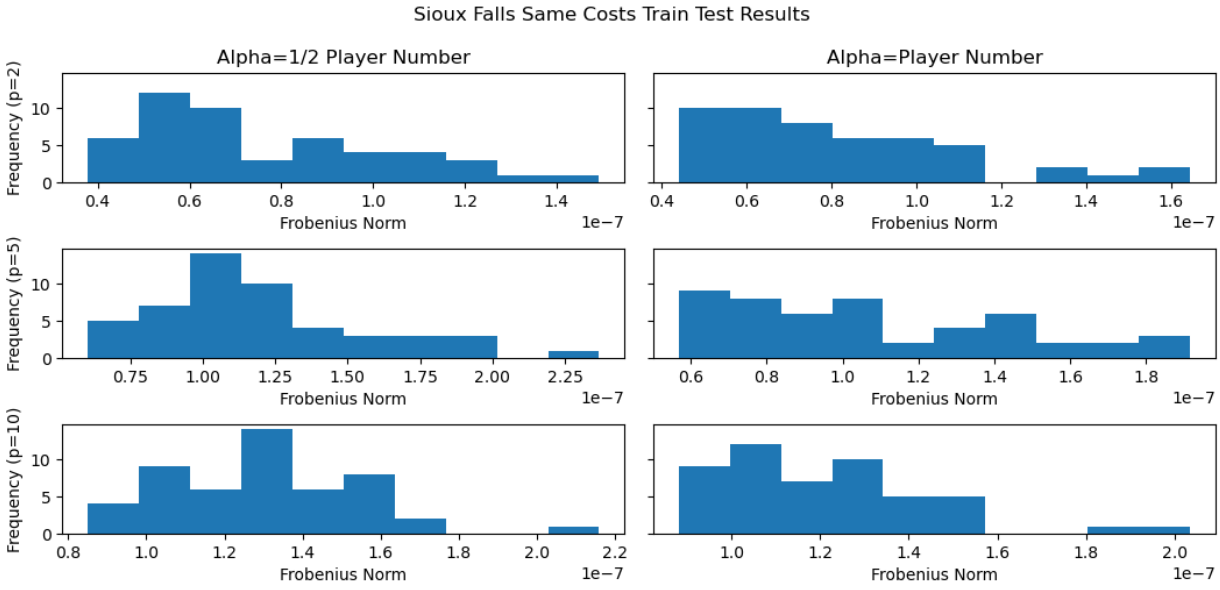


Figure 3.9: Sioux Falls Train Test Same Costs

3.5.6 Different Randomized Costs: Part B

As would be expected, for all of the flow errors in Figures 3.10-3.12 below, the flow error range end point increases as the number of players increases since there is more opportunity for mistaken flows. However, we are for the most part satisfied with the results because different costs is a significantly more difficult problem than same costs, and the errors for the most part are low, especially for the Sioux Falls graph. Interestingly, for the 2×2 Grid and 4×4 Grid results below in Figures 3.10-3.11, the α capacity parameter seems to have some effect on the recovery of the flow patterns by our parameterizations because the $\alpha = \frac{1}{2}N$ produces lower flow error ranges for the Grids. For the Sioux Falls graph, the results are not as clear; the α does not appear to have a clear effect one way or another.

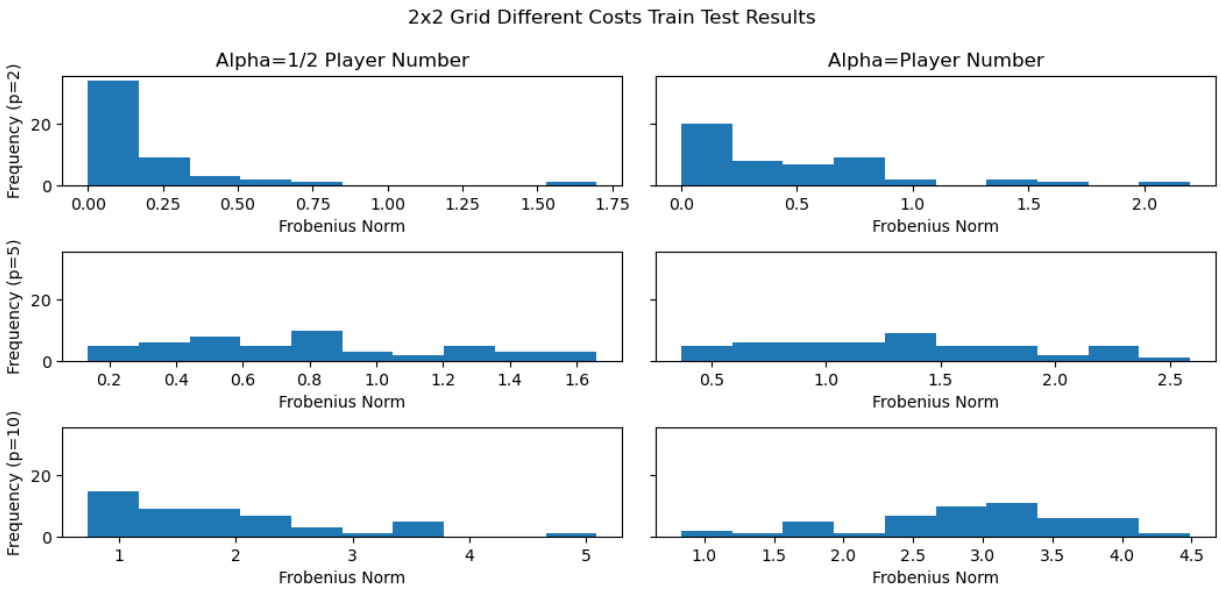


Figure 3.10: 2×2 Grid Train Test Different Costs

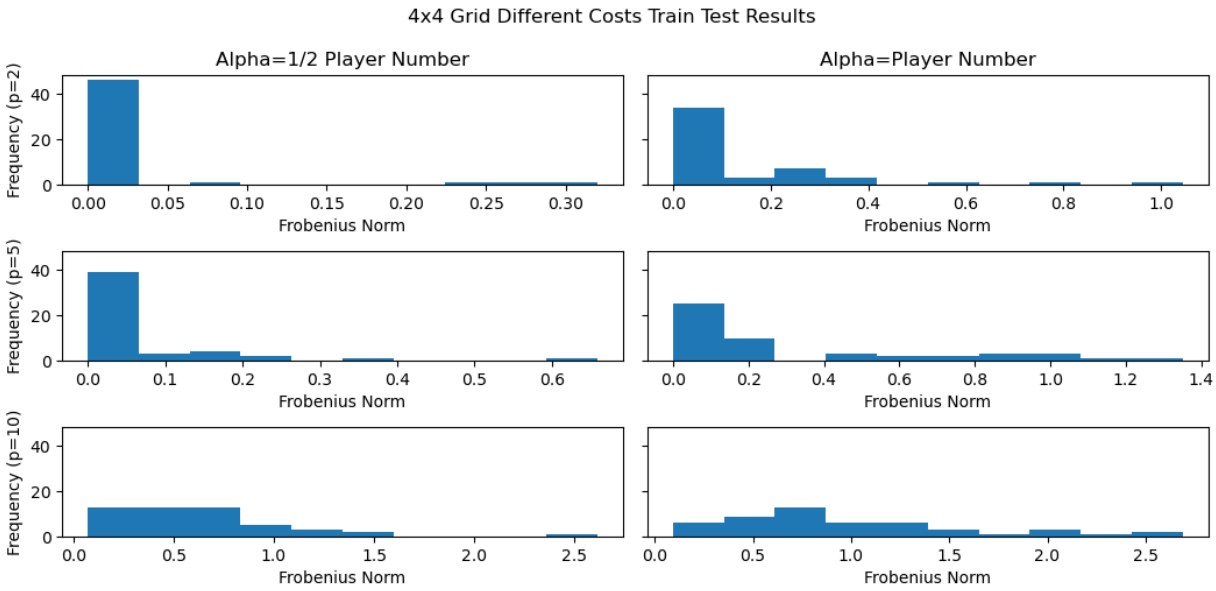


Figure 3.11: 4×4 Grid Train Test Different Costs

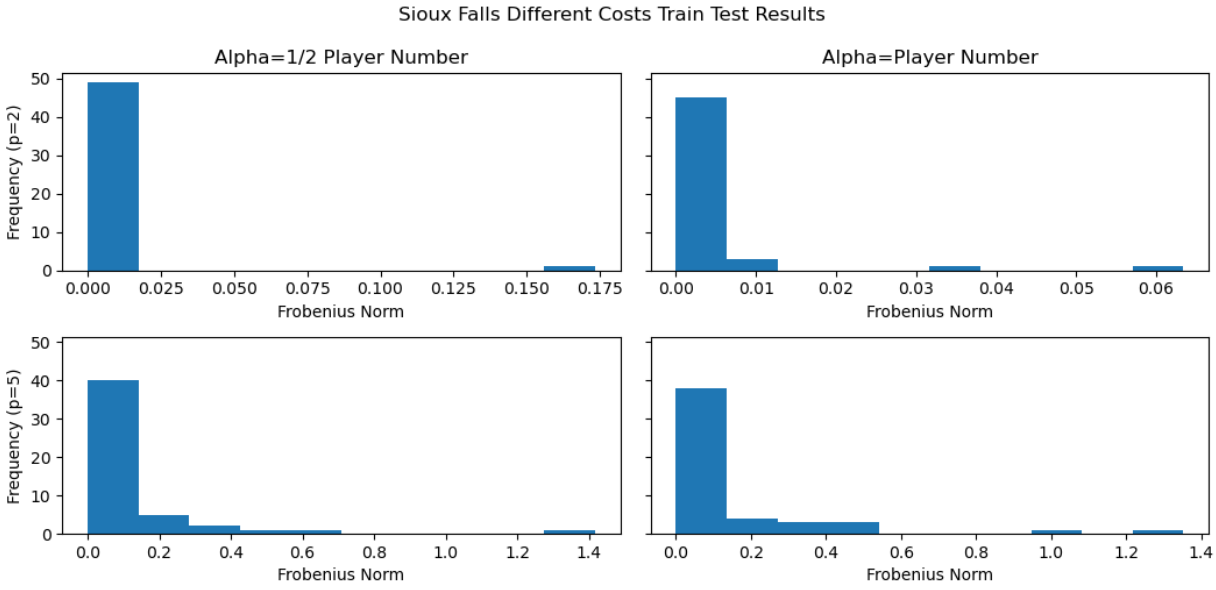


Figure 3.12: Sioux Falls Train Test Different Costs

3.6 Conclusions & Future Research

Ratliff et al. [154] applied the framework due to [112] to multi-player Nash games. We extended this framework to the case of jointly convex generalized Nash games by proving that we can use the VI KKT conditions in conjunction with a residual model to recover a parameterization for the objective functions of the players in these games. For our transportation game example, we have seen that, although our model may not recover the original parameterization, it recovers a parameterization that produces the same flow patterns as the original parameterization. This holds true across multiple grid sizes, the Sioux Falls network, different assumptions regarding players' perceived costs, and the majority of restrictive α capacity settings and the associated numbers of players. When we apply a train test procedure from [84] to our experiments, the results overall are encouraging, showing that, for same costs, our framework produces overall low flow error results and, for the difficult problem of different costs, our framework is promising.

Further research could extend our model to the setting where the multipliers for the shared constraint are not assumed to be equal, especially in the case when the cost functions differ between the players [75]. For more on this, we suggest to readers the work of [138]. In addition, we could extend the research to a real-world traffic situation with real data, in which we would likely need to incorporate more players into our framework and different origin-destination traffic patterns, including ones in which not all the players leave from the same origin node and head to the same destination node. Furthermore, we could attempt to address some of the issues with the Nash approach, including looking at a low regret condition on the players as in [140, 193, 194]. We could also explore making our model more robust to the issues indicated by [151] such as rationality assumptions and incorrect models. Finally, we could explore the scalability

of the method further. As noted in the experimentation section, some of the larger examples did not finish running, so we would want to explore other ways of making the method more computationally efficient.

3.7 Chapter Acknowledgements

Allen was partially funded by a Graduate Fellowship in STEM Diversity while completing this research (formerly known as a National Physical Science Consortium Fellowship). Allen was also supported by a Flagship and Dean's Fellowships from the University of Maryland-College Park and has worked for Johns Hopkins University Applied Physics Lab in the summers. Gabriel was supported by the following grant: Research Council of Norway, "Trans-Atlantic Cooperation on Energy Market Modeling," 2018-2022. Dickerson was supported in part by NSF CAREER Award IIS-1846237, NSF D-ISN Award #2039862, NIST MSE Award #20126334, and DARPA GARD #HR00112020007.

We would like to thank Michael Curry of University of Maryland, College Park for helpful comments earlier on in this project. We would like to thank Luke Evans of University of Maryland, College Park for reading an early draft of this chapter.

Chapter 4: Inverse Optimization for Parameterization of Linear Complementarity Problems and for Incentive Design in Markets

Note: We note that a shorter version of this article will appear on arXiv soon. We thank our co-authors on this chapter: Steven Gabriel and Nathan Boyd. Stephanie Allen did the vast majority of the work on this project, including the new theory and proofs, the majority of the coding for the case study, and the writing of the paper. Steven Gabriel provided some suggestions for additional avenues to explore for the theory as well as provided much editorial feedback. Nathan Boyd helped Stephanie Allen with understanding the application behind the case study as well as interpreting results obtains from solving the linear complementarity problems involved. Nathan Boyd also provided editorial feedback on the case study section.

4.1 Introduction

Mathematical programming allows us to propose a model of the world and, then, find the decisions that minimize cost, maximize benefit, or produce an equilibrium. Market models, which are built using mathematical programming, allow us to explore the incentives at work that cause players to make decisions that optimize their objectives in reaction to the objectives of other players. These models along with other equilibrium and optimization problems can be encapsulated by mixed complementarity problems [75]. However, this thesis chapter focuses

on linear complementarity problems, a subset of the mixed complementarity problems that still covers many optimization and equilibrium problems.

This linear complementarity problem can be termed the "forward problem," [37] where we obtain the decisions made by a singular entity, multiple players, or a collective group of players. In one paradigm of inverse optimization, we take *observed* decisions made in the world by the singular decision-maker, multiple players, or collective group of players and use them to parameterize the forward problem. Used in this way, inverse optimization helps us form models when the decision solutions are observable, but not all of the parameters of the model are not [3, 21]. Traffic modeling is a perfect example because we see the flow of traffic every day, but we cannot see the decision function in people's minds as they make driving choices [3, 21]. Inverse optimization has also sought to *influence* the decisions of users whose choices can be modeled by optimization and/or equilibrium problems. Ahuja and Orlin [3] in their seminal work on inverse optimization point to the idea of influencing traffic flow using inverse optimization by taking a system optimal solution and figuring out the correct tolls to levy on drivers to push them from their user solutions to the system optimal. Ahuja and Orlin cite work by [29, 30, 53] as examples of papers in this area. As another inverse optimization paper in this vein, Bertsimas et al. [21] mention in their introduction the idea of "design applications" in which one wishes to adjust a system toward a specific equilibrium. Overall, we see the dual purpose of inverse optimization in: (a) uncovering how systems work and (b) designing "better" systems [3, 20].

This chapter focuses on parameterizing linear complementarity problems using sets of solutions that either are observed or are desirable. We define conditions on these sets of solutions that make the inverse optimization problem feasible or infeasible and that produce certain parameterizations that possess advantageous properties. We also address when these conditions fail. What's more,

we consider the interaction of the given problem data with the parameterization goal. As the literature review reveals, linear complementarity problems have rarely been considered in the abstract in the inverse optimization literature, so this work breaks new ground by considering many of the aspects of their parameterization.

To ground this research in reality, we introduce a new application to the inverse optimization literature in the form of a water supply application. We build off Boyd et al. [25] by taking a simplified water supply market example from their paper and using inverse optimization to both: (a) parameterize the problem using observed solution(s) and (b) choose the parameters of market model such that a system optimal is an equilibrium point of the market.

4.2 Literature Review

The majority of the inverse optimization (IO) literature has concentrated on parameterizing objective functions in optimization problems. Interested readers can see the following significant foundational IO works to read more on this subject: [3, 8, 37, 88, 112, 132, 164]. These works have lead to many applications of inverse optimization in areas such as healthcare [35], energy [115], and transportation [207]. Recently, there has been some work done in estimating other parameters in optimization problems, specifically in linear programs. Černỳ et al. [33] estimate the cost vector in $c^T\mathbf{x}$ and the right-hand side in $A\mathbf{x} = b$ when an interval is given for each of the parameters using a "union of polytopes corresponding to optimal bases" and then using parametric programming to move around the solution space. Saez-Gallego and Morales [161] estimate the cost vector in $c^T\mathbf{x}$ and right-hand side for constraints $A\mathbf{x} \leq b$ in a linear program using a set of sequential linear programs along with cross validation, with the application of

demand response. Chan and Kaw [36] parameterize A in constraints $A\mathbf{x} \geq b$ and c in objective function $c^T\mathbf{x}$; they use projection/optimality condition techniques as well as robust optimization in their approach. Ghobadi & Mahmoudzadeh [82] parameterize the A and b in constraints $A\mathbf{x} \geq b$ using both single point and multi-point data and using multiple types of loss functions, with an application to the diet recommendation problem. Tan et al. [173] estimate A, b in constraints $A\mathbf{x} \geq b$ and c in objective function $c^T\mathbf{x}$ using backpropagation and unrolling, and they also estimate "parametric" optimization problems. Finally, Tan et al. [174] estimate A, b in constraints $A\mathbf{x} \geq b$, G, h in constraints $G\mathbf{x} = h$, and c in objective function $c^T\mathbf{x}$ using backpropagation and implicit differentiation. Thus, there has been a recent literature in estimating parameters besides the ones in the objective function.

However, all of the papers in the previous paragraph focus on optimization problems, and none of them focus on game theory models involving multiple players. Furthermore, they do not propose adjusting the incentives of the entities involved to improve societal outcomes. There has been some work in inverse optimization for game theory models and even a bit of work done in adjusting incentives in these models. Bertsimas et al. [20] use inverse optimization to generate new estimation models for finding covariance, mean return, and level of risk based on observed investment decisions in markets; the researchers present these models as "a novel, richer, reformulation of the Black-Litterman framework" that overall help investors in environments where there is imperfect information. Chow et al. [43] adjust the capacity coefficients in a transportation/freight equilibrium problem formulation by building on the method of [3] to produce a convex nonlinear program. Ratliff et al. [154] use inverse optimization techniques from [112] in an energy game. Chen et al. [39] use the inverse optimization method of [21] to find the cost functions for electricity suppliers that are operating in a market. Birge et al. [24] employ

inverse optimization to "recover parameters of transmission and related constraints that are not revealed to market participants but explain the price variation" in an electricity market. Awasthi [10] finds the costs for a dynamic game involving control theory for multiple players. Kovács [116] "introduces an inverse optimization approach to eliciting the parameters of electricity consumer models formulated as linear programs from the historic samples." Hong et al. [99] use a biquadratic program to represent the interaction among non-atomic players in a game in which "disutility is determined only by the number of agents choosing each alternative." They find the different sets of parameters representing the different players' preferences. Huang [103] explores the computational complexity of the inverse linear complementarity problem and some different forms of solving this problem.

With regard to the papers that deal with changing incentives in the market models, the applications vary from line shipping to electricity. Zhou et al. [209] explicitly use a bilevel inverse optimization problem strategy to find the right "incentives" to push a system operator to use more renewable energy by proposing the solution they would like to see and then using the inverse optimization bilevel model to find the right coefficients. The lower-level is the system level operator that is trying to minimize costs, while the upper-level player is the policy maker trying to choose the right set of "incentives" to get the system operator to choose a certain level of renewable energy. The researchers experiment with different kinds of incentives including subsidies and taxes, and then they unite these incentives with "mandatory policies" to provide a mix. Without these interventions, on their small case study, they show that the system operator chooses a very low amount of renewable energy. Agarwal and Ergun [2] use inverse optimization to try to "drive each individual [line shipping] carrier's linear program toward the collaborative optimal solution", which is the solution in which there is complete

coordination between all of the players involved and in which they all are working to maximize total profit (also known as the "centralized problem"). They create a linear program representing each carrier's shipping decisions, and then they use IO techniques of relating primal and dual relationships to find a cost vector that causes each player to do their part in the "collaborative optimal solution." Once they find this cost, this tells the researchers the side payments that must be used as a "mechanism" to keep all of the carriers in line toward this collaborative optimal solution. Thus, this chapter is very much in line with the theme of our water market case study. Tan and Aviso [172] adjust the cost coefficients in the objective functions of a Stackelberg game according to the method of [3] to produce a socially optimal solution in the area of environmental regulation. Nguyen et al. [141] allow prosumers (agents that consume and produce energy) to specify a price range and a range on the amount of energy they would like to buy or sell, and then the researchers design algorithms to generate the cost function parameters in the market such that these ranges can be respected. Some of these algorithms even preserve the privacy of the prosumer information. Therefore, this is a kind of market adjusting analysis because the individuals in the market give feedback regarding what they would like to see in the market, and then parameters are chosen taking this into consideration (as long as the intervals satisfy certain conditions like being "overlapping"). Aswani et al. [9] parameterize a "principal-agent model" representing the partnerships encouraged by the Medicare Shared Savings Program (MSSP) by using inverse optimization to find the appropriate savings payment and subsidy rate for new medical investments that the program should provide to these partnerships. They essentially designed the subsidy rate mechanism as a new part of the MSSP and, then, use inverse optimization to find a rate based on noisy data. Overall, these papers demonstrate that system optimal solution(s) or desirable solution(s) can be utilized by inverse optimization techniques to

design better systems.

4.2.1 Contribution

We address both the question of estimating other parameters besides the parameters in the objective function as well as the question of parameterizing other problems besides optimization problems. In particular, we examine the linear complementarity problem (LCP), which generalizes optimization problems via their Karush-Kuhn-Tucker (KKT) conditions, encompasses game theory models, and represents a host of other problems in engineering and economics [75]. Thus, this LCP problem class is versatile and worthy of further study, and it is not a class that has been explored very much in the inverse optimization literature before. The only paper that explores inverse optimization of the LCP problem explicitly is Huang [103]. Huang [103] investigates the computational complexity of inverse linear complementarity problems and propose the same inverse optimization quadratic program that we propose, but they do not address simplification measures for the LCP inverse optimization problem, the possibility of being given multiple solutions, or the complementarity cone and complete information concepts that we address in this chapter. We address both theoretical and practical considerations when parameterizing an LCP from either observed or desirable solutions. We define conditions on these sets of solutions that make the inverse optimization problem feasible or infeasible and that produce parameterizations that have noteworthy properties. We also discuss when these conditions fail to be satisfied. Furthermore, we analyze the interaction of the given problem data with the parameterization goal, examining single-point, multi-point, and infinite-point data cases. These cases correspond to situations in which we have a unique solution, a finite set of solutions, or a convex set of

infinite solutions to our LCP. To apply our new found knowledge to an application, we address the idea of using "system optimal" solutions to adjust parameters in a water supply market case study, which is the first time inverse optimization has been applied to a water supply market. In this case study, we draw on the transportation literature's discuss regarding system optimal versus user equilibrium solutions [165] along with previous water supply work [28]. We demonstrate that we can use inverse optimization to adjust incentives in a market to achieve the system optimal solution that increases total additive market benefit.

4.3 Background

In order to discuss inverse optimization for LCPs, we first provide some background regarding LCPs and the theory of their solutions. With this background, we then discuss the inverse optimization problem at a high level at the end of this section.

To document a few notation conventions, we use subscripts to identify elements in matrices and vectors. For matrices $A_{i,:}$ denotes the ith row of A, $A_{:,i}$ denotes the ith column of A, and $A_{i,j}$ denotes the (i,j)th element of A. We use superscripts on vectors to identify specific vectors. Thus, $\mathbf{z}_i^k$ denotes the ith element of the kth vector.

4.3.1 Basic Linear Complementarity Problem

A linear complementarity problem with only nonnegative variables can be defined as finding a vector $\mathbf{z} \in \mathbb{R}^n_{\geq 0}$, given a matrix of coefficients $M \in \mathbb{R}^{n \times n}$ and a vector of coefficients $q \in \mathbb{R}^n$, such that the following relationship holds [48]:

$$0 \leq \mathbf{z} \perp M\mathbf{z} + q \geq 0 \tag{4.3.1}$$

The $\perp$ indicates that the inner product $\mathbf{z}^T(M\mathbf{z}+q) = 0$, and this entire problem can be written in short-hand as LCP(q, M) [48]. The set $\mathcal{I} = \{1, ..., n\}$ is an index set for the linear complementary problem, often used to refer to rows of the matrix M, elements of the vector $\mathbf{z}$, or elements of the vector q. The set of solutions to this problem can be written as $SOL(q, M)$ [48]. The LCP(q, M) (4.3.1) is sometimes termed the "forward problem" [37] because it produces the $\mathbf{z}$ solutions. There are a few more terms related to the LCP that should be defined.

Definition 5 ([48]). *A vector* $\mathbf{z}$ *that satisfies* $0 \leq \mathbf{z}$ *and* $M\mathbf{z} + q \geq 0$ *is said to be feasible. A LCP*(q, M) *is feasible if a feasible vector exists.*

Definition 6 ([48]). *A vector* $\mathbf{z}$ *such that* $\mathbf{z}^T(M\mathbf{z} + q) = 0$ *is called complementary. A vector is a solution of the LCP*(q, M) *if it is both feasible and complementary. A LCP*(q, M) *is said to be solvable if it has a solution.*

Definition 7 ([134]). *A solution of the LCP*(q, M) *is degenerate if a component* $\mathbf{z}_i = 0$ *and the associated* $M_{i,:}\mathbf{z} + q_i = 0$.

Definition 8. *A "*$\mathbf{z}$ *solution" or an "LCP* $\mathbf{z}$ *solution" refers to a solution to the LCP*(q, M).

Having defined some terms we use throughout the chapter in reference to LCPs, we state that the goal of our inverse optimization problem is to find a q given a set of $\mathbf{z}$ solutions. We define this formally in Section 4.3.3.

A natural question might surround the multiplicity of $\mathbf{z}$ solutions to this problem. This is dependent on the properties of the matrix M and the vector q. Indeed, a linear complementarity

problem can have a unique, a finite, an infinite, or unknown number of solutions for a given M and q [48]. Thus, in the following subsections, we examine the LCP from both the perspective of the M matrix and the q vector.

4.3.1.1 Multiplicity of $\mathbf{z}$ Solutions based on the M Matrix

First, we provide some background regarding when the M matrix induces a unique, a finite, an infinite number, or an unknown number of $\mathbf{z}$ solutions based on the properties of the M matrix.

Case 1 $M \in \mathbf{P}$: To have a unique solution for every right hand-side q, the LCP M matrix needs to be a $\mathbf{P}$-matrix [48]. This is a complete characterization. Before we state the definition of a $\mathbf{P}$-matrix, we need to define the concepts of principle submatrices and principle minors.

Definition 9 (Definition 2.2.1 [48]). *Let $A \in \mathbb{R}^{m \times n}$ be given. For index sets $\alpha \subseteq \{1, ..., m\}$ and $\beta \subseteq \{1, .., n\}$, the submatrix $A_{\alpha,\beta}$ of A is the matrix whose entries lie in the rows of A indexed by α and the columns index by β. If $\alpha = \{1, ..., m\}$, we denote the submatrix $A_{\alpha,\beta}$ by $A_{:,\beta}$; similarly, if $\beta = \{1, ..., n\}$, we denote $A_{\alpha,\beta}$ by $A_{\alpha,:}$. If $m = n$ and $\alpha = \beta$, the submatrix $A_{\alpha,\alpha}$ is called a principal submatrix of A; the determinant of $A_{\alpha,\alpha}$ is called a principal minor of A.*

Definition 10 (Definition 3.3.1 [48]). *A matrix $M \in \mathbb{R}^{n \times n}$ is said to be a $\mathbf{P}$-matrix if all its principal minors are positive. The class of such matrices is denoted $\mathbf{P}$.*

The corresponding theorem is the following.

Theorem 5 (Theorem 3.3.7 [48]). *A matrix $M \in \mathbb{R}^{n \times n}$ is a $\mathbf{P}$-matrix if and only if the LCP(q, M) has a unique solution for all vectors $q \in \mathbb{R}^n$.*

Therefore, we know that, if we have a $\mathbf{P}$-matrix, then all $q \in \mathbb{R}^n$ each map to one $\mathbf{z}$ vector. It is possible that multiple q could map to the same $\mathbf{z}$, which means that, when finding a q for $\mathbf{z}$, there

may be multiple potential q that could lead to that $\mathbf{z}$. An example of this comes from the M and q defined below:

$$M = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \; q = \begin{bmatrix} -8 \\ q_2 \geq -4 \end{bmatrix} \tag{4.3.2}$$

This M is a $\mathbf{P}$-matrix, and all of the q defined in (4.3.2) map to the $\mathbf{z}$ solution $\mathbf{z} = [4, \; 0]^T$. This fits with the overall setting of an inverse problem in that inverse problems often produce multiple solutions, with solutions here refering to q solutions. We continue this discussion in Section 4.4 when discussing an algorithm for finding q in this case. We also note for this problem that, if $-16 < q_2 < -4$, then $\mathbf{z}_1 = \frac{q_2+16}{3}$, $\mathbf{z}_2 = \frac{-8-2q_2}{3}$ and, if $q_2 \leq -16$, $\mathbf{z}_1 = 0$, $\mathbf{z}_2 = \frac{-q_2}{2}$. Therefore, the $\mathbf{z}$ solution changes as q_2 changes, but we still obtain a unique solution for this matrix M.

Case 2 Nondegenerate M: For this case, the class of matrices includes the $\mathbf{P}$-matrices, since a finite number does indeed include the case of 1 solution. For this broader class of matrices, we assume M is nondegenerate, defined as follows:

Definition 11 (Definition 3.6.1 [48]). *A matrix $M \in \mathbb{R}^{n \times n}$ is called nondegenerate if all its principal minors are nonzero.*

We then have the following theorem:

Theorem 6 (Theorem 3.6.3 [48]). *Let $M \in \mathbb{R}^{n \times n}$. The following statements are equivalent*

1. *M is nondegenerate.*
2. *For all vectors q, the LCP(q, M) has a finite number (possibly zero) of solutions.*
3. *For all vectors q, any solution of the LCP(q, M), if it exists, must be locally unique.*

Thus, from this definition and theorem, we see that nondegenerate matrices have a finite number of $\mathbf{z}$ solutions.

Case 3 Positive Semi-Definite M: We consider the case where there are potentially an infinite number of solutions forming a convex set. This leads us to deal with the case of positive semi-definite matrices, which can be defined as follows:

Definition 12 (Definition 2.2.15 [48]). *A matrix $A \in \mathbb{R}^{n\times n}$ is said to be positive semi-definite if $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n$.*

We then have the following theorem:

Theorem 7 (Theorem 3.1.7 [48]). *Let $M \in \mathbb{R}^{n\times n}$ be positive semi-definite, and let $q \in \mathbb{R}^n$ be arbitrary. The following hold:*

1. *If $\mathbf{z}^1$ and $\mathbf{z}^2$ are two solutions of LCP(q, M), then:*

$$(\mathbf{z}^1)^T(q + M\mathbf{z}^2) = (\mathbf{z}^2)^T(q + M\mathbf{z}^1) = 0 \tag{4.3.3}$$

2. *If $\mathbf{z}^* \in$ SOL(q, M) has the property that (i) $\mathbf{z}^*$ is nondegenerate and (ii) $M_{\alpha\alpha}$ is nonsingular where*

$$\alpha = \{i : \mathbf{z}_i^* > 0\} \tag{4.3.4}$$

   then $\mathbf{z}^*$ is the unique solution of LCP(q, M).

3. *If LCP(q, M) has a solution, then SOL(q, M) is polyhedral and equal to*

$$P = \{\mathbf{z} \in \mathbb{R}^n_{\geq 0} : q + M\mathbf{z} \geq 0,\ q^T(\mathbf{z} - \bar{\mathbf{z}}) = 0,\ (M + M^T)(\mathbf{z} - \bar{\mathbf{z}}) = 0\} \tag{4.3.5}$$

where $\bar{\mathbf{z}}$ is an arbitrary solution.

4. *If M is symmetric (as well as positive semi-definite), then $M\mathbf{z}^1 = M\mathbf{z}^2$ for any two solutions $\mathbf{z}^1$ and $\mathbf{z}^2$.*

Therefore, positive semi-definite M matrices can produce convex sets of solutions as demonstrated in part 3 of Theorem 7 with the polyhedral set. One example of a positive semi-definite but not a $\mathbf{P}$-matrix M with related q that produces a convex set of solutions is the following problem:

$$M = \begin{bmatrix} 1 & 2 \\ \\ 2 & 4 \end{bmatrix}, \quad q = \begin{bmatrix} -4 \\ \\ -8 \end{bmatrix} \tag{4.3.6}$$

The two extreme points of the convex solution set, meaning the points that cannot be constructed from convex combinations of other points, are $(4, 0)$ and $(0, 2)$. If we take a convex combination of these two solutions, we also obtain a solution to the above problem, as demonstrated by the following for $\theta \in [0, 1]$:

$$M\mathbf{z} + q = \begin{bmatrix} 4\theta + (2)(2(1-\theta)) - 4 \\ \\ (2)(4\theta) + (4)(2)(1-\theta) - 8 \end{bmatrix} = \begin{bmatrix} 4\theta + 4 - 4\theta - 4 \\ \\ 8\theta + 8 - 8\theta - 8 \end{bmatrix} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix}. \tag{4.3.7}$$

Thus, this problem has an infinite number of solutions that forms a convex set.

Case 4 Unknown Structure for M: Finally, the matrix M may satisfy none of the properties addressed in the previous three cases. Then, depending on the problem size, it may or may not be feasible to obtain all of the solutions. With a limited problem size, we can check the 2^n cases of potential $\mathbf{z}$ solutions being 0 or strictly positive for each variable. For larger problems, we may have to

engage in searching techniques where we warm start an LCP solver to find multiple potential $\mathbf{z}$ solutions [75], meaning that we initialize the LCP model with a starting point from which the Lemke [48], PATH [54, 70], or other LCP algorithms can work to find a solution.

4.3.1.2 Multiplicity via the q Vector

Having examined multiplicity via the M matrix, we now discuss multiplicity through the q vector. The overarching idea is that, given a fixed M matrix, different q in the space of possible q vectors induce different sets of $\mathbf{z}$ solutions in the LCP(q, M). This is important for our inverse optimization problem formulation in the next section. Before we can approach this, we need to define a few additional terms.

Definition 13 (Cone and Convex Cone: Defintion 1.3.1 [48])**.** *We say that a nonempty set X in $\mathbb{R}^n$ is a cone if, for any $x \in X$ and any $t \geq 0$, we have $tx \in X$. If a cone X is, in addition, a convex set, then we say that X is a convex cone.*

We can then define an important cone that is be relevant to our M matrix.

Definition 14 (Finite Cone [48])**.** *A matrix $A \in \mathbb{R}^{m \times p}$ generates a convex cone obtained by taking nonnegative linear combinations of the columns of A. This cone, denoted pos A, is given by*

$$\textit{pos } A = \{w \in \mathbb{R}^m : w = Av \textit{ for some } v \in \mathbb{R}^p_{\geq 0}\} \tag{4.3.8}$$

This definition is related to the definition of a complementary matrix of M and allows us to express the set of q for a given matrix M that lead to a solution of LCP(q, M).

Definition 15 (Definition 1.3.2 [48]). *Given* $M \in \mathbb{R}^{n\times n}$ *and* $\alpha \subseteq \{1, ..., n\}$, *we define* $C_M(\alpha) \in \mathbb{R}^{n\times n}$ *as*

$$C_M(\alpha)_{:,i} = \begin{cases} -M_{:,i} & \text{if } i \in \alpha \\ I_{:,i} & \text{if } i \notin \alpha \end{cases} \tag{4.3.9}$$

$C_M(\alpha)$ *is then called a complementary matrix of* **M**. *The associated cone, pos* $C_M(\alpha)$, *is called a complementary cone (relative to* **M**).

As discussed in Cottle et al. [48], we can form the union of all of the complementary cones associated with M to produce the following set in which the set $\mathcal{A}$ contains all 2^n possible index sets for the complementary cones involving matrix M:

$$K(M) = \bigcup_{\alpha\in\mathcal{A}} \{\hat{q} : C_M(\alpha)v = \hat{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} = \{q : SOL(q, M) \neq \emptyset\} \tag{4.3.10}$$

This provides us with the insight that the complementary cones can identify the set of q that produce solutions $\mathbf{z}$ to the LCP(q, M) [48]. The interactions of the cones with each other also identify when a q produces, for a fixed M, different numbers of LCP $\mathbf{z}$ solutions [48]. However, finding these interactions becomes exponentially difficult in the worst case because there are 2^n number of complementary cones for a matrix M. Nevertheless, M and q interact to produce the set of $\mathbf{z}$ solutions for the LCP they define. These insights are important as we move in reverse, having a set of $\mathbf{z}$ solutions and attempting to produce a q associated with this set of $\mathbf{z}$ solutions. We expand upon this interaction in Section 4.4. First, however, we define a set of examples that we utilize throughout the rest of the chapter to illustrate various points.

4.3.2 Examples for the Different Cases

We provide examples of the four $\mathbf{z}$ solution cases for the different M matrix types before continuing to the next section. We note for these examples that the q vector and $\mathbf{z}$ solution(s) pairs do not necessarily imply uniqueness of the q vector, meaning that multiple potential q vectors could have potentially produced the $\mathbf{z}$ solution(s).

- *Case 1 $M \in \mathbf{P}$:*

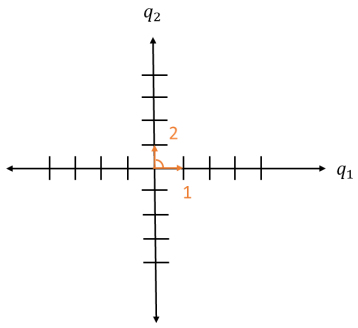
  Example 1.1: For a non-symmetric example of a $M \in \mathbf{P}$, we have:

$$M = \begin{bmatrix} 2 & 2 \\ 1 & 4 \end{bmatrix} \tag{4.3.11}$$

  The clearest demonstration that any potential q only associates with one solution $\mathbf{z}$ comes from the diagrams below which show each of the complementary cones for this example, with the last diagram showing all of the complementary cones together. The diagrams also showcase that the complementary cones cover the whole $\mathbb{R}^2$ space and, thus, all potential $q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$. The vectors in the diagrams are numbered such that:

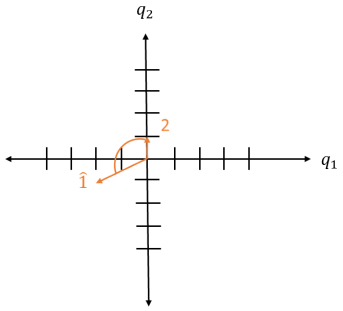
$$1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ 2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ \hat{1} = -M_{:,1},\ \hat{2} = -M_{:,2} \tag{4.3.12}$$

$$\alpha = \emptyset \tag{4.3.13a}$$



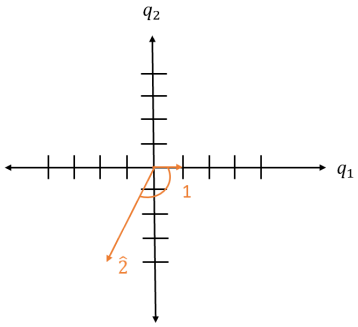
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.13b}$$

$$\alpha = \{1\} \tag{4.3.14a}$$



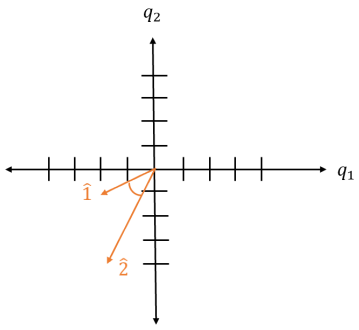
$$C_M(\alpha) = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix} \tag{4.3.14b}$$

$$\alpha = \{2\} \tag{4.3.15a}$$



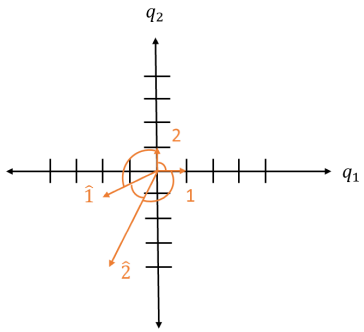
$$C_M(\alpha) = \begin{bmatrix} 1 & -2 \\ 0 & -4 \end{bmatrix} \tag{4.3.15b}$$

$$\alpha = \{1, 2\} \tag{4.3.16a}$$



$$C_M(\alpha) = \begin{bmatrix} -2 & -2 \\ -1 & -4 \end{bmatrix} \tag{4.3.16b}$$

All the complementary cones.



From these complementary cone diagrams, we can see that each $q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$ corresponds with one complementary cone and thus with one solution. To demonstrate this numerically, we have Table 4.1 which shows sample q from each divided region created by the cones and the associated singular solution to the LCP for each q. We verified by hand each $\mathbf{z}$ solution set.

q Vector	$\mathbf{z}_1, \mathbf{z}_2$ Solutions	Associated Conic Regions
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
$\begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = \frac{1}{4}, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$
$\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = \frac{1}{6}, \mathbf{z}_2 = \frac{1}{12}$	$pos\ C_M(\{1,2\})$
$\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = \frac{1}{8}$	$pos\ C_M(\{2\})$

Table 4.1: q Vectors and Associated $\mathbf{z}$ Solutions for Example 1.1

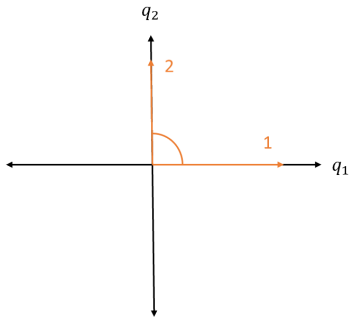
- *Case 2 Nondegenerate M:*

  Example 2.1: For the case of a finite number of solutions, we utilize two examples. Both are nondegenerate matrices. The first example M is:

$$\begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4.3.17}$$

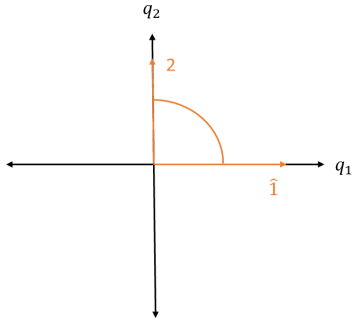
  The associated complementary cone diagrams are as follows:

$$\alpha = \emptyset \tag{4.3.18a}$$



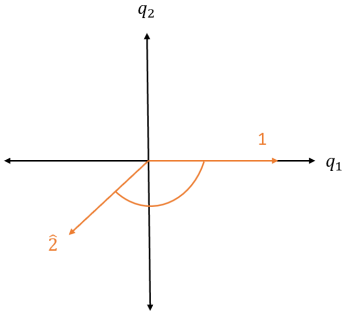
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.18b}$$

$$\alpha = \{1\} \tag{4.3.19a}$$



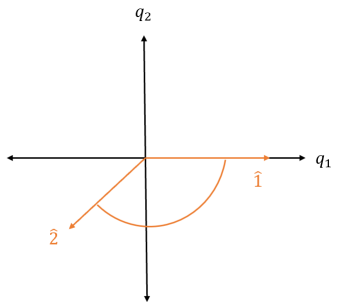
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.19b}$$

$$\alpha = \{2\} \tag{4.3.20a}$$



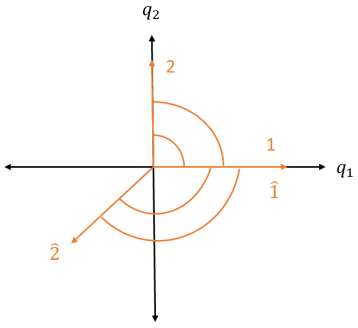
$$C_M(\alpha) = \begin{bmatrix} 1 & -1 \\ 0 & -1 \end{bmatrix} \tag{4.3.20b}$$

$$\alpha = \{1, 2\} \tag{4.3.21a}$$



$$C_M(\alpha) = \begin{bmatrix} 1 & -1 \\ 0 & -1 \end{bmatrix} \tag{4.3.21b}$$

All the complementary cones.



Unlike the previous example, the complementary cones do overlap and do not cover all of $\mathbb{R}^2$. This means not every $q \in \mathbb{R}^2$ produces solutions to LCP(q, M) for this M and, for this particular example, the $q \in \mathbb{R}^2$ that do produce solutions produce a finite number greater than 1.[1] We numerically demonstrate the multiple solutions in Table 4.2 by choosing a q from each of the two regions.

[1] We see in the next example that, for nondegenerate matrices, you could have some regions that only produce unique **z** solutions.

q Vector	$\mathbf{z}_1, \mathbf{z}_2$ Solutions	Associated Conic Regions
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
	$\mathbf{z}_1 = 0.5, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$
$\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0.5$	$pos\ C_M(\{2\})$
	$\mathbf{z}_1 = 1, \mathbf{z}_2 = 0.5$	$pos\ C_M(\{1,2\})$

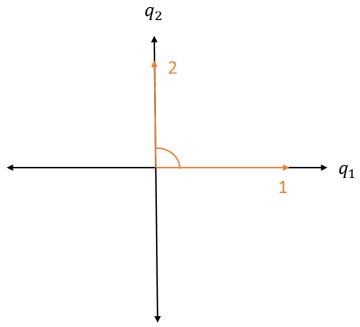
Table 4.2: q Vectors and Associated $\mathbf{z}$ Solutions for Example 2.1

Example 2.2: The second example is from Cottle et al. [48] page 22. This M matrix is as follows:

$$M = \begin{bmatrix} -0.25 & 0.5 \\ 0.5 & -0.25 \end{bmatrix} \tag{4.3.22}$$

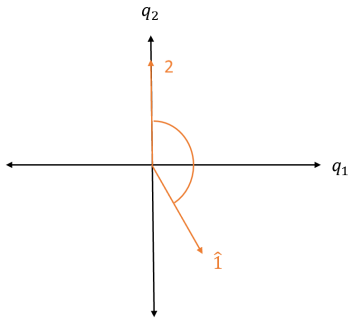
The complementary cone diagrams for this example are as follows:

$$\alpha = \emptyset \tag{4.3.23a}$$



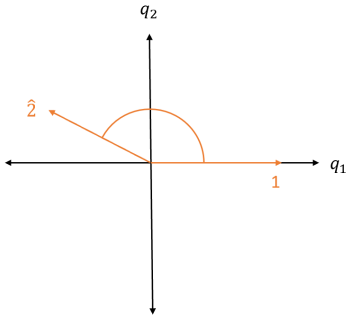
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.23b}$$

$$\alpha = \{1\} \tag{4.3.24a}$$



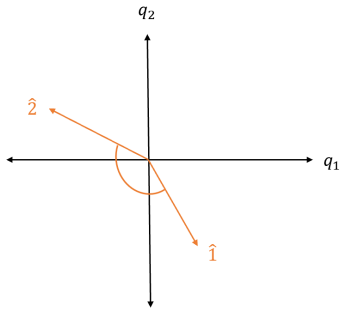
$$C_M(\alpha) = \begin{bmatrix} 0.25 & 0 \\ -0.5 & 1 \end{bmatrix} \tag{4.3.24b}$$

$$\alpha = \{2\} \tag{4.3.25a}$$



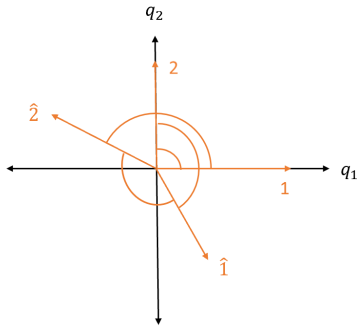
$$C_M(\alpha) = \begin{bmatrix} 1 & -0.5 \\ 0 & 0.25 \end{bmatrix} \tag{4.3.25b}$$

$$\alpha = \{1, 2\} \tag{4.3.26a}$$



$$C_M(\alpha) = \begin{bmatrix} 0.25 & -0.5 \\ -0.5 & 0.25 \end{bmatrix} \tag{4.3.26b}$$

All of the complementary cones.



We see that, unlike the previous example, the complementary cones coincidentally span the entirety of $\mathbb{R}^2$, and there is a mix of regions that have one solution and three solutions. Table 4.3 below verifies these facts numerically with four q values from the four different regions.

q Vector	$\mathbf{z}_1, \mathbf{z}_2$ Solutions	Associated Conic Regions
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
	$\mathbf{z}_1 = 2, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$
	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 2$	$pos\ C_M(\{2\})$
$\begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 2$	$pos\ C_M(\{2\})$
$\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 2, \mathbf{z}_2 = 2$	$pos\ C_M(\{1,2\})$
$\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 2, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$

Table 4.3: q Vectors and Associated $\mathbf{z}$ Solutions for Example 2.2

- *Case 3 Positive Semi-Definite M:*

Example 3.1: To find a positive semi-definite matrix, which is valuable because it produces a convex set of solutions, we turn to a linear programming problem:

$$\min_{\mathbf{x}_1,\mathbf{x}_2} 2\mathbf{x}_1 + 3\mathbf{x} \tag{4.3.27a}$$

$$4\mathbf{x}_1 + \mathbf{x}_2 \geq 4 \quad (\lambda_1) \tag{4.3.27b}$$

$$2\mathbf{x}_1 + 3\mathbf{x}_2 \geq 6 \quad (\lambda_2) \tag{4.3.27c}$$

$$\mathbf{x}_1 \geq 0,\ \mathbf{x}_2 \geq 0 \tag{4.3.27d}$$

with the following M matrix and q vector for the associated necessary and sufficient KKT conditions:

$$M = \begin{bmatrix} 0 & 0 & -4 & -2 \\ 0 & 0 & -1 & -3 \\ 4 & 1 & 0 & 0 \\ 2 & 3 & 0 & 0 \end{bmatrix}, \quad q = \begin{bmatrix} 2 \\ 3 \\ -4 \\ -6 \end{bmatrix} \tag{4.3.28}$$

We verify that this matrix is positive semi-definite by checking that $\frac{1}{2}(M + M^T)$ is positive semi-definite, as Cottle et al. [48] on page 66 states we can do. We also know this matrix is positive semi-definite because it is bisymmetric and the upper left hand block is all 0s.

Since this is a 4×4 matrix, we cannot draw the complementary cones, but we know from constructing the example ourselves that the $\mathbf{z} = [\mathbf{x}_1,\ \mathbf{x}_2,\ \lambda_1,\ \lambda_2]^T$ solution space is a line segment (and thus a convex set), with end points of:

$$\hat{\mathbf{z}}^1 = \begin{bmatrix} 0.6 \\ 1.6 \\ 0 \\ 1 \end{bmatrix}, \quad \hat{\mathbf{z}}^2 = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.3.29}$$

which can be seen from the following diagram in which the blue and orange lines represent the constraints at equality, the black dotted lines represent the intersection of all the constraint half-spaces, and the green lines represent different instances of the objective function (also known as the isocost curves):

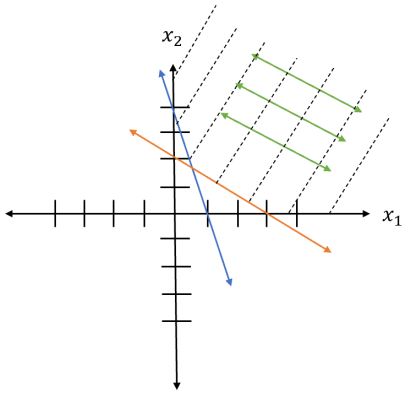


Figure 4.1: Linear Program Diagram for Example 3.1

Readers can see that the line segment between where the blue line intersects the orange line and where the orange line intersects the $\mathbf{x}_1$ axis is the set of solutions because the objective function isocost curves reach a minimum on that line segment, since the isocost curves are

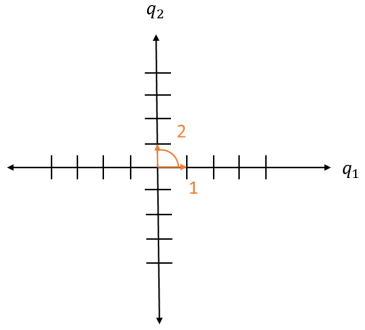
parallel to that segment. The two ends of the line segment match the $\hat{\mathbf{z}}^1$ and $\hat{\mathbf{z}}^2$ listed above.

Example 3.2: The second example is again positive semi-definite, and M is:

$$M = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \tag{4.3.30}$$

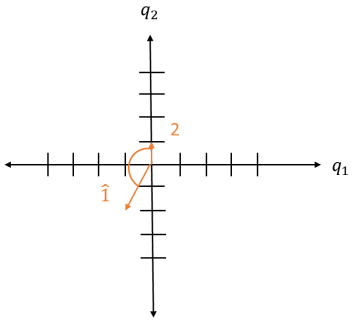
The associated complementary cones are:

$$\alpha = \emptyset \tag{4.3.31a}$$



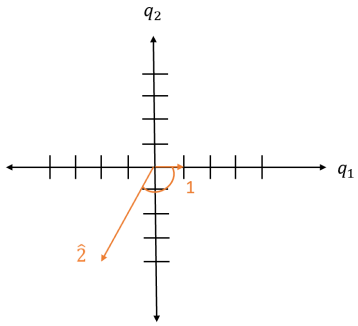
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.31b}$$

$$\alpha = \{1\} \tag{4.3.32a}$$



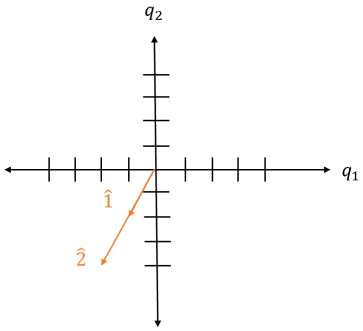
$$C_M(\alpha) = \begin{bmatrix} -1 & 0 \\ -2 & 1 \end{bmatrix} \tag{4.3.32b}$$

$$\alpha = \emptyset \qquad (4.3.33a)$$



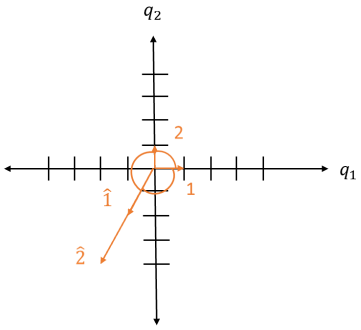
$$C_M(\alpha) = \begin{bmatrix} 1 & -2 \\ 0 & -4 \end{bmatrix} \qquad (4.3.33b)$$

$$\alpha = \{1, 2\} \qquad (4.3.34a)$$



$$C_M(\alpha) = \begin{bmatrix} -1 & -2 \\ -2 & -4 \end{bmatrix} \qquad (4.3.34b)$$

All the complementary cones.



We see that, for the cone $pos\ C_M(\{1, 2\})$, we have a ray, which denotes linearly dependent columns of M since the negative columns of M form vectors that overlap each other. This

gives rise to a convex set of solutions for the q in the cone created by the negative columns of M because, since M has linearly dependent columns, this means the system $-M\mathbf{z} = q$ has either 0 or an infinite number of solutions, as presented in the well-known linear algebra textbook by Strang [170]. We know the system has at least one solution because the q in that cone are a multiples of the columns of M, which means the system has an infinite number of solutions.[2] This is verified in Table 4.4 below.

q Vector	$\mathbf{z}_1, \mathbf{z}_2$ Solutions	Associated Conic Regions
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
$\begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0.5, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$
$\begin{bmatrix} -0.25 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0.25, \mathbf{z}_2 = 0$ $\mathbf{z}_1 = 0, \mathbf{z}_2 = \frac{1}{8}$ $\mathbf{z}_1, \mathbf{z}_2 \geq 0$ s.t. $\mathbf{z}_1 + 2\mathbf{z}_2 = 0.25$	$pos\ C_M(\{1, 2\})$
$\begin{bmatrix} -0.5 \\ -1 \end{bmatrix}$	$\mathbf{z}_1 = 0.5, \mathbf{z}_2 = 0$ $\mathbf{z}_1 = 0, \mathbf{z}_2 = \frac{1}{4}$ $\mathbf{z}_1, \mathbf{z}_2 \geq 0$ s.t. $\mathbf{z}_1 + 2\mathbf{z}_2 = 0.5$	$pos\ C_M(\{1, 2\})$
$\begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = \frac{1}{8}$	$pos\ C_M(\{2\})$

Table 4.4: q Vectors and Associated $\mathbf{z}$ Solutions for Example 3.2

- *Case 4 Unknown Structure for M:*

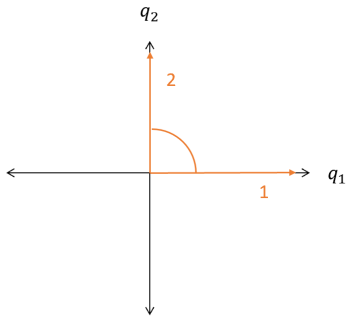
[2] Technically, in all cases in Table 4.4, there is a convex set of solutions because a single solution vacuously also constitutes a convex set. This matches with Theorem 7.

Example 4.1: We take an adjusted example from page 22 of [48] with the following M matrix:

$$M = \begin{bmatrix} 0 & 0.5 \\ -1 & -0.5 \end{bmatrix} \tag{4.3.35}$$

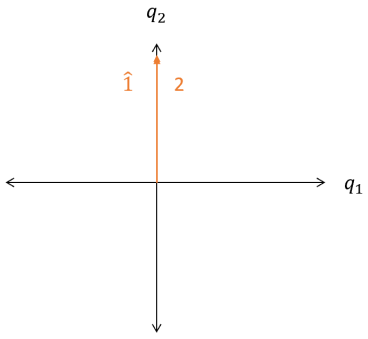
We see that this matrix does not fulfill any of the properties of the previous three cases because its principle minors are: 0, -0.5, 0.5 and because $\frac{1}{2}(M + M^T)$ has a negative eigenvalue. This M matrix's associated complementary cones are:

$$\alpha = \emptyset \tag{4.3.36a}$$



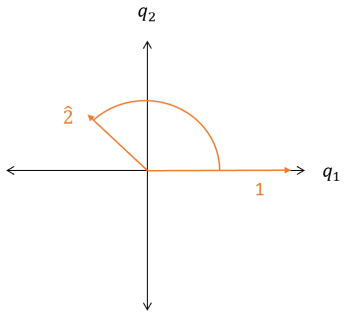
$$C_M(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.3.36b}$$

$$\alpha = \{1\} \tag{4.3.37a}$$



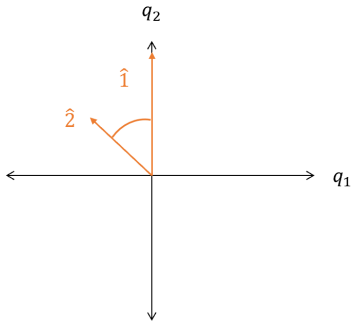
$$C_M(\alpha) = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{4.3.37b}$$

$$\alpha = \{2\} \tag{4.3.38a}$$



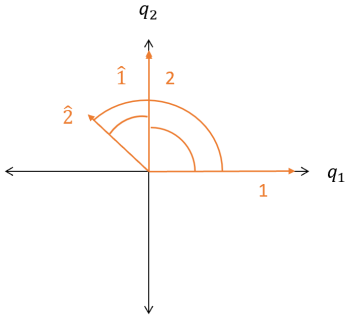
$$C_M(\alpha) = \begin{bmatrix} 1 & -0.5 \\ 0 & 0.5 \end{bmatrix} \tag{4.3.38b}$$

$$\alpha = \{1, 2\} \tag{4.3.39a}$$



$$C_M(\alpha) = \begin{bmatrix} 0 & -0.5 \\ 1 & 0.5 \end{bmatrix} \tag{4.3.39b}$$

All of the complementary cones.



Again, we see that there is a ray formed for the $pos\ C_M(\{1\})$ cone, which leads to a convex set of solutions as seen in Table 4.5 below. This results for the same reasons as Example

3.2; namely, the $C_M(\{1\})v = q$ system has linearly dependent columns in $C_M(\{1\})$ and the q lies in the cone created by $C_M(\{1\})$, so there is an infinite number of solutions to $C_M(\{1\})v = q$.

q Vector	$\mathbf{z}_1, \mathbf{z}_2$ Solutions	Associated Conic Regions
$\begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 1$	$pos\ C_M(\{2\})$
	$0 < \mathbf{z}_1 \leq 0.5, \mathbf{z}_2 = 0$	$pos\ C_M(\{1\})$
$\begin{bmatrix} -0.25 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 1$	$pos\ C_M(\{2\})$
	$\mathbf{z}_1 = \frac{1}{4}, \mathbf{z}_2 = \frac{1}{2}$	$pos\ C_M(\{1,2\})$
$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 0$	$pos\ C_M(\emptyset)$
	$\mathbf{z}_1 = 0, \mathbf{z}_2 = 1$	$pos\ C_M(\{2\})$

Table 4.5: q Vectors and Associated $\mathbf{z}$ Solutions for Example 4.1

Example 4.2: In this example, we expand the linearly dependent column idea from Example 3.2 to three dimensions, with an M and q as follows:

$$M = \begin{bmatrix} 1 & 4 & 14 \\ 2 & 5 & 19 \\ 3 & 6 & 24 \end{bmatrix}, \quad q = \begin{bmatrix} -12 \\ -16.5 \\ -21 \end{bmatrix} \tag{4.3.40}$$

This M matrix is not a $\mathbf{P}$ matrix because its determinant is 0, is degenerate because its determinant is 0, and is not positive definite because two of $\frac{1}{2}(M + M^T)$ eigenvalues are negative, with the eigenvalues being -2.42410512, -0.30252309, and 32.72662821. The solutions for this LCP(q, M) are:

1. $\mathbf{z}_1 = 12, \mathbf{z}_2 = 0, \mathbf{z}_3 = 0 \quad ((+, 0, 0)$ case)

2. $\mathbf{z}_1 = 0, \mathbf{z}_2 = 0, \mathbf{z}_3 = 0.875 \quad ((0, 0, +)$ case)

3. $\mathbf{z}_1 = 2, \mathbf{z}_2 = 2.5, \mathbf{z}_3 = 0 \quad ((+, +, 0)$ case)

4. $\mathbf{z}_1 = \frac{1}{3}, \mathbf{z}_2 = 0, \mathbf{z}_3 = \frac{5}{6} \quad ((+, 0, +)$ case)

5. $\mathbf{z}_1 + 2\mathbf{z}_3 - 2 = 0,\ \mathbf{z}_2 + 3\mathbf{z}_3 - 2.5 = 0$, for $\mathbf{z}_1 > 0, \mathbf{z}_2 > 0, \mathbf{z}_3 > 0 \quad ((+, +, +)$ case)

which we verified by hand by checking all 2^3 cases of variables being set to either 0 or a strictly positive number. We see that there is a convex set of solutions in item 5 because there are only two equations defining the three strictly positive variables, leading to an underdetermined system. This results from column 3 of M being a linear combination of the other two columns; specifically, column 3 of M is a combination of: $2M_{:,1} + 3M_{:,2} = M_{:,3}$. We also see that solutions in number 5 in the list above can be expressed as the convex combination of solution numbers 3 and 4 in the list above or, in other words,

$$\{\mathbf{z} : \mathbf{z}_1 + 2\mathbf{z}_3 - 2 = 0,\ \mathbf{z}_2 + 3\mathbf{z}_3 - 2.5 = 0,\ \mathbf{z} \geq 0\} = \{\mathbf{z} : \theta \in [0, 1], \mathbf{z} = \theta \begin{bmatrix} 2 \\ 2.5 \\ 0 \end{bmatrix} + (1 - \theta) \begin{bmatrix} \frac{1}{3} \\ 0 \\ \frac{5}{6} \end{bmatrix} \}$$

which we arrive at by taking the extreme points, which are not linear combinations of other points, of the solutions generated by the equations in number 5.

- **Example 4.3:** Finally, in this example, we demonstrate a solution set that is the union of two convex sets, but this union does not produce a convex set. We have the following M

and q:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 10 \\ 3 & 7 & 9 \end{bmatrix}, \quad q = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix} \tag{4.3.41}$$

This M has principle minors ranging from -34 to 9, including 0, and it has eigenvalues from -2.2093 to 16.0968. Therefore, it does not satisfy any of the properties previously outlined in Section 4.3.1.1. Instead, its solutions are the union of the two convex sets in (4.3.42), which we obtained by solving the problem by hand and was informed by the fact that various submatrices of M have linearly dependent columns. The submatrices having linearly dependent columns forces either 0 or an infinite number of solutions [170]. Specifically, for index sets $\alpha_1 = \{1, 2\}$ and $\alpha_2 = \{1, 3\}$, M_{α_1,α_1} and M_{α_2,α_2} form submatrices that have linearly dependent columns that correspond to entries in q that produce solution sets that are infinite. In turn, since these two convex sets correspond to solutions to the LCP, we can write that the entire solution set of the LCP is the union of these two sets:

$$\left(\theta^1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + (1 - \theta^1) \begin{bmatrix} 0 \\ 0.5 \\ 0 \end{bmatrix} \right) \cup \left(\theta^2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + (1 - \theta^2) \begin{bmatrix} 0 \\ 0 \\ \frac{1}{3} \end{bmatrix} \right), \quad \theta^1, \theta^2 \in [0, 1] \tag{4.3.42}$$

We see that this union does not produce a convex set because we cannot take the convex combination of an element from the first term and an element from the second term and guarantee that it is an element of the solution set of the LCP.

Having established these examples, we present one definition before moving on to our inverse optimization problem. We define what it means for a solution $\mathbf{z}$ to correspond to a complementary cone.

Definition 16. *A solution* $\mathbf{z}$ *corresponds to a complementary cone if* $\{k : \mathbf{z}_k > 0\} = \alpha^{\mathbf{z}}$ *for* $pos\ C_M(\alpha^{\mathbf{z}})$.

An example of this would be in Example 4.1 in which given a solution $\mathbf{z} = (\frac{1}{4}, \frac{1}{2})$, the corresponding complementary cone is $pos\ C_M(\{1, 2\})$. We may now proceed to the definition of the inverse optimization problem.

4.3.3 Basic Inverse Optimization Problem for LCP

Our inverse optimization problem for the LCP is to find a q that induces the set of $\mathbf{z}$ solutions that we observe. We define this set of $\mathbf{z}$ solutions as follows:

Definition 17. *The set* Z_{obs} *is the set of* $\mathbf{z}$ *solutions that we observe for the inverse optimization problem.*

These observed solutions can either be viewed in the world (such as via noting the flow of traffic in a traffic equilibrium problem), obtained from simulation, or gotten from solving a system optimal model. Given a set of LCP solutions that we observe, Z_{obs}, the set of q that could have possibly produced this set Z_{obs} can be written as:

$$Q_{IO} = \{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \tag{4.3.43}$$

We can also define the following set for a particular $\mathbf{z} \in Z_{obs}$:

$$Q_{IO_{\mathbf{z}}} = \{q^{\mathbf{z}} : \mathbf{z} \in SOL(q^{\mathbf{z}}, M)\} \tag{4.3.44}$$

Two examples to demonstrate these two sets come from Section 4.3.2. We first explore a Z_{obs} constructed from Example 1.1 in Section 4.3.2. Say $Z_{obs} = \{(\frac{1}{6}, \frac{1}{12})\}$. In this case, since $M \in \mathbf{P}$, $Q_{IO} = Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} -0.5 \\ \\ -0.5 \end{bmatrix} \right\}$. In the second example, we explore a Z_{obs} formed from Example 2.2 from Section 4.3.2. Say $Z_{obs} = \{(0,0), (2,0), (0,2)\}$. Then, $Q_{IO} = \left\{ \begin{bmatrix} 0.5 \\ \\ 0.5 \end{bmatrix} \right\}$. If we take a particular $\mathbf{z} \in Z_{obs}$ such as $(0,2)$, we have $Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} q_1 \geq -1 \\ \\ 0.5 \end{bmatrix} \right\}$.[3] We obtain these Q_{IO} and $Q_{IO_{\mathbf{z}}}$ by using the feasibility conditions from the forthcoming quadratic program (4.3.46). We notice from these examples that:

$$Q_{IO} \subseteq Q_{IO_{\mathbf{z}}} \tag{4.3.45}$$

When we want to refer to the solutions from the inverse optimization problem, we use the following terms.

Definition 18. *The terminology "q solution" or "IO q solution" refers to a solution found in either the set Q_{IO} or the set $Q_{IO_{\mathbf{z}}}$. The terminology is meant to denote a solution to the inverse optimization problem for the set Z_{obs}.*[4]

There are several considerations on Z_{obs} that we define in Section 4.4.1 that affect the q parameters to be obtained, including if the q generated from the set Z_{obs} by inverse optimization

[3] Determining these sets came from using Theorem 8 and, thus, the quadratic program (4.3.46) later defined. Specifically, we look at the feasibility constraints of this quadratic program (4.3.46), and the feasibility constraints produce the $Q_{IO_{\mathbf{z}}}$ and Q_{IO}.

[4] This Z_{obs} could contain a singular $\mathbf{z}$.

techniques produces extra $\mathbf{z}$ solutions not present in Z_{obs} when LCP(q, M) is solved (defined in Definition 19) and if the solutions in the set Z_{obs} correspond to some intersecting set of complementary cones (termed Propositions 2 and 3 and relationships (4.4.19) and (4.4.20)). Sections 4.4.2-4.4.5 focus on the characteristics of M and how they shape the ways we search for solutions. In Sections 4.4.2-4.4.5, we also address the considerations on the Z_{obs} set from Section 4.4.1 and their interactions with each of the M cases.

As an overarching technique and to close this section, we introduce a quadratic program that is central to finding q. It is specified and further refined in the coming sections, but it is worth defining here because we use it as a test for a property of Z_{obs} in Section 4.4.1. Given some starting guess of q, q^0, and the set Z_{obs} of $\mathbf{z}$ solutions for our candidate LCP$(\cdot, M)$, we draw on previous work on embedding equilibrium/optimization problems within an optimization program that calculates differences from a starting point such as minimizing the l_2 norm [55, 76] to produce a quadratic program as follows, with $\mathcal{J} = \{1, ..., |Z_{obs}|\}$ acting as an index set for Z_{obs} and $|\mathcal{J}| = |Z_{obs}|$ possibly being infinite. The use of q^0 allows for case-specific fine-tuning of problem (4.3.46). We note also that this quadratic program matches one of the quadratic programs discussed in Huang [103]:

$$\min_{q \in \mathcal{Q}} ||q - q^0||_2^2 \tag{4.3.46a}$$

$$(\mathbf{z}^j)^T (q + M\mathbf{z}^j) = 0, \quad \forall j \in \mathcal{J} \tag{4.3.46b}$$

$$q + M\mathbf{z}^j \geq 0, \quad \forall j \in \mathcal{J} \tag{4.3.46c}$$

Importantly, constraint (4.3.46b) represents the inner product between the jth $\mathbf{z}^j$ data vector and the vector $q + M\mathbf{z}^j$. $\mathcal{Q}$ represents any constraints we have placed on the variable q, which might include specific bounds on the elements of q and/or might include relational constraints among the q variables. For instance, say we have the M from Example 2.2 in Section 4.3.2, and say $Z_{obs} = \{(0, 2)\}$. Our quadratic program with $\mathcal{Q} = \mathbb{R}^n$ is:

$$\min_q ||q - q^0||_2^2 \tag{4.3.47a}$$

$$2q_2 - 1 = 0 \tag{4.3.47b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} \tag{4.3.47c}$$

If we wanted to restrict q to be in the non-negative orthant, we could write the problem with that following restriction:

$$\min_q ||q - q^0||_2^2 \tag{4.3.48a}$$

$$2q_2 - 1 = 0 \tag{4.3.48b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -1 \\ 0.5 \end{bmatrix} \tag{4.3.48c}$$

$$\mathcal{Q} = \{q : q \geq 0\} \tag{4.3.48d}$$

We may wish to restrict $q \geq 0$ if q represents some kind of non-negative item/quantity.

4.4 Algorithms for Finding q

We next discuss how to find q given the four cases for the matrix M discussed in Section 4.3.1.1. However, before we consider these cases in more depth, we need to consider the complementary cone effects on IO q solutions.

4.4.1 The Complete Information Property and Complementary Cone Effects on Possible IO q Solutions

The interaction between M and Z_{obs} takes up much of Section 4. In this subsection, we examine the complete information property as well as the way in which the complementary cones associated with matrix M can affect the possible q's that we can recover from Z_{obs}. To better understand this, we utilize Example 4.1 from Section 4.3.2. First, we see that one problem that can arise is when our chosen q returns more $\mathbf{z}$ solutions than are contained within Z_{obs}. To show this, we observe if $Z_{obs} = \{(0, 1)\}$, that solution set could have originated from either $\bar{q} = [0,\ 0.5]^T$, $\tilde{q} = [-0.25,\ 0.5]^T$, or $\hat{q} = [0.5,\ 0.5]^T$. However, as we know from Table 4.5, the first and third of these q vectors produce two different sets of $\mathbf{z}$ solutions, with $Z_{obs} \subset SOL(\hat{q}, M) \subset SOL(\bar{q}, M)$. For the second of these q vectors, $Z_{obs} \subset SOL(\tilde{q}, M)$. Therefore, we must distinguish between when the set Z_{obs} is guaranteed to find a q that only returns the set Z_{obs} in the forward problem and when it finds a q that produces a set $Z_{obs+extra}$ such that

$Z_{obs} \subset Z_{obs+extra}$.

Before proceeding with this definition, we explain the meaning of the following set intersection: $\bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\}$ using Example 2.2 in Section 4.3.2. Say $Z_{obs} = \{(0,0), (2,0), (0,2)\}$; the individual $Q_{IO_{\mathbf{z}}} = \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\}$ sets for each of the $\mathbf{z} \in Z_{obs}$ are as follows, which we obtained by looking at the feasible region of the quadratic program (4.3.46)[5]:

$$\mathbf{z} = (0,0) : Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} q_1 \geq 0 \\ q_2 \geq 0 \end{bmatrix} \right\} \tag{4.4.1a}$$

$$\mathbf{z} = (2,0) : Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} 0.5 \\ q_2 \geq -1 \end{bmatrix} \right\} \tag{4.4.1b}$$

$$\mathbf{z} = (0,2) : Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} q_1 \geq -1 \\ 0.5 \end{bmatrix} \right\} \tag{4.4.1c}$$

while the intersection of these $Q_{IO_{\mathbf{z}}}$ is:

$$\bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} = \bigcap_{\mathbf{z} \in Z_{obs}} Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right\} \tag{4.4.2}$$

With this set explained, we can define what it means for a set to have complete information.

Definition 19 (Complete Information)**.** *A set Z_{obs} has complete information when the $q \in \bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\}$ are such that $SOL(q, M) = Z_{obs}$.*

[5] which we know is equivalent to there being a solution to the IO problem by Theorem 8 which will be proven below.

To continue the example from above, we know $\bigcap_{\mathbf{z}\in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} = \left\{ \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right\}$. From Table 4.3, we know $SOL(q, M)$ for this q is equal to Z_{obs}, which means Z_{obs} has complete information. An example of a Z_{obs} without complete information is $Z_{obs} = \{(0,0),(2,0)\}$. The associated $Q_{IO_{\mathbf{z}}}$ are defined in (4.4.1), and the intersection of the $Q_{IO_{\mathbf{z}}}$ is:

$$\bigcap_{\mathbf{z}\in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} = \bigcap_{\mathbf{z}\in Z_{obs}} Q_{IO_{\mathbf{z}}} = \left\{ \begin{bmatrix} 0.5 \\ q_2 \geq 0 \end{bmatrix} \right\} \tag{4.4.3}$$

One of the qs in that intersection is $\left\{ \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right\}$ which we know from Table 4.3 produces the extra solution $(0,2)$. This means that Z_{obs} does not have complete information. We do point out that, just because the feasible region of the IO quadratic program (4.3.46) with the Z_{obs} solution set generates just one q vector, this does not mean that Z_{obs} has complete information. We can see this point if $Z_{obs} = \{(2,0),(0,2)\}$ for Example 2.2 because this Z_{obs} produces $q = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, but this IO q solution returns back the solution of $(0,0)$ when the LCP is solved in addition to the two other solutions.

In theory, one way to test if Z_{obs} has complete information is to solve (4.3.46) and then generate the solutions for the 2^n cases of zero and strictly positive $\mathbf{z}_i$ variables. One would then check to see if the $\mathbf{z}$ solutions recovered match the Z_{obs} set. This is feasible for smaller problems, but quickly becomes infeasible for larger problems. Another approach to test if Z_{obs} has complete information would be to generate the 2^n set of complementary cones and see if the q we obtain from (4.3.46) appears only in the complementary cones corresponding to the solutions present in

Z_{obs} [48]. This would require the formulation of 2^n feasibility problems involving the following constraints: $C_M(\alpha)v = q,\ v \geq 0$ so, again, the scalability is in question [48].

Second, we see that, if our observed solution set Z_{obs} has $\mathbf{z}$ solutions that correspond to non-overlapping complementary cones, this presents a problem for finding a q vector. For example, using the mathematical program defined in (4.3.46), we obtain a contradiction in the mathematical program when $Z_{obs} = \{(0,0),(0,1),(\frac{1}{4},\frac{1}{2})\}$ for Example 4.1 in Section 4.3.2. The constraints (without the objective function) in the mathematical program according to (4.3.46) are as follows for this set Z_{obs} and the example's M:

$$q_2 - 0.5 = 0 \quad \text{(from (0,1))} \tag{4.4.4a}$$

$$\frac{1}{4}q_1 + \frac{1}{2}q_2 - \frac{1}{8} - \frac{1}{16} = 0 \quad \text{(from } \left(\frac{1}{4},\frac{1}{2}\right)\text{)} \tag{4.4.4b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{(from (0,0))} \tag{4.4.4c}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \quad \text{(from (0,1))} \tag{4.4.4d}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -0.25 \\ 0.5 \end{bmatrix} \quad \text{(from } \left(\frac{1}{4},\frac{1}{2}\right)\text{)} \tag{4.4.4e}$$

In the equality constraints (4.4.4a)-(4.4.4b) created by (4.3.46b), $q_1 = \frac{-1}{4}, q_2 = \frac{1}{2}$, but this contradicts the lower bound in the inequality constraint (4.4.4c) (created by (4.3.46c)) that requires

$q_1 \geq 0$. Furthermore, if we graph the potential solution $q = [\frac{-1}{4} \ \frac{1}{2}]$ vector in the graph depicting all of the complementary cones, it only is in two out of the three regions to which the $\mathbf{z}$ solutions in Z_{obs} correspond.

As a result of this problem, when we have multiple $\mathbf{z}$ solutions in Z_{obs}, we may need to further subset them according to if they produce a feasible q solution to (4.3.46), which can be used as a test to see if the $\mathbf{z}$ solutions in Z_{obs} correspond to overlapping complementary cones as can be seen in Proposition 3. We discuss in the coming subsections the ways in which we address the complementary cone considerations for each case. Working up to Proposition 3, we formalize the knowledge using Proposition 2 and relationships (4.4.19) and (4.4.20) that, if the inverse optimization problem has a q solution for a given Z_{obs}, then Z_{obs} corresponds to overlapping complementary cones and, if the corresponding complementary cones for Z_{obs} do not overlap, then there is no q solution to the inverse optimization problem, unless M is a $\mathbf{P}$ matrix which means there should only be one solution in Z_{obs} since each q in $\mathbb{R}^n$ is only associated with one $\mathbf{z}$ for this type of M. In the situation in which the corresponding complementary cones for Z_{obs} do not overlap, we would try different (potentially mutually exclusive) subsets of Z_{obs} in the quadratic program (4.3.46) until we obtained a q for each subset such that union of the subsets resulted in Z_{obs}.

We then define some sets that are important for Propositions 2-3 and relationships (4.4.19)-(4.4.20):

- The set $\{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\}$ is the complementary cone corresponding to the solution $\mathbf{z}$. For example, for the solution $\bar{\mathbf{z}} = [0, \ 1]^T$ from Table 4.5 for Example 4.1

in Section 4.3.2, the corresponding complementary cone is:

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{v}_1 + \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \mathbf{v}_2, \quad \mathbf{v}_1, \mathbf{v}_2 \geq 0 \tag{4.4.5}$$

We prove in Proposition 2 that

$$\{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} \subseteq \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \tag{4.4.6}$$

As an example, of this, we return to the $\bar{\mathbf{z}} = [0,\ 1]^T$ for Example 4.1, and see for $\{\hat{q}^{\mathbf{z}} : \bar{\mathbf{z}} \in SOL(\hat{q}^{\mathbf{z}}, M)\}$ that this set is $\{\hat{q} : \hat{q}_1 \geq -0.5,\ \hat{q}_2 = 0.5\}$ for this $\bar{\mathbf{z}}$. The set $\{\bar{q} : C_M(\alpha^{\bar{\mathbf{z}}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\}$ is defined in (4.4.5). We see that (4.4.6) holds based on these two set definitions, and we can see it pictorially in the following Figure 4.2. The $Q_{IO_{\mathbf{z}}}$ is in blue as a ray, and the corresponding complementary cone for $\mathbf{z}$ is in orange; we see the blue ray is contained within the orange cone.

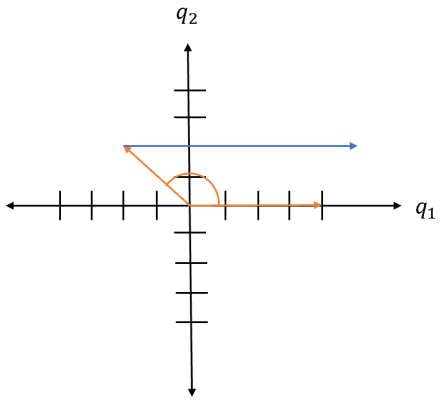


Figure 4.2: $Q_{IO_{\mathbf{z}}}$ as a Subset of the Corresponding Complementary Cone for $\mathbf{z}$

- For the set $\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\}$, this set denotes the intersection

of the overlapping complementary cones corresponding to the solutions in Z_{obs}. The $\mathbf{z}$ solution set $Z_{obs} = \{(0,1), (\frac{1}{4}, \frac{1}{2})\}$ from Example 4.1 in Section 4.3.2 corresponds to complementary cones $pos\ C_M(\{2\})$ and $pos\ C_M(\{1,2\})$, and we can visualize the intersection of these overlapping cones pictorially in Figure 4.3.

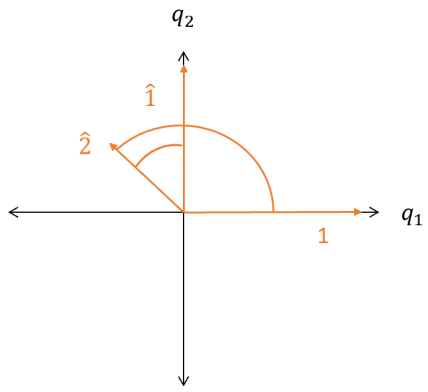


Figure 4.3: Overlapping Complementary Cones

The intersection of the two cones $pos\ C_M(\{2\})$ and $pos\ C_M(\{1,2\})$ in the 2nd quadrant of $\mathbb{R}^2$ space defines the set $\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q}$ for some $v \in \mathbb{R}^n_{\geq 0}\}$. Thus, this set includes the q that are the IO q solutions for the Z_{obs} set, but it includes other q as well. Specifically, $\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q}$ for some $v \in \mathbb{R}^n_{\geq 0}\}$ for Z_{obs} can be defined as:

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{v}_1 + \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \mathbf{v}_2, \quad \mathbf{v} \geq 0 \tag{4.4.7}$$

while the q that are the IO q solutions for the Z_{obs} set are:

$$q = \begin{bmatrix} -0.25 \\ 0.5 \end{bmatrix} \tag{4.4.8}$$

which we obtain from looking at the feasible region of quadratic program (4.3.46). We prove in Proposition 2 that:

$$\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \subseteq \bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \quad (4.4.9)$$

With these sets defined and described, we have Proposition 2.

Proposition 2 (IO q Solutions as a Subset of Corresponding Complementary Cones)**.** *We have the following subset relationship between the IO q solutions to the problem (4.3.43) and the corresponding overlapping complementary cones to the set Z_{obs}:*

$$\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \subseteq \bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \textit{ for some } v \in \mathbb{R}^n_{\geq 0}\} \quad (4.4.10)$$

We also establish that:

$$\{q : Z_{obs} \subseteq SOL(q, M)\} = \bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} \quad (4.4.11)$$

Proof. Each solution $\mathbf{z} \in Z_{obs}$ corresponds to $pos\ C_M(\alpha^{\mathbf{z}})$ for the index set $\alpha^{\mathbf{z}}$. We know

$$\{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} \subseteq \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \quad (4.4.12)$$

because of the following logic. For the set $\{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\}$, we know for a given $\hat{q}$ in this set that $0 \leq M\mathbf{z} + \hat{q} \perp \mathbf{z} \geq 0$ holds. This leads to the following two cases:

- For $\mathbf{z}_i > 0$, we write $\hat{q}_i = -M_{i,:}\mathbf{z}$ from the LCP conditions above. However, for $j \neq i$

$\mathbf{z}_j = 0$, we can replace $-M_{i,j} = 0$ and achieve the same $\hat{q}_i$

- For $\mathbf{z}_i = 0$, we write $\hat{q}_i \geq -M_{i,:}\mathbf{z}$ from the LCP conditions above. We can write $-M_{i,:}\mathbf{z}$ as follows:

$$-M_{i,:}\mathbf{z} = -\left(\sum_{j \neq i} M_{i,j}\mathbf{z}_j + M_{i,i}\mathbf{z}_i\right) = -\left(\sum_{j \neq i} M_{i,j}\mathbf{z}_j - \mathbf{z}_i\right) \tag{4.4.13}$$

  We see that $M_{i,i}$ can be replaced with a -1, since $\mathbf{z}_i = 0$. However, since $\hat{q}_i \geq -M_{i,:}\mathbf{z}$, we see that we can write $\hat{q}_i$ as follows:

$$\hat{q}_i = -\left(\sum_{j \neq i} M_{i,j}\mathbf{z}_j - w_i\right) \tag{4.4.14}$$

  for $w_i \geq 0$. Because we replaced $-M_{i,j} = 0$ for $\mathbf{z}_j = 0$ in the previous bullet, this current w_i does not affect the $\hat{q}_i$ for the $\mathbf{z}_i > 0$ above.

Thus, employing these observations, we can construct a new matrix A such that $A^{\mathbf{z}}w = \hat{q}$, for a $w \geq 0$ and, using h as the index for the elements of $\mathbf{z}$ and h as an index for the columns of M, such that

$$A^{\mathbf{z}}_{:,h} = \begin{cases} -M_{:,h} & \text{if } \mathbf{z}_h > 0 \\ I_{:,h} & \text{if } \mathbf{z}_h = 0 \end{cases} \tag{4.4.15}$$

We see that $A^{\mathbf{z}}$ matches $C_M(\alpha^{\mathbf{z}})$, and w matches the v for the corresponding complementary cone of $\mathbf{z}$ because w is forced to be non-negative to fulfill the bullet points above. Specifically, the i for which $\mathbf{z}_i > 0$ results in $w_i = \mathbf{z}_i$ but, for the i in which $\mathbf{z}_i = 0$, $w_i \geq 0$ since $\hat{q}_i \geq -M_{i,:}\mathbf{z}$.

Therefore $\hat{q} \in \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\}$ thus proving (4.4.12). If we have the following set relationships $U \subseteq V$ and $X \subseteq Y$, then we can say $U \cap X \subseteq V \cap Y$. This allows us to say that:

$$\bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} \subseteq \bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \tag{4.4.16}$$

As an intermediate step to proving the proposition, we can say that:

$$\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} = \bigcap_{\mathbf{z} \in Z_{obs}} \{\hat{q}^{\mathbf{z}} : \mathbf{z} \in SOL(\hat{q}^{\mathbf{z}}, M)\} \tag{4.4.17}$$

because a q such that $Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)$ would be an element of the intersection of the $Q_{IO_{\mathbf{z}}}$ over the $\mathbf{z} \in Z_{obs}$, and a q that is an element of the intersection of $Q_{IO_{\mathbf{z}}}$ over $\mathbf{z} \in Z_{obs}$ would be a q that would make $Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)$. We can then say that:

$$\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \subseteq \bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \tag{4.4.18}$$

which proves (4.4.10). □

From Proposition 2, we use set relationship knowledge to make the following statements:

$$\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset \Longrightarrow \bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} \neq \emptyset \tag{4.4.19}$$

$$\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q} \text{ for some } v \in \mathbb{R}^n_{\geq 0}\} = \emptyset \Longrightarrow \{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} = \emptyset \tag{4.4.20}$$

These relationships tell us that, if the IO problem has a q solution, then the Z_{obs} involved corresponds to overlapping complementary cones and, if the corresponding complementary cones to Z_{obs} do not overlap, then the IO problem does not have a q solution.

Working up to a test regarding if a set Z_{obs} corresponds to overlapping complementary cones, we have the following theorem.

Theorem 8. *The quadratic program (4.3.46) is feasible for a given set Z_{obs} and set $\mathcal{Q} = \mathbb{R}^n$ if and only if $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$*

Proof. $\Rightarrow$ If the quadratic program is feasible, then it means there exists a q such that the LCP conditions of $0 \leq M\mathbf{z} + q \perp \mathbf{z} \geq 0$ have been met for all the $\mathbf{z} \in Z_{obs}$. This implies that there is a q such that $Z_{obs} \subseteq SOL(q, M)$.

$\Leftarrow$ If $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$, then there exists a q that meets the LCP conditions $0 \leq M\mathbf{z} + q \perp \mathbf{z} \geq 0$ for all $\mathbf{z} \in Z_{obs}$. These are exactly the conditions for feasibility for the quadratic program (4.3.46) when $\mathcal{Q} = \mathbb{R}^n$. □

Using relationship (4.4.19) and Theorem 8, we can use the feasibility of the quadratic program (4.3.46) when $\mathcal{Q} = \mathbb{R}^n$ to see if a set Z_{obs} corresponds to overlapping complementary cones. We see this in the next proposition.

Proposition 3. *If the quadratic program (4.3.46) is feasible for a set Z_{obs} and $\mathcal{Q} = \mathbb{R}^n$, then Z_{obs} corresponds to overlapping complementary cones, meaning $\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q}$ for some $v \in \mathbb{R}^n_{\geq 0}\} \neq \emptyset$.*

Proof. We have from Theorem 8 that, if the quadratic program (4.3.46) is feasible for a set Z_{obs} and set $\mathcal{Q} = \mathbb{R}^n$, then $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$. We then have from relationship

(4.4.19) that, if $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$, then $\bigcap_{\mathbf{z} \in Z_{obs}} \{\bar{q} : C_M(\alpha^{\mathbf{z}})v = \bar{q}$ for some $v \in \mathbb{R}^n_{\geq 0}\} \neq \emptyset$.

□

Thus, we can use the feasibility of quadratic program (4.3.46) with $\mathcal{Q} = \mathbb{R}^n$ to see for a set Z_{obs} (a) if there is a q solution to the inverse problem as seen in Theorem 8 and (b) if the set Z_{obs} corresponds to a set of overlapping complementary cones as seen in Proposition 3.

4.4.2 Case 1 $M \in \mathbf{P}$

To begin this section, we assume that we have a set Z_{obs} consisting of one LCP $\mathbf{z}$ solution and that $M \in \mathbf{P}$. Given that we know our LCP(q, M) produces a unique $\mathbf{z}$ solution for every q due to $M \in \mathbf{P}$, we can form a quadratic program to find a q given this unique $\mathbf{z}$ solution. Using the quadratic program (4.3.46), we adapt it to the situation when $M \in \mathbf{P}$ in which we have $|Z_{obs}| = 1$ and thus $Z_{obs} = \mathbf{z}^1$ as follows:

$$\min_{q \in \mathcal{Q}} ||q - q^0||_2^2 \tag{4.4.21a}$$

$$(\mathbf{z}^1)^T(q + M\mathbf{z}^1) = 0 \tag{4.4.21b}$$

$$q + M\mathbf{z}^1 \geq 0 \tag{4.4.21c}$$

If M is a $\mathbf{P}$-matrix and $\mathbf{z}^1 \geq 0$, then we know that, if this quadratic program is feasible, it produces a q that induces $\mathbf{z}^1$ to be the solution of LCP(q, M).

Proposition 4. *Let M be a* ***P****-matrix and $\mathbf{z}^1 \geq 0$. If the quadratic program (4.4.21) is feasible, then it produces a $q^{Z_{obs}}$ such that $\mathbf{z}^1 = SOL(q^{Z_{obs}}, M)$.*

Proof. If (4.4.21) is feasible, then this means there exists a $q^{Z_{obs}}$ such that $(\mathbf{z}^1)^T(q^{Z_{obs}} + M\mathbf{z}^1) = 0$ and $q^{Z_{obs}} + M\mathbf{z}^1 \geq 0$. We are already given that $\mathbf{z}^1 \geq 0$. From Section 4.3.1, we know that these are the three conditions that define an LCP. Therefore, we know $\mathbf{z}^1$ is a solution of LCP$(q^{Z_{obs}}, M)$ for the feasible vector q. We also know, by Theorem 5, that $\mathbf{z}^1$ is the only solution since M is a **P**-matrix. Therefore, $\mathbf{z}^1 = \text{SOL}(q^{Z_{obs}}, M)$ □

For $M \in \mathbf{P}$, we could solve (4.4.21) and find our q directly. However, we notice that we can simplify (4.4.21) through a few observations, which we generalize with multiple solutions in the next section. We note first that, for each variable i in $\mathbf{z}^1$ that is strictly positive, $q_i = -M_{i,:}\mathbf{z}^1$. This is because of the complementarity relationship between the components of $\mathbf{z}^1$ and $q + M\mathbf{z}^1$, meaning that, if $z_i^1 > 0$, this implies $q_i + M_{i,:}\mathbf{z}^1 = 0$ which means $q_i = -M_{i,:}\mathbf{z}^1$. We then observe that, if $\mathbf{z}_i^1 = 0$, $q_i + M_{i,:}\mathbf{z}^1 \geq 0$, since the inner product of the equality constraint (4.4.21b) forces one of each of the nonnegative component pairs to be zero, so $q_i \geq -M_{i,:}\mathbf{z}^1$ since $z_i^1 = 0$ already. We can then form two sets $\mathcal{Z}^+$ and $\mathcal{Z}^0$, with $i \in \mathcal{Z}^+$ corresponding to the indices of $\mathbf{z}^1$ where $\mathbf{z}_i^1 > 0$ and $k \in \mathcal{Z}^0$ corresponding to the indices of $\mathbf{z}^1$ where $\mathbf{z}_k^1 = 0$. We then have the following mathematical program:

$$\min_{q \in \mathcal{Q}} ||q - q^0||_2^2 \tag{4.4.22a}$$

$$q_i = -M_{i,:}\mathbf{z}^1, \quad \forall i \in \mathcal{Z}^+ \tag{4.4.22b}$$

$$q_k \geq -M_{k,:}\mathbf{z}^1, \quad \forall k \in \mathcal{Z}^0 \tag{4.4.22c}$$

Proposition 5. *Given that M is a **P**-matrix and $\mathbf{z}^1 \geq 0$, if the quadratic program (4.4.22) is feasible, then it produces a q such that $\mathbf{z}^1 = SOL(q^{Z_{obs}}, M)$.*

Having discussed the IO q solution methods for $M \in \mathbf{P}$, we remember the considerations of Section 4.4.1, and see their relation to this case. We know that a Z_{obs} with one $\mathbf{z}$ solution has complete information for $M \in \mathbf{P}$ because, as discussed in Section 4.3.1.2, the complementary cones for a $\mathbf{P}$-matrix do not overlap and cover the entire space of $\mathbb{R}^n$ [48]. This means also that, if $|Z_{obs}| > 1$, then we have a problem because each q maps to one $\mathbf{z}$ solution; this speaks to the idea that Z_{obs} for $M \in \mathbf{P}$ would not have overlapping complementary cones and, thus, relationship (4.4.20) states there is a not an inverse optimization q solution for this set Z_{obs}. Consequently, in a situation in which $|Z_{obs}| > 1$ for $M \in \mathbf{P}$, we need to form separate mathematical programs (4.4.21)-(4.4.22) for each $\mathbf{z} \in Z_{obs}$ and, thus, generate separate q vectors for each of the LCP $\mathbf{z} \in Z_{obs}$. This situation could occur if there is a change in the system or market being modeled by the LCP such that the q vector changes, and $\mathbf{z}$ solution changes as a result, meaning that there are multiple solutions in Z_{obs}.

4.4.3 Case 2 Nondegenerate Matrix M

To begin this section, we assume that the set Z_{obs} of LCP solutions satisfies $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ meaning the quadratic program (4.3.46) is feasible and Z_{obs} corresponds to overlapping complementary cones, or i.e., relationship (4.4.19) and Theorem 8 are satisfied. We address when this may fail at the end of the section. We do *not* assume that the set Z_{obs} has

complete information according to Definition 19. We discuss an example of this lack of complete information at the end of this subsection. In this subsection, we assume M is nondegenerate, so we know our LCP$(\cdot, M)$ produces a finite number of $\mathbf{z}$ solutions. To demonstrate the need to consider all of the $\mathbf{z}$ solutions carefully in Z_{obs} when constructing the quadratic program (4.3.46), we consider Example 2.1 in Section 4.3.2 with a specified q that is different from the ones found in Table 4.2

$$M = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, \quad q = \begin{bmatrix} 0 \\ -3 \end{bmatrix} \tag{4.4.23}$$

This LCP(q, M) has exactly two solutions:

$$\bar{\mathbf{z}}^1 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}, \quad \bar{\mathbf{z}}^2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \tag{4.4.24}$$

If we use just $\bar{\mathbf{z}}^1$ in the IO quadratic program (4.3.46), we obtain the following program:

$$\min_{q \in \mathcal{Q}} (q_1 - q_1^0)^2 + (q_2 - q_2^0)^2 \tag{4.4.25a}$$

$$3q_2 + 9 = 0 \tag{4.4.25b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -3 \\ -3 \end{bmatrix} \tag{4.4.25c}$$

Note that q_1 does have a lower bound, but its value varies depending on the q_1^0 value chosen

because q_1 does not appear in equation (4.4.25b). If $q_1^0 > -3$, then $q_1 = q_1^0$ and, if $q_1^0 \leq -3$, then $q_1 = -3$. We know $q_2 = -3$ based on (4.4.25b). If $q = [-3,\ -3]^T$, we see that $\bar{\mathbf{z}}^2$ is not a solution to the LCP with this new q because:

$$\begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \end{bmatrix} + \begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \end{bmatrix} \not\geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.4.26}$$

This points to the importance of including all of the finite number of $\mathbf{z}$ solutions from Z_{obs} in the mathematical program that is recovering q if a reader desires to find a q such that $Z_{obs} \subseteq SOL(q, M)$. We discuss ways of reworking the constraints in (4.3.46) to eliminate both some of the constraints and some of the $\mathbf{z} \in Z_{obs}$ used in the quadratic program feasible region. For illustrative purposes for now, we see that, when we use all of the $\mathbf{z}$ solutions in the quadratic program (4.3.46), we obtain the following program:

$$\min_{q \in \mathcal{Q}} (q_1 - q_1^0)^2 + (q_2 - q_2^0)^2 \tag{4.4.27a}$$

$$3q_2 + 9 = 0 \quad \text{(from } \bar{\mathbf{z}}^1\text{)} \tag{4.4.27b}$$

$$3q_1 + 3q_2 + 9 = 0 \quad \text{(from } \bar{\mathbf{z}}^2\text{)} \tag{4.4.27c}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -3 \\ -3 \end{bmatrix} \quad \text{(from } \bar{\mathbf{z}}^1\text{)} \tag{4.4.27d}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ -3 \end{bmatrix} \quad \text{(from } \bar{\mathbf{z}}^2\text{)} \tag{4.4.27e}$$

Equations (4.4.27b) and (4.4.27c) come from the complementarity relationships for $\bar{\mathbf{z}}^1$ and $\bar{\mathbf{z}}^2$, respectively. As can be seen, (4.4.27e) acts as a floor for the q_1 and q_2 values that can satisfy (4.4.27c) which forces the IO solution of $q = [0, -3]^T$. The reader may notice that, with regard to the inequality constraints found in (4.3.46), we only need the inequality constraints generated from $\bar{\mathbf{z}}^2$, as well as the equality constraints generated from the two $\mathbf{z}$ solutions. This suggests that $\bar{\mathbf{z}}^2$ is able to provide more definitive information than $\bar{\mathbf{z}}^1$, which we discuss shortly.

Having worked through this example, we then discuss the mathematical program that can recover the q that produced all of the finite number of $\mathbf{z}$ solutions in a general Z_{obs}. It is simply (4.3.46) with the set Z_{obs}. We then have the following corollary to Proposition 4.

Corollary 2. *Given that M is nondegenerate, $\mathbf{z}^j \geq 0, \forall j \in \mathcal{J}$, if the quadratic program (4.3.46) is feasible, then it produces a $q^{Z_{obs}}$ such that $Z_{obs} = \{\mathbf{z}^j : j \in \mathcal{J}\} \subseteq SOL(q^{Z_{obs}}, M)$.*

Proof. If (4.3.46) is feasible, then that means there exists a $q^{Z_{obs}}$ such that $(\mathbf{z}^j)^T(q^{Z_{obs}} + M\mathbf{z}^j) = 0, \forall j \in \mathcal{J}$ and $q^{Z_{obs}} + M\mathbf{z}^j \geq 0, \forall j \in \mathcal{J}$. We are already given that $\mathbf{z}^j \geq 0, \forall j \in \mathcal{J}$. From Section 4.3.1, we know these are the conditions that define an LCP so, since each $\mathbf{z}^j$ fulfills these conditions, each $\mathbf{z}^j$ is a solution to the LCP. Therefore, we know $Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)$.

□

Similarly to Section 4.4.2, we can now ask the question: is there a more efficient way to formulate this quadratic program? The answer is yes, and we tackle this question in three steps. First, we establish Propositions 6 and 7 which establish some properties of the quadratic program and the

interactions between the $\mathbf{z}$ solutions in Z_{obs} and the M matrix. Second, we establish specific constraint relationships in Propositions 8 and 9 that can be derived using the relations in the quadratic program and the previous propositions. Third, we take Propositions 8 and 9 and define the simplified version of quadratic program (4.3.46) in Theorem 9.

Proposition 6. *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, if* $\mathbf{z}_i^k > 0$ *for every* $k \in \mathcal{K} \subseteq \mathcal{J}$ *and for a given* $i \in \mathcal{I}$*, then* $M_{i,:}\mathbf{z}^k = c, \forall k \in \mathcal{K}$ *where* c *is some constant number.*

Proof. For all $q^h \in \{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\}$ and all $\mathbf{z}^j \in Z_{obs}$, we know

$$0 \leq M\mathbf{z}^j + q^h \perp \mathbf{z}^j \geq 0 \tag{4.4.28}$$

because that is the definition of $Z_{obs} \subseteq SOL(q^h, M)$. For all $k \in \mathcal{K}$ and a given $i \in \mathcal{I}$, if $\mathbf{z}_i^k > 0$, then $M_{i,:}\mathbf{z}^k + q_i^h = 0$. Say we have $\mathbf{z}_i^1 > 0$ and $\mathbf{z}_i^2 > 0$ with $\mathbf{z}^1, \mathbf{z}^2 \in Z_{obs}$ and we have q^1, $q^2 \in \{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\}$. Also, say $M_{i,:}\mathbf{z}^1 = c$. Then, then we can write:

$$M_{i,:}\mathbf{z}^1 + q_i^1 = 0, \quad M_{i,:}\mathbf{z}^2 + q_i^2 = 0 \tag{4.4.29a}$$

$$M_{i,:}\mathbf{z}^1 + q_i^2 = 0, \quad M_{i,:}\mathbf{z}^2 + q_i^1 = 0 \tag{4.4.29b}$$

which means

$$-M_{i,:}\mathbf{z}^1 = -M_{i,:}\mathbf{z}^2 = -c \tag{4.4.30}$$

□

An example of this proposition is seen in the quadratic program above (4.4.27) for Example 2.1's M in which, for both $\bar{\mathbf{z}}^1, \bar{\mathbf{z}}^2$ solutions in (4.4.24), $\bar{\mathbf{z}}_2^1 > 0$ and $\bar{\mathbf{z}}_2^2 > 0$, and the associated $-M_{2,:}\bar{\mathbf{z}}^1$ and $-M_{2,:}\bar{\mathbf{z}}^1$ are equal according to the quadratic program.

Proposition 7. *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, if, for a given* $i \in \mathcal{I}$*, we have* $\mathbf{z}_i^k > 0$ *for all* $k \in \mathcal{K} \subseteq \mathcal{J}$ *and if we have* $\mathbf{z}_i^w = 0$ *for all* $w \in \mathcal{W} = \mathcal{J} - \mathcal{K}$*, then* $M_{i,:}\mathbf{z}^w \geq M_{i,:}\mathbf{z}^k, \forall k \in \mathcal{K}, w \in \mathcal{W}$.

Proof. Given our assumption that $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$, then for any $q \in \{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\}$:

$$M_{i,:}\mathbf{z}^w + q_i \geq 0, \quad \forall w \in \mathcal{W} \tag{4.4.31}$$

and $\forall k \in \mathcal{K}\ M_{i,:}\mathbf{z}^k + q_i = 0$ since $\mathbf{z}_i^k > 0$. Therefore, we can write:

$$-M_{i,:}\mathbf{z}^w \leq q_i = -M_{i,:}\mathbf{z}^k, \quad \forall k \in \mathcal{K}, w \in \mathcal{W} \tag{4.4.32a}$$

$$M_{i,:}\mathbf{z}^w \geq M_{i,:}\mathbf{z}^k, \quad \forall k \in \mathcal{K}, w \in \mathcal{W} \tag{4.4.32b}$$

□

An example of this can be see again in the quadratic program (4.4.27) in which, for q_1, $M_{1,:}\bar{\mathbf{z}}^1 = 3$, while $M_{1,:}\bar{\mathbf{z}}^2 = 0$, thus satisfying the inequality in the proposition above.

Before moving to Theorem 9, we formalize the idea that, when $\mathbf{z}_i^j = 0$ for all $j = 1, ..., \mathcal{J}$ and for a given $i \in \mathcal{I}$, then the quadratic program (4.3.46) only contains lower bounds for the

associated q_i.

Proposition 8. *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, if* $\mathbf{z}_i^j = 0$ *for all* $j \in \mathcal{J}$ *for a given entry* i*, then the only constraints on the associated* q_i *for quadratic program (4.3.46) are:*

$$q_i \geq -M_{i,:}\mathbf{z}^j, \quad j \in \mathcal{J} \tag{4.4.33}$$

Proof. To see this, we examine quadratic program (4.3.46). For the equality constraints (4.3.46b), if for all $j \in \mathcal{J}$ and for a given $i \in \mathcal{I}$, $\mathbf{z}_i^j = 0$, then the associated q_i is zeroed out for all j in constraint (4.3.46b). Therefore, we are left with the lower bounds on q_i in the inequality constraints (4.3.46c). □

To see an example of this proposition, we turn to Example 2.2 from Section 4.3.2. We select $Z_{obs} = \{(0,0),(2,0)\}$. The constraints are as follows:

$$2q_1 - 1 = 0 \quad \text{(from (2,0))} \tag{4.4.34a}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{(from (0,0))} \tag{4.4.34b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} 0.5 \\ -1 \end{bmatrix} \quad \text{(from (2,0))} \tag{4.4.34c}$$

We see that q_2 is not involved in the equality constraints, only in the inequality constraints.

Continuing to the final proposition before the main theorem, we propose the idea that, if an

entry i of a solution $\mathbf{z} \in Z_{obs}$ is strictly positive, then we can use this $\mathbf{z}$ to determine the value of q_i.

Proposition 9. *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, if there exists a subset* $\mathcal{K} \subseteq \mathcal{J}$ *of* $\mathbf{z}^k \in Z_{obs}$ *such that* $\mathbf{z}_i^k > 0$ *for a given entry* $i \in \mathcal{I}$ *and for all* $k \in \mathcal{K}$*, then:*

$$q_i = -M_{i,:}\mathbf{z}^k \quad \forall k \in \mathcal{K} \tag{4.4.35}$$

Proof. We know the constraints of quadratic program (4.3.46) state the following for $\mathbf{z}^j$ solutions $j \in \mathcal{J}$:

$$(\mathbf{z}^j)^T(q + M\mathbf{z}^j) = 0, \quad \forall j \in \mathcal{J} \tag{4.4.36a}$$

$$q + M\mathbf{z}^j \geq 0, \quad j \in \mathcal{J} \tag{4.4.36b}$$

which we know, according to Theorem 8, has a q solution because we assumed that the IO problem has a q solution. We know from Proposition 8 that, if for a given $i \in \mathcal{I}$, $\mathbf{z}_i^j = 0$ for all $j \in \mathcal{J}$, then we have only the constraints $q_i \geq -M_{i,:}\mathbf{z}^j$, $\forall j \in \mathcal{J}$ for that q_i. If, however, for a given entry $i \in \mathcal{I}$, there exists $\mathbf{z}^k \in Z_{obs}$ for all $k \in \mathcal{K} \subseteq \mathcal{J}$ such that $\mathbf{z}_i^k > 0$, then by equality constraint (4.4.36a), $q_i = -M_{i,:}\mathbf{z}^k$ due to the fact that the inner product of two nonnegative vectors forces at least one element in each corresponding pair to be 0. We know from Proposition 6 that, for all $k \in \mathcal{K}$ such that $\mathbf{z}_i^k > 0$, $M_{i,:}\mathbf{z}^k = c$, which means we only need one of those $k \in \mathcal{K}$ $\mathbf{z}$ solutions to produce the q_i value. We also know from Proposition 7 that for any $w \in \mathcal{W} = \mathcal{J} - \mathcal{K}$ such that $\mathbf{z}_i^w = 0$, there is the relationship that $M_{i,:}\mathbf{z}^k \leq M_{i,:}\mathbf{z}^w \Rightarrow -M_{i,:}\mathbf{z}^k \geq -M_{i,:}\mathbf{z}^w$.

This means we can eliminate the $q_i \geq -M_{i,:}\mathbf{z}^w$ constraints. Thus, if there exists $\mathbf{z}^k \in Z_{obs}$ $\forall k \in \mathcal{K} \subseteq \mathcal{J}$ such that $\mathbf{z}_i^k > 0$, then $q_i = -M_{i,:}\mathbf{z}^k, \forall k \in \mathcal{K}$.

□

An example of this proposition comes from the quadratic program in the example at the beginning of this subsection, i.e., (4.4.27). In this quadratic program, the q solution is defined, but we see that the program could have been reduced to the constraints $q_1 = 0, q_2 = -3$ based on the fact that we had the $\bar{\mathbf{z}}^2$ solution with both entries as positive.

With these last two propositions, we can simplify the IO quadratic program (4.3.46) dramatically.

Theorem 9 (Simplifying Quadratic Program (4.3.46))**.** *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, the associated quadratic program (4.3.46) feasible region can be simplified for* $\mathbf{z} \in Z_{obs}$ *according to the following rules for all* $i \in \mathcal{I}$ *(with* $\mathcal{I}$ *representing the index set for* q *and for the entries in* $\mathbf{z}$*):*

Rule 1 *If* $\mathbf{z}_i^j = 0$ *for all* $j \in \mathcal{J}$*, then we have the following constraints on* q_i*:* $q_i \geq -M_{i,:}\mathbf{z}^j$, $\forall j \in \mathcal{J}$.

Rule 2 *If* $\mathbf{z}_i^k > 0$ *for some* $k \in \mathcal{K}_i \subseteq \mathcal{J}$*, with* $\mathcal{K}_i$ *representing the index set of* $\mathbf{z}$ *solutions that have a positive* i*th entry, then choose one* $k \in \mathcal{K}_i$ *and the constraint for* q_i *becomes:* $q_i = -M_{i,:}\mathbf{z}^k$.

If there exists a $\mathbf{z}^k \in Z_{obs}$ *such that* $\mathbf{z}_i^k > 0$ *for all* $i \in \mathcal{I}$*, or in other words* $k \in \bigcap_{i \in \mathcal{I}} \mathcal{K}_i \neq \emptyset$*, then* $q = -M\mathbf{z}^k$.

Proof. According to Theorem 8, the fact that the inverse optimization q solution exists implies that there exists a feasible quadratic program (4.3.46). Rule 1 comes from Proposition 8. Rule 2

comes from Proposition 9. The last statement comes from applying Rule 2 for all $i \in \mathcal{I}$.

□

A simple example of using this Theorem 9 is to apply it to quadratic program (4.4.27), which appears as follows after applying Theorem 9:

$$\min_{q \in \mathcal{Q}} (q_1 - q_1^0)^2 + (q_2 - q_2^0)^2 \tag{4.4.37a}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \end{bmatrix} \tag{4.4.37b}$$

Since $\bar{\mathbf{z}}^2 > 0$, we use the last part of Theorem 9 to produce the above quadratic program.

Having proven this theorem, we move into a discussion of when Z_{obs} does not satisfy relationship (4.4.20), meaning that the LCP solutions in Z_{obs} do not have overlapping complementary cones. We use Example 2.2 in Section 4.3.2 to again illustrate this point for this case. If we were to form a $Z_{obs} = \{(2,0),(2,2)\}$ which consists of two $\mathbf{z}$ solutions that do not have corresponding overlapping complementary cones, our quadratic program (4.3.46) does not produce a solution because the feasibility region of the quadratic program looks as follows:

$$-1 + 2q_1 = 0 \quad \text{(from (2,0))} \tag{4.4.38a}$$

$$1 + q_1 + q_2 = 0 \quad \text{(from (2,2))} \tag{4.4.38b}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} 0.5 \\ -1 \end{bmatrix} \quad \text{(from (2,0))} \tag{4.4.38c}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \geq \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \quad \text{(from (2,2))} \tag{4.4.38d}$$

The q solution we obtain from the complementarity constraints (4.4.38a)-(4.4.38b) is $q = [\frac{1}{2} \ \frac{-3}{2}]$ which contradicts the bounds obtained from the equality constraints (4.4.38d) which require $q_2 \geq -0.5$. Thus, relationship (4.4.20) comes into play because having non-overlapping complementary cones implies there is no q solution to the inverse optimization problem.

One question worth asking is in what possible situations we are to encounter a set Z_{obs} with $\mathbf{z}$ solutions that correspond to non-overlapping complementary cones? In the case that we *observe* $\mathbf{z}$ solutions and aim to parameterize a system from these $\mathbf{z}$ solutions, as long as the parameters to be estimated can be assumed to not be changing over the time we observe the $\mathbf{z}$ solutions, then we can be confident that our set Z_{obs} corresponds to a set of overlapping complementary cones. This is because, if the parameters are not changing, then one q is creating the $\mathbf{z}$ solutions and, therefore, there is a q such that $\{q : Z_{obs} \subseteq SOL(q, M)\} \neq \emptyset$ which implies the corresponding complementary cones to Z_{obs} are overlapping according to relationship (4.4.19). However, if the parameters of the system do change over a period of time of observation or if we are trying to *design* (as discussed in [3, 21]) or formulate a system such that it produces a set of $\mathbf{z}$ solutions, then we could run into the issue of proposing a Z_{obs} such that the corresponding complementary cones are not overlapping. For the former concern, if we know when the system changes, we can note the change and subset the $\mathbf{z}$ solutions according to when the system changes. For the latter,

we have to be careful regarding the **z** solutions we utilize for inverse optimization, and we may have to undergo a trial and error process. Luckily, we can use the feasibility of (4.3.46) as a test for our Z_{obs} to see if $\{q : Z \subseteq SOL(q, M)\} \neq \emptyset$.

To close this section, we consider the case when the set Z_{obs} does not have complete information (Definition 19). In the Example 2.2 from Section 4.3.2, if $Z_{obs} = \{(0, 2)\}$, we can see in Figure 4.4 below that this solution corresponds to region $pos\ C_M(\{2\})$, and the q in this region do not all lead to a recovery of just the set Z_{obs}. This is apparent in the first quadrant in which $pos\ C_M(\{2\})$ intersects with two other complementary cones, indicating that in this region, the qs would lead to more solutions than just Z_{obs}.

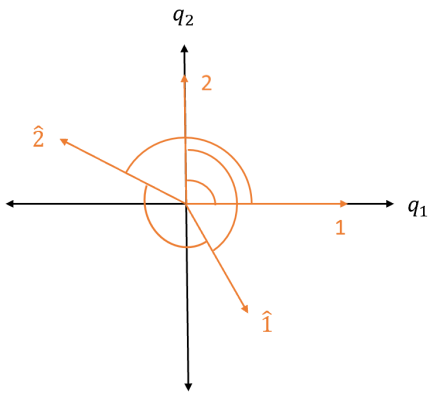


Figure 4.4: Example 2.2 Complementary Cones

Consequently, when we have Z_{obs} and obtain our q, we may desire to perform some searching to see if new solutions to the LCP(q, M) arise. As discussed in Section 4.4.1, we can also check if the resulting q lies within any complementary cones that correspond to solutions we deem undesirable, which would be based upon such criteria as the solutions having positive values where there should be zero values based on application knowledge. If any of these solutions are undesirable, we may have to restrict $\mathcal{Q}$ further or change q^0 to find a new q.

4.4.4 Case 3 Positive Semi-Definite M

As in previous sections, we assume that the set Z_{obs} of LCP $\mathbf{z}$ solutions satisfies $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$, meaning the quadratic program (4.3.46) is feasible and Z_{obs} corresponds to overlapping complementary cones, or i.e. relationship (4.4.19) and Theorem 8 apply. For this section, we assume that M is a positive semi-definite matrix, which means that, if a q produces a set of $\mathbf{z}$ solutions in LCP(q, M), then this set of solutions is a convex set as we see from this chapter's Theorem 7 that we take from [48]. We are particularly interested in the case when this set of $\mathbf{z}$ solutions contains an infinite number of solutions because this could potentially present a problem for the IO quadratic program (4.3.46) in recovering a q solution. Importantly, we assume that the set Z_{obs} contains the entire set of $\mathbf{z}$ solutions that could have been produced by an LCP(q, M). If this convex set of infinite $\mathbf{z}$ solutions contains a $\mathbf{z}$ such that $\mathbf{z} > 0$, then we can use Theorem 9 to reduce the infinite set of $\mathbf{z}$ solutions to this single $\mathbf{z}$ solution that determines the exact values of all the elements of q. This can be seen in Example 3.2 from Section 4.3.2 for $q = [-0.5, \ -1]^T$. If we are given the $Z_{obs} = \{\mathbf{z}_1, \mathbf{z}_2 \geq 0 \text{ s.t. } \mathbf{z}_1 + 2\mathbf{z}_2 = 0.5\}$, we can choose $\mathbf{z}_1 = \frac{1}{4}, \mathbf{z}_2 = \frac{1}{8}$ and we see:

$$q = -M\mathbf{z} = -\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} \frac{1}{4} \\ \frac{1}{8} \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \end{bmatrix} \tag{4.4.39}$$

Thus, we recover the q using this strictly positive $\mathbf{z}$ solutions from the convex set of infinite solutions Z_{obs}. However, as we know from both Proposition 8 and Theorem 9, if there is an entry i for the entire convex set of $\mathbf{z}$ solutions Z_{obs} in which $\mathbf{z}_i = 0$, then we have a problem because we need to express all of the lower bounds that these solutions produce according to Proposition

8 and Theorem 9, which is impossible because then there would be an infinite number of lower bounds. As example of just a few of the bounds generated for $\mathbf{z}_3$ in Example 3.1 from Section 4.3.2, these are the bounds for different $\lambda \in [0, 1]$ values along the line $\lambda\hat{\mathbf{z}}^1 + (1 - \lambda)\hat{\mathbf{z}}^2$

$$\lambda = 0 : \; q_3 \geq -12 \tag{4.4.40a}$$

$$\lambda = 0.25 : \; q_3 \geq -10 \tag{4.4.40b}$$

$$\lambda = 0.5 : \; q_3 \geq -8 \tag{4.4.40c}$$

$$\lambda = 0.75 : \; q_3 \geq -6 \tag{4.4.40d}$$

$$\lambda = 1 : \; q_3 \geq -4 \tag{4.4.40e}$$

Therefore, we need a way to obtain the tightest lower bound on q_i. Luckily, we can construct a relatively simple linear program to find this tightest lower bound. However, in order for this linear program to produce a lower bound that is associated with one of the solutions in the convex set, we must again assume that we have knowledge of the convex set, meaning that we need to have the vertex points that define the set, since we know from Theorem 7 that the convex set is a convex polyhedron [48]. This is a significant assumption. Call φ the number of vertices, and let us denote these vertices using $\mathbf{y}$ such that $\mathbf{y}^h, \forall h = 1, ..., \varphi$ denotes all of the vertices. We

can then construct the linear program as follows for each of the indices i in which $\mathbf{z}_i = 0$ for all $\mathbf{z} \in Z_{obs}$:

$$\max_{\mathbf{z},\lambda} -M_{i,:}\mathbf{z} \tag{4.4.41a}$$

$$\sum_{h=1}^{\varphi} \lambda_h \mathbf{y}^h = \mathbf{z} \tag{4.4.41b}$$

$$\mathbf{z} \geq 0 \tag{4.4.41c}$$

$$\sum_{h=1}^{\varphi} \lambda_h = 1 \tag{4.4.41d}$$

$$\lambda \geq 0 \tag{4.4.41e}$$

To explain this program, we start with the objective function (4.4.41a). We know for the entry i that is 0 across all $\mathbf{z} \in Z_{obs}$ that the lower bounds will be of the form $q_i \geq -M_{i,:}\mathbf{z}$. Consequently, since we want the tightest lower bound, we maximize the right hand side of that inequality. The first constraint (4.4.41b) ensures the $\mathbf{z}$ chosen is part of the convex set of solutions. Constraint (4.4.41c) is the nonnegativity constraint for $\mathbf{z}$. Finally, constraints (4.4.41d) and (4.4.41e) ensure that the λ vector sums to 1 and is greater than or equal to 0.

We can once again use Example 3.1 to form this linear program using the $\hat{\mathbf{z}}^1$ and $\hat{\mathbf{z}}^2$ vertex points from that example. The linear program for $i = 3$, since the third entry for the $\hat{\mathbf{z}}^1$ and $\hat{\mathbf{z}}^2$

vertex points is 0, is as follows:

$$\max_{\mathbf{z},\lambda} -\begin{bmatrix} 4 \\ 1 \\ 0 \\ 0 \end{bmatrix}^T \mathbf{z} \tag{4.4.42a}$$

$$\lambda_1 \hat{\mathbf{z}}^1 + \lambda_2 \hat{\mathbf{z}}^2 = \mathbf{z} \tag{4.4.42b}$$

$$\lambda_1 + \lambda_2 = 1 \tag{4.4.42c}$$

$$\lambda_1, \lambda_2 \geq 0 \tag{4.4.42d}$$

We note that this method does not only work for linear programs; it works for any LCP that has a positive semi-definite M matrix with a convex set of solutions in Z_{obs}.

Once we have the $\mathbf{z}$ we need for each relevant entry i, which we denote as $\tilde{\mathbf{z}}$, we can form the simplified version of quadratic program (4.3.46) using a few rules similar to Theorem 9, which we formalize in the following corollary.

Corollary 3 (Simplified Quadratic Program (4.3.46) for Convex Polyhedron Z_{obs})**.** *Given* $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ *or, in other words, the IO problem has a* q *solution, the associated quadratic program (4.3.46) feasible region can be simplified for a set of* $\mathbf{z}$ *solutions* Z_{obs} *that (a) is a convex polyhedron of* $\mathbf{z}$ *solutions and that (b) includes all the vertices of the convex polyhedron according to the following rules for all* $i \in \mathcal{I}$ *(with* $\mathcal{I}$ *representing the index set for* q *and for the*

entries in $\mathbf{z}$*):*

Rule 1 *If* $\mathbf{z}_i = 0$ *for all* $\mathbf{z} \in Z_{obs}$*, solve linear program (4.4.41) to find the* $\tilde{\mathbf{z}}$ *that maximizes the objective function. The constraint for* q_i *is then* $q_i \geq -M_{i,:}\tilde{\mathbf{z}}$*.*

Rule 2 *If* $\mathbf{z}_i > 0$ *for some* $\mathbf{z} \in Z_{obs}$*, then the constraint for* q_i *becomes:* $q_i = -M_{i,:}\mathbf{z}$*.*

Proof. We prove this corollary by addressing the two rules it codifies. For the first rule, if $\mathbf{z}_i = 0$ for all $\mathbf{z} \in Z_{obs}$, then the linear program (4.4.41) provides the tightest lower bound for q_i because (a) the objective function maximizes this lower bound and (b) constraint (4.4.41b) ensures that the $\tilde{\mathbf{z}}^i$ chosen is a member of the convex set of solutions defined by Z_{obs}. For the second rule, this is just a restatement of the second rule of Theorem 9, so we have already proven this.

□

A reader may wonder at this point about what happens if Z_{obs} does not have all of the solutions, meaning it would not have all of the vertices that make up the convex polyhedron. This could mean that we have a subset of the vertices which, in that case, we can solve the linear program (4.4.41) for this subset. We may also have a set of $\mathbf{z}$ solutions from which we need to construct the convex hull in order to use the linear program (4.4.41) to find $\tilde{\mathbf{z}}$. We refer readers to the following recent papers that discuss efficient methods for generating convex hulls: [12, 110]. If we have just one of the vertices, then we advise solving the quadratic program (4.3.46) as if it has a finite set Z_{obs}. Indeed, this vertex may indeed be the only solution to the LCP because, even though M is positive semi-definite, this does not guarantee an infinite convex set of solutions; see Example 3.2 in Section 4.3.2.

As in Section 4.4.3, if we do not have complete information in Z_{obs}, we may need to do some searching techniques to see what other solutions we obtain for a given q. With regard to the

assumption of overlapping complementary cones, we refer readers to the discussion at the end of Section 4.4.3. To close this section, we note that, often, we likely are to encounter a Z_{obs} that does not have all of the vertices of a convex infinite set. In particular, in a situation in which we observe solutions, we may only obtain one observation; for example, in traffic, we only see one set of flows for a given state of the system.

4.4.5 Case 4 Unknown M Matrix Structure

In this section, we once again assume that the set Z_{obs} of LCP $\mathbf{z}$ solutions satisfies $\{q^{Z_{obs}} : Z_{obs} \subseteq SOL(q^{Z_{obs}}, M)\} \neq \emptyset$ meaning the quadratic program (4.3.46) is feasible and Z_{obs} corresponds to overlapping complementary cones, or i.e., relationship (4.4.19) and Theorem 8 hold true. We are *not* going to assume complete information in this section. The main point of this section is to handle cases when we either know M does not satisfy any of the previous three cases or M is too large to check for some or all of those properties.

We proved most of the propositions, theorems, and corollaries from the previous three subsections without using any specific properties of M. We mainly used the previous properties of M to help guide us in thinking about the types of solutions structures we might find in Z_{obs}. In this section, Z_{obs} can be structured in a variety of ways that include similar structures that we found before as well as combinations of these structures. For instance, in Table 4.5 for Example 4.1 in Section 4.3.2, we see that, for $q = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^T$, the set of solutions includes both a convex set ($0 < \mathbf{z}_1 \leq 0.5, \mathbf{z}_2 = 0$) of solutions and a isolated solution ($\mathbf{z}_1 = 0, \mathbf{z}_2 = 1$) whereas, for $q = \begin{bmatrix} -0.25 & 0.5 \end{bmatrix}^T$ and $q = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^T$, the solution sets are comprised of a finite number of solutions. For Example 4.2 in Section 4.3.2, the solution set for that problem is also composed of

a convex set of solutions as well as some isolated solutions. Finally, in Example 4.3 from Section 4.3.2, the solution set is a union of convex sets that is not itself a convex set. Therefore, we need to use knowledge gained from the previous sections to parameterize an LCP with an M that does not have any of the previous properties, which we outline in the paragraphs below.

First, we note that, if any of the solutions in Z_{obs} are strictly positive for all entries, then we can apply the result from Theorem 9 and Corollary 3 to say that this strictly positive $\mathbf{z}$ solution can exactly produce the q. Furthermore, if a group of solutions from Z_{obs} can be identified such that their positive entries cover the entire index set of $\mathcal{I}$ for q, then we can apply the result from Theorem 9 and Corollary 3 to obtain a q. However, if there is an entry i or multiple entries for which $\mathbf{z}_i = 0$ for all $\mathbf{z} \in Z_{obs}$, then we have to consider two sub-cases:

- Sub-case 1: If Z_{obs} has a finite set of solutions, we use Theorem 9 and produce a simplified version of quadratic program (4.3.46).

- Sub-case 2: If Z_{obs} has as part of it a convex set of infinite solutions or multiple convex sets of solutions, then we need to solve a modified version of the linear program (4.4.41) for each entry i. The modified version looks as follows:

$$\max_{\mathbf{z}} -M_{i,:}\mathbf{z} \tag{4.4.43a}$$

$$\mathbf{z} \in Z_{obs} \tag{4.4.43b}$$

The most challenging part is constraint (4.4.43b) because it could be difficult to express this depending on the form of Z_{obs}. It might require disjunctive programming, especially

if Z_{obs} is a union of multiple convex sets or multiple convex sets and isolated points. We would need to set up a disjunctive constraint such that the $\tilde{\mathbf{z}}$ comes from one of the subsets that comprises Z_{obs}.

4.5 Case Study: Water Supply Market

Having explored the theoretical nature of parameterizing a LCP, we turn to our case study in which we apply our knowledge to a water supply market. We demonstrate that we can use inverse optimization for the purposes of adjusting a market toward the system optimal.

4.5.1 Water Supply Background

As illustrated by Boyd et al. [25], water users on a river can be modeled as competing non-cooperatively on a line graph, which is a "linear ordering on a set of players" [181], to obtain the necessary water to satisfy the demand of their regions. However, Boyd et al. [25] also demonstrate that these water resources can be facilitated among players in the game through markets, and the researchers present two types of markets in their paper: a general commodity market (GCM) and a cost-sharing market (CSM). The central difference between these two markets is the delineation of who has a right to use the water as it flows downstream. In the GCM, downstream users specifically purchase certain quantities of water from upstream users that any intermediate players are not allowed to utilize; the pricing, as a result, occurs at the supplier nodes, where they receive one price for their water resources. In the CSM, downstream users simply demand a certain amount of water from upstream users, so the pricing occurs at the downstream user nodes. As Boyd et al. [25] point out, this allows the water resource to be utilized

by multiple users. These users would include municipal, local, and city governments, essentially entities that withdraw water and make recovery decisions about the water. We note that the water that is available to be purchased comes from consumptive-loss reductions, such as fixing leaking pipes and rerouting water to sewers, which upstream players implement in order to have water to sell to downstream players [25]. The recovered water is returned to the original source through wastewater networks [25]. The water recovered from consumptive-loss reductions is then sold from upstream players to downstream players.

These market problems are interesting from an inverse optimization standpoint because we can actually form system optimal versions of the problems [28]. By a system optimal $\mathbf{z}$ solution, we mean a solution that would be chosen if a centralized figure were maximizing the total benefits across all of the players, making decisions about how much water each player should withdraw and recover [28]. It would be as if all the players were working as one player to maximize total benefit summed across all players [28, 47, 113]. Then, these system optimal $\mathbf{z}$ solutions can be used in inverse optimization formulations of the original market problems to tell us what parameter values in the market problems we would need to change to achieve these system optimal $\mathbf{z}$ solutions or, in other words, the new taxes or subsidies that would need to be implemented to adjust the market to these system optimal $\mathbf{z}$ solutions. This is the role of inverse optimization in *influencing* models that we discuss in Section 4.1, that is the adjusting of incentives to achieve a system optimal $\mathbf{z}$ solution. A system optimal $\mathbf{z}$ solution does not naturally occur in the GCM and CSM markets because, in a market, each player is attempting to maximize their own benefit in reaction to the actions of other players [75]; more specifically, in this application, the upstream players do not consider the harm to downstream players of their withdrawals, which does allow the system optimal solution to occur. To learn more about

influencing models via system optimal solutions, we direct readers to Section 4.5.2.2.

However, our first task is to use inverse optimization to take *observed* $\mathbf{z}$ solutions from the market models and, then, recover the qs for the LCP formulations of both market models. In this test case, observed $\mathbf{z}$ solutions come from simulating/solving the market models and then seeing the q that are returned from the IO quadratic program (4.3.46) solve. The q in this application contains the per unit costs of water extraction plus components of demand, the per unit costs of water recovery efforts, as well as the water inflows and flow constraints.

The full formulation details can be found in [25]. We provide an abridged version of the formulation details for each market, including only the variables that appear in the toy problem we modify from [25] as well as only the related constraints. Specifically, the major element missing is the capital improvements portion of the model, which readers can refer to [25] for more information. We do note that, unlike the toy model in [25], we only use one time period in this toy model. The following sets, variables, and parameters relate to the models, and the descriptions are taken from [25]:

Sets: These are the sets for the GCM and CSM models.

- $i, j, k \in \{I\}$ - indexed users of the river numbered from upstream to downstream
- $U_i \subset I$ - upstream nodes of i, where $j \in U_i$ is a typical node index
- $D_i \subset I$ - downstream nodes of i, where $k \in D_i$ is a typical node index
- $c \in \{C\}$ - classes of water loss reductions in ascending order of expense. Examples of these classes of water loss reductions include fixing leaking pipes and rerouting water to sewers.

Primal Variables: These are the primal variables for the GCM and CSM models. If a variable is specifically for one model or the other, this is noted. Descriptions are taken from [25]. The sets ζ_i^{GCM} and ζ_i^{CSM} refer to the set of primal variables for each of the two models for player i.

- W_i^D - player i's direct water withdrawal from the river (volume/day). This is composed of water purchased plus free flow water. Thus, to obtain free flow water, one subtracts water purchased from water withdrawn. Water purchases for the two markets are defined in the coming definitions.

- $L_{i,c}^R$ - player i's incremental water loss reductions in class c (volume/day)

- W_i^P - player i's water purchases from upstream in the CSM formulation to reduce asymmetric access to water (volume/day). This ability of downstream users to purchase water from upstream users "reduces asymmetric access" because it incentivizes upstream users to engage in consumptive-loss reductions and thus make water available for downstream users to purchase.

- $W_{i,j}^P$ - player i's purchases from an upstream player j (volume/day). This is relevant for the GCM model. This ability of downstream users to purchase water from upstream users "reduces asymmetric access" because it incentivizes upstream users to engage in consumptive-loss reductions and thus make water available for downstream users to purchase.

- $W_{k,i}^P$ - water sales to player k downstream from i (volume/day). This is relevant for the GCM model.

Dual Variables: These are the dual variables are they are defined in the same conventions as the primal variables.

- $\gamma_{i,c}^{loss}$ - non-negative shadow price for loss reductions (cost/unit flow)
- γ_{i}^{flow} - non-negative shadow price for withdrawal limitations (cost/unit flow)
- π_i - non-negative user i's price for water (cost/unit flow). The relevant i depends upon which market model is being used (upstream for GCM and downstream for CSM).

Parameters: These are the relevant parameter values taken directly from [25].

- c_i^{ops} - player i's unit operating costs to withdraw water from the river (cost/unit flow)
- $c_{i,c}^{cu}$ - player i's unit costs for consumptive-loss reductions in class c (cost/unit flow)
- $\delta_{ds_{k,i}}^{all} \in \{0,1\}$ - logical parameter specifying if player k is downstream of player i
- $\delta_{us_{j,i}}^{all} \in \{0,1\}$ - logical parameter specifying if player j is upstream of user i
- $lf_{c,i}$ - player i's estimated fractions of water losses in class c (%)
- n_i - local water inflow at player i independent of upstream water releases (e.g., such as from side streams) (volume/day)
- r_i^{fc} - player i's regulator imposed flow constraint (volume/day). This is the flow rate of water a given player is legally obligated to leave in the river.
- α_i - inverse water demand intercept for player i (cost/unit flow)
- β_i - inverse water demand linear slope for player i (cost/unit flow)

There are three players in the toy example we modify from [25], and our formulations of both the ith player optimization problems as well as the LCP conditions below incorporate the

presence of three players into the formulations through the limits on the various sums in the formulations. However, to make the problem more concrete, this is the line graph diagram for the three players:

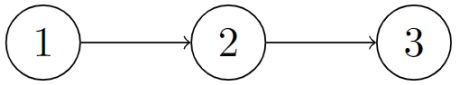


Figure 4.5: Three Node Line Graph Flow

The diagram represents the flow of water as it moves downstream from player to player, left to right. The player i optimization models and the LCPs for the GCM and CSM markets encode the sequential relationship shown in the diagram. To make the models more concrete, the parameters values for the three-player example for the GCM and CSM market models are as follows:

Parameter	Player(s)	Class(es)	Value
β_i	1,2,3	-	3, 3, 3
α_i	1,2,3	-	21, 31, 41
c_i^{ops}	1,2,3	-	1, 1, 1
n_i	1,2,3	-	14, 0, 0
r_i^{fc}	1,2,3	-	4, 4, 4
$lf_{1,i}$	1,2,3	1	0.13, 0.1, 0.07
$lf_{2,i}$	1,2,3	2	0.13, 0.1, 0.07
$c_{1,i}^{cu}$	1,2,3	1	0.67, 1.00, 1.33
$c_{2,i}^{cu}$	1,2,3	2	3.33, 5.00, 6.67

Table 4.6: Parameter Values for Three Node Model

These parameters differ somewhat from the toy model in [25], and they are synthetic data.

General Commodity Model: There are three players in this case study, but we provide the general formulation for a player i.

$$\max_{\zeta_i^{GCM}} \int_0^{W_i^D} (\alpha_i - \beta_i x_i) dx_i + \pi_i \sum_{c=1}^{2} L_{c,i}^R - \left(c_i^{ops} W_i^D + \sum_{c=1}^{2} c_{c,i}^{cu} L_{c,i}^R + \sum_{j=1}^{3} \pi_j \delta_{us_{j,i}}^{all} W_{i,j}^P \right) \quad (4.5.1a)$$

$$L_{c,i}^R \leq lf_{c,i} W_i^D, \quad \forall c \quad (\gamma_{c,i}^{loss}) \quad (4.5.1b)$$

$$W_i^D \leq n_i + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} W_{i,j}^P - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right), \quad (\gamma_i^{flow}) \quad (4.5.1c)$$

$$W_i^D, W_i^P \geq 0, \quad L_{c,i}^R \geq 0, \quad \forall c \quad (4.5.1d)$$

The objective function is the net benefit and, thus, it is the sum of benefits which include consumer surplus from meeting water demand plus payments for water recovery efforts (with the price at the supplier node) minus the costs which are the cost of withdrawing water, the cost of water recovery efforts, and the cost of water purchasing efforts. The first constraint requires the flow of recovered water to be less than or equal to the amount lost. The second constraint is a flow-balance constraint that ensures that the amount of water withdrawn leaves enough water above a certain threshold r_i^{fc}. The n_i is the water from side streams, the term involving $W_{i,j}^P$ is the water purchased from upstream players, and the last term is the net flow from upstream players that comes naturally to player i. We can form a LCP for this market because the KKT

conditions are necessary and sufficient for the players in this market. The KKT conditions are necessary because, in each player's problem, the constraints are linear, thus satisfying a constraint qualification [19, 26]. In each players problem, we are also maximizing a concave function subject to a linear feasible region, which means we have a convex program, thus making the conditions sufficient [19, 26]. Furthermore, according to Kim and Ferris [113], even though this problem is a generalized Nash equilibrium problem due to constraint (4.5.1c), we can still concatenate the KKT conditions and form a complementarity problem since we satisfy Slater's condition and convexity assumptions and produce solutions to the GNEP. Thus, we can concatenate the KKT conditions from each player along with the market-clearing conditions to form an LCP [75]. The objective function for the players has to be negated in order to form a minimization problem for the KKT conditions and for the concatenation of the KKT conditions along with market-clearing conditions [75]. We also have two classes of improvements, hence the summation index c going from 1 to 2 in the summation over class in the optimization problem above and the LCP below. Furthermore, we have three players, hence the summation indices k and j going from 1 to 3 in the summation over players in the optimization problem above and the LCP below. The LCP for this market is as follows for player i:

$$0 \leq \beta_i W_i^D - \alpha_i + c_i^{ops} - \sum_{c=1}^{2} \gamma_{c,i}^{loss} lf_{c,i} + \gamma_i^{flow} \perp W_i^D \geq 0 \tag{4.5.2a}$$

$$0 \leq \delta_{us_{j,i}}^{all} (\pi_j - \gamma_i^{flow}) \perp W_{i,j}^P \geq 0, \quad j \neq i \tag{4.5.2b}$$

$$0 \leq -\pi_i + c_{c,i}^{cu} + \gamma_{c,i}^{loss} \perp L_{c,i}^R \geq 0, \quad \forall c \tag{4.5.2c}$$

$$0 \leq \sum_{c=1}^{2} L_{c,i}^{R} - \sum_{k=1}^{3} \delta_{ds_{k,i}}^{all} W_{k,i}^{P} \perp \pi_i \geq 0 \tag{4.5.2d}$$

$$0 \leq lf_{c,i} W_i^D - L_{c,i}^R \perp \gamma_{c,i}^{loss} \geq 0, \quad \forall c \tag{4.5.2e}$$

$$0 \leq n_i + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} W_{i,j}^{P} - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right) - W_i^D \perp \gamma_i^{flow} \geq 0 \tag{4.5.2f}$$

The first three sets of complementarity constraints involve the KKT stationarity conditions for the three sets of variables. For the first of the stationarity conditions associated with W_i^D, the $c_i^{ops} + \gamma_i^{flow}$ portion represents the marginal costs, with c_i^{ops} as the cost for extracting the water and γ_i^{flow} as the value or opportunity cost of the water. The marginal benefit terms are $\beta_i W_i^D - \alpha_i - \sum_{c=1}^{2} \gamma_{c,i}^{loss} lf_{c,i}$, with $\beta_i W_i^D - \alpha_i$ as the marginal consumer surplus and $-\sum_{c=1}^{2} \gamma_{c,i}^{loss} lf_{c,i}$ as the profit obtained from water recovery efforts, since we have the opportunity cost of the recovered water multiplied by the loss fraction (which represents the water that could be recovered). In the second stationarity constraint associated with $W_{i,j}^P$, we have the marginal cost of water purchased as π_j, and the marginal benefit as the value of the additional water γ_i^{flow}. For the third stationarity constraint associated with $L_{c,i}^R$, we have the marginal cost of water recovery in extraction cost $c_{c,i}^{cu}$ and in $\gamma_{c,i}^{loss}$ as the opportunity cost/value of the recovered water, and we have the marginal benefit of water recovery in the price paid for the water recovered π_i. The next complementarity relationship is the market-clearing condition, which says that the supply from player i of water to be sold in the form of recovery efforts must be greater than or equal to the demand from

downstream players for water; this must have a complementary relationship with the price at player i's node that player i receives for the water. The final two sets of complementarity constraints are the feasibility conditions, which require a complementarity relationship between the inequality constraints and their multipliers [19, 26].

Cost-Sharing Model: There are three players, but we provide the general formulation for a player i.

$$\max_{\zeta_i^{CSM}} \int_0^{W_i^D} (\alpha_i - \beta_i x_i) dx_i + \sum_{k=1}^{3} \pi_k \delta_{ds_{k,i}}^{all} \sum_{c=1}^{2} L_{i,c}^R - \left(c_i^{ops} W_i^D + \sum_{c=1}^{2} c_{i,c}^{cu} L_{i,c}^R + \pi_i W_i^P \right) \quad (4.5.3a)$$

$$L_{c,i}^R \leq lf_{c,i} W_i^D, \quad \forall c \quad (\gamma_{i,c}^{loss}) \quad (4.5.3b)$$

$$W_i^D \leq n_i + W_i^P - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right), \quad (\gamma_i^{flow}) \quad (4.5.3c)$$

$$W_i^D, W_i^P \geq 0, \quad L_{i,c}^R \geq 0, \quad \forall c \quad (4.5.3d)$$

The objective function illustrates the net benefit and, thus, it is the benefits minus the costs in the form of consumer surplus gained by meeting demand plus the payments by other players from water recovery efforts minus the cost of water withdrawn, the cost to make water recovery improvements, and price of water purchases from upstream users. The first constraint defines the amount of water that can be recovered; it is bounded by the loss factor multiplied by the amount

of water withdrawn. The second constraint has to do with accounting for how much water can be taken out of the river; it is very similar to the related constraint in the GCM model. The discussion above from the GCM market regarding the lead up to the LCP applies here for the CSM LCP as well including the remarks about the KKT conditions, the concatenation, and the limits on the summations. The LCP for this market is as follows for player i:

$$0 \leq \beta_i W_i^D - \alpha_i + c_i^{ops} - \sum_{c=1}^{2} lf_{c,i}\gamma_{c,i}^{loss} + \gamma_i^{flow} \perp W_i^D \geq 0 \tag{4.5.4a}$$

$$0 \leq \pi_i - \gamma_i^{flow} \perp W_i^P \geq 0 \tag{4.5.4b}$$

$$0 \leq c_{c,i}^{cu} - \sum_{k=1}^{3} \pi_k \delta_{ds_{k,i}}^{all} + \gamma_{c,i}^{loss} \perp L_{c,i}^R \geq 0, \quad \forall c \tag{4.5.4c}$$

$$0 \leq \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \sum_{c=1}^{2} L_{c,j}^R - W_i^P \perp \pi_i \geq 0 \tag{4.5.4d}$$

$$0 \leq lf_{c,i} W_i^D - L_{c,i}^R \perp \gamma_{c,i}^{loss} \geq 0, \quad \forall c \tag{4.5.4e}$$

$$0 \leq n_i + W_i^P - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right) - W_i^D \perp \gamma_i^{flow} \geq 0 \tag{4.5.4f}$$

The first three complementarity constraints correspond with the KKT stationarity conditions for those variables. The same analysis holds for these conditions as in the GCM model except for in the third stationarity condition in which the marginal benefit is a summation of downstream

prices for the recovered water. The constraint after the first three is the market-clearing constraint in which the supply of water from upstream users must be greater than the demand from user i. The price, unlike the previous model, is at the demand player's node, and it is the price paid to upstream players for their water. The final two complementarity constraints refer to feasibility.

4.5.2 Water Supply Inverse Optimization

Section 4.5.2.1 focuses on observed $\mathbf{z}$ solutions to the GCM and CSM market models, and Section 4.5.2.2 focuses on using system optimal $\mathbf{z}$ solutions to influence the GCM and CSM market models. Both the observed and system optimal $\mathbf{z}$ solution defintions can be found in the beginning of Section 4.5.1.

4.5.2.1 Observed $\mathbf{z}$ Solutions Inverse Optimization

Our first set of inverse optimization experiments focuses on recovering the q back from LCP solution(s) to each of the two GCM and CSM market models. This set of experiments is essentially a check on the inverse optimization method. As far as we can tell, neither of the two models' M matrices demonstrate properties that would allow us to say anything about the $\mathbf{z}$ solution structure of the resulting LCP. For the GCM model, the principle minors range from -0.08070000000000006 to 27 and the values do include 0, thus the matrix is degenerate and the negative values plus the 0s mean the matrix is not part of the class $\mathbf{P}$. The algorithm we used to find the principle minors can be found in the Appendix C.1. What's more, we know from Cottle et al. [48] that all positive semi-definite matrices are members of the class of matrices $\mathbf{P_0}$, which means their principle minors are nonnegative. Thus, since we have negative principle

minors, M is also not positive semi-definite. This is further confirmed by the fact that the eigenvalues of $\frac{1}{2}(M^{GCM} + M^{GCM^T})$ range from -1.31173554e-02 to 3.01311736 and include 0, again meaning the matrix is not positive semi-definite (see page 66 of [48]). For the CSM matrix, the principle minors have the same value range as the GCM matrix, and the eigenvalues of $\frac{1}{2}(M^{CSM} + M^{CSM^T})$ are the same as for $\frac{1}{2}(M^{GCM} + M^{GCM^T})$. Consequently, we need to use the material from Section 4.4.5 to solve the inverse optimization problem.

For the first experiment of this section, we find one of potentially multiple $\mathbf{z}$ by using the PATH solver [54, 70] in GAMS with a convergence tolerance of $1e{-}12$. These are the $\mathbf{z}$ solutions, with $\mathbf{z}^{GCM}$ representing the solution we find for the GCM market and $\mathbf{z}^{CSM}$ representing the solution we find for the CSM market. We find these $\mathbf{z}$ solutions by using a multi-point starting method in which we construct a scheme of 2024 random starting points, with the first 1000 points being random vectors of 0s and 10s, the next 1000 points being random vectors of numbers between 0 and 10, and the last 24 points being the coordinate axes multipled by 10. We choose to scale vectors by 10 because this is the largest number that is present in the initial solutions obtained from solving the LCPs with an all 0s starting point. We check that the solutions we obtain from the random starting point scheme match the solution we obtain from the all 0s starting point for both the GCM and CSM markets. Thus, we have one solution to present in Table 4.7 below.

Variable	Explanation	$\mathbf{z}^{GCM}$	$\mathbf{z}^{CSM}$
W_1^D	water withdrawal (volume/day), player 1	7.36	7.4813
W_2^D	water withdrawal (volume/day), player 2	8.0864	10.0
W_3^D	water withdrawal (volume/day), player 3	10.0	10.0
$W_{2,1}^P$ or W_1^P	water purchases (volume/day)	0.0	0.0
$W_{3,1}^P$ or W_2^P	water purchases (volume/day)	1.9136	1.9451
$W_{3,2}^P$ or W_3^P	water purchases (volume/day)	1.6173	3.9451
$L_{1,1}^R$	loss reductions (volume/day), class 1, player 1	0.9568	0.9726
$L_{1,2}^R$	loss reductions (volume/day), class 1, player 2	0.8086	1.0
$L_{1,3}^R$	loss reductions (volume/day), class 1, player 3	0.0	0.0
$L_{2,1}^R$	loss reductions (volume/day), class 2, player 1	0.9568	0.9726
$L_{2,2}^R$	loss reductions (volume/day), class 2, player 2	0.8086	1.0
$L_{2,3}^R$	loss reductions (volume/day), class 2, player 3	0.0	0.0
π_1	price for reductions (cost/unit flow)	9.9999	0.0
π_2	price for reductions (cost/unit flow)	9.9999	1.4
π_3	price for reductions (cost/unit flow)	0.0	10.0
$\gamma_{1,1}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 1	9.3332	10.7332
$\gamma_{1,2}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 2	8.9999	9.0
$\gamma_{1,3}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 3	0.0	0.0
$\gamma_{2,1}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 1	6.6666	8.0666
$\gamma_{2,2}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 2	4.9999	4.9999
$\gamma_{2,3}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 3	0.0	0.0
γ_1^{flow}	value of additional water to player 1 (cost/unit flow)	0.0	0.0
γ_2^{flow}	value of additional water to player 2 (cost/unit flow)	7.1408	1.4
γ_3^{flow}	value of additional water to player 3 (cost/unit flow)	9.9999	9.9999

Table 4.7: PATH Solver [54, 70] Generated Market Solutions

We see that the two markets do indeed produce different outcomes for the players. Players 1 and 2 withdraw less water in the GCM market compared to the CSM market. Player 3 purchases more water in the CSM compared to the GCM market. What's more, π_3 is 0 in the GCM market because player 3 has no downstream players to whom they can sell water, and the π_1 is 0 in the CSM market because player 1 has no upstream players from whom they can buy water.

Since we have one solution for each of the markets, this situation fits into the finite solution case from Section 4.4.5, and we take the $\mathbf{z}$ solution we have and construct the quadratic program (4.3.46). Our q^0 is a vector of 0s, the reasoning for which is that it aids us in choosing a q that has 0s in the places in q that we expect. The resulting q^{GCMIO}, which is the IO q solution using $Z_{obs} = \{\mathbf{z}^{GCM}\}$, and q^{CSMIO}, which is the IO q solution using $Z_{obs} = \{\mathbf{z}^{CSM}\}$, are in Table 4.8. The table also includes, for the sake of comparison, the original q used to generate the $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ solutions, and this q is denoted $q^{GCM} = q^{CSM}$ in order to express that the same q is used in both markets to produce the $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ solutions for the sake of comparison between the markets. The row names denote the coefficients to which the q entries are associated. The 0s in certain row names indicate that there are 0 coefficient values for those respective equations:

Coefficient	Explanation	$q^{GCM} = q^{CSM}$	q^{GCMIO}	q^{CSMIO}
$-\alpha_1 + c_1^{ops}$	negative demand intercept plus cost of water consumption (cost/unit flow), player 1	-20.0	-20.0	-20.0
$-\alpha_2 + c_2^{ops}$	negative demand intercept plus cost of water consumption (cost/unit flow), player 2	-30.0	-30.0	-30.0
$-\alpha_3 + c_3^{ops}$	negative demand intercept plus cost of water consumption (cost/unit flow), player 3	-39.9999	-39.9999	-39.9999
0	water purchase equilibrium (cost/unit flow)	0.0	0.0	0.0
0	water purchase equilibrium (cost/unit flow)	0.0	0.0	0.0
0	water purchase equilibrium (cost/unit flow)	0.0	0.0	0.0
$c_{1,1}^{cu}$	cost of loss reductions (cost/unit flow), class 1, player 1	0.6667	0.6667	0.6667
$c_{1,2}^{cu}$	cost of loss reductions (cost/unit flow), class 1, player 2	1.0	1.0	1.0
$c_{1,3}^{cu}$	cost of loss reductions (cost/unit flow), class 1, player 3	1.3333	0.0	0.0
$c_{2,1}^{cu}$	cost of loss reductions (cost/unit flow), class 2, player 1	3.3333	3.3333	3.3333
$c_{2,2}^{cu}$	cost of loss reductions (cost/unit flow), class 2, player 2	5.0	5.0	5.0
$c_{2,3}^{cu}$	cost of loss reductions (cost/unit flow), class 2, player 3	6.6667	0.0	0.0
0	market clearing (volume/day)	0.0	0.0	0.0
0	market clearing (volume/day)	0.0	0.0	0.0
0	market clearing (volume/day)	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 1, player 1	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 1, player 2	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 1, player 3	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 2, player 1	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 2, player 2	0.0	0.0	0.0
0	feasibility loss reductions (cost/unit flow), class 2, player 3	0.0	0.0	0.0
$n_1 - r_1^{fc}$	flow constraint (volume/day), player 1	10.0	7.36	7.4813
$n_1 + n_2 - r_2^{fc}$	flow constraint (volume/day), player 2	10.0	10.0	10.0
$n_1 + n_2 + n_3 - r_3^{fc}$	flow constraint (volume/day), player 3	10.0	10.0	10.0

Table 4.8: q from Observed $\mathbf{z}$ Table

We know from Proposition 9 that any positive entries i in $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ fully determine the associated individual values for q_i in the IO q^{GCMIO} and q^{CSMIO} solutions. In contrast, as we

know from Proposition 8, when $\mathbf{z}_i = 0$ for $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$, the associated q_i entry can *in theory* take on a range of values, with a lower bound established by $q_i \geq -M_{i,:}\mathbf{z}$. We identify the q_i below in (4.5.5) that have lower bounds for the $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ inputs to the IO quadratic program (4.3.46). The relevant q_i are numbered as well as listed as equal to their respective coefficient value from Table 4.8. The lower bounds for each entry i are obtained by calculating $-M^{GCM}_{i,:}\mathbf{z}^{GCM}$ for the GCM market and $-M^{CSM}_{i,:}\mathbf{z}^{CSM}$ for the CSM market.

$$\begin{bmatrix} q_4^{GCMIO} = 0 \\ q_9^{GCMIO} = c_{1,3}^{cu} \\ q_{12}^{GCMIO} = c_{2,3}^{cu} \\ q_{15}^{GCMIO} = 0 \\ q_{18}^{GCMIO} = 0 \\ q_{21}^{GCMIO} = 0 \\ q_{22}^{GCMIO} = n_1 - r_1^{fc} \end{bmatrix} \geq \begin{bmatrix} -2.8591 \\ 0 \\ 0 \\ 0 \\ -0.7 \\ -0.7 \\ 7.3600 \end{bmatrix}, \quad \begin{bmatrix} q_4^{CSMIO} = 0 \\ q_9^{CSMIO} = c_{1,3}^{cu} \\ q_{12}^{CSMIO} = c_{2,3}^{cu} \\ q_{13}^{CSMIO} = 0 \\ q_{18}^{CSMIO} = 0 \\ q_{21}^{CSMIO} = 0 \\ q_{22}^{CSMIO} = n_1 - r_1^{fc} \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -0.7 \\ -0.7 \\ 7.4813 \end{bmatrix} \tag{4.5.5}$$

As we see in this application, there are certain q_i coefficient values we expect to be 0 (such as q_4^{GCMIO} and q_4^{CSMIO} which correspond to the water purchase equilibrium for $W_{2,1}^P$ for GCM and W_1^P for CSM respectively) due to the specification of the GCM and CSM models, even if they technically could take on different values based on the theoretical inverse optimization analysis. Interestingly, if these q_i values were to turn out not to be zero, this might open up the idea of introducing new parameters to the IO model that would represent different incentives that could influence the market toward a $\mathbf{z}$ solution. The $q = q^{GCM} = q^{CSM}$, which produced $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$, matches the q^{GCMIO} and q^{CSMIO} in the i entries where $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ have positive entries. The $q = q^{GCM} = q^{CSM}$ also satisfies the lower bounds from (4.5.5) for the i entries where $\mathbf{z}^{GCM}$ and $\mathbf{z}^{CSM}$ have zero entries.

Feeding the q^{GCMIO} and q^{CSMIO} back into the market models to verify that they produce

the $\mathbf{z}$ with which we started, we find that, for the GCM market, we obtain back the same $\mathbf{z}^{GCM}$ solution. For the CSM market, we just about get back the $\mathbf{z}^{CSM}$ solution, with the exception of the 13th and 22nd entries which are $0.9992656e-4$ instead of 0. We credit this to numerical precision error.

4.5.2.2 Using System Optimal $\mathbf{z}$ Solutions to Influence GCM and CSM Market Models via Inverse Optimization

For the inverse optimization experiment in this section, we find the GCM and CSM *system* $\mathbf{z}$ solution(s) and use them to parameterize the GCM and CSM market models. To find these system optimal solutions, we draw on the methods of Britz et al. [28] to form what these researchers call an "aggregate optimization" problem in which we combine all of the players' problems into one problem. The constraints from all of the players are brought into the same optimization problem, and the objectives from all of the players are summed together. Summing these objectives together pools the benefit such that the goal becomes to maximize total, communal benefit, which is the view of a social planner. Indeed, the aggregate problem itself can be looked at as a single-player game with market-clearing constraints, as opposed to the multi-player game with market clearing constraints that comprise the GCM and CSM markets [47, 113]. The aggregate GCM model looks as follows:

$$\max_{\zeta_i^{GCM}} \sum_{i=1}^{3} \left(\int_0^{W_i^D} (\alpha_i - \beta_i x_i) dx_i + \pi_i \sum_{c=1}^{2} L_{c,i}^R - \left(c_i^{ops} W_i^D + \sum_{c=1}^{2} c_{c,i}^{cu} L_{c,i}^R + \sum_{j=1}^{3} \pi_j \delta_{us_{j,i}}^{all} W_{i,j}^P \right) \right) \tag{4.5.6a}$$

$$L_{c,i}^R \leq lf_{c,i} W_i^D, \quad \forall c, i \quad (\gamma_{c,i}^{loss}) \tag{4.5.6b}$$

$$W_i^D \leq n_i + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} W_{i,j}^P - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right), \quad \forall i \quad (\gamma_i^{flow}) \tag{4.5.6c}$$

$$W_i^D, W_i^P \geq 0, \quad \forall i, \quad L_{c,i}^R \geq 0, \quad \forall c, i \tag{4.5.6d}$$

The aggregate CSM model looks as follows:

$$\max_{\zeta_i^{CSM}} \sum_{i=1}^{3} \left(\int_0^{W_i^D} (\alpha_i - \beta_i x_i) dx_i + \sum_{k=1}^{3} \pi_k \delta_{ds_{k,i}}^{all} \sum_{c=1}^{2} L_{i,c}^R - \left(c_i^{ops} W_i^D + \sum_{c=1}^{2} c_{i,c}^{cu} L_{i,c}^R + \pi_i W_i^P \right) \right) \tag{4.5.7a}$$

$$L_{c,i}^R \leq lf_{c,i} W_i^D, \quad \forall c, i \quad (\gamma_{i,c}^{loss}) \tag{4.5.7b}$$

$$W_i^D \leq n_i + W_i^P - r_i^{fc} + \sum_{j=1}^{3} \delta_{us_{j,i}}^{all} \left(n_j - \sum_{c=1}^{2} lf_{c,j} W_j^D \right), \quad \forall i \quad (\gamma_i^{flow}) \tag{4.5.7c}$$

$$W_i^D, W_i^P \geq 0, \quad \forall i, \quad L_{i,c}^R \geq 0, \quad \forall c, i \tag{4.5.7d}$$

The only difference in the LCP model for the GCM and CSM market models versus these two GCM and CSM aggregate models lies in the stationarity conditions for W_i^D, which become for the two aggregate models the following:

$$0 \leq \beta_i W_i^D - \alpha_i + c_i^{ops} - \sum_{c=1}^{2} lf_{c,i}\gamma_{c,i}^{loss} + \gamma_i^{flow} + \sum_{c=1}^{2} lf_{c,i} \sum_{k=1}^{3} \delta_{ds_{k,i}}^{all} \gamma_k^{flow} \perp W_i^D \geq 0, \quad \forall i \quad (4.5.8)$$

The new term of $\sum_{c=1}^{2} lf_{c,i} \sum_{k=1}^{3} \delta_{ds_{k,i}}^{all} \gamma_k^{flow}$ represents the marginal value of water lost to the system or lost to the rest of downstream players k if player i takes more water from the system. Thus, the centralized planner for the aggregate model is trying to find the W_i^D that balances the marginal benefit to player i with the marginal cost to the system as Britz et al. discuss in [28].

We use the same q as is used in the GCM and CSM market models and which can be found in Table 4.8 to produce the following $\mathbf{z}$ solutions that can be found in Table 4.9:

- $\mathbf{z}^{GCM}$ - the GCM market solution
- $\mathbf{z}^{AGGCM}$ - the GCM aggregate solution
- $\mathbf{z}^{CSM}$ - the CSM market solution
- $\mathbf{z}^{AGCSM}$ - the CSM aggregate solution

Again, as in the GCM and CSM market model solves, we use one solution of potentially multiple solutions that we find employing PATH. We use the same starting point technique and starting points as in Section 4.5.2.1. We do find that, for the W_2^P variable value for the CSM aggregate market, this variable can take on values ranging from 1.4883 to 1.6883. However, we still use only one of these aggregate solutions for this CSM model for this section. The matrices for the aggregate GCM and CSM models do have different properties than the GCM and CSM market models. For the aggregate GCM and CSM matrices, the principle minors lie in the range of 0

to 27, and the unique eigenvalues are 0 and 3, which repeat. This means that both matrices are positive semi-definite, which we know means they have a convex set of solutions as Theorem 7 shows. The starting-point methods provide some evidence that there may be only one solution for the GCM model, which is a convex set of solutions, while the starting point method for the CSM model suggests that there may be multiple solutions involving the W_2^P variable. Tables 4.10 and 4.11 compare the objective function totals across all three players as well as the individual player objective function values for the GCM and CSM market and aggregate models.

Variable	Explanation	$\mathbf{z}^{GCM}$	$\mathbf{z}^{AGGCM}$	$\mathbf{z}^{CSM}$	$\mathbf{z}^{AGCSM}$
W_1^D	water demand (volume/day), player 1	7.36	6.1309	7.4813	6.4933
W_2^D	water demand (volume/day), player 2	8.0864	8.406	10.0	9.8
W_3^D	water demand (volume/day), player 3	10.0	10.0	10.0	10.0
$W_{2,1}^P$ or W_1^P	water purchases (volume/day)	0.0	0.0	0.0	0.0
$W_{3,1}^P$ or W_2^P	water purchases (volume/day)	1.9136	1.594	1.9451	1.6883
$W_{3,2}^P$ or W_3^P	water purchases (volume/day)	1.6173	1.6812	3.9451	3.6483
$L_{1,1}^R$	loss reductions (volume/day), class 1, player 1	0.9568	0.797	0.9726	0.8441
$L_{1,2}^R$	loss reductions (volume/day), class 1, player 2	0.8086	0.8406	1.0	0.98
$L_{1,3}^R$	loss reductions (volume/day), class 1, player 3	0.0	0.0	0.0	0.0
$L_{2,1}^R$	loss reductions (volume/day), class 2, player 1	0.9568	0.797	0.9726	0.8441
$L_{2,2}^R$	loss reductions (volume/day), class 2, player 2	0.8086	0.8406	1.0	0.98
$L_{2,3}^R$	loss reductions (volume/day), class 2, player 3	0.0	0.0	0.0	0.0
π_1	price for reductions (cost/unit flow)	9.9999	10.0	0.0	0.0
π_2	price for reductions (cost/unit flow)	9.9999	10.0	1.4	0.0
π_3	price for reductions (cost/unit flow)	0.0	0.0	10.0	10.0
$\gamma_{1,1}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 1	9.3332	9.3333	10.7332	9.3333
$\gamma_{1,2}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 2	8.9999	9.0	9.0	9.0
$\gamma_{1,3}^{loss}$	opportunity cost of reductions (cost/unit flow), class 1, player 3	0.0	0.0	0.0	0.0
$\gamma_{2,1}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 1	6.6666	6.6667	8.0666	6.6667
$\gamma_{2,2}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 2	4.9999	5.0	4.9999	5.0
$\gamma_{2,3}^{loss}$	opportunity cost of reductions (cost/unit flow), class 2, player 3	0.0	0.0	0.0	0.0
γ_1^{flow}	value of additional water to player 1 (cost/unit flow)	0.0	0.0	0.0	0.0
γ_2^{flow}	value of additional water to player 2 (cost/unit flow)	7.1408	4.1821	1.4	0.0
γ_3^{flow}	value of additional water to player 3 (cost/unit flow)	9.9999	10.0	9.9999	10.0

Table 4.9: Comparing $\mathbf{z}$ Solutions Between Market and Aggregate Models

	GCM	CSM
Market Model	451.7738	455.7809
Aggregate Model	454.1930	457.3051

Table 4.10: Sum of Objective Function Values Market vs Aggregate Model (cost/day)

	Player 1	Player 2	Player 3
GCM Market Model	81.2542	155.8280	214.6916
GCM Aggregate Model	78.9883	157.9570	217.2478
CSM Market Model	83.9553	161.2766	210.5489
CSM Aggregate Model	80.1277	163.6600	213.5173

Table 4.11: Individual Objective Function Values Market vs Aggregate Models (cost/day)

Comparing the aggregate and market solutions in Tables 4.9-4.11 provides several important insights regarding the water withdrawals of the three players. The unconstrained demand (which is the demand of the players given no water scarcity[6]) values of the three players are 6.6667, 10, and 13.3333 volume/day, respectively [25]. Noting these values helps with the water withdrawal analysis for certain players.

For both the GCM and CSM models, it is better for the system as a whole according to the aggregate objectives (4.5.6a) and (4.5.7a) and thus according to Tables 4.9-4.10 to have player 1 withdraw less water W_1^D and also recover less water to be able to sell to downstream players in the form of $L_{1,1}^R, L_{2,1}^R$. This results mathematically because, in the aggregate models, there is now the term $\sum_{c=1}^{2} lf_{c,i} \sum_{k=1}^{3} \delta_{ds_{k,i}}^{all} \gamma_k^{flow}$ in the W_1^D stationarity complementarity equation (4.5.8) that forces

[6]Water scarcity for player i is defined as the constraint associated with the γ_i^{flow} dual variable being tight.

player 1 to take into consideration the "damage" it causes downstream players by withdrawing too much. We note for player 1 in the market and aggregate models that W_1^D represents only the removal of free-flow water and not purchases because there is no one upstream from whom to purchase water. Furthermore, we indicate that, in both the GCM and CSM markets, player 1 withdraws more water than their unconstrained demand in order to sell recovered water to downstream players. However, in both of the aggregate models, player 1 withdraws less water than their unconstrained demand, indicating that this player is taking into consideration the affect of their water withdrawals on the players downstream. What's more, we notice in Table 4.11 that player 1's net benefit decreases in the aggregate model compared to the market model for both the GCM and CSM models. This is because player 1 was over withdrawing in the market models but, in the aggregate models, they are correcting for this over-withdrawal.

In the case of player 2, the story is more complicated. For the GCM model, player 2 withdraws more water in the aggregate model compared to the market model as shown in Table 4.9. For the CSM model, player 2 withdraws a little less water in the aggregate model compared to the market model as shown in Table 4.9. In both GCM and CSM aggregate models, the objective function for player 2 does increase as seen in Table 4.11. For the GCM model, the increase in the aggregate model for player 2 occurs because player 2 can satisfy more demand by removing more free-flowing water (i.e. $W_2^D - W_{2,1}^P$). Player 2 does not purchase from upstream player 1 in either the market or aggregate models, so player 2 is able to remove more free flow water and then recover more water to sell to downstream player 3. For the CSM model, the increase in the objective function in the aggregate model for player 2 comes from the water that is accessed freely (i.e., $\pi_2 = 0$) from player 1 as it flows downstream to player 3. Thus, CSM player 2 is able to consume a significant amount of water for "free," except for treatment and

distribution costs (i.e., c_2^{ops}). Indeed, one theory for why the W_2^P value may take on a range of solution values as we reported above might have to do with this "free" effect because the player may be indifferent between the water it can take from the free flow and water it can "purchase" from upstream players, since this water is the same cost through both means.

As shown in Table 4.11, the objective function values for player 3 are higher in the aggregate model versus the market model. For both the GCM and CSM markets, however, the level of water withdrawn by player 3 stays the same when moving from the market to the aggregate models as Table 4.9 demonstrates with the value of W_3^D being consistently 10. For both the GCM and CSM models, the increase in player 3's objective occurs because they remove more free-flowing water to satisfy demand which is less expensive, since free-flowing water only costs c_3^{ops} while purchasing water requires an additional cost in the form of π_1 for the GCM market for player 3 and π_3 for the CSM market for player 3. As a result, they purchase less from upstream players.

Since we now have aggregate market $\mathbf{z}^{AGGCM}$ and $\mathbf{z}^{AGCSM}$ solutions, we can use these solutions in our inverse optimization quadratic program (4.3.46) to parameterize the original GCM and CSM market models as if the $\mathbf{z}^{AGGCM}$ and $\mathbf{z}^{AGCSM}$ are solutions to these market models, thus pushing the GCM and CSM models toward these system optimal solutions. For the formulation of this inverse optimization problem, we run a couple of different trials:

1. The first trial fixes q^0 to zeros for both the GCM and CSM inverse optimization problems.

2. The second trial fixes q^0 equal to the $q^{GCM} = q^{CSM}$ vector from the last set of experiments for both the GCM and CSM inverse optimization problems. The $q^{GCM} = q^{CSM}$ vector is also used to produce the aggregate solutions.

We define

- $q^{AGGCMIO}$ as the IO q solution to the GCM market model using the $\mathbf{z}^{AGGCM}$ solution
- $q^{AGCSMIO}$ as the IO q solution to the CSM market model using the $\mathbf{z}^{AGCSM}$ solution

In Table 4.12, we showcase the relevant subset of the q coefficients from the IO quadratic program trials. q^0 is denoted as either being zeros, q^{GCM}, or q^{CSM}. We call the following experiments (with the different settings for q^0) the IO quadratic program trials.

Coefficient	$q^{GCM} = q^{CSM}$	$q^{AGGCMIO}, q^0$ zeros	$q^{AGGCMIO}, q^0 = q^{GCM}$	$q^{AGCSMIO}, q^0$ zeros	$q^{AGCSMIO}, q^0 = q^{CSM}$
$-\alpha_1 + c_1^{ops}$	-20.0	-16.3127	-16.3127	-17.4	-17.4
$-\alpha_2 + c_2^{ops}$	-30.0	-28.0	-28.0	-28.0	-28.0
$-\alpha_3 + c_3^{ops}$	-39.9999	-40.0	-40.0	-40.0	-40.0

Table 4.12: Selected IO q Entries for GCM and CSM Markets Using Aggregate $\mathbf{z}$ Solution

We choose to display these coefficients of the q vectors because they are the most consequential coefficients of the experiments. This is due to the fact that they have the same values between the two settings of the q^0 and due to the meaning of the change for the GCM and CSM markets. For the table that lists all of the q coefficient values, see Appendix C.2. These q coefficients are important because they present policy implications for the GCM and CSM markets. Recall that the purpose of these IO quadratic program trials is to see if the GCM and CSM markets can be adjusted to produce the respective aggregate model solutions. For the $q^{AGGCMIO}$ and $q^{AGCSMIO}$ coefficient values compared to the original $q^{GCM} = q^{CSM}$ coefficient values in Table 4.12, the values are lower for players 1 and 2, which means c_1^{ops} and c_2^{ops} have to be increased to achieve the aggregate $\mathbf{z}^{AGGCM}$ and $\mathbf{z}^{AGCSM}$ solutions in the GCM and CSM markets. Since c_1^{ops} and c_2^{ops} represent the cost of withdrawing water, increasing their values could be done through a form of taxation on water withdrawals. Thus, inverse optimization has *suggested a policy mechanism* to move the markets toward system optimal and thus to maximize additive net benefit for the entire

market.

We verify that the q coefficient values obtained from all of the IO quadratic program trials in Table 4.12 for both the GCM and CSM markets return the aggregate solutions when used in the GCM and CSM market LCPs. See Appendix C.3 for more information.

4.6 Conclusions & Future Work

In conclusion, we explore the parameterization of the q vector of the general linear complementarity problem and then apply that knowledge to a water supply market case study. In the theoretical portion of the chapter, we establish general concepts such as complete information and overlapping complementary cones that allow us to analyze properties of and evaluate solutions from the IO quadratic program (4.3.46). We also cover the simplification of this general quadratic program under different solution structure cases, thus providing a general but practical IO solution framework. In the case study, the most important result we find is that the water supply market evaluated can be influenced by levying taxes on upstream players to move the market toward system optimal. This case study provides a blueprint for other markets to discover the incentives that must be shifted in order to obtain system optimal by demonstrating the way in which to collapse a market to one player, obtain system optimal solution(s), and then use those solution(s) in the inverse optimization quadratic program (4.3.46) to find the right incentives.

With regard to future work, we would expand upon the idea of complete information from both theoretical and practical points of view, seeing if we could find necessary and/or sufficient conditions for complete information. We might also explore the degeneracy properties of $\mathbf{z}$ solutions in the context of finding q. We would also explore more regarding the complementary

cones and their role in the parameterization of the LCP problem. Furthermore, we would potentially investigate if we could use this work to detect changes in equilibrium phenomenon. A change in this phenomenon might be suggested if Z_{obs} is forced to be subset into different parts due to the feasibility check provided by the IO quadratic program (4.3.46). Finally, we would experiment with more case studies to further demonstrate the power of this tool for market design.

4.7 Chapter Acknowledgements

Stephanie Allen was partially funded by a Graduate Fellowship for STEM Diversity, a University of Maryland Flagship Fellowship, and NSF Grant #2113891 Division of Civil, Mechanical and Manufacturing Innovation. Steven Gabriel and Nathan Boyd were funded by NSF Grant #2113891 Division of Civil, Mechanical and Manufacturing Innovation.

Chapter 5: Conclusion and Future Work

Having reached the end of this dissertation, we can discuss some important themes and overarching concepts that this work suggests. First, as the first two projects demonstrate, when using inverse optimization in conjunction with observational or simulation data, it is important to design a train-test framework as discussed in [84] to ensure that that the parameters obtained from inverse optimization can perform well on unseen data. Second, we see that inverse optimization, as in the first two projects, can be used with simulation or observational data to find the hidden parameters in models such as traffic models, but inverse optimization can also be used with idealized or system optimal solutions, as in project three, to find the parameters that push the equilibrium problem towards system optimal. In fact, project three's methods could be used in conjunction with project two's concepts to improve overall traffic flow by aggregating the players in project two's transportation game, finding a system optimal solution, and then using the system optimal solution to parameterize the original game theory model. Third, we see that there is still much work to be done in parameterizing different equilibrium problems. Project one illustrates this well in its sensitivity to parameterizations for some of the experiments. Inverse optimization certainly helped in finding the "correct" protection decisions, but the stochastic mathematical program with complementarity constraints was still quite sensitive to the parameterization even being slightly off from the original.

With regard to future work, we would certainly take the market design vision from the third project and see if it could be applied to the first two projects. We would also explore parameterization of additional classes of equilibrium problems in general, such as mixed complementarity problems and potentially mathematical programs with equilibrium constraints. Overall, we would further expand the conversation of these three projects with each other, such as exploring multiplicity of solutions in projects one and two and considering expanded computational experiments in project three.

For project one specifically, we would expand the traffic equilibrium model to handle the stochastic user equilibrium case original proposed by [51], which would likely require more computational innovations due to the presence of additional random terms in the formulation. We would also explore if it would be possible to develop a stochastic network protection model that could have scenario dependent traffic parameters and see what role inverse optimization would play in those scenario dependent traffic parameters, potentially drawing upon methods from [121] who use inverse optimization to parameterize risk functions. We do not believe this work [121] is directly transferable because one would need to know the outcomes under each scenario, which is impossible in a disaster relief context. However, perhaps this project paired with simulation and other tools could give us new insights into estimating more detailed traffic parameters.

For project two, we would apply the jointly convex GNEP framework to more applications to further demonstrate its utility. We would also expand the traffic example to incorporate a more sophisticated cost function and interaction function, such as the BPR function employed in the first project, and to potentially integrate a more complex jointly convex constraint that would

measure capacity as less of a ceiling and more of a gradual function.[1] It would also be interesting to expand other inverse optimization frameworks besides just [112] to the context of Nash and generalized Nash models, particularly frameworks that deal with noisy observations [8, 13, 61].

For project three, it would be interesting to expand the parameterization of the LCP to the M matrix, with the q fixed, which Huang [103] start to address. We could see if we could generate simplification rules for the IO quadratic program that we and Huang [103] propose, and we imagine that this work would take us into the world of semi-definite programming from some very preliminary work we did on the subject in the beginning of the project. It would also be interesting to tackle the idea of noisy observations in equilibrium problems in general, using [8, 13, 61] as a guide.

In conclusion, this dissertation has applied inverse optimization to new applications and expanded its reach to new classes of equilibrium problems. Our research presents new pathways for using inverse optimization for the social good, and we aim to carry this social good focus forward in our subsequent work beyond these pages.

[1] We credit anonymous reviewers for inspiring these insights regarding complicating the traffic model.

Appendix A: A Hybrid Inverse Optimization-Stochastic Programming Framework for Network Protection

Note: The contents of this appendix come mostly from a paper we have published on arXiv, referenced here: [5]. There are certain additions that were made in response to the first round of reviews from an academic journal.

A.1 Data Analysis Component: Inverse Optimization

A.1.1 Proof of $\mathbf{d}^w - N\mathbf{x}^w = 0$

Lemma 2. *For the following complementarity problem:*

$$0 \leq \mathbf{c}(\mathbf{x}^w) + N^T\mathbf{y}^w \perp \mathbf{x}^w \geq 0,\ \forall w \in \mathcal{W} \tag{A.1.1a}$$

$$0 \leq \mathbf{d}^w - N\mathbf{x}^w \perp \mathbf{y}^w \geq 0,\ \forall w \in \mathcal{W} \tag{A.1.1b}$$

$\mathbf{d}^w - N\mathbf{x}^w = 0$ *when there is a solution for (A.1.1) and when we assume that the* $\mathbf{c}(\mathbf{x})$ *function is greater than 0 for all* $\mathbf{x}^w \geq 0$.

Proof. This proof is adapted from a proof seen in [15]. Assume that the $\mathbf{c}(\mathbf{x}^w)$ function is greater

than 0 for all $\mathbf{x}^w \geq 0$. Also assume for the sake of contradiction that

$$0 < d_i^w - \left(\sum_{l:(l,i)} x_{(l,i)}^w - \sum_{j:(i,j)} x_{(i,j)}^w \right) \tag{A.1.2}$$

for some destination $w \in \mathcal{W}$ and for some $i \in \mathcal{N}$. $\sum_{l:(l,i)} x_{(l,i)}^w$ represents the inflow at node i, and $\sum_{j:(i,j)} x_{(i,j)}^w$ represents the outflow at node i. We know $y_i^w = 0$ by complementarity in (A.1.1b). We can rearrange the inequality in (A.1.2) to say:

$$0 \leq \sum_{l:(l,i)} x_{(l,i)}^w < d_i^w + \sum_{j:(i,j)} x_{(i,j)}^w \tag{A.1.3}$$

for some $w \in \mathcal{W}$ (representing the final destination) and for some $i \in \mathcal{N}$. We have that the $\sum_{l:(l,i)} x_{(l,i)}^w$ term is greater than or equal to 0 because we know all $\mathbf{x}^w \geq 0$. The inequality in (A.1.3) produces three difference cases:

- Case 1: $d_i^w = 0$. This means at least one link in the $\sum_{j:(i,j)} x_{(i,j)}^w$ sum must be positive because $0 < d_i^w + \sum_{j:(i,j)} x_{(i,j)}^w$. Therefore, for such a link (i,j), $x_{(i,j)}^w > 0$ forces the following equality:

  $$\mathbf{c}_{(i,j)}(\mathbf{x}^w) + y_j^w - y_i^w = 0 \tag{A.1.4}$$

  We know $y_i^w = 0$, so we have $\mathbf{c}_{(i,j)}(\mathbf{x}^w) + y_j^w = 0$. Since $y_j^w \geq 0$, both components must be zero but that contradicts the assumption that $\mathbf{c}(\mathbf{x})$ is greater than 0 for all $\mathbf{x} \geq 0$, thereby contradicting (A.1.2).

- Case 2: d_i^w is negative. This means at least one $x_{(i,j)}^w > 0$ and contradiction follows as in Case 1.

- Case 3: d_i^w is greater than 0. This implies that $i = w$ because only the final destination has a positive value. There are some sub-cases to this case, but we first note that we know for a general node $k \neq i$:

$$\sum_{l:(l,k)} x_{(l,k)}^w - \sum_{j:(k,j)} x_{(k,j)}^w = d_k^w \tag{A.1.5}$$

for the cases of $d_k^w = 0$ or when d_k^w is negative, based on Cases 1 and 2.

  - Sub-Case A: $\sum_{j:(i,j)} x_{(i,j)}^w > 0$. In this case, we arrive at the same contradictions we arrived at for the previous two cases.

  - Sub-Case B: $\sum_{j:(i,j)} x_{(i,j)}^w = 0$ and $\sum_{l:(l,i)} x_{(l,i)}^w = 0$. In this case, we know there is some d_k^w that is negative, which by (A.1.5) means $\sum_{j:(k,j)} x_{(k,j)}^w > 0$. For any connecting nodes q between node k and node i, $\sum_{l:(l,q)} x_{(l,q)}^w = \sum_{j:(q,j)} x_{(q,j)}^w > 0$. Therefore, for node i, the inflow sum $\sum_{l:(l,i)} x_{(l,i)}^w$ must be greater than 0, contradicting our assumption. Overall, we arrive at the contradiction because flow begins at a node, which produces outflow to neighboring nodes, and this in turn produces inflow at node i.

  - Sub-Case C: $\sum_{j:(i,j)} x_{(i,j)}^w = 0$ and $\sum_{l:(l,i)} x_{(l,i)}^w > 0$ but less than d_i^w. For any k nodes in which d_k^w is negative, we know the absolute sum over these k nodes is equal to d_i^w when $i = w$. Therefore, as in Sub-Case B, there are outflows at these k nodes due to the relationship (A.1.5). There is also conservation of flow at the q nodes in which $d_q^w = 0$. Therefore, for the q nodes connected to node i, the outflow from those nodes must match the inflow from previous nodes, which if taken back to the k nodes, would equal the total d_i^w sum. Therefore, for node i, the inflow sum $\sum_{l:(l,i)} x_{(l,i)}^w$ must be equal to d_i^w. This contradicts our assumption that the inflow would be less than d_i^w. Overall,

we arrive at the contradiction because flow would be pushed toward the i destination in order to satisfy the relationships established by (A.1.5).

Consequently, we have shown that a solution to the traffic equilibrium problem will result in the $\mathbf{d}^w - N\mathbf{x}^w = 0$ if we assume the $\mathbf{c}(\mathbf{x}^w)$ function is greater than 0 for all $\mathbf{x}^w \geq 0$. □

A.1.2 Explanation of Forming the Inverse Model from Bertsimas et al. [21]

To form the inverse optimization mathematical program from Bertsimas et al. [21] for our traffic equilibrium problem, we return to the VI formulation of the problem and notice the structure (A.1.6)

$$\mathbf{c}(\mathbf{x}^*)^T\mathbf{x} \geq \mathbf{c}(\mathbf{x}^*)^T\mathbf{x}^* - \epsilon, \ \forall \mathbf{x} \in \mathcal{F} \tag{A.1.6a}$$

$$\mathcal{F} = \left\{\mathbf{x} : \mathbf{x} \in \mathbb{R}_+^{|\mathcal{A}|} \ s.t. \ N\mathbf{x} \leq \mathbf{d}\right\} \tag{A.1.6b}$$

Note, the $\mathcal{F}$ set is written slightly differently here than how it was initially introduced in equation (2) in Section 3.1 in order to mirror the complementarity problem (2.3.3) and in order to represent the fact that we only are working with one destination at a time (hence we do not need the w index). We notice that we can turn the left hand side of the (A.1.6a) inequality into a minimization problem

$$\min_{\mathbf{x} \in \mathcal{F}} \mathbf{c}(\mathbf{x}^*)^T\mathbf{x} \tag{A.1.7}$$

in which $\mathbf{x}^*$ is fixed, and this minimization problem forms the tightest upper bound on the right hand side of (A.1.6a) since we are choosing the $\mathbf{x}$ to minimize the left hand side of (A.1.6a) [21, 64]. Because (A.1.7) is a linear program, we know strong duality holds, which means we

can find the dual of this problem and know that there is no duality gap between the primal and the dual [137, 197]. The dual of this problem is:

$$\max_{\mathbf{y}} (-\mathbf{d})^T \mathbf{y} \tag{A.1.8a}$$

$$-N^T \mathbf{y} \leq \mathbf{c}(\mathbf{x}^*) \tag{A.1.8b}$$

$$\mathbf{y} \geq 0 \tag{A.1.8c}$$

As in Bertsimas et al. [21], we then equate the dual objective and the primal objective, and our final set of constraints representing the satisfaction of the variational inequality in (A.1.6) are:

$$\mathbf{c}(\mathbf{x}^*)^T \mathbf{x}^* + (\mathbf{d})^T \mathbf{y} \leq \epsilon \tag{A.1.9a}$$

$$-N^T \mathbf{y} \leq \mathbf{c}(\mathbf{x}^*) \tag{A.1.9b}$$

$$\mathbf{y} \geq 0 \tag{A.1.9c}$$

which include the equating of the dual and primal objectives (A.1.9a) as well as the dual feasibility constraints (A.1.9b-A.1.9c). Through this process, we have created a set of conditions that encodes the satisfaction of the ϵ VI [21, 38]. Using these conditions, we can then form an

optimization problem including each data point $\hat{\mathbf{x}}^j$ representing the flow on the network such that:

- There is one OD pair for each instance $\hat{\mathbf{x}}^j$.
- There is the same node-arc incidence matrix N for each $\hat{\mathbf{x}}^j$.

The optimization problem becomes for J data points, parameters $\theta \in \Theta$ with Θ as a convex subset of $\mathbb{R}^Z$ (Z representing a number of parameters), $\mathbf{y}^j \in \mathbb{R}^{|\mathcal{N}|}$, and $\epsilon \in \mathbb{R}^J$:

$$\min_{\theta \in \Theta, \mathbf{y}, \epsilon} ||\epsilon||_2^2 \tag{A.1.10a}$$

$$-(N)^T \mathbf{y}^j \leq \mathbf{c}(\hat{\mathbf{x}}^j; \theta),\ j = 1, ..., J, \tag{A.1.10b}$$

$$\mathbf{y}^j \geq 0,\ j = 1, ..., J, \tag{A.1.10c}$$

$$\mathbf{c}(\hat{\mathbf{x}}^j; \theta)^T \hat{\mathbf{x}}^j + (\mathbf{d}^j)^T \mathbf{y}^j \leq \epsilon^j,\ j = 1, ..., J, \tag{A.1.10d}$$

The set Θ is determined by the lower and upper bounds on the parameter values found in Table 2.1. This optimization problem tries to find the parameterization that minimizes the ϵ needed from the approximate equilibrium [21, 38].

A.2 Normative Model: Two-Stage Stochastic Model

A.2.1 Two-Stage Stochastic Model: Parameter Details

We adopt the notation from Fan and Liu [65], with the exception of the parameters in the $h_a^s(u_a)$ function which, although inspired by [65], is of a different form:

- $\mathcal{A}$: the set of network arcs, and m as the number of arcs.
- $\mathcal{N}$: the set of network nodes, and n as the number of nodes.
- K: the number of destinations of flow in the network.
- $\mathcal{S}$: the scenario set
- $x_a^{k,s}$: the flow on arc a that is destined for the kth destination in scenario s. The vector $\mathbf{x}^{k,s} \in \mathbb{R}^m$ denotes the flow on all arcs that is headed for the kth destination in scenario s. (units=thousands of vehicles)
- f_a^s: the total flow on arc a in scenario s, and $\mathbf{f}^s$ as the vector containing all of the f_a^s decision variables for scenario s. (units=thousands of vehicles)
- u_a: the decision variable controlling resources used to protect an arc a against a crisis. Some examples of potential protection decisions include protective measures against landslides and flash floods as in the case of Nepal [152]. (units=proportion of necessary resources needed to fully insure the arc)
- W: the node-link adjacency matrix. We use the definition from [129]'s work of this matrix which states $W \in \{-1, 0, 1\}^{|N|\times|A|}$ such that, for a given column (representing an arc),

there is a -1 at the node in which the arc begins and a 1 at the node in which the arc ends.

- $\mathbf{q}^k \in \mathbb{R}^n$: designates the amount of flow originating at each node that is headed to destination k. We based our construction of the $\mathbf{q}^k$ vectors upon the set-up from [129]'s such that negative entries within the vector indicate the presence of and amount of demand at those nodes and such that a single positive entry denotes location of the demand (and is the absolute sum of the negative entries). (units = thousands of vehicles)

- $h_a^s(u_a)$: the capacity of an arc a given first stage decision u_a under scenario s:

$$h_a^s(u_a) = \begin{cases} \text{cap}_a \ \text{ if } a \notin \bar{\mathcal{A}} \\ \text{cap}_a - m_a^s(1-u_a) \ \text{ if } a \in \bar{\mathcal{A}} \end{cases} \tag{A.2.1}$$

  with cap_a representing capacity of the arc without it being affected by a disaster, m_a^s representing the amount of damage done to arc a in scenario s if not protected, and $\bar{\mathcal{A}}$ represents the set of arc vulnerable to the disaster. Note that m_a^s could be 0 in certain scenarios. (units = thousands of vehicles)

- $t_a(\mathbf{f}^s)$ represents the time per vehicle along arc a [124, 125] as a function of the flows $\mathbf{f}^s$ in scenario s. We explore multiple different forms for t_a.

- $\lambda_i^{k,s}$ as "the minimum time from node i to destination k" in scenario s [65]. (units=travel time)

- $\mathbf{d}^{k,s}$ as the vector of extra variables that acts as a buffer for any flow that cannot be properly apportioned. (units=thousands of vehicles).

- p^s as the probability of each scenario s

A.2.2 Calculating the $M_{i,j}^{k,s}$ Values

The $M_{i,j}^{k,s}$ values are the numbers utilized in the disjunctive constraints in Section 2.3.2.1. To calculate the $M_{i,j}^{k,s}$ values, we use the following reasoning. First, we know that the maximum flow on a given arc is 8. We also know from Table 2.1 that the maximum value of ϕ_a and β_a is 10. Therefore, we input $x_a = 8$ into $\phi_a x_a + \beta_a$, obtain 90, and then multiply by the number of arcs to obtain an upper bound, which can be increased if desired. We decide to increase it by multiplying by 2. The resulting value represents an upper bound on the maximum travel time between an origin and destination point in the networks under the linear cost function. It also works for the BPR cost function because if we take the maximum value of that function for a given arc, we would get 12, which is significantly below 90. The final value of $M_{i,j}^{k,s}$ is $90(m)(2)$, with m as the number of arcs.

A.3 Results

A.3.1 Flow Error under IO α

Allen et al. [4] define a flow error metric to evaluate whether or not an IO parameterization is valid for their application. Since the α values (for the BPR functions) imputed through IO are different from the original α values, we evaluate the flow error for each network used. This flow error metric is the Frobenius norm between the flow values across all the arcs for all of the OD pairs in a given trial. The flow error metrics for the two networks can be seen in Table A.1 below:

4x4 Grid (Experiment IIA)	Nguyen & Dupuis Network (Experiment IVA)
0.0002	1.57e-05
0.0001	2.93e-05
0.0002	7.11e-05
0.0002	3.08e-05
5.73e-05	8.62e-05
0.0007	0.0004
0.0001	4.02e-05
6.39e-05	4.43e-05
0.0001	2.26e-05
7.82e-05	6.06e-05

Table A.1: Flow Errors for BPR Functions on the Two Networks

From the small magnitude of these values, we see that the α recovered by the IO model produce flow values that are very close to the flow values produced by the original α values.

A.3.2 Median/Min-Max Tables and Nguyen & Dupuis Boxplots

When examining the medians as a percentage of the budget for Experiments IA-IIIA in Tables A.2 and A.3, the O-IO metric medians are quire small compared to the U-IO & U-O metric medians, thus again supporting the claim that IO can be used to recover the original cost protection decisions and that the protection decisions made under IO and original costs are different from the protection decisions made under uniform cost.

Looking at Figure A.1, the metric data are not overlapping which supports the idea that the

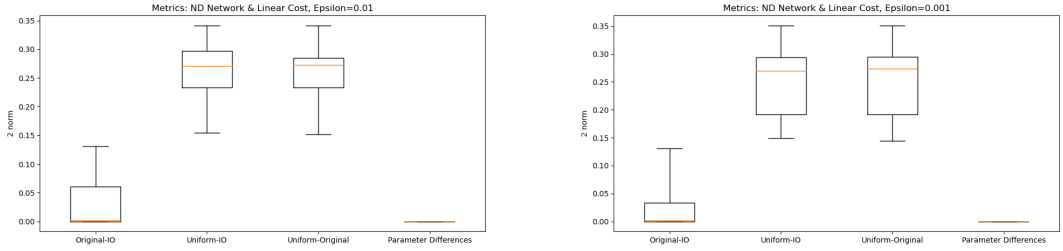
protection decisions under IO or original costs differ when compared to the protection decisions under uniform or baseline cost parameters. In Figure A.3, we see that the decisions under uniform cost do not differ from the IO imputed cost decisions in Experiment IVA as much as in other experiments. However, this could be a result of the small interval in which α was allowed to vary. In future work, it would be interesting to experiment with wider intervals to further understand this behavior. At the same time, these results do not take away from our conclusion that IO is able to impute costs that lead to protection decisions similar to those of the original cost.

	Experiment IA		**Experiment IIA**		**Experiment IIIA**		**Experiment IVA**	
	Med	(Min, Max)	Med	(Min, Max)	Med	(Min, Max)	Med	(Min, Max)
O-IO	0.0 0.0%	(0.0, 0.0004)	0.0006 0.01%	(0.0, 0.0076)	0.0027 0.05%	(0.0, 0.1309)	0.0014 0.02%	(0.0, 0.0181)
U-IO	0.3456 5.76%	(0.2208, 0.4372)	0.0446 0.74%	(0.0108, 0.1641)	0.2704 4.51%	(0.154, 0.3416)	0.0112 0.19%	(0.0, 0.1819)
U-O	0.3456 5.76%	(0.2208, 0.4371)	0.0447 0.74%	(0.0083, 0.164)	0.2729 4.55%	(0.1513, 0.3416)	0.004 0.07%	(0.0, 0.1819)

Table A.2: Medians, Medians as Percentage of $I = 6$ Budget, and Ranges for Experiments, $\epsilon = 0.01$

	Experiment IA		Experiment IIA		Experiment IIIA		Experiment IVA	
	Med	(Min, Max)	Med	(Min, Max)	Med	(Min, Max)	Med	(Min, Max)
O-IO	0.0 0.0%	(0.0, 0.0004)	0.0006 0.01%	(0.0, 0.0041)	0.0012 0.02%	(0.0, 0.1309)	0.001 0.02%	(0.0, 0.0941)
U-IO	0.3134 5.22%	(0.2194, 0.4717)	0.0472 0.79%	(0.0092, 0.2037)	0.2699 4.5%	(0.1494, 0.351)	0.0095 0.16%	(0.0, 0.1209)
U-O	0.3134 5.22%	(0.2194, 0.4716)	0.0472 0.79%	(0.007, 0.2032)	0.2733 4.56%	(0.1446, 0.3511)	0.0075 0.13%	(0.0, 0.1251)

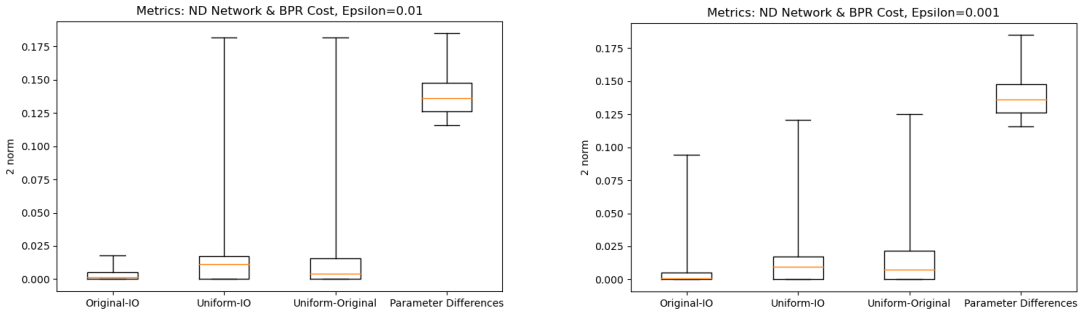
Table A.3: Medians, Medians as Percentage of $I = 6$ Budget, and Ranges for Experiments A, $\epsilon = 0.001$



(a) Experiment IIIA Results for $\epsilon = 0.01$

(b) Experiment IIIA Results for $\epsilon = 0.001$

Figure A.1: Experiment IIIA Results: Nguyen & Dupuis Network with Linear Cost. The parameter differences refer to the ϕ differences.



(a) Experiment IVA Results for $\epsilon = 0.01$

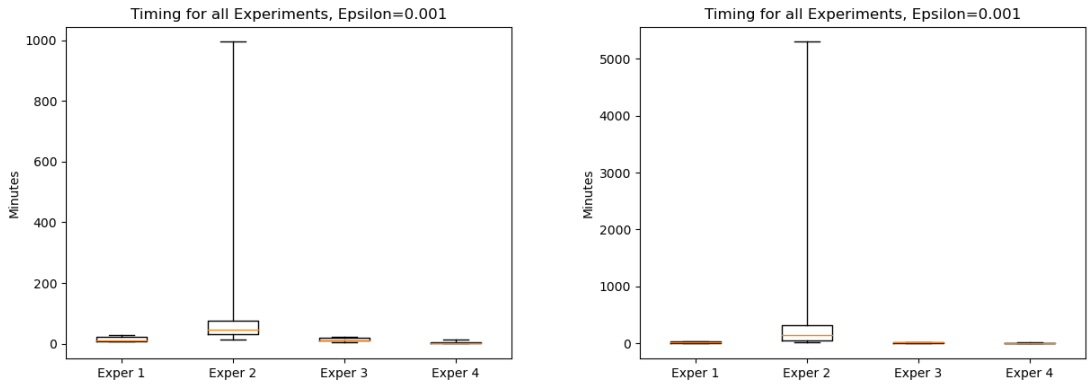
(b) Experiment IVA Results for $\epsilon = 0.001$

Figure A.2: Experiment IVA Results: Nguyen & Dupuis Network with BPR. The parameter differences refer to the α differences.

A.3.3 Run Time Results

The following box-plots illustrate the run time data for Experiments IA-IVA and for both values of ϵ, which is the value of the $g^{(k)}$ error metric in which the iterations could stop (or if 300 iterations occurred). As a reminder, all of the experiments were run on an 8 core machine.

For $\epsilon = 0.01$, most of the trials of the experiments were below 200 minutes and, for $\epsilon = 0.001$, most of the trials of the experiments were below 500 minutes. It can be seen that there were some outliers for Experiment II, which was likely due to the additional variables needed to estimate the BPR function and the larger graph size.



(a) Experiment A Timing for $\epsilon = 0.01$

(b) Experiment A Timing for $\epsilon = 0.001$

Figure A.3: Experiment A Timing Results (Minutes)

A.4 Code Attribution

Below are the various code resources, packages, etc. that we utilized over the course of the project:

- Python (Version 3.8.5) Package
  - `pyomo` 5.7.1 [95, 96]
  - `pysp` 5.7.1 [192]
  - `networkx` 2.5 [163]
  - `pandas` 1.1.3 [131]
  - `numpy` 1.19.2 [145, 184, 189]
  - `scipy` 1.5.2 [184]
  - `matplotlib` 3.3.2 [104]

  - `scikit-learn` 0.23.2 [150]

- Solvers

  - `gurobi` Version 9.1.1 [90]

  - `ipopt` [185]

- MATLAB 9.8.0.1417392 (R2020a) Update 4 [130]

- GAMS [78] with PATH solver [54, 70] (PATH website [71])

- Wolfram Alpha [198]

- Data:

  - Nguyen & Dupuis Network [142]

- Important Sites with Example Code for `pysp` Implementation in Scripts:

  - `https://projects.coin-or.org/Pyomo/browser/pyomo/trunk/examples/pysp/farmer/concrete/ReferenceModel.py?rev=9358`

  - `https://github.com/Pyomo/pysp/blob/master/examples/farmer/concreteNetX/ReferenceModel.py`

  - `https://pyomo.readthedocs.io/en/stable/advanced_topics/pysp_rapper/demorapper.html#ph`

Appendix B: Using Inverse Optimization to Learn Cost Functions in Generalized Nash Games

Note: Most of the contents of this appendix come from our article published in *Computers & Operations Research*: [4]. There have been a few additions due to the experiments in part B.

B.1 Variable Number Calculations for Lemma 1

In this section, we will provide the exact variable counts for problem (3.2.2) as the grid size increases from 2x2 to 3x3 to 4x4, and so on, as well as a more general variable count for graphs involving an arbitrary number of arcs. These variable counts are polynomial in the problem input size and, when compared with the polynomial-time complexity of solving linear programs, supports Lemma 1 in the main chapter. We will define N as the number of players and m as the total number of nodes.

Grid: We first focus on the Grid family of networks, as used in Section 3.5 in the main chapter. We note the following:

$$\text{Number of Origin-Destination Pairs: } (2)\binom{m}{2} \tag{B.1.1a}$$

Number of Arcs in a $\sqrt{m}x\sqrt{m}$ Grid: $(\sqrt{m}-1)(\sqrt{m})(2)(2)$ (B.1.1b)

For the same costs across all players, we have to consider the variables diag$(\mathcal{C}), \bar{c}, v_i^k, u_i^k, \bar{u}^k, \ \forall i, k$ along with the variables utilized in creating the 1-norms for the residual model. This variable count comes out to be for our code:

$$13Nm^3 - 12Nm^{5/2} - 13Nm^2 + 12Nm^{3/2} + 16m^3 - 16m^{5/2} - 16m^2 + 16m^{3/2} \quad \text{(B.1.2)}$$

For different costs across all players, we consider the variables diag$(\mathcal{C}_i), \bar{c}_i, v_i^k, u_i^k, \bar{u}^k, \ \forall i, k$ along with the variables utilized in creating the 1 norms for the residual model. This variable count comes out to be for our code:

$$21Nm^3 - 20Nm^{5/2} - 21Nm^2 + 20Nm^{3/2} + 8m^3 - 8m^{5/2} - 8m^2 + 8m^{3/2} \quad \text{(B.1.3)}$$

Therefore, in big O notation, the order of both variable counts is $\mathcal{O}(Nm^3)$.

General graphs: For any graph, we now define a as the number of arcs, with N as the number of players and m as the total number of nodes. Again, the number of OD pairs will be $(2)\binom{m}{2}$. The variable count for the same costs across players is for our code:

$$3aNm^2 - 3aNm + m^3N - m^2N + 4am^2 - 4am \quad \text{(B.1.4)}$$

The variable count for not the same costs across all players is for our code:

$$5aNm^2 - 5aNm + m^3N - m^2N + 2am^2 - 2am \tag{B.1.5}$$

Therefore, in big O notation, the order of both variable counts is $\mathcal{O}(aNm^2)$.

B.2 Uniqueness Considerations for the VI

Furthermore, according to [92] and [75], we can obtain a unique solution to this VI if the set $\mathbf{X}$ is compact and convex and if $\mathbf{F}$ defined by (3.3.4) is strictly monotone, which is defined in the following definition:

Definition 20 (Strict Monotonicity ([75, 93, 147]))**.** *A function* $\mathbf{F}$ *is strictly monotone if*

$$(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) > 0,\ \forall \mathbf{x}, \mathbf{y} \in \mathbf{X},\ \mathbf{x} \neq \mathbf{y} \tag{B.2.1}$$

We achieve both criteria with our traffic game, as defined and discussed in Section 3.2, because the equality constraints are linear, the player specific inequality constraints are convex, the "coupled" constraints are convex [63, 92], the constraints create bounds on the feasible region, and the $\mathbf{F}$ defined in (3.3.3) composed of the gradients of the objective functions of the players is strictly monotone [75, 92]. Strict monotonicity of our $\mathbf{F}$ defined in (3.3.3) is implied by showing it is strongly monotone [75, 93, 147], a property captured by the following definition:

Definition 21 (Strong Monotonicity ([75, 93, 147]))**.** *A function* $\mathbf{F}$ *is strongly monotone if for some* $\alpha > 0$

$$(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq \alpha||\mathbf{x} - \mathbf{y}||_2^2,\ \forall \mathbf{x}, \mathbf{y} \in \mathbf{X},\ \mathbf{x} \neq \mathbf{y} \tag{B.2.2}$$

We recognize that a similar proposition regarding players that all have the same cost functions and same origin-destination pairs appeared in [146] as described by [46], but our proof is specialized to this problem.

Lemma 3. *The* $\mathbf{F}$ *defined by (3.3.3) and created by the gradients of the objective functions defined in (3.2.1) is strongly monotone if we assume* $\mathcal{C}_1 = \mathcal{C}_2 = ... = \mathcal{C}_N$.

See Appendix B.3 for the proof. Note though that, just because the VI form for the GNEP has one solution (due to the strong monotonicity property), this does not mean that the original generalized Nash problem without the assumption of equal multipliers for the "coupled" constraints would not have other solutions (see [63] for a simple example in Section 1). Furthermore, if the $\mathcal{C}_i$ matrices are not all equal, then we are not guaranteed to have a strongly monotone $\mathbf{F}$.

Lemma 4. *The* $\mathbf{F}$ *defined by (3.3.3) and created by the gradients of the objective functions defined in (3.2.1) is not in general strongly monotone if we assume the* $\mathcal{C}_i$ *are not all equal.*

See Appendix B.4 for the counterexample.

B.3 Proof of Lemma 3

Proof. $\mathbf{F}$ is defined for (3.2.1) as follows:

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} 2\mathcal{C}_1 x_1 + \mathcal{C}_1 x_2 + ... + \mathcal{C}_1 x_N + \bar{c}_1 \\ \mathcal{C}_2 x_1 + 2\mathcal{C}_2 x_2 + ... + \mathcal{C}_2 x_N + \bar{c}_2 \\ \vdots \\ \mathcal{C}_N x_1 + \mathcal{C}_N x_2 + ... + 2\mathcal{C}_N x_N + \bar{c}_N \end{bmatrix} \tag{B.3.1}$$

In order to prove strong monotonicity, we must show that there exists a scalar $\alpha > 0$ such that

$$\left(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\right)^T (\mathbf{x} - \mathbf{y}) \geq \alpha ||\mathbf{x} - \mathbf{y}||^2 \; \forall \mathbf{x}, \mathbf{y} \in \mathbf{X}, \; \mathbf{x} \neq \mathbf{y}. \tag{B.3.2}$$

Through some algebra, it is not difficult to show that:

$$\left(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\right)^T (\mathbf{x} - \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \begin{bmatrix} 2\mathcal{C}_1 & \mathcal{C}_1 & \dots & \mathcal{C}_1 \\ \mathcal{C}_2 & 2\mathcal{C}_2 & \dots & \mathcal{C}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C}_N & \mathcal{C}_N & \dots & 2\mathcal{C}_N \end{bmatrix}^T (\mathbf{x} - \mathbf{y}) \tag{B.3.3}$$

If we assume that $\mathcal{C}_1 = \mathcal{C}_2 = ... = \mathcal{C}_N = \mathcal{C}$, we have

$$= (\mathbf{x} - \mathbf{y})^T \begin{bmatrix} 2\mathcal{C} & \mathcal{C} & \dots & \mathcal{C} \\ \mathcal{C} & 2\mathcal{C} & \dots & \mathcal{C} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C} & \mathcal{C} & \dots & 2\mathcal{C} \end{bmatrix} (\mathbf{x} - \mathbf{y}) \tag{B.3.4}$$

From Proposition 2.2.10 in [48], we know that, for symmetric, real valued matrices M and the smallest eigenvalue of M referenced as λ_1, we can write $\lambda_1 ||x||^2 \leq x^T M x, \; \forall x$. Therefore, to prove the matrix in (B.3.4) is positive definite, we simply need to prove that its smallest

eigenvalue is positive. We begin this process by splitting the matrix in (B.3.4) into two pieces

$$A = \begin{bmatrix} \mathcal{C} & 0 & \dots & 0 \\ 0 & \mathcal{C} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathcal{C} \end{bmatrix}, \; B = \begin{bmatrix} \mathcal{C} & \mathcal{C} & \dots & \mathcal{C} \\ \mathcal{C} & \mathcal{C} & \dots & \mathcal{C} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C} & \mathcal{C} & \dots & \mathcal{C} \end{bmatrix} \tag{B.3.5}$$

We state that $A + B = M$, with M as the original matrix, and we note that $A, B, M \in \mathbb{R}^{nN \times nN}$, with n as the number of arcs and N as the number of players. First, we prove that the eigenvalues of B are $\{0, ..., 0, N(eig(C))\}$. For this proof, we notice that the eigenvalue-eigenvector equation for B is

$$Bz = \lambda z \tag{B.3.6}$$

$$\mathcal{C}z_1 + ... + \mathcal{C}z_N = \lambda z_i, \; i = 1, ..., N \tag{B.3.7}$$

such that $z_i \in \mathbb{R}^n, \; \forall i$ (not all $z_i = 0$) and $\lambda \in \mathbb{R}$. Because $\forall i$ the left-hand sides of (B.3.7) are the same, we can equate the right hand sides to say:

$$\lambda z_1 = \lambda z_2 = ... = \lambda z_N \tag{B.3.8}$$

This provides two cases for the eigenvalues. Either $\lambda = 0$, or the z_i are such that

$$z_1 = z_2 = ... = z_N = w \neq 0 \tag{B.3.9}$$

for some $w \in \mathbb{R}^n$. This w can be substituted back into the equation in (B.3.7) to say

$$N\,\mathcal{C}w = \lambda w \tag{B.3.10}$$

which means the λ are the eigenvalues of $N\mathcal{C}$ and, since we already know that $\mathcal{C}$ is a positive diagonal matrix, the eigenvalues are just those entries on the diagonal multiplied by N. Because these diagonal values are positive and N is positive, then the resultant eigenvalues are positive.

Next, we use this information to prove that M is a positive definite matrix. Since we know that A and B are real, symmetric matrices, Weyl's Inequality [102, Theorem 4.3.1] states:

$$\lambda_1(A) + \lambda_1(B) \leq \lambda_1(A+B) = \lambda_1(M) \tag{B.3.11}$$

We know from above that $\lambda_1(B) = 0$. We also know that $\lambda_1(A) > 0$ because it is a diagonal matrix with the $\mathcal{C}$ matrices on its diagonal and, since $\mathcal{C}$ has all positive values, then we know all of the eigenvalues of A are positive. Consequently, $\lambda_1(A) + \lambda_1(B) > 0$ which means $\lambda_1(M) > 0$. As a result, we can write

$$(\mathbf{x}-\mathbf{y})^T \begin{bmatrix} 2\mathcal{C} & \mathcal{C} & \dots & \mathcal{C} \\ \mathcal{C} & 2\mathcal{C} & \dots & \mathcal{C} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C} & \mathcal{C} & \dots & 2\mathcal{C} \end{bmatrix} (\mathbf{x}-\mathbf{y}) \geq \lambda_1(M)||\mathbf{x}-\mathbf{y}||^2 \tag{B.3.12}$$

and we know $\lambda_1(M)$ is a positive number. Thus, we have proven strong monotonicity of $\mathbf{F}$. $\square$

B.4 Counterexample and MATLAB Code for Lemma 4

We will provide a counter example to demonstrate this lemma. In MATLAB, if we set the `rng` seed to 1 with the `'twister'` option, we draw four arcs using the `rand` function and multiply these numbers by 1000. We then form a $\mathcal{C}$ matrix by putting these four numbers along the diagonal. Next, we form the following matrix:

$$A = \begin{bmatrix} 2\mathcal{C} & \mathcal{C} & \mathcal{C} & \mathcal{C} \\ (500.1)\mathcal{C} & 2(500.1)\mathcal{C} & (500.1)\mathcal{C} & (500.1)\mathcal{C} \\ (600.7)\mathcal{C} & (600.7)\mathcal{C} & 2(600.7)\mathcal{C} & (600.7)\mathcal{C} \\ (700.8)\mathcal{C} & (700.8)\mathcal{C} & (700.8)\mathcal{C} & 2(700.8)\mathcal{C} \end{bmatrix} \tag{B.4.1}$$

Therefore, we see that we have formed a four player matrix game in which all of the players do have different costs due to the different factors multiplying the $\mathcal{C}$ matrices. According to [109, 196], a matrix with real values A is positive definite if and only if $\frac{1}{2}(A + A^T)$ is positive definite. We can check the smallest eigenvalue of this resultant matrix to find out if it is positive definite. Using MATLAB's `eig` function, we discover that the smallest eigenvalue of the matrix $\frac{1}{2}(A + A^T)$ is -7.7307e+04. Therefore, A is not positive definite, thus demonstrating that, when the costs are not the same across all players, we are not guaranteed to have a strongly monotone $\mathbf{F}$ as defined by (3.3.3). See below for MATLAB code.

This is the MATLAB code utilized for the counterexample for Lemma 4:

```
rng(1,'twister');
v=1000*rand(1,4);
```

```
C=diag(v);

M=[2*C   C    C    C

   500.1*C  2*500.1*C   500.1*C     500.1*C

   600.7*C  600.7*C  2*600.7*C      600.7*C

   700.8*C   700.8*C    700.8*C    2*700.8*C];

w=eig(0.5*(M+M'));

minval=min(w)
```

B.5 Additional Experimental Set-Up Details

The PATH solver [54, 70] in GAMS generates the data used in the inverse optimization residual model [112, 154]. The only default altered for the PATH solver was the tolerance threshold for the solver to finish, which changed from 1e-6 to 1e-8. In the scripts, the PATH solver can begin at a few different starting points in case one of the start points fails due to solver error. For the case in which the costs are the same across all players, $\mathbf{F}$ defined by (3.3.3) is strictly monotone, so the problem has a unique solution, which means the starting point does not matter. For the case in which the costs are not the same across all players, multiple starting points are still provided to ensure that a solution is obtained to the problem. Interestingly, all of the different cost matrices for our experiments are positive definite, except for one in grid size 5x5 and player number 10.[1] Then, the `pyomo` [95, 96] package in Python is utilized to construct the inverse optimization residual model to find the parameterizations. The Gurobi solver [90] is

[1] In different costs, we use slightly different starting point options between when we run the simulation for the original costs and when we run the simulation for the IO costs. This doesn't appear to affect the results, as will be seen in the next few sections. We also did a check for $\alpha = 10$, grid 5x5, and $N = 10$ with the start points set to the same values for the original costs and IO costs, and the resulting errors were the same as before.

utilized to solve the pyomo models, and the BarQCPConvTol and the BarConvTol parameters are set to 1e-14. The documentation for BarConvTol states that smaller values for this parameter can lead to "a more accurate solution" [89]. The experiments are run on machines with 4 cores and 32 GB of RAM.

The original costs under which the simulation data is generated have to be selected, and this was done randomly. MATLAB [130] was utilized to generate the random sets of numbers needed, specifically the `unifrnd` random number function with 5 as the `rng` seed. This function draws from a uniform distribution the original costs for the $\mathcal{C}_i$ and $\bar{c}_i$ parameters in the simulation model outlined in (3.4.1). For the same costs across all players set-up, one $\mathcal{C}$ and one $\bar{c}$ are drawn but, when the costs vary across all the players, a $\mathcal{C}_i$ and a $\bar{c}_i$ are drawn for each player i. For each experimental set up with a specified graph, number of players, and α level (the capacity level for the "coupled" constraint), 10 trials are run.

For the different costs case, it should be noted that all of the $\mathbf{F}$ functions as defined by (3.3.3) across the number of arcs present in the grids (2x2-5x5) and in Sioux Falls as well as the number of players, with the exception of one trial in the 5x5 grid and 10 players case, are strongly monotone functions. As indicated in Lemma 4 in Appendix B.2, we cannot make this guarantee in general for players with different interaction costs. This matters because the different starting point scheme for solving the simulation model is bolstered by the positive definiteness assumption and, as can be seen in Appendix B.6, there is less of a chance that the matrix will be positive definite as the number of players increases. Therefore, this suggests further computational work that could be done to see if non-positive definite matrices of our type work in general in our simulation framework.[2]

[2]Note that the one non-positive definite matrix in the 5x5 grid with 10 players and $\alpha = 10$ example appears to

B.6 Positive Definiteness of Different Costs Matrices with MATLAB rng(5)

In Figure B.1, we plot a sampling of the minimum eigenvalues for the "symmetric part" [196] of random matrices of the form:

$$A = \begin{bmatrix} 2\mathcal{C}_1 & \mathcal{C}_1 & \dots & \mathcal{C}_1 \\ \mathcal{C}_2 & 2\mathcal{C}_2 & \dots & \mathcal{C}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C}_N & \mathcal{C}_N & \dots & 2\mathcal{C}_N \end{bmatrix} \tag{B.6.1}$$

in which $\mathcal{C}_i$ are diagonal matrices whose diagonals are chosen as random numbers from the uniform distribution from 1 to 5 using the MATLAB unifrnd function. We choose to set the arc number to 76 (same number as the Sioux Falls arc number), and we gradually increase the number of players N from 2 to 15. We do use the seed of 5 for rng but, since we draw all of the numbers in one continuous session, the minimum eigenvalues here do not necessarily match the eigenvalues of the matrices that correspond with our experiments. As discussed in Appendix B.4 and according to [109, 196], we can check the positive definiteness of the matrix A by finding the minimum eigenvalue of $0.5(A + A^T)$, which is what we do for the 10 random matrices corresponding with each player setting. In return, we obtain Figure B.1, and we see that, as player number increases, there is more and more of a chance that the minimum eigenvalue is negative, thus making the matrix not positive definite.

This is important because we can only guarantee unique solutions to our problem when the A matrix is positive definite, as explained in Section 3.3.1. This allows us to have multiple

work in the simulation framework because the flow error and objective function values are both low.

starting points in our code for the simulation experiments.

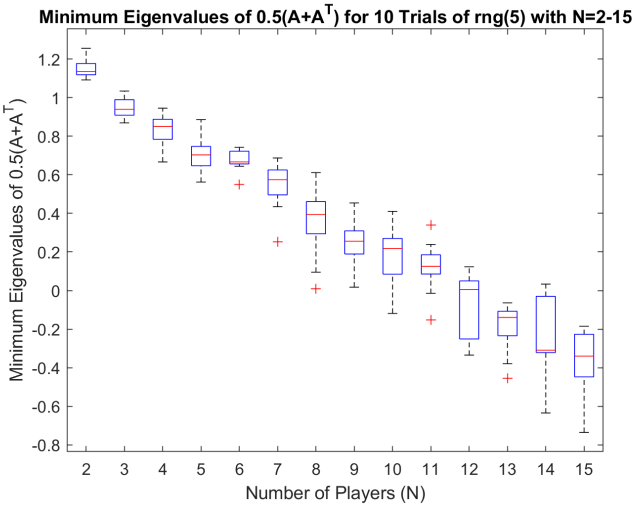


Figure B.1: Minimum Eigenvalues for 10 Trials, $N = 2-15$, rng(5), and Arc Number=76, which is the same number of arcs as Sioux Falls

B.7 Objective Function Values for the Experiment Groups

Overall, it is important to remember that, for all the experiments, the objective function values are small, on the order of 1e-6 or lower. However, it is also relevant to identify patterns where present. For experiment group 1, in Figure B.2, increasing grid size sometimes leads to a higher median objective function value in certain subsets of the experiments, including when there are 2 players and $\alpha = 1$, when there are 5 players and $\alpha = 2.5$, and when there are 10 players and $\alpha = 5$. These are all subsets in which alpha was cut in half according to the number of players present, which indicates that there could be a mild trend of increasing objective function values as grid size increases when the problem is more constrained. Overall, however, the objective function values are small, with the highest outlier in terms of absolute value being a little over 2.5e-6 in the case of a 5x5 grid with 10 players and $\alpha = 5$.[3] For experiment group 2, in

[3]Note that there are a few negative values, but this isn't concerning because their corresponding flow errors are still low.

Figure B.3, the objective values are overall quite low, with one outlier for 10 players and $\alpha = 5$ above 8e-6. They over all appear comparable to the grid objective function values once the scales of the two graphs are taken into account.[4] For experiment group 3, in Figure B.4, although the objective function values of the inverse optimization residual model are on the order of 1e-6, the median values under the $\alpha = (0.5)N$ capacity are bigger than the $\alpha = N$ median values. One theory for this trend is that the more constrained problems are more difficult to solve and hence lead to larger objective function values. The ranges for these objective function values is about 1.5e-6 higher in the positive direction than the corresponding range for the values in Experiment group 1 (see Figure B.2). For experiment group 4, Figure B.5 demonstrates that most of the values fall below 1e-6, except for the $N = 5, \alpha = 2.5$ experiment, whose maximum is close to 4e-6. This does appear to mirror the the objective function values for experiment group 2 in terms of the general range, which also fell below 1e-6.

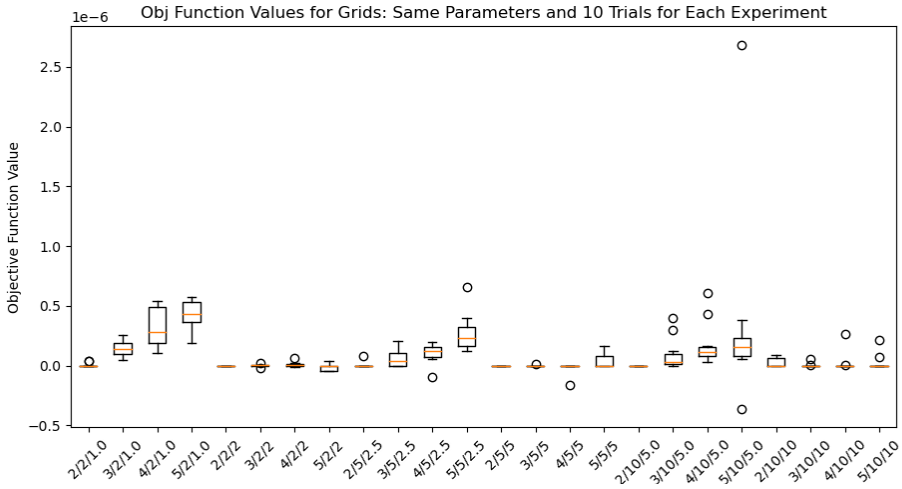


Figure B.2: Objective Function Values for Experiment Group 1: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Grid Size/Number of Players/Alpha Value

[4]There do again appear to be some negative values but, again, this isn't concerning because their corresponding flow errors are still low.

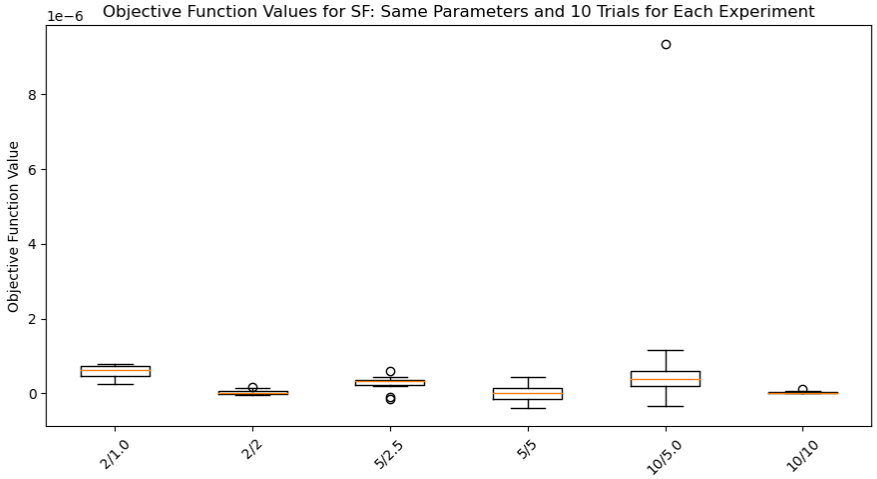


Figure B.3: Objective Function Values for Experiment Group 2: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Number of Players/Alpha Value

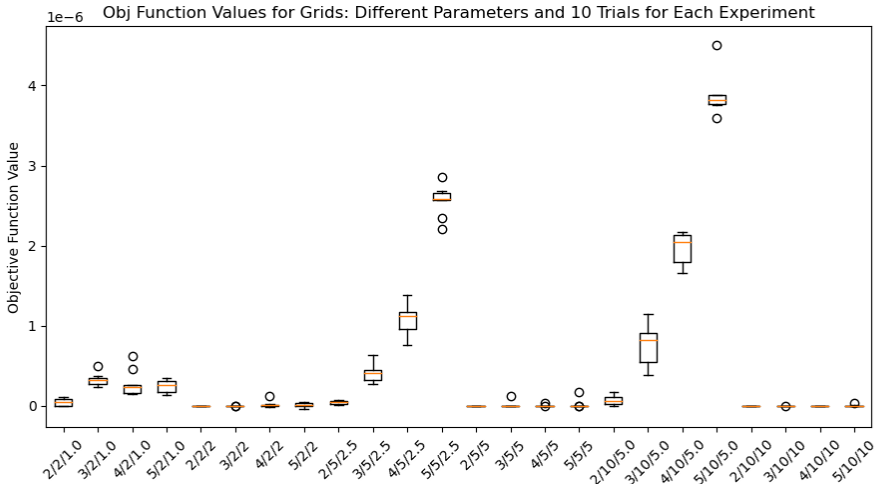


Figure B.4: Objective Function Values for Experiment Group 3: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Number of Players/Alpha Value. Note: Only 6 trials are included for 5/10/5.0, see note in the text.

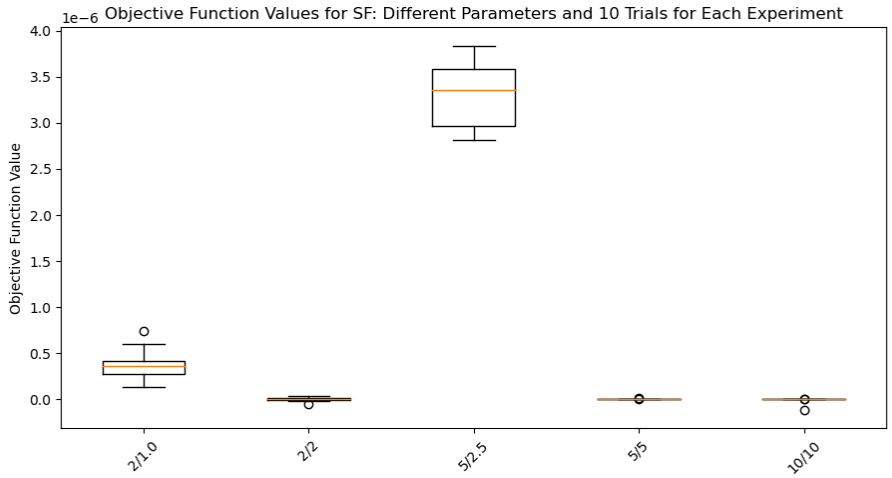


Figure B.5: Objective Function Values for Experiment Group 4: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Number of Players/Alpha Value. Note: The trials did not finish in time for $N = 10$ and $\alpha = 5$, hence that boxplot is not included.

B.8 Timing for the Experiment Groups

In this section of the Appendix, we present the timing graphs for the experiment groups. In Figure B.6, timing increases with grid size for all of the subsets of α and number of players. In Figure B.7, the timing increases as the player number increases. Both of these trends are understandable because of the effect on variable count that both grid size and player number have, as demonstrated by Appendix B.1. In Figure B.8, we see that all of the timing data is obscured by the large outlier of the experiment with grid 5x5, 10 players, and $\alpha = 5$. This experiment took quite a while and, as previously noted, only 6 of the trials for the experiment finished in under 24 hours (the 1st, 2nd, 3rd, 4th, 5th, and 7th). We hypothesize that having the largest grid, the most number of players, the more constrained α value of the two, and the more difficult task of determining different sets of cost parameters for each player caused the large time amounts. In Figure B.9, again noting that the trials did not finish in time for $N = 10$ and

$\alpha = 5$, it appears that there are not trends to discuss, although note that all the experiments but $N = 5, \alpha = 2.5$ have consistent timing data across trials.

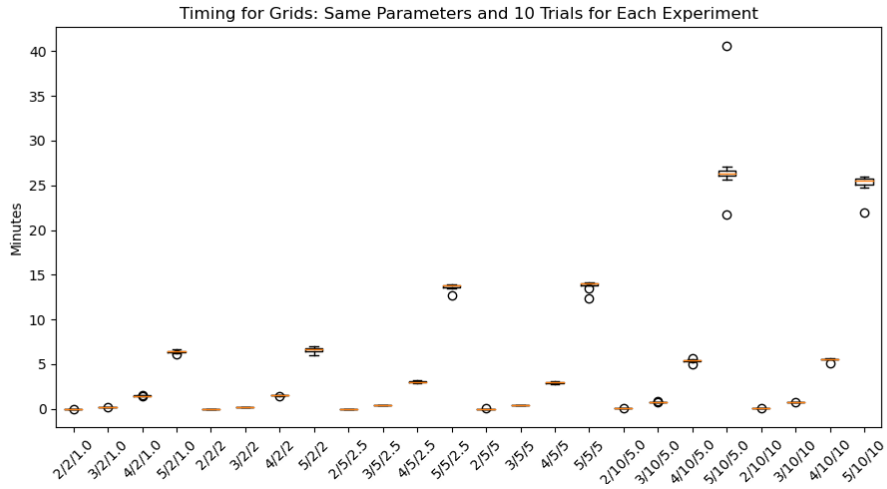


Figure B.6: Timing for Experiment Group 1: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Grid Size/Number of Players/Alpha Value

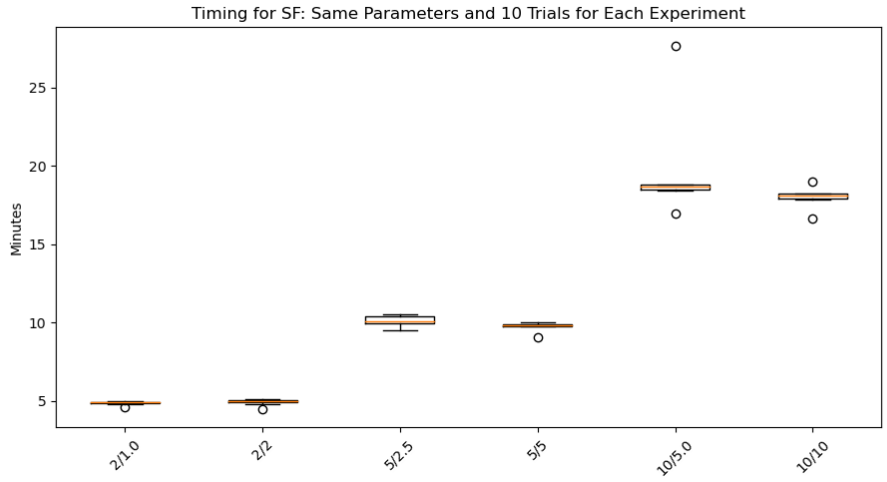


Figure B.7: Timing for Experiment Group 2: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Number of Players/Alpha Value

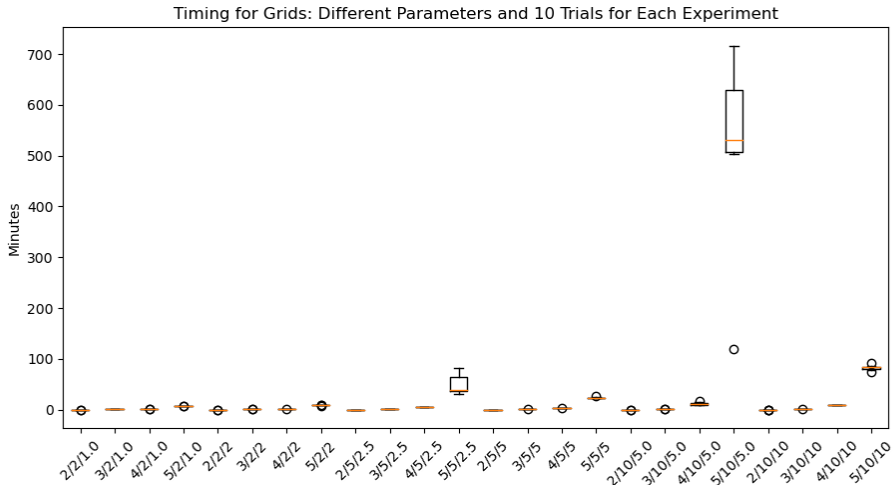


Figure B.8: Timing for Experiment Group 3: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Grid Size/Number of Players/Alpha Value. Note: Only 6 trials are included for 5/10/5.0, see note in the text.

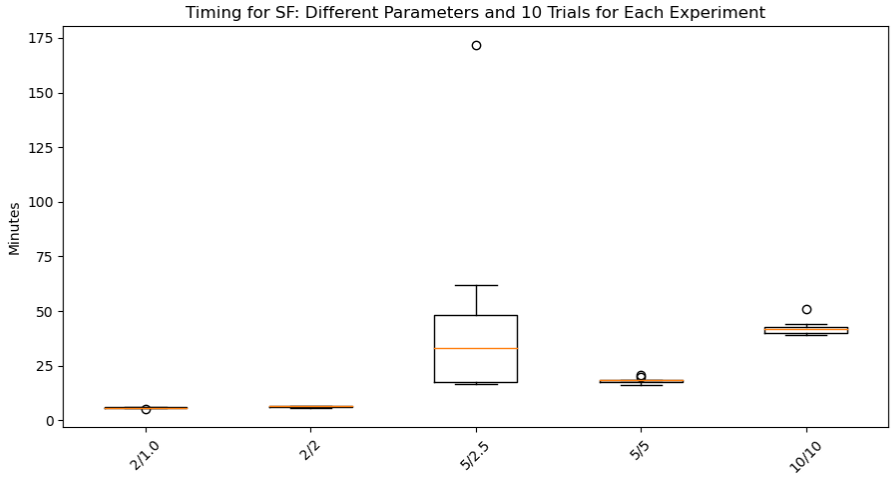


Figure B.9: Timing for Experiment Group 4: The labeling at the bottom of the graph indicates attributes of the boxplot, specifically the Number of Players/Alpha Value. Note: The trials did not finish in time for $N = 10$ and $\alpha = 5$, hence that boxplot is not included.

B.9 Part B: Additional Results

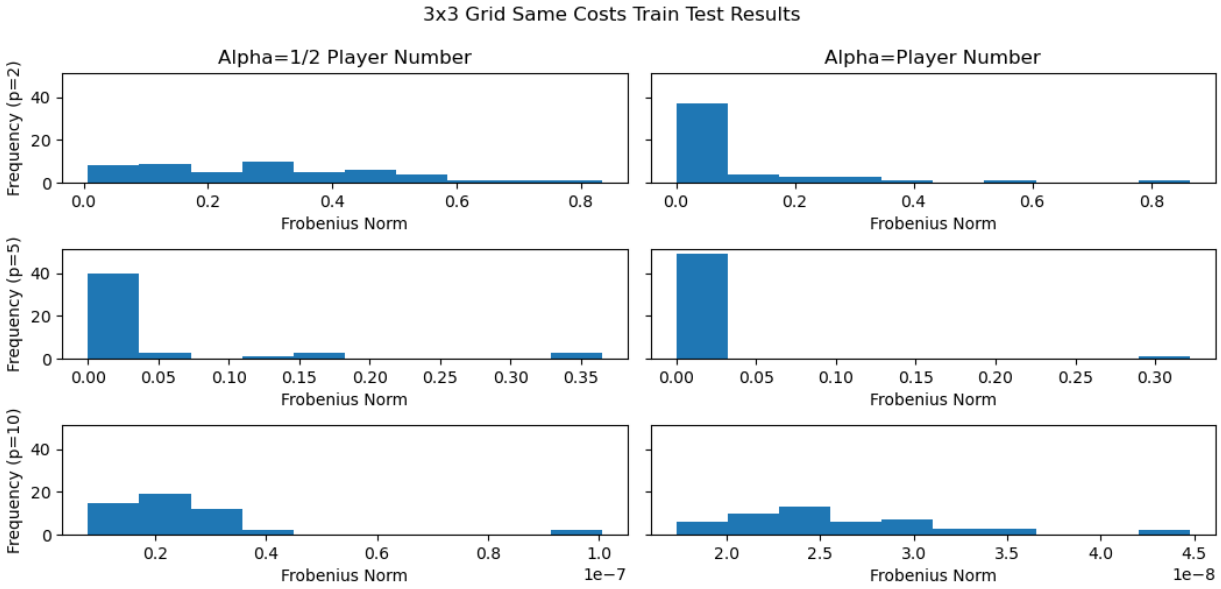


Figure B.10: 3×3 Grid Train Test Same Costs

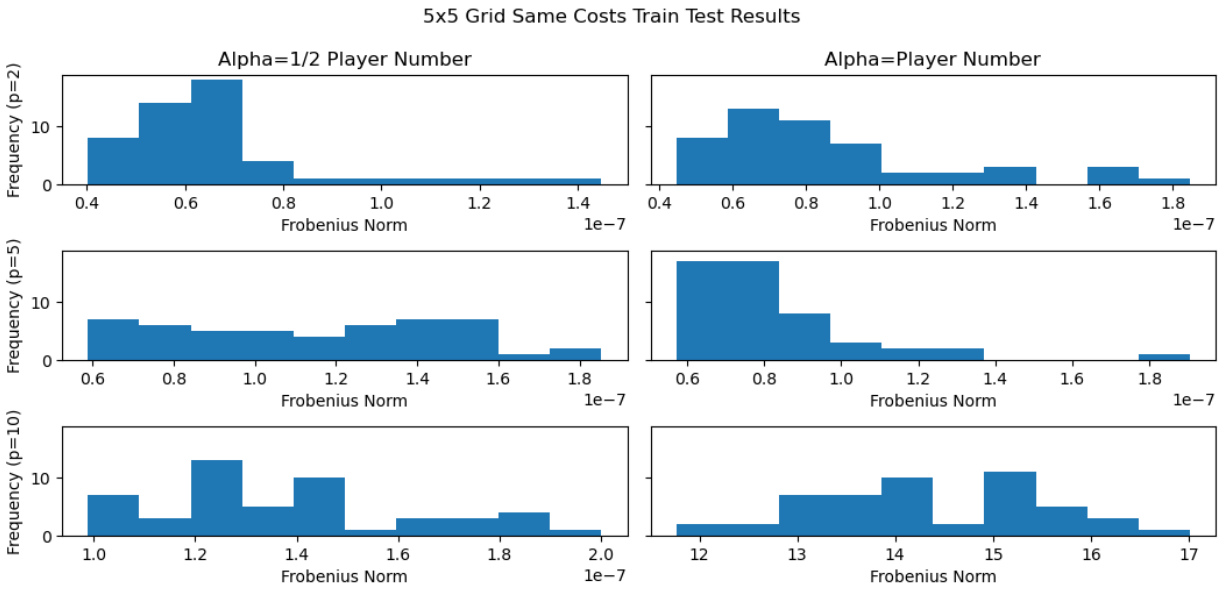


Figure B.11: 5×5 Grid Train Test Same Costs

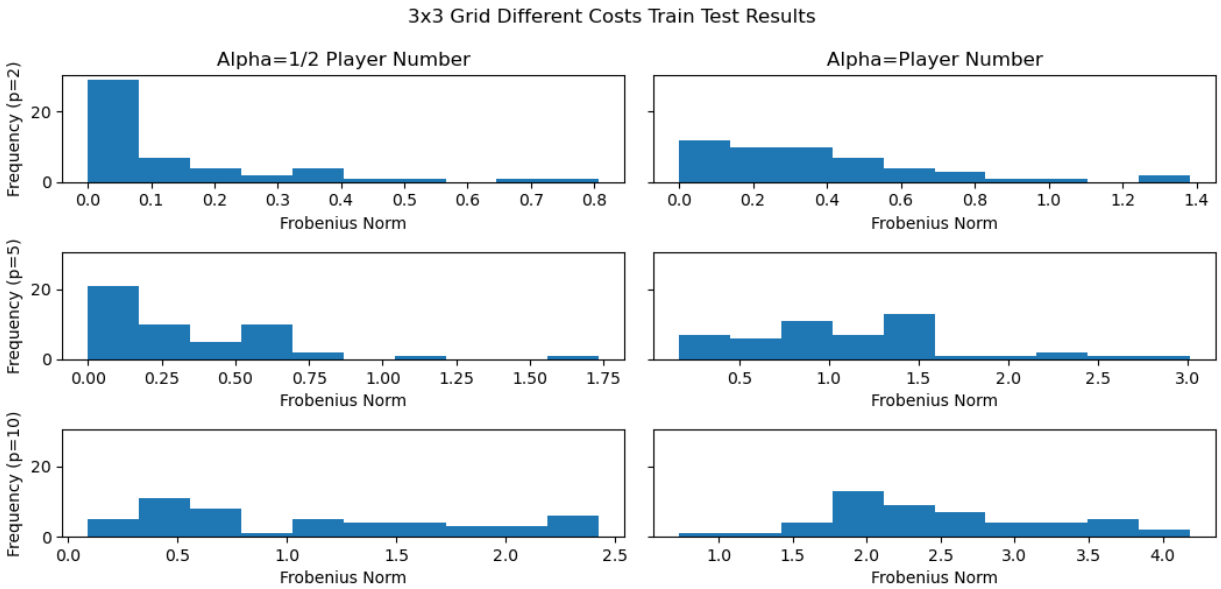


Figure B.12: 3×3 Grid Train Test Different Costs

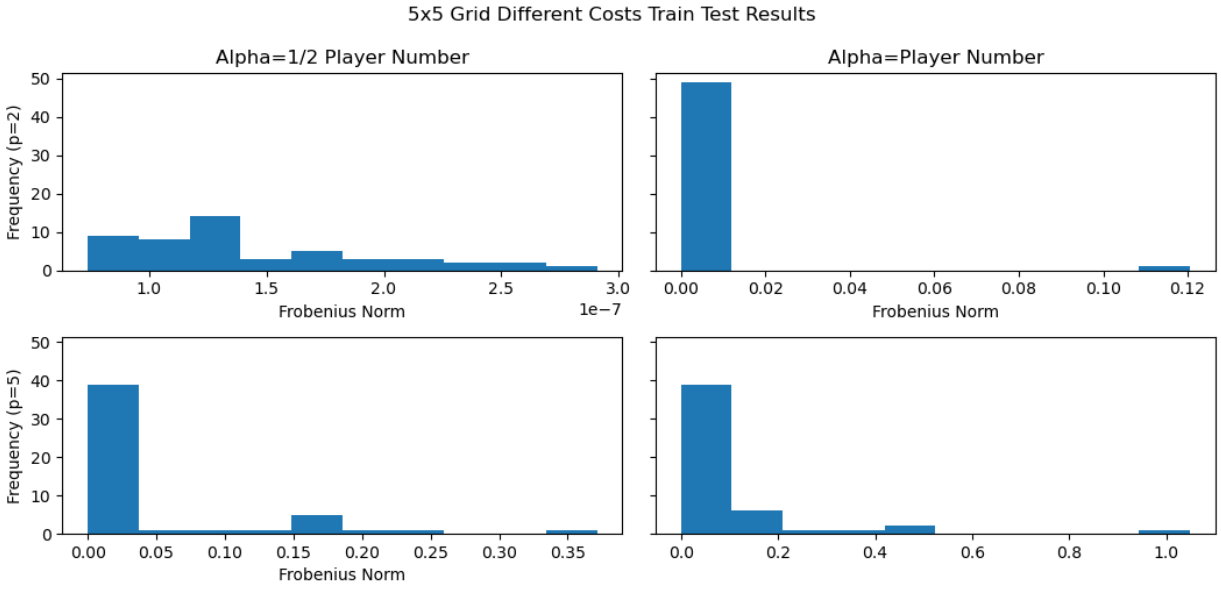


Figure B.13: 5×5 Grid Train Test Different Costs

B.10 Code Attribution

In this section, we would like to cite the various packages and software that we used in the project. We would also like to note that the code for this project was built off of another code base that is for another on-going project (not published yet).

- Python Packages:
  - `pyomo` [95, 96]
  - `networkx` [91]
  - `matplotlib` [104]
  - `numpy` [145, 184, 189]
  - `scipy` [184]
  - `pandas` [131]
- MATLAB 9.8.0.1323502 (R2020a) [130]
- GAMS [77, 78] with PATH solver [54, 70] (PATH website [71])
- Solvers:
  - `gurobi` Version 9.1.1 [90]; Gurobi Documentation [123]
- Data: Sioux Falls data [120, 179]

These are some of the helpful resources we utilized for writing our code:

- These articles [23, 128] were helpful in determining some of the constraints that equalized necessary variables in my code.

- We learned much about simulation from [55] and [17, 18], specifically about running multiple trials and the importance of randomization.

Appendix C: Inverse Optimization for Linear Complementarity Problems and for Incentive Design in Markets

C.1 Principle Minors Algorithm

In Section 4.5.2.1, we mention finding the principle minors for the M^{GCM} and M^{CSM} matrices. This is the brute-force algorithm we devised to find all of the principle minors for the two matrices:

Algorithm 6: Principle Minor Brute-Force Algorithm

Data: Initialize the M matrix.

Generate all the combinations of indices for creating principle submatrices in set $\mathcal{I}_{PM}$

Initialize determinant_list

for $i \in \mathcal{I}_{PM}$ **do**

| determinant_list.append(determinant($M[i, i]$))

end

C.2 Full q Table from Aggregate $\mathbf{z}$ Solution IO Quadratic Program Solve

	$q^{GCM} = q^{CSM}$	$q^{AGGCMIO}$, q^0 zeros	$q^{AGGCMIO}$, $q^0 = q^{GCM}$	$q^{AGCSMIO}$, q^0 zeros	$q^{AGCSMIO}$, $q^0 = q^{CSM}$
$-\alpha_1 + c_1^{ops}$	-20.0	-16.3127	-16.3127	-17.4	-17.4
$-\alpha_2 + c_2^{ops}$	-30.0	-28.0	-28.0	-28.0	-28.0
$-\alpha_3 + c_3^{ops}$	-39.9999	-40.0	-40.0	-40.0	-40.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
$c_{1,1}^{cu}$	0.6667	0.6667	0.6667	0.6667	0.6667
$c_{1,2}^{cu}$	1.0	1.0	1.0	1.0	1.0
$c_{1,3}^{cu}$	1.3333	0.0	1.3333	0.0	1.3333
$c_{2,1}^{cu}$	3.3333	3.3333	3.3333	3.3333	3.3333
$c_{2,2}^{cu}$	5.0	5.0	5.0	5.0	5.0
$c_{2,3}^{cu}$	6.6667	0.0	6.6667	0.0	6.6667
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0
$n_1 - r_1^{fc}$	10.0	6.1309	10.0	6.4933	10.0
$n_1 + n_2 - r_2^{fc}$	10.0	10.0	10.0	9.8	10.0
$n_1 + n_2 + n_3 - r_3^{fc}$	10.0	10.0	10.0	10.0	10.0

Table C.1: q Recovered from IO for GCM and CSM Markets Using One Aggregate $\mathbf{z}$ Solutions

C.3 Verifying the q for System Optimal IO Solve

We verify that the q coefficient values obtained from all of the IO quadratic program trials in Table 4.12 for both the GCM and CSM markets return the aggregate solutions when used in the GCM and CSM market LCPs. We return the GCM aggregate solution using the GCM market model in conjunction with the $q^{AGGCMIO}$ from Table 4.12. For the CSM model, using $q^{AGCSMIO}$ when q^0 was set to 0s, results in a $\mathbf{z}$ solution from the CSM market model that has

0.0001 in positions 13 and 22 in which the value should be a 0 (see Table 4.9). We attribute this to numerical precision issues. For the CSM model, using $q^{AGCSMIO}$ when q^0 was set to q^{CSM} resulted in one alteration for the W_2^P value, changing it from 1.6883 to 1.6867. As was mentioned before, the W_2^P variable can take on a range of solution values, so its change to a different value inside the expected range we reported from the starting point method is not alarming.

C.4 Code Attribution

Below are the code resources we utilized for this project:

- Python (Version 3.8.5) Packages
  - `pyomo` 5.7.1 [95, 96]
  - `pandas` 1.3.4 [131]
  - `numpy` 1.19.2 [145, 184, 189]
  - `scipy` 1.5.2 [184]
- Solvers
  - `gurobi` Version 9.1.1 [90]; Gurobi Documentation [123]
- MATLAB 9.8.0.1323502 (R2020a) [130]
- GAMS [79] with PATH solver [54, 70] (PATH website [71])

Bibliography

[1] Shadi Abpeykar and Mehdi Ghatee. Supervised and unsupervised learning dss for incident management in intelligent tunnel: A case study in tehran niayesh tunnel. *Tunnelling and Underground Space Technology*, 42:293–306, 2014.

[2] Richa Agarwal and Özlem Ergun. Network design and allocation mechanisms for carrier alliances in liner shipping. *Operations research*, 58(6):1726–1742, 2010.

[3] Ravindra K Ahuja and James B Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001.

[4] Stephanie Allen, Steven A Gabriel, and John P Dickerson. Using inverse optimization to learn cost functions in generalized nash games. *Computers & Operations Research*, page 105721, 2022.

[5] Stephanie Allen, Daria Terekhov, and Steven A Gabriel. A hybrid inverse optimization-stochastic programming framework for network protection. *arXiv preprint arXiv:2110.00488*, 2021.

[6] Daniel Alvear, Orlando Abreu, Arturo Cuesta, and Virginia Alonso. Decision support system for emergency management: Road tunnels. *Tunnelling and underground space technology*, 34:13–21, 2013.

[7] Ali Asadabadi and Elise Miller-Hooks. Optimal transportation and shoreline infrastructure investment planning under a stochastic climate future. *Transportation Research Part B: Methodological*, 100:156–174, 2017.

[8] Anil Aswani, Zuo-Jun Shen, and Auyon Siddiq. Inverse optimization with noisy data. *Operations Research*, 66(3):870–892, 2018.

[9] Anil Aswani, Zuo-Jun Max Shen, and Auyon Siddiq. Data-driven incentive design in the medicare shared savings program. *Operations Research*, 67(4):1002–1026, 2019.

[10] Chaitanya Awasthi. *Forward and inverse methods in optimal control and dynamic game theory*. PhD thesis, University of Minnesota, 2019.

[11] Chaitanya Awasthi and Andrew Lamperski. Inverse differential games with mixed inequality constraints. In *2020 American Control Conference (ACC)*, pages 2182–2187. IEEE, 2020.

[12] Pranjal Awasthi, Bahman Kalantari, and Yikai Zhang. Robust vertex enumeration for convex hulls in high dimensions. In *International Conference on Artificial Intelligence and Statistics*, pages 1387–1396. PMLR, 2018.

[13] Aaron Babier, Timothy CY Chan, Taewoo Lee, Rafid Mahmood, and Daria Terekhov. An ensemble learning framework for model fitting and evaluation in inverse linear optimization. *Informs Journal on Optimization*, 3(2):119–138, 2021.

[14] Jeff X Ban, Henry X Liu, Michael C Ferris, and Bin Ran. A general mpcc model and its solution algorithm for continuous network design problem. *Mathematical and Computer Modelling*, 43(5-6):493–505, 2006.

[15] Xuegang Jeff Ban. *Quasi-variational inequality formulations and solution approaches for dynamic user equilibria*. The University of Wisconsin-Madison, 2005.

[16] Gulay Barbarosoglu and Yasemin Arda. A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the operational research society*, 55(1):43–53, 2004.

[17] Andreas Bärmann, Alexander Martin, Sebastian Pokutta, and Oskar Schneider. An online-learning approach to inverse optimization. *arXiv preprint arXiv:1810.12997*, 2018.

[18] Andreas Bärmann, Sebastian Pokutta, and Oskar Schneider. Emulating the expert: inverse optimization through online learning. In *International Conference on Machine Learning*, pages 400–410, 2017.

[19] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[20] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. Inverse optimization: A new perspective on the black-litterman model. *Operations research*, 60(6):1389–1403, 2012.

[21] Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2):595–633, 2015.

[22] Umang Bhaskar, Katrina Ligett, Leonard J Schulman, and Chaitanya Swamy. Achieving target equilibria in network routing games without knowing the latency functions. *Games and Economic Behavior*, 118:533–569, 2019.

[23] John R Birge and Derek F Holmes. Efficient solution of two-stage stochastic linear programs using interior point methods. *Computational Optimization and Applications*, 1(3):245–276, 1992.

[24] John R Birge, Ali Hortaçsu, and J Michael Pavlin. Inverse optimization for the recovery of market structure from market outcomes: An application to the miso electricity market. *Operations Research*, 65(4):837–855, 2017.

[25] Nathan Boyd, Steven Gabriel, George Rest, and Tom Dumm. Generalized nash equilibrium models for asymmetric, non-cooperative games on line graphs: Application to water resource systems. *arXiv preprint arXiv:2206.07102*, 2022.

[26] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[27] David Branston. Link capacity functions: A review. *Transportation research*, 10(4):223–236, 1976.

[28] Wolfgang Britz, Michael Ferris, and Arnim Kuhn. Modeling water allocating institutions based on multiple optimization problems with equilibrium constraints. *Environmental Modelling & Software*, 46:196–207, 2013.

[29] Didier Burton and Ph L Toint. On an instance of the inverse shortest paths problem. *Mathematical programming*, 53(1):45–61, 1992.

[30] Didier Burton and Ph L Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1):1–22, 1994.

[31] Victor Cantillo, Luis F Macea, and Miguel Jaller. Assessing vulnerability of transportation networks for disaster response operations. *Networks and Spatial Economics*, 19(1):243–273, 2019.

[32] P-L Carpentier, Michel Gendreau, and Fabian Bastin. Long-term management of a hydroelectric multireservoir system under uncertainty using the progressive hedging algorithm. *Water Resources Research*, 49(5):2812–2827, 2013.

[33] Michal Černỳ and Milan Hladík. Inverse optimization: towards the optimal parameter set of inverse lp with interval coefficients. *Central European Journal of Operations Research*, 24(3):747–762, 2016.

[34] Timothy CY Chan, Tim Craig, Taewoo Lee, and Michael B Sharpe. Generalized inverse multiobjective optimization with application to cancer therapy. *Operations Research*, 62(3):680–695, 2014.

[35] Timothy CY Chan, Maria Eberg, Katharina Forster, Claire Holloway, Luciano Ieraci, Yusuf Shalaby, and Nasrin Yousefi. An inverse optimization approach to measuring clinical pathway concordance. *Management Science*, 68(3):1882–1903, 2022.

[36] Timothy CY Chan and Neal Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.

[37] Timothy CY Chan, Taewoo Lee, and Daria Terekhov. Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Science*, 65(3):1115–1135, 2019.

[38] Timothy CY Chan, Rafid Mahmood, and Ian Yihang Zhu. Inverse optimization: Theory and applications. *arXiv preprint arXiv:2109.03920*, 2021.

[39] Ruidi Chen, Ioannis Ch Paschalidis, and Michael C Caramanis. Strategic equilibrium bidding for electricity suppliers in a day-ahead market using inverse optimization. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 220–225. IEEE, 2017.

[40] Yang Chen and Michael Florian. Congested od trip demand adjustment problem: bilevel programming formulation and optimality conditions. In *Multilevel Optimization: Algorithms and Applications*, pages 1–22. Springer, 1998.

[41] Joseph YJ Chow and Shadi Djavadian. Activity-based market equilibrium for capacitated multimodal transport systems. *Transportation Research Procedia*, 7:2–23, 2015.

[42] Joseph YJ Chow and Will W Recker. Inverse optimization with endogenous arrival time constraints to calibrate the household activity pattern problem. *Transportation Research Part B: Methodological*, 46(3):463–479, 2012.

[43] Joseph YJ Chow, Stephen G Ritchie, and Kyungsoo Jeong. Nonlinear inverse optimization for parameter estimation of commodity-vehicle-decoupled freight assignment. *Transportation Research Part E: Logistics and Transportation Review*, 67:71–91, 2014.

[44] James C Chu and Shih-Chi Chen. Optimization of transportation-infrastructure-system protection considering weighted connectivity reliability. *Journal of Infrastructure Systems*, 22(1):04015008, 2016.

[45] Marius Cioca, Lucian-Ionel Cioca, and Sabin-Corneliu Buraga. Spatial [elements] decision support system used in disaster management. In *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference*, pages 607–612. IEEE, 2007.

[46] Roberto Cominetti, José R Correa, and Nicolás E Stier-Moses. Network games with atomic players. In *International Colloquium on Automata, Languages, and Programming*, pages 525–536. Springer, 2006.

[47] GAMS Development Corporation. Jams and logmip, 2022.

[48] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Academic Press, Inc, 1992.

[49] Teodor Gabriel Crainic, Xiaorui Fu, Michel Gendreau, Walter Rei, and Stein W Wallace. Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124, 2011.

[50] ARTURO Cuesta, DANIEL Alvear, ORLANDO Abreu, and DELFÍN Silió. Real-time stochastic evacuation models for decision support in actual emergencies. *Fire Safety Science*, 11:1063–1076, 2014.

[51] Carlos F Daganzo and Yosef Sheffi. On stochastic models of traffic assignment. *Transportation science*, 11(3):253–274, 1977.

[52] Department of Homeland Security. National infrastructure protection plan: Partnering for critical infrastructure security and resilience. `https://www.cisa.gov/sites/default/files/publications/national-infrastructure-protection-plan-2013-508.pdf`, 2013.

[53] Robert B Dial. Minimal-revenue congestion pricing part i: A fast algorithm for the single-origin case. *Transportation Research Part B: Methodological*, 33(3):189–202, 1999.

[54] Steven P Dirkse and Michael C Ferris. The path solver: a nommonotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5(2):123–156, 1995.

[55] Chaosheng Dong, Yiran Chen, and Bo Zeng. Generalized inverse optimization through online learning. In *Advances in Neural Information Processing Systems*, pages 86–95, 2018.

[56] Alper Döyen and Necati Aras. An integrated disaster preparedness model for retrofitting and relief item transportation. *Networks and Spatial Economics*, 19(4):1031–1068, 2019.

[57] CW Duin and A Volgenant. Some inverse optimization problems under the hamming distance. *European Journal of operational research*, 170(3):887–899, 2006.

[58] B Curtis Eaves. On the basic theorem of complementarity. *Mathematical Programming*, 1(1):68–75, 1971.

[59] Ronald T Eguchi, James D Goltz, Hope A Seligson, Paul J Flores, Neil C Blais, Thomas H Heaton, and Edward Bortugno. Real-time loss estimation as an emergency response decision support system: the early post-earthquake damage assessment tool (epedat). *Earthquake Spectra*, 13(4):815–832, 1997.

[60] Rune Elvik. A review of game-theoretic models of road user behaviour. *Accident Analysis & Prevention*, 62:388–396, 2014.

[61] Peyman Mohajerin Esfahani, Soroosh Shafieezadeh-Abadeh, Grani A Hanasusanto, and Daniel Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1):191–234, 2018.

[62] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. On generalized nash games and variational inequalities. *Operations Research Letters*, 35(2):159–164, 2007.

[63] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, 2010.

[64] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[65] Yueyue Fan and Changzheng Liu. Solving stochastic transportation network protection problems using the progressive hedging-based method. *Networks and Spatial Economics*, 10(2):193–208, 2010.

[66] Reza Faturechi, Shabtai Isaac, Elise Miller-Hooks, and Lei Feng. Risk-based models for emergency shelter and exit design in buildings. *Annals of Operations Research*, 262(1):185–212, 2018.

[67] Reza Faturechi and Elise Miller-Hooks. Travel time resilience of roadway networks under disaster. *Transportation research part B: methodological*, 70:47–64, 2014.

[68] FEMA. National response framework: Fourth edition. `https://www.fema.gov/sites/default/files/2020-04/NRF_FINALApproved_2011028.pdf`, October 28, 2019.

[69] Ricardo Fernández-Blanco, Juan Miguel Morales, and Salvador Pineda. Forecasting the price-response of a pool of buildings via homothetic inverse optimization. *Applied Energy*, 290:116791, 2021.

[70] Michael C Ferris and Todd S Munson. Complementarity problems in gams and the path solver. *Journal of Economic Dynamics and Control*, 24(2):165–188, 2000.

[71] Michael C. Ferris and Todd S. Munson. Path 4.7. `https://www.gams.com/latest/docs/S_PATH.html`, 2020.

[72] Audrey Fertier, Anne-Marie Barthe-Delanoë, Aurélie Montarnal, Sébastien Truptil, and Frédérick Bénaben. A new emergency decision support system: the automatic interpretation and contextualisation of events to model a crisis situation in real-time. *Decision Support Systems*, page 113260, 2020.

[73] Christian Fikar, Manfred Gronalt, and Patrick Hirsch. A decision support system for coordinated disaster relief distribution. *Expert Systems with Applications*, 57:104–116, 2016.

[74] José Fortuny-Amat and Bruce McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the operational Research Society*, 32(9):783–792, 1981.

[75] Steven A Gabriel, Antonio J Conejo, J David Fuller, Benjamin F Hobbs, and Carlos Ruiz. *Complementarity modeling in energy markets*, volume 180. Springer Science & Business Media, 2012.

[76] Steven A Gabriel, Marina Leal, and Martin Schmidt. On linear bilevel optimization problems with complementarity-constrained lower levels. *Journal of the Operational Research Society*, pages 1–11, 2021.

[77] GAMS Development Corporation. General algebraic modeling system (gams) release 33.2.0. `https://www.gams.com/download/`, 2020.

[78] GAMS Development Corporation. General algebraic modeling system (gams) release 34.1.0. https://www.gams.com/download/, 2021.

[79] GAMS Development Corporation. General algebraic modeling system (gams) release 37.1.0. https://www.gams.com/download/, 2021.

[80] John Geweke. Exact inference in the inequality constrained normal linear regression model. *Journal of Applied econometrics*, 1(2):127–141, 1986.

[81] Kimia Ghobadi, Taewoo Lee, Houra Mahmoudzadeh, and Daria Terekhov. Robust inverse optimization. *Operations Research Letters*, 46(3):339–344, 2018.

[82] Kimia Ghobadi and Houra Mahmoudzadeh. Inferring linear feasible regions using inverse optimization. *European Journal of Operational Research*, 290(3):829–843, 2021.

[83] Raphael EC Gonçalves, Erlon Cristian Finardi, and Edson Luiz da Silva. Applying different decomposition schemes using the progressive hedging algorithm to the operation planning problem of a hydrothermal system. *Electric power systems research*, 83(1):19–27, 2012.

[84] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[85] Emilia Grass and Kathrin Fischer. Two-stage stochastic programming in disaster management: A literature survey. *Surveys in Operations Research and Management Science*, 21(2):85–100, 2016.

[86] Serhat Gul, Brian T Denton, and John W Fowler. A progressive hedging approach for surgery planning under uncertainty. *INFORMS Journal on Computing*, 27(4):755–772, 2015.

[87] Anxin Guo, Zhenliang Liu, Suchao Li, and Hui Li. Seismic performance assessment of highway bridge networks considering post-disaster traffic demand of a transportation system in emergency conditions. *Structure and Infrastructure Engineering*, 13(12):1523–1537, 2017.

[88] Rishabh Gupta and Qi Zhang. Decomposition and adaptive sampling for data-driven inverse linear optimization. *arXiv preprint arXiv:2009.07961*, 2020.

[89] LLC Gurobi Optimization. Barconvtol, 2021.

[90] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[91] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[92] Patrick T Harker. Generalized nash games and quasi-variational inequalities. *European journal of Operational research*, 54(1):81–94, 1991.

[93] Patrick T Harker and Jong-Shi Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 48(1-3):161–220, 1990.

[94] William E. Hart, Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, second edition, 2017.

[95] William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, and John D Siirola. *Pyomo—Optimization Modeling in Python: Second Edition*, volume 67. Springer Optimization and Its Applications, 2017.

[96] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219, 2011.

[97] Philip Hartman and Guido Stampacchia. On some non-linear elliptic differential-functional equations. *Acta mathematica*, 115(1):271–310, 1966.

[98] Raymond Hemmecke, Shmuel Onn, and Robert Weismantel. Nash-equilibria and n-fold integer programming. *arXiv preprint arXiv:0903.4577*, 2009.

[99] Sung-Pil Hong, Kyung Min Kim, and Suk-Joon Ko. Estimating heterogeneous agent preferences by inverse optimization in a randomized nonatomic game. *Annals of Operations Research*, 307(1):207–228, 2021.

[100] Flávio EA Horita and João Porto de Albuquerque. An approach to support decision-making in disaster management based on volunteer geographic information (vgi) and spatial decision support systems (sdss). In *ISCRAM*, 2013.

[101] Flávio EA Horita, João Porto de Albuquerque, Lívia C Degrossi, Eduardo M Mendiondo, and Jó Ueyama. Development of a spatial decision support system for flood risk management in brazil that combines volunteered geographic information with wireless sensor networks. *Computers & Geosciences*, 80:84–94, 2015.

[102] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[103] Siming Huang. The inverse problems of some mathematical programming problems. *arXiv preprint arXiv:2103.15337*, 2021.

[104] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[105] John Hunter, Darren Dale, Eric Firing, Michael Droettboom, and Matplotlib development team. matplotlib.pyplot.boxplot. `https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.boxplot.html`, April 2020.

[106] Lars Magnus Hvattum and Arne Løkketangen. Using scenario trees and progressive hedging for stochastic inventory routing problems. *Journal of Heuristics*, 15(6):527, 2009.

[107] Jairo Inga, Esther Bischoff, Timothy L Molloy, Michael Flad, and Sören Hohmann. Solution sets for inverse non-cooperative linear-quadratic differential games. *IEEE Control Systems Letters*, 3(4):871–876, 2019.

[108] Sanjay Jain and Nitin Arya. An inverse capacitated transportation problem. *IOSR Journal of Mathematics*, 5(4):24–27, 2013.

[109] CHARLES R Johnson. Positive definite matrices. *The American Mathematical Monthly*, 77(3):259–264, 1970.

[110] Bahman Kalantari and Yikai Zhang. Algorithm 1024: Spherical triangle algorithm: A fast oracle for convex hull membership queries. *ACM Transactions on Mathematical Software (TOMS)*, 48(2):1–32, 2022.

[111] Arash Kaviani, Russell G Thompson, Abbas Rajabifard, Ged Griffin, and Yiqun Chen. A decision support system for improving the management of traffic networks during disasters. In *37th Australasian Transport Research Forum (ATRF), Sydney, New South Wales, Australia*, 2015.

[112] Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *2011 IEEE International Symposium on Intelligent Control*, pages 613–619. IEEE, 2011.

[113] Youngdae Kim and Michael C Ferris. Solving equilibrium problems using extended mathematical programming. *Mathematical programming computation*, 11(3):457–501, 2019.

[114] Ioannis C Konstantakopoulos, Andrew R Barkan, Shiying He, Tanya Veeravalli, Huihan Liu, and Costas Spanos. A deep learning and gamification approach to improving human-building interaction and energy efficiency in smart infrastructure. *Applied energy*, 237:810–821, 2019.

[115] Ioannis C Konstantakopoulos, Lillian J Ratliff, Ming Jin, S Shankar Sastry, and Costas J Spanos. A robust utility learning framework via inverse optimization. *IEEE Transactions on Control Systems Technology*, 26(3):954–970, 2017.

[116] András Kovács. Inverse optimization approach to the identification of electricity consumer models. *Central European Journal of Operations Research*, 29(2):521–537, 2021.

[117] Volodymyr Kuleshov and Okke Schrijvers. Inverse game theory: Learning utilities in succinct games. In *International Conference on Web and Internet Economics*, pages 413–427. Springer, 2015.

[118] Ibad Kureshi, Georgios Theodoropoulos, Eleni Mangina, Gregory O'Hare, and John Roche. Towards an info-symbiotic decision support system for disaster risk management. In *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 85–91. IEEE, 2015.

[119] Amina Lamghari and Roussos Dimitrakopoulos. Progressive hedging applied as a metaheuristic to schedule production in open-pit mines accounting for reserve uncertainty. *European Journal of Operational Research*, 253(3):843–855, 2016.

[120] Larry J LeBlanc, Edward K Morlok, and William P Pierskalla. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation research*, 9(5):309–318, 1975.

[121] Jonathan Yu-Meng Li. Inverse optimization of convex risk functions. *Management Science*, 67(11):7113–7141, 2021.

[122] Chong Kiew Liew. Inequality constrained least-squares estimation. *Journal of the American Statistical Association*, 71(355):746–751, 1976.

[123] Gurobi Optimization LLC. Documentation. `https://www.gurobi.com/documentation/`, 2020.

[124] Jie Lu, Sez Atamturktur, and Yongxi Huang. Bi-level resource allocation framework for retrofitting bridges in a transportation network. *Transportation Research Record*, 2550(1):31–37, 2016.

[125] Jie Lu, Akshay Gupte, and Yongxi Huang. A mean-risk mixed integer nonlinear program for transportation network protection. *European Journal of Operational Research*, 265(1):277–289, 2018.

[126] Zhaoyang Lu, Qiang Meng, and Gabriel Gomes. Estimating link travel time functions for heterogeneous traffic flows on freeways. *Journal of Advanced Transportation*, 50(8):1683–1698, 2016.

[127] Paramet Luathep, Agachai Sumalee, William HK Lam, Zhi-Chun Li, and Hong K Lo. Global optimization method for mixed transportation network design problem: a mixed-integer linear programming approach. *Transportation Research Part B: Methodological*, 45(5):808–827, 2011.

[128] Irvin J Lustig, John M Mulvey, and Tamra J Carpenter. Formulating two-stage stochastic programs for interior point methods. *Operations Research*, 39(5):757–770, 1991.

[129] Patrice Marcotte and Michael Patriksson. Traffic equilibrium. *Handbooks in Operations Research and Management Science*, 14:623–713, 2007.

[130] MATLAB. *Version 9.8.0.1323502 (R2020a)*. The MathWorks, Inc., Natick, Massachusetts, 2020.

[131] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[132] Peyman Mohajerin Esfahani, Soroosh Shafieezadeh-Abadeh, Grani A Hanasusanto, and Daniel Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1):191–234, 2018.

[133] Reza Mohammadi, SMT Fatemi Ghomi, and F Jolai. Prepositioning emergency earthquake response supplies: A new multi-objective particle swarm optimization algorithm. *Applied Mathematical Modelling*, 40(9-10):5183–5199, 2016.

[134] SR Mohan. Degeneracy in linear complementarity problems: a survey. *Annals of Operations Research*, 46(1):179–194, 1993.

[135] Timothy L Molloy, Jason J Ford, and Tristan Perez. Inverse noncooperative differential games. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5602–5608. IEEE, 2017.

[136] John M Mulvey and Hercules Vladimirou. Applying the progressive hedging algorithm to stochastic generalized networks. *Annals of Operations Research*, 31(1):399–424, 1991.

[137] Katta G Murty. *Linear programming*. Springer, 1983.

[138] Koichi Nabetani, Paul Tseng, and Masao Fukushima. Parametrized variational inequality approaches to generalized nash equilibrium problems with shared constraints. *Computational Optimization and Applications*, 48(3):423–452, 2011.

[139] NCEI. U.s. billion-dollar weather and climate disasters. `https://www.ncdc.noaa.gov/billions/`, DOI: `10.25921/stkw-7w73`, 2021.

[140] Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 1–18, 2015.

[141] Dinh Hoa Nguyen. Cooperative learning for p2p energy trading via inverse optimization and interval analysis. *arXiv preprint arXiv:2011.02609*, 2020.

[142] Sang Nguyen and Clermont Dupuis. An efficient method for computing traffic equilibria in networks with asymmetric transportation costs. *Transportation Science*, 18(2):185–202, 1984.

[143] Thai Dung Nguyen. *Application of robust and inverse optimization in transportation*. PhD thesis, Massachusetts Institute of Technology, 2010.

[144] Nilay Noyan. Risk-averse two-stage stochastic programming with an application to disaster management. *Computers & Operations Research*, 39(3):541–559, 2012.

[145] Travis E. Oliphant. *A guide to NumPy*. USA: Trelgol Publishing, 2006.

[146] Ariel Orda, Raphael Rom, and Nahum Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on networking*, 1(5):510–521, 1993.

[147] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[148] Reece P Otsuka, Daniel B Work, and Junho Song. Estimating post-disaster traffic conditions using real-time data streams. *Structure and Infrastructure Engineering*, 12(8):904–917, 2016.

[149] Olafur P Palsson and Hans F Ravn. Stochastic heat storage problem—solved by the progressive hedging algorithm. *Energy conversion and management*, 35(12):1157–1171, 1994.

[150] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[151] Alexander Peysakhovich, Christian Kroer, and Adam Lerer. Robust multi-agent counterfactual prediction. In *Advances in Neural Information Processing Systems*, pages 3077–3087, 2019.

[152] World Food Programme. 2.3 nepal road network. `https://dlca.logcluster.org/display/public/DLCA/2.3+Nepal+Road+Network`, 2019.

[153] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE, 2013.

[154] Lillian J Ratliff, Ming Jin, Ioannis C Konstantakopoulos, Costas Spanos, and S Shankar Sastry. Social game for building energy efficiency: Incentive design. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1011–1018. IEEE, 2014.

[155] Simon Risanger, Stein-Erik Fleten, and Steven A Gabriel. Inverse equilibrium analysis of oligopolistic electricity markets. *IEEE Transactions on Power Systems*, 35(6):4159–4166, 2020.

[156] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[157] Oscar Rodríguez-Espíndola, Pavel Albores, and Christopher Brewster. Disaster preparedness in humanitarian logistics: A collaborative approach for resource management in floods. *European Journal of Operational Research*, 264(3):978–993, 2018.

[158] J Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

[159] Tim Roughgarden. Routing games. In *Algorithmic Game Theory*. Cambridge University Press.

[160] Sarah M Ryan, Roger J-B Wets, David L Woodruff, César Silva-Monroy, and Jean-Paul Watson. Toward scalable, parallel progressive hedging for stochastic unit commitment. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, pages 1–5. IEEE, 2013.

[161] Javier Saez-Gallego and Juan M Morales. Short-term forecasting of price-responsive loads using inverse optimization. *IEEE Transactions on Smart Grid*, 9(5):4805–4814, 2017.

[162] Navid Sahebjamnia, S Ali Torabi, and S Afshin Mansouri. A hybrid decision support system for managing humanitarian relief chains. *Decision Support Systems*, 95:12–26, 2017.

[163] Daniel A Schult and P Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in science conferences (SciPy 2008)*, volume 2008, pages 11–16. Pasadena, CA, 2008.

[164] Zahed Shahmoradi and Taewoo Lee. Quantile inverse optimization: Improving stability in inverse linear programming. *Operations Research*, 2021.

[165] Yosef Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice-Hall, Inc., 1985.

[166] Enrico Siri, Silvia Siri, and Simona Sacone. A progressive traffic assignment procedure on networks affected by disruptive events. In *2020 European Control Conference (ECC)*, pages 130–135. IEEE, 2020.

[167] Michael J Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research Part B: Methodological*, 13(4):295–304, 1979.

[168] Oliver Stein and Nathan Sudermann-Merx. The noncooperative transportation problem and linear generalized nash games. *European Journal of Operational Research*, 266(2):543–553, 2018.

[169] Josef Stoer. On the numerical solution of constrained least-squares problems. *SIAM journal on Numerical Analysis*, 8(2):382–411, 1971.

[170] Gilbert Strang. *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006.

[171] K Tai, A Kizhakkedath, J Lin, RLK Tiong, and MS Sim. Identifying extreme risks in critical infrastructure interdependencies. In *Proceedings of the International Symposium of Next Generation Infrastructure*, pages 1–4. SMART Infrastructure Facility, University of Wollongong, Australia, 2013.

[172] Raymond R Tan and Kathleen B Aviso. An inverse optimization approach to inducing resource conservation in eco-industrial parks. In *Computer Aided Chemical Engineering*, volume 31, pages 775–779. Elsevier, 2012.

[173] Yingcong Tan, Andrew Delong, and Daria Terekhov. Deep inverse optimization. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 540–556. Springer, 2019.

[174] Yingcong Tan, Daria Terekhov, and Andrew Delong. Learning linear programs from optimal decisions. *arXiv preprint arXiv:2006.08923*, 2020.

[175] Jérôme Thai and Alexandre M Bayen. Learnability of edge cost functions in routing games. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6422–6429. IEEE, 2017.

[176] Jérôme Thai and Alexandre M Bayen. Imputing a variational inequality function or a convex objective function: A robust approach. *Journal of Mathematical Analysis and Applications*, 457(2):1675–1695, 2018.

[177] Jérôme Thai, Rim Hariss, and Alexandre Bayen. A multi-convex approach to latency inference and control in traffic equilibria from sparse data. In *2015 American Control Conference (ACC)*, pages 689–695. IEEE, 2015.

[178] Ezio Todini. An operational decision support system for flood risk mapping, forecasting and management. *Urban Water*, 1(2):131–143, 1999.

[179] Transportation Networks for Research Core Team. Transportation networks for research. `https://github.com/bstabler/TransportationNetworks`, June 2020.

[180] Urban Planning Division US Department of Commerce. Bureau of public roads: Traffic assignment manual, 1964.

[181] René van den Brink, Gerard van der Laan, and Valeri Vasil'ev. Component efficient solutions in line-graph games with applications. *Economic Theory*, 33(2):349–364, 2007.

[182] Kasper van Zuilekom, Martin van Maarseveen, and Marcel van der Doef. A decision support system for preventive evacuation of people. In *Geo-information for disaster management*, pages 229–253. Springer, 2005.

[183] Fernando Badilla Veliz, Jean-Paul Watson, Andres Weintraub, Roger J-B Wets, and David L Woodruff. Stochastic optimization models in forest planning: a progressive hedging solution approach. *Annals of Operations Research*, 232(1):259–274, 2015.

[184] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[185] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[186] Andreas Waechter. [ipopt] optimality guarantees for convex problems? https://list.coin-or.org/pipermail/ipopt/2009-November/001733.html.

[187] William A Wallace and Frank De Balogh. Decision support systems for disaster management. *Public Administration Review*, pages 134–146, 1985.

[188] Raymond H. Myers Sharon L. Myers Keying Ye Walpole, Ronald E. *Probability & Statistics for Engineers & Scientists: Ninth Edition*. Prentice Hall: Pearson Education Inc, 2012.

[189] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[190] John Glen Wardrop. Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1(3):325–362, 1952.

[191] Jean-Paul Watson and David L Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370, 2011.

[192] Jean-Paul Watson, David L Woodruff, and William E Hart. Pysp: modeling and solving stochastic programs in python. *Mathematical Programming Computation*, 4(2):109–149, 2012.

[193] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. Computational rationalization: The inverse equilibrium problem. *28th International Conference on Machine Learning*, 2011.

[194] Kevin Waugh, Brian D Ziebart, and J Andrew Bagnell. Computational rationalization: The inverse equilibrium problem. *arXiv preprint arXiv:1308.3506*, 2013.

[195] Eric W. Weisstein. Matrix norm. https://mathworld.wolfram.com/MatrixNorm.html, From MathWorld–A Wolfram Web Resource.

[196] Eric W. Weisstein. Positive definite matrix. https://mathworld.wolfram.com/PositiveDefiniteMatrix.html, From MathWorld–A Wolfram Web Resource.

[197] Wayne L. Winston. *Operations Research: Applications and Algorithms*. Wadsworth Inc., Belmont, CA, 1994.

[198] Wolfram—Alpha. Wolframalpha computational intelligence. https://www.wolframalpha.com/, 2021.

[199] Wai Wong and SC Wong. Network topological effects on the macroscopic bureau of public roads function. *Transportmetrica A: Transport Science*, 12(3):272–296, 2016.

[200] Susan Jia Xu, Mehdi Nourinejad, Xuebo Lai, and Joseph YJ Chow. Network learning via multiagent inverse transportation problems. *Transportation Science*, 52(6):1347–1364, 2018.

[201] Zhongzhen Yang, Liquan Guo, and Zaili Yang. Emergency logistics for wildfire suppression based on forecasted disaster evolution. *Annals of Operations Research*, 283(1):917–937, 2019.

[202] Zafer Yilmaz, Ayyuce Aydemir-Karadag, and Serpil Erol. Finding optimal depots and routes in sudden-onset disasters: An earthquake case for erzincan. *Transportation journal*, 58(3):168–196, 2019.

[203] He Zhang, Yuelong Su, Lihui Peng, and Danya Yao. A review of game theory applications in transportation analysis. In *2010 International Conference on Computer and Information Application*, pages 152–157. IEEE, 2010.

[204] Hongjun Zhang and Stephen G Ritchie. Real-time decision-support system for freeway management and control. *Journal of Computing in Civil Engineering*, 8(1):35–51, 1994.

[205] Jian-zhong Zhang, Jin-bao Jian, and Chun-ming Tang. Inverse problems and solution methods for a class of nonlinear complementarity problems. *Computational Optimization and Applications*, 49(2):271–297, 2011.

[206] Jing Zhang and Ioannis Ch Paschalidis. Data-driven estimation of travel latency cost functions via inverse optimization in multi-class transportation networks. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6295–6300. IEEE, 2017.

[207] Jing Zhang, Sepideh Pourazarm, Christos G Cassandras, and Ioannis Ch Paschalidis. The price of anarchy in transportation networks: Data-driven evaluation and reduction strategies. *Proceedings of the IEEE*, 106(4):538–553, 2018.

[208] Yu-Jun Zheng and Hai-Feng Ling. Emergency transportation planning in disaster relief supply chain management: a cooperative fuzzy optimization approach. *Soft Computing*, 17(7):1301–1314, 2013.

[209] Ying Zhou, Lizhi Wang, and James D McCalley. Effective incentives design for renewable energy generation expansion planning: An inverse optimization approach. In *IEEE PES General Meeting*, pages 1–7. IEEE, 2010.