A hybrid inverse optimization-stochastic programming framework for network protection

Stephanie Allen\textsuperscript{a}, Daria Terekhov\textsuperscript{b} and Steven A. Gabriela

\textsuperscript{a}University of Maryland, College Park, MD, USA; \textsuperscript{b}Concordia University, Montreal, Canada

ABSTRACT
Disaster management is a complex problem demanding sophisticated modeling approaches. We propose utilizing a hybrid method involving inverse optimization to parameterize the cost functions for a road network's traffic equilibrium problem and employing a modified version of a two-stage stochastic model to make protection decisions using the information gained from inverse optimization. Inverse optimization allows users to take observations of solutions of optimization and/or equilibrium problems and estimate the parameter values of the functions defining them. In the case of multi-stage stochastic programs for disaster relief, using inverse optimization to parameterize the cost functions can prevent users from making incorrect protection decisions. We demonstrate the framework using two types of cost functions for the traffic equilibrium problem and two different networks. We showcase the value of inverse optimization by demonstrating that, in most of the experiments, different decisions are made when the stochastic network protection problem is parameterized by inverse optimization versus when it is parameterized using a uniform cost assumption. We also demonstrate that similar decisions are made when the stochastic network protection problem is parameterized by inverse optimization versus when it is parameterized by the original/“true” cost parameters.

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 NCEI (2021), 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” (FEMA, 2019), 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.

As a response to these events, researchers have developed disaster support systems (DSS), which include systems that evacuate people out of tunnels (Alvear et al., 2013), deliver supplies to affected individuals (Fikar et al., 2016), decide where to place supplies in anticipation of a disaster (Rodriguez-Espíndola et al., 2018), and extinguish wildfires (Yang et al., 2019). Wallace & De Balogh (1985) 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 paper 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 (Department of Homeland Security, 2013) 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 paper, 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 (Ahuja & Orlin, 2001; Bertsimas et al., 2015; Zhang et al., 2011). 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 paper, 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 (Ahuja & Orlin, 2001; Bertsimas et al. 2015) 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 Goodfellow et al., 2016) for more information on deep learning). There are techniques such as structural estimation that can be compared to inverse optimization but, as Bertsimas et al. (2015) 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 Bertsimas et al. (2015) for more details).

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. This paper’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 paper is organized as follows. Section 2 investigates the literature related to our problem. Section 3 provides appropriate mathematical background. Section 4 explains the experimental structure, and Section 5 explains the results. Section 6 discusses our conclusions and ideas for future work.

2. Literature review

The literature in this section demonstrates that multi-stage disaster relief models and DSS models have not used inverse optimization previously. We also start by reviewing general inverse optimization literature to provide context for this paper and the field of inverse optimization.

2.1. General inverse optimization

Inverse optimization involving multi-point data sets is a growing field with researchers examining the problem from multiple angles. Our efforts in this paper are to introduce inverse optimization to the disaster relief and multi-stage program for disaster relief community, but it is also important to provide readers with context for the state of the literature at the present time to aid in the expansion of our work in the future. We focus in this review on papers parameterizing objective functions, but there has been some fruitful recent work in parameterizing feasible regions as well [see Ghalbadi and Mahmoudzadeh (2021) and Chan and Kaw (2020) for more on this]. Keshavarz et al. (2011) is an early work that proposes minimizing the residuals of the KKT conditions in finding a parameterization for an objective function. Bertsimas et al. (2015) extend multi-point inverse optimization to the class of variational inequality and thus equilibrium problems. Aswani et al. (2018) provide a bilevel formulation for finding the parameters of the objective function.
in which the upper level minimizes the distance between noisy observations of solutions and “denoised” solutions and in which the lower level find the solution of the optimization problem being parameterized. They prove that this formulation is risk-consistent, meaning, as data increases, it finds the parameters for the IO problem that minimize the error between the denoised and the noisy data as if one had the true distribution of the noisy data. They also prove this formulation is estimation-consistent, meaning that, if there is a unique parameterization to be found, this formulation finds it. Mohajerin Esfahani et al. (2018) use a risk measurement (such as CVaR) constructed according to a sample data distribution in a “worst case risk” problem in which the risk, according to out-of-sample data, is minimized in the space of probability distributions that are within a given Wasserstein distance from the sample data distribution. This creates a bound on the “out-of-sample risk” with some probability. This is all in an effort to handle data obtained in situations with imperfect information, namely incorrect forms of the objective function, human error, and noise. Babier et al. (2020) provide a framework for multi-point inverse optimization in which they expand upon their single-point geometric work from Chan et al. (2019).

Li (2021) engages in inverse optimization for risk functions in stochastic programming applications in which one knows the decisions of people under the different potential outcomes of the random variable(s). This is not completely realistic for our application because we would need to know the traffic patterns given all scenarios of disasters we propose, but it is interesting research in the area of IO for stochastic programming applications and could potentially be used to extend our work. Shahmorad and Lee (2022) define two important concepts for inverse optimization: inverse stability and forward stability. Inverse stability contends that an inverse optimization model has higher stability if the minimum distance between the input data set and some perturbation of the data set such that the inverse optimization model returns none of the same cost vectors between the input and the perturbation is higher. Forward stability states that a problem with a specific cost vector is more forward stable if the maximum distance between the forward solutions to the problem with that cost vector and the original data set is lower. Shahmorad and Lee (2022) then design a mathematical program whose feasible region increases inverse stability through quantile regression (in which they have a subset/fraction/quantile of the data have “optimality errors” below a certain bound) and whose objective function increases forward stability.

This literature provides some important ideas for future inverse optimization studies that deal with real data in the inverse optimization for disaster relief context. The point of this paper is to introduce inverse optimization to disaster relief and multi-stage for disaster relief researchers. We choose Bertsimas et al. (2015)’s inverse optimization framework because it is one of the only general approaches from the literature that handles equilibrium problems, a key part of the multi-stage network protection model we use in this paper.

2.2. 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. (2015) 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 and Bayen (2018) use a combination of methods by Bertsimas et al. (2015) and Chen and Florian (1998) 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. (2015) use their inverse variational inequality problem along with kernel methods to estimate the cost functions. Zhang and Paschalidis (2017) and Zhang et al. (2018) follow Bertsimas et al. (2015)’s methodology, with Zhang and Paschalidis (2017) involving different categories of vehicles and Zhang et al. (2018) emphasizing recovering both cost function and origin-destination matrices from real-world traffic data. Chow et al. (2014) use techniques from Ahuja and Orlin (2001) but augment them to handle the non-linear nature of their problem. Hong et al. (2017) use inverse optimization for “taste heterogeneity” parameters in a routing game in which there is a probability distribution over these parameters for travelers. Hong et al. (2021) then build upon this work for “discontinuous taste parameters” and produce a biquadratic program. Finally, Allen et al. (2022) extend Ratliff et al. (2014)’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.
2.3. Multi-stage disaster relief models

There is a substantial literature on multi-stage stochastic programs for disaster relief and protection; see Grass and Fischer (2016). 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 3.1.1), and stochastic programming. None of them use inverse optimization to estimate costs, which is what we propose in this paper.

Zheng and Ling (2013) 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 (2004) use road information and Euclidean distances between points for their cost parameters in their stochastic model pertaining to distributing supplies after natural disasters. Chu and Chen (2016) 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 (2012) and Doyen and Aras (2019) use a mixture of road data along with scenario dependent costs to form their cost functions, while Mohammadi et al. (2016) use exclusively scenario dependent costs.

Fan and Liu (2010) 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 (Asadabadi & Miller-Hooks, 2017; Futurechi et al., 2018; Futurechi & Miller-Hooks, 2014; Lu et al., 2016, 2018). 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.

2.4. 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 (Alvare et al., 2013; Cuesta et al., 2014; Sahebjamnia et al., 2017; Eguchi et al., 1997; Fikar et al., 2016; Todini, 1999; Yilmaz et al., 2019; Kureshi et al., 2015; Yang et al., 2019; van Zuilekom et al., 2005). Second, DSS papers can also determine parameters via data processing as in Fertier et al. (2020), Horita et al. (2015), and Zhang and Ritchie (1994), or they can utilize machine learning to determine modeling structures as in Abpeykar and Ghatee (2014). Other DSS papers use geographic information system (GIS) techniques to estimate parameters (Cioca et al., 2007; Rodriguez-Espindola et al., 2018). Horita and de Albuquerque (2013) combine GIS and sensor information to estimate parameters. In addition, some papers propose data fusion techniques such as ensemble Kalman filters (Otsuka et al. (2016) and gradient based methods (Kaviani et al., 2015). Inverse optimization allows a user to propose a model of the system and parameterize the model using data/simulated solutions and optimality conditions (Ahuja & Orlin, 2001; Bertsimas et al. 2015; Chan et al., 2019; Zhang et al., 2011), 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.

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. (2015)’s 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. The normative model comes from Fan and Liu (2010) 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 $\theta$ representing the collection of parameters to be estimated by the inverse optimization model:

1. Input data $x^j, j = 1..J$ into inverse optimization model (4) and obtain $\theta$
2. Form stochastic network protection problem (SNPP) (8) with $\theta$
3. Solve the SNPP (8) and obtain protection decisions $u$.

3.1. Data analysis component: Inverse optimization

Bertsimas et al. (2015) utilize variational inequalities (VI) to represent optimization and equilibrium problems. Bertsimas et al. (2015) assume that the following extended variational inequality describes
the $\epsilon$ equilibrium of a system, with $F : \mathbb{R}^n \to \mathbb{R}^n, F \subseteq \mathbb{R}^n, x \in \mathcal{F}$, and $\epsilon \in \mathbb{R}_+$:

$$F(x)^T(x - x^*) \geq -\epsilon, \forall x \in \mathcal{F}. \quad (1)$$

When $\epsilon = 0$, we recover the classical VI. 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 (Lu et al. 2016, 2018). Therefore, expression (1) states that $x^*$ solves the VI if the inner product between $F$ at $x^*$ and the difference between any point in $\mathcal{F}$ and $x^*$ is greater than a small, negative number.

Bertsimas et al. (2015) describe the Wardrop traffic equilibrium with nodes $N$ and arcs $\mathcal{A}$ as having a node-arc incidence matrix $N \in \{-1, 0, 1\}^{[W] \times [A]}$ (Marcotte & Patriksson, 2007), vectors $d^w$ 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 (Marcotte & Patriksson, 2007), and a feasible set $\mathcal{F}$. Bertsimas et al. (2015) define the $\mathcal{F}$ set as:

$$\mathcal{F} = \left\{ x : \exists x^w \in \mathbb{R}_+^{[A]}; s.t. x = \sum_{w \in \mathcal{W}} x^w, N x^w = d^w \forall w \in \mathcal{W} \right\} \quad (2)$$

in which $x^w \in \mathbb{R}_+^{[A]}$ represents the flow between origin and destination $w$ and $x \in \mathbb{R}_+^{[A]}$ represents the composite flow vector. The corresponding $F$ function for the variational inequality is defined as $c(x)$ such that $c_a : \mathbb{R}_+^{[A]} \to \mathbb{R}_+$ 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 $w$ (Fan and Liu 2010). Therefore, we need to turn the equalities in the $\mathcal{F}$ set into inequalities (Ban et al., 2006; Ban 2005). Respecting the definition of $N$ and $d^w$, our traffic equilibrium problem in complementarity form is then (Ban et al., 2006; Ban 2005; Bertsimas et al. 2015; Fan & Liu 2010; Gabriel et al., 2012; Sheffi, 1985).:

$$0 \leq c(x) + N^T y^w \perp x^w \geq 0, \forall w \in \mathcal{W} \quad (3a)$$

$$0 \leq d^w - N x^w \perp y^w \geq 0, \forall w \in \mathcal{W} \quad (3b)$$

We can show that $d^w - N x^w = 0$ when there is a solution for (3) and when we assume that the $c(x)$ function is greater than 0 for all $x \geq 0$ in a proof which is adapted from Ban (2005). See Appendix A.1. We use (3) to generate data for the inverse optimization part of the framework, and the data generation process can be found in Section 4.1.

We take a moment here to discuss this Wardrop traffic equilibrium problem (Wardrop, 1952) in the context of the rest of the literature. For example, Du et al. (2021) proposes line search methods with the Barzilai-Borwein coefficient for the stochastic user equilibrium (SUE) problem originally proposed by Daganzo and Sheffi (1977), and they use both a SUE variant that just has a random error term known as the multinomial logit model (in which they cite (Dial, 1971; Fisk, 1980) and a SUE variant that takes into consideration “overlapping paths” known as cross-nested logit model (in which they cite Vovsha (1997)). In another example, Wang et al. (2019) employs a cross-nested logit model for vehicles with human drivers and user equilibrium for autonomous vehicles, thus creating a blended model. What’s more, Guo et al. (2018) discuss various algorithms for a dynamic user equilibrium approach, demonstrating that there is much work to be done in this field. Even some of the earlier works suggest more complicated models involving non-additive path costs (Dafermos & Sparrow, 1969) and interactions among flow between different arcs (Gabriel & Bernstein, 1997). Given all of this, a reader might wonder if it is valid to choose this equilibrium model. It is valid because of the computational considerations that we elaborate upon in Section 3.2.1 regarding the stochastic network protection model into which the equilibrium conditions are integrated. The considerations include handling the stochastic nature of the problem, transforming the equilibrium constraints, and linearizing one of the cost functions we utilize. Thus, in this proof-of-concept paper in which we bring together inverse optimization with a complex optimization model, keeping the traffic equilibrium piece simpler saves on computation.

For the inverse optimization model in Bertsimas et al. (2015), we can then form an optimization model including each data point $x^{\check{i}}; j = 1, \ldots, J$ (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 $x^{\check{i}}$. There is the same node-arc incidence matrix $N$ for each $x^{\check{i}}$.

The inverse optimization model for $J$ data points (corresponding to each of the $x^{\check{i}}$ flow patterns), parameters $\theta \in \Theta$ with $\Theta$ as a convex subset of $\mathbb{R}^Z$ (Z representing a number of parameters), $\psi \in \mathbb{R}^{[N\times]}$, and $\epsilon \in \mathbb{R}^J$ is:

$$\min_{\theta \in \Theta, y_j, \epsilon} ||\epsilon||^2 \quad (4a)$$

$$-(N)^T y \leq c(x^{\check{i}}; \theta), j = 1, \ldots, J, \quad (4b)$$

$$y_j \geq 0, j = 1, \ldots, J, \quad (4c)$$
\[ c(\tilde{x}^j; \theta)^T \tilde{x}^j + (d^j)^Ty^j \leq \epsilon^j, j = 1, \ldots, J, \]  

(4d)

The derivation of this mathematical program can be found in Appendix A.2. We note that the \( y^j \) dual variables are important because they correspond, as in the complementarity problem above, to the travel distance to destination \( j \) (Fan & Liu 2010). We solve this mathematical program using the ipopt solver (Wächter & Biegler 2006). There can be multiple forms for the vector-valued arc cost function \( c(\mathbf{x}; \theta) \), 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 Goodfellow et al. (2016) would not be able to express the constraint relationships in (4b)-(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 Bertsimas et al. (2015). Even in linear regression’s constrained form (as expressed by these papers: Geweke, 1986; Liew, 1976; Stoer, 1971), the constraints are on the parameters themselves, which \( \Theta \) in (4a) can already express. Consequently, since even constrained linear regression cannot accommodate the relationships expressed in (4b)-(4d), then we need a more complicated model to parameterize our cost functions. The equations in (4b)-(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 (4) is the set of dual variables \( y^j \) which must satisfy two sets of inequality relationships (4b) and (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 (4b)-(4d) encode the variational inequality conditions seen in (1) and, thus, the equilibrium relationship needed for the problem.

### 3.1.1. Different types of cost functions

We propose two different formulations for the vector valued function \( c(\mathbf{x}) \) which represents the time per vehicle along each arc in the set of arcs \( \mathcal{A} \) (Lu et al. 2016, 2018). Note that \( \theta \) will represent the collection of all parameters for a given function.

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

\[
c(x) = \begin{bmatrix} \phi_1 & \cdots & \phi_{|\mathcal{A}|} \\ \vdots & \ddots & \vdots \\ \phi_{|\mathcal{A}|} & \cdots & \phi_{|\mathcal{A}|} \end{bmatrix} x + \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{|\mathcal{A}|} \end{bmatrix}
\]

(5)

This function has been used for representing travel times such as in Siri et al. (2020). Siri et al. (2020) labels the \( \beta_a \) as the free flow travel time i.e. travel times without any interaction with other travelers (Bertsimas et al. 2015; Chow et al., 2014; Thai et al., 2015; Zhang & Paschalidis, 2017; Zhang et al., 2018). \( \phi_a \) is the factor of additional time of having one more unit of flow on the arc. In this paper, we assume that the free flow travel times are given, and our goal is to estimate \( \phi_a \) for all \( a \in \mathcal{A} \) (see Section 4.5).

- **Bureau of Public Roads Function:** The Bureau of Public Roads function (BPR) (Branston, 1976; US Department of Commerce 1964) is a common function utilized by transportation researchers when modeling flow along arcs in a network (Bertsimas et al. 2015; Thai & Bayen, 2018; Zhang & Paschalidis, 2017; Zhang et al., 2018). From Sheffi (1985), the BPR function for arc \( a \) is:

\[
c_a(\mathbf{x}_a) = \lambda^0_a \left( 1 + \phi \mathbf{x}_a \frac{\mathbf{x}_a}{c_a} \right)^\beta_a
\]

(6)

The \( \lambda^0_a \) is the free-flow travel time, \( c_a^0 \) is the “practical capacity” which we just take as the normal capacity, and \( \phi_a, \beta_a \) are parameters which, following Sheffi (1985), are commonly assumed to be 0.15 and 4 respectively, regardless of the arc. In contrast, in this paper, we will assume that the \( \phi_a \) parameter is different for each arc and that it is the quantity we estimate with inverse optimization (see Section 4.5). We linearize the BPR function using standard techniques (Luathep et al., 2011; Winston 1994).

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 (3) and with the protection decisions when a user assumes uniform cost, meaning average \( \phi \) for the linear cost function and 0.15 for the \( \phi \) in the BPR cost function. Section 4.2 will explain this further.

### 3.2. Normative model: two-stage stochastic model

For the stochastic network protection model portion of the framework, we implement Fan and Liu (2010)’s 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 Fan and Liu (2010) and extended definitions for these terms can be found in Appendix B.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.
- $x_{k,s}^a$: the flow on arc $a$ that is destined for the $k$th destination in scenario $s$. The vector $x_{k,s} \in \mathbb{R}^m$ denotes the flow on all arcs. (units $=$ thousands of vehicles)
- $f_{a,s}^s$: 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)
- $h^a(u_a)$: the capacity of an arc $a$ given first stage decision $u_a$ under scenario $s$:

$$h^a(u_a) = \begin{cases} c_{a} & \text{if } a \notin \bar{A} \\ c_{a} - m_a^s(1 - u_a) & \text{if } a \in \bar{A} \end{cases}$$

(7) with $c_a$ 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 $\bar{A}$ 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 $a$ (Lu et al. 2016, 2018) as a function of the flows $\mathbf{f}^s$ in scenario $s$. We explore multiple different forms for $t_a$, described in Section 3.1.1.
- $\lambda_{i,k}^{a,s}$ as the minimum travel time from node $i$ to node $k$ in scenario $s$ (Fan & Liu 2010). (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 (2010)’s model with a modification to the complementarity constraints based on work by Ban (2005) and Ban et al. (2006) is thus:

$$\min \sum_{a \in \bar{A}} p_s Q^s(x,u,f^s)$$

s.t. $u \in \mathcal{D}$

(8a)

$$f_{a,s}^s = \sum_{k=1}^{K} x_{k,s}^{a} \leq h^a(u_a), a \in \mathcal{A}, s \in \mathcal{S}$$

(8b)

$$0 \leq x_{k,s}^{a} \leq (t_a(\mathbf{f}^s) + \lambda_{k}^{a,s} - \lambda_{i}^{a,s}) \geq 0, \forall(i,j) \in \mathcal{A}, \forall k$$

(8c)

$$0 \leq q^k + d^{k,s} - W x^{k,s} \leq 0$$

(8d)

$$\forall k = 1...K, \forall s \in \mathcal{S}$$

The $Q^s(x,u,f^s)$ function (8a) in the objective function has the following form:

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

The first term denotes the total cost of protection (with $\psi$ as the dollar amount it costs to protect each arc fully); this term differs from the Fan and Liu (2010) 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 $\gamma$, which transforms travel time to financial units (Fan and Liu 2010), keeping in the same units as the first term. Note, $t(f^s)$ corresponds to the $c(x)$ function from Section 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 (8e) constraints. Constraints (8b) represent the budgetary and technological restrictions that Fan and Liu (2010) define. We specified this further as just budgetary constraints of the form:

$$\sum_{a \in \bar{A}} u_a \leq I$$

(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 (8c) have an $s$ dependence for the $h^a$ functions because the $m_a^s \forall a \in \bar{A}$ are scenario-dependent. The constraints in (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 (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 (2010) 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 paper:

- The cap\(a\) value is set to 8 for all arcs \(a\).
- The \(m_j^k\) value (amount of damage) is set to 8.
- In the objective function, we set \(\gamma = 1\) (following Fan & Liu 2010) and set the \(\psi\) vector to 1 because we do not want cost to be prohibitive.

### 3.2.1. Big M method for complementarity constraints and progressive hedging algorithm for solving stochastic network protection problem

Fan and Liu (2010) 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 (Fortuny-Amat & McCarl, 1981; Hart et al., 2017b). As an example, we take the complementarity condition from (8d) and produce a series of constraints:

\[
\begin{align*}
    x_{ij}^{k,s} &\geq 0 & (10a) \\
    t_a(f^s) + z_{ij}^{k,s} - \bar{z}_{ij}^{k,s} &\geq 0 & (10b) \\
    x_{ij}^{k,s} &\leq M_{ij}^{k,s}(b_{ij}^{k,s}) & (10c) \\
    (t_a(f^s) + z_{ij}^{k,s} - \bar{z}_{ij}^{k,s}) &\leq M_{ij}^{k,s}(1 - b_{ij}^{k,s}) & (10d)
\end{align*}
\]

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 (10c) or (10d) to be 0. We repeat the same procedure for the complementarity constraints in (8e). See Appendix B.2 for information on calculating the \(M_{ij}^{k,s}\) values. To handle the stochasticity of this problem, we follow Fan and Liu (2010) by employing the progressive hedging (PH) algorithm. Proposed by Rockafellar and Wets (1991), 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 (Rockafellar and Wets 1991) that is computed after each iteration of the algorithm. See the following references for more information using the algorithm and setting its parameters: (Carpentier et al., 2013; Crainic et al., 2011; Fan & Liu 2010; Gonçalves et al., 2012; Gul et al., 2015; Hvattum & Lokketangen, 2009; Lamghari & Dimitrakopoulos, 2016; Mulvey & Vladimirou, 1991; Palsson & Ravn, 1994; Ryan et al., 2013; Veliz et al., 2014; Watson & Woodruff, 2011). We use the implementation of the PH algorithm found in the pysp extension (Watson et al., 2012) of the pyomo package (Hart et al., 2011, 2017b) in Python. We use gurobi for the mixed-integer quadratic programming subproblems 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 (8d) and (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.

### 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.

#### 4.1. Data generation

We generate data (observations of flow \(\hat{x}^j\) for all \(j = 1, ..., J\), as discussed in Section 2.1, using the forward problem in the form of the complementarity model (3); we solve (3) using PATH (Dirkse & Ferris 1995; Ferris & Munson 2000) in GAMS. The set \(\Lambda\) represents the origin-destination pairs utilized for each run of the complementarity model. For this paper, \(\Lambda\) 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, \(\Lambda\) would
consist of multiple different OD pairs per run of the complementarity model. The algorithm below illustrates generating the \( \hat{x}^j \) data for \( j = 1, \ldots, J \) given a set of configurations \( \Lambda \) with \( |\Lambda| = J \).

**Algorithm 1**: Generating the Data

**Data**: The set \( \Lambda \) of configurations

1. for \( j = 1:|\Lambda| \) do
2. Build the traffic equilibrium model (3) with the \( j \)th configuration of OD pair(s)
3. Solve the traffic equilibrium model (3) with PATH in GAMS
4. Store optimal \( \hat{x}^j \)
5. end

In part A of the experiments, the generated data \( \hat{x}^j, j = 1 \ldots J \) is used as input into the inverse optimization model to determine the parameters \( \theta \) for the cost function. In part B of the experiments, a subset of the generated data \( \hat{x}^j, j = 1 \ldots J \) is used as input into the inverse optimization model to determine the parameters \( \theta \) for the cost function. Namely, for each of the OD pairs, we leave out the data corresponding to that OD pair to obtain parameters \( \theta \) that correspond to that OD pair. This parallels the training-testing framework used in machine learning (Goodfellow et al., 2016). For both experimental setups, we then carry through with the rest of the hybrid framework described at the beginning of Section 3 to obtain the protection decisions. See Section 4.3 for more details.

### 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 \( u \), 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 \( \theta \).
- We define the original information protection decisions, denoted \( \hat{u} \), as the protection decisions that the two-stage model would make if it were directly given the original cost structure that was used in (3) to generate the data \( \hat{x}^j, j = 1 \ldots J \). 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 \( \theta \) is at recovering parameters leading to the original (assumed correct) decisions. The cost vector corresponding to the original cost structure is \( \theta \).
- We define the uniform information protection decisions, denoted \( \hat{u} \), 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 \( \theta \).

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

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

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:

\[
\text{Flow Error} : || \hat{x} - x ||_2 \quad (11)
\]

### 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

1. for \( i = 1:10 \) do
2. Generate the data \( \hat{x}^j, j = 1 \ldots J \) according to Algorithm 1
3. Input data \( \hat{x}^j, j = 1 \ldots J \) into inverse optimization model (4) and obtain \( \theta \)
4. \( u = \text{SNPP}(\theta) \)
5. \( \hat{u} = \text{SNPP}(\theta) \)
6. \( \hat{u} = \text{SNPP}(\theta) \)
7. Calculate the performance metrics according to Section 4.2
8. 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

1. for \( i = 1:10 \) do
2. Generate the data \( \vec{x}^j, j = 1...J \) according to Algorithm 1
3. for \( k = 1:\text{number of OD pairs} \) do
4. Input data \( \vec{x}^j, j = 1...J \) with the \( \vec{x}^j \) corresponding to the \( k \)th OD pair removed into inverse optimization model (4) and obtain \( \theta \)
5. Calculate the flow error by generating the flow for the \( k \)th OD pair under the \( \theta \) and compare this flow with \( \vec{x}^k \)
6. end
7. for \( k \in \text{subset of OD pairs} \) do
8. \( u = \text{SNPP}(\vec{x}) \)
9. \( \hat{u} = \text{SNPP}(\hat{x}) \)
10. \( \hat{u} = \text{SNPP}(\hat{x}) \)
11. Calculate the performance metrics according to Section 4.2
12. end
13. 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 \( k \)th OD pair and (b) the performance metrics from Section 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 \( \vec{x}^k \) when calculating \( \theta \) for that OD pair and, then, we take the 2-norm between the flow obtained from Algorithm 1 and the flow obtained for the \( k \)th OD pair under \( \theta \) (Allen et al. 2022; Goodfellow et al., 2016).

### 4.4. Networks and scenarios

We consider two networks on which to test our hybrid framework: a 4 x 4 grid in Figure 1a and the Nguyen & Dupuis network (Nguyen & Dupuis 1984) in Figure 1b, both of which we make bidirectional.

#### 4.4.1. Experiment I

In the first experiment, the linear cost function along with the 4 x 4 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 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.

Furthermore, for the experiments involving the linear cost function, only the \( \phi_a \) parameters are estimated. \( \beta_a \), 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 (Bertsimas et al. 2015; Chow et al., 2014; Thai & Bayen, 2018; Thai et al., 2015; Zhang & Paschalidis, 2017).

---

4x4 Directied Grid Network, 16 Nodes & 48 Arcs

**Figure 1.** Illustrative road networks used for experiments.
Table 1. Experiment descriptions.

| Experiment # | I                  | II                  | III                | IV                 |
|--------------|--------------------|--------------------|--------------------|--------------------|
| Network      | 4 × 4 Grid         | 4 × 4 Grid         | N & D             | N & D             |
| Cost Function| Linear BPR         | BPR                | Linear BPR        | BPR                |
| Parameters   | \(\phi_a \sim U(2, 10)\) | \(\phi_a \sim U(0, 1, 0.2)\) | \(\phi_a \sim U(2, 10)\) | \(\phi_a \sim U(0, 1, 0.2)\) |
|              | \(\beta_a \sim U(2, 10)\) | \(\beta_a \sim U(2, 10)\) | \(\beta_a \sim U(2, 10)\) | \(\beta_a \sim U(2, 10)\) |
| \(c'_a\)     | 6                  | 8                  | 8                  | 8                  |
| # Scenarios  | 12                 | 12                 | 9                  | 9                  |
| \(\mu\) Used| 5                  | 5                  | 5                  | 5                  |
| Number of Cores | 8              | 8                  | 8                  | 8                  |

**Figure 3.** Scenarios for Experiments III & IV.

Zhang et al. (2018). Consequently, for the \(\theta\) protection model, the \(\phi_a\) terms are the same across all of them. For the \(\theta\) protection model, the \(\phi_a\) terms are the original cost values generated by the unifrnd MATLAB function, as further discussed in Section 4.5. For the \(\theta\) protection model, the \(\phi_a\) terms come from the inverse optimization model (4). Finally, for the \(\theta\) protection model, the \(\phi_a = 6\) for all \(a\), as indicated in Table 1 in Section 4.5.

4.4.2. Experiment II

In Experiment II, the inverse optimization algorithm computes the \(z_a\) parameters for each arc \(a\) for the BPR cost function. Wong and Wong (2016) support having different \(z_a\) parameters across the network because they vary the \(z\) based on the structure of the network involved. Lu et al. (2016) create their BPR function such that the \(z\) parameter value differs for each type of vehicle in their simulation, thus again showing that the \(z\) parameter can be different than the standard uniform 0.15 noted in Section 3.1.1. The IO model is given the randomly chosen \(t^0_a\) parameters and the capacity levels (all set to 8), and it is asked to estimate the \(z_a\) values. The uniform parameter value that is chosen for the BPR experiments is 0.15 because that is the traditionally chosen parameter value (Sheffi, 1985). The scenario set up is the same as in Experiment I (see Figure 2).

4.4.3. Experiment III

Experiment III uses the linear cost function for the Nguyen & Dupuis network. Figure 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.

The notes about the linear cost function discussed in Experiment I hold for this experiment; only the network has changed along with the \(q^k\) for part A of the SNPP solves. See the beginning of Section 4.5 for the note about \(q^k\) for part A of the experiments.

4.4.4. 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 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 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 \(\phi_a\), \(\beta_a\) and \(t^0_a\) 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 \(\alpha_a\) to be between 0.1 and 0.2 because typically \(z_a\) is set to 0.15 (Sheffi, 1985), 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.

- \( \rho \) Used: Indicates a parameter value in the PH Algorithm. We obtain this value after many runs of the PH Algorithm with various values of \( \rho \).
- Number of Cores: Refers to the number of computer cores utilized for the experiments.

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 \((k = 1)\).

### 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 \( \tilde{x}^j \) for \( j = 1 \ldots J \). 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), the type of cost function (linear vs. BPR functions), and the origin-destination specification (see Table 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 (random) cost parameterization.
- For part A of the experiments, the \( k \) for \( q^k \) 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 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 \( k \)th 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 and 3. Each pair of arcs has an equal chance of failing.
- The budget for constraint (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 (Watson & Woodruff, 2011); we utilize the default error metric outlined in the pysp documentation in Hart et al. (2017a). 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.

### 5. Experimental results

#### 5.1. Experiment IA-IVA

The results of the experiments are evaluated with respect to two hypotheses regarding confidence intervals of the means and to a direct comparison of the means of the O-IO, U-IO, and U-O metrics defined in Section 4.2.4 See Appendix C.2 for information on the medians and minimum/maximums of the data and see Appendix C.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 3 and 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 3 and 4 indicate that Experiments IA-III A present evidence

| Table 2. Experiment matrix. |
|-----------------------------|
| A                           | B                           |
| I                           | Experiment IA: Linear, 4 × 4, & \( k = 2 \) | Experiment IB: Linear, 4 × 4, & \( k = 1 \) |
| II                          | Experiment IIA: BPR, 4 × 4, & \( k = 2 \) | Experiment IIB: BPR, 4 × 4, & \( k = 1 \) |
| III                         | Experiment IIA: Linear, N & D, & \( k = 2 \) | Experiment IIB: Linear, N & D, & \( k = 1 \) |
| IV                          | Experiment IVA: BPR, N & D, & \( k = 2 \) | Experiment IVB: BPR, N & D, & \( k = 1 \) |
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-III A demonstrate that using \(\bar{\theta}\) (uniform cost) leads to different protection decisions than the \(\tilde{\theta}\) or \(\theta\) costs. Comparing the means as a percentage of the total budget in Tables 3 and 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 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 4 and 5 (along with the boxplots for Experiment III A in Appendix C.2) tell the same story.

### 5.2. Experiments IB-IVB

From Algorithm 3, we obtain results for Experiments IB-IVB. We calculate the metrics described in Section 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 6–9, we plot histograms of the decision difference metrics between \(\mathbf{u}, \tilde{\mathbf{u}}, \mathbf{u}\) outlined in the first part of Section 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.

Finally, in Figures 10 and 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.

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.

6. Conclusions & future work

In this paper, 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 Goodfellow et al. (2016).

This paper 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 paper 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 Cantillo et al. (2019) and Guo et al. (2017). 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.

Notes
1. See Tai et al. (2013) for an example using proposed solutions.
2. We define the $\mathcal{F}$ set for the inverse optimization model differently; see Appendix A.2.
3. We keep the row in $N$ that contains the destination, which is different from X. J. Ban (2005) and J. X. Ban et al. (2006).
4. See Appendix C.1 for the flow error metrics for the IO $z$ values because, as can be seen from Figure 5,
the IO \( x \) values are different from the original \( x \) values.

**Acknowledgements**

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

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

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.

**References**

Abpeykar, S., & Ghatee, M. (2014). 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. https://doi.org/10.1016/j.tust.2014.03.008

Ahuja, R. K., & Orlin, J. B. (2001). Inverse optimization. *Operations Research*, 49(5), 771–783. https://doi.org/10.1287/opre.49.5.771.10607

Allen, S., Gabriel, S. A., & Dickerson, J. P. (2022). Using inverse optimization to learn cost functions in generalized Nash games. *Computers & Operations Research*, 142, 105721. https://doi.org/10.1016/j.cor.2022.105721

Alvear, D., Abreu, O., Cuesta, A., & Alonso, V. (2013). Decision support system for emergency management: Road tunnels. *Tunnelling and Underground Space Technology*, 34, 13–21. https://doi.org/10.1016/j.tust.2012.10.005

Asadabadi, A., & Miller-Hooks, E. (2017). Optimal transportation and shoreline infrastructure investment planning under a stochastic climate future. *Transportation Research Part B: Methodological*, 100, 156–174. https://doi.org/10.1016/j.trb.2016.12.023

Aswani, A., Shen, Z.-J., & Siddiq, A. (2018). Inverse optimization with noisy data. *Operations Research*, 66(3), 870–892. https://doi.org/10.1287/opre.2017.1705

Babier, A., Chan, T. C., Lee, T., Mahmood, R., & Terekhov, D. (2020). An ensemble learning framework for model fitting and evaluation in inverse linear optimization. *Inform Journal on Optimization*, 3(2), 119–138.

Ban, X. J. (2005). Quasi-variational inequality formulations and solution approaches for dynamic user equilibria. The University of Wisconsin-Madison.

Ban, J. X., Liu, H. X., Ferris, M. C., & Ran, B. (2006). A general mpcc model and its solution algorithm for continuous network design problem. *Mathematical and Computer Modelling*, 435(6), 493–505. https://doi.org/10.1016/j.mcm.2005.11.001

Barbarosoglu, G., & Arda, Y. (2004). A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the Operational Research Society*, 55(1), 43–53. https://doi.org/10.1057/palgrave.jors.2601652

Bertsimas, D., Gupta, V., & Paschalidis, I. C. (2015). Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2), 595–633. https://doi.org/10.1007/s10107-014-0819-4

Brantson, D. (1976). Link capacity functions: A review. *Transportation Research*, 10(4), 223–236. https://doi.org/10.1016/0041-1647(76)90055-1

Cantillo, V., Macea, L. F., & Jaller, M. (2019). Assessing vulnerability of transportation networks for disaster response operations. *Networks and Spatial Economics*, 19(1), 243–273. https://doi.org/10.1007/s11096-017-9382-x

Carpentier, P.-L., Gendreau, M., & Bastin, F. (2013). Long-term management of a hydroelectric multireservoir system under uncertainty using the progressive hedging algorithm. *Water Resources Research*, 49(5), 2812–2827. https://doi.org/10.1002/wrcr.20254

Chan, T. C., & Kaw, N. (2020). Inverse optimization to report. *Journal of the Operational Research Society*, 71(12), 2278–2290. https://doi.org/10.1080/01605682.2020.1795505

Chan, T. C., Lee, T., & Terekhov, D. (2019). Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Science*, 65(3), 1115–1135. https://doi.org/10.1287/mnsc.2017.2992

Chen, Y., & Florian, M. (1998). Congested O-D trip demand adjustment problem: Bilevel programming formulation and optimality conditions. *Mathematical Programming*, 87(1), 19–46. https://doi.org/10.1007/s10107-016-1385-1

Chow, J. Y., Ritchie, S. G., & Jeong, K. (2014). Nonlinear inverse optimization for parameter estimation of commodity-vehicle-decoupled freight assignment. *Transportation Research Part E: Logistics and Transportation Review*, 67, 71–91. https://doi.org/10.1016/j.tre.2014.04.004

Chu, J. C., & Chen, S. (2016). Optimization of transportation-infrastructure-system protection considering weighted connectivity reliability. *Journal of Infrastructure Systems*, 22(1), 04105008. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000264

Cioca, M., Cioca, L., Buraga, S. (2007). Spatial [elements] decision support system used in disaster management. In *2007 Inaugural Ieee-Ies Digital Ecosystems and Technologies Conference* (pp. 607–612). https://doi.org/10.1109/DEST.2007.372045

Crainic, T. G., Fu, X., Gendreau, M., Rei, W., & Wallace, S. W. (2011). Progressive hedging-based metaheuristics in actual emergencies. *Fire Safety Science*, 11, 1063–1076. https://doi.org/10.3801/IAFSS.FSS.11-1063

Dafermos, S. C., & Sparrow, F. T. (1969). The traffic assignment problem for a general network. *Journal of Research of the National Bureau of Standards, Section**
Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Ratliff, I. J., Jin, M., Konstantakopoulos, I. C., Spanos, C., & Sastry, S. S. (2014). Social game for building energy efficiency: Incentive design. In 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 1011–1018). https://doi.org/10.1109/ALLERTON.2014.7028565

Rockafellar, R. T., & Wets, R. J. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1), 119–147. https://doi.org/10.1287/moor.16.1.119

Rodríguez-Espindola, O., Albores, P., & Brewster, C. (2018). Disaster preparedness in humanitarian logistics: A collaborative approach for resource management in floods. *European Journal of Operational Research*, 264(3), 978–993. https://doi.org/10.1016/j.ejor.2017.01.021

Ryan, S. M., Wets, R. J., Woodruff, D. L., Silva-Monroy, C., & Watson, J. (2013). Toward scalable, parallel progressive hedging for stochastic unit commitment. In *Power and Energy Society General Meeting (PES)*, 2013 IEEE (pp. 1–5).

Sahebjamnia, N., Torabi, S. A., & Mansouri, S. A. (2017). A hybrid decision support system for managing humanitarian relief chains. *Decision Support Systems*, 95, 12–26. https://doi.org/10.1016/j.dss.2016.11.006

Schult, D. A., Swart, P. (2008). Exploring network structure, dynamics, and function using networks. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)* (Vol. 2008, pp. 11–16).

Shahmoradi, Z., & Lee, T. (2022). Quantile inverse optimization: Improving stability in inverse linear programming. *Operations Research*, 70(4), 2538–2562. https://doi.org/10.1287/opre.2021.2143

Sheffi, Y. (1985). *Urban transportation networks: Equilibrium analysis with mathematical programming methods*. Prentice-Hall, Inc.

Siri, E., Siri, S., & Sacone, S. (2020). A progressive traffic assignment procedure on networks affected by disruptive events. In *2020 European Control Conference (ECC)* (pp. 130–135). https://doi.org/10.23919/ECC51009.2020.9143945

Stoer, J. (1971). On the numerical solution of constrained least-squares problems. *SIAM Journal on Numerical Analysis*, 8(2), 382–411. https://doi.org/10.1137/0708038

Tai, K., Kizhakkedath, A., Lin, J., Tiong, R., & Sim, M. (2013). Identifying extreme risks in critical infrastructural events. In *Proceedings of the International Symposium of Next Generation Infrastructure* (pp. 1–4).

Thai, J., & Bayen, A. M. (2018). Imputing a variational inequality function or a convex objective function: A robust approach. *Journal of Mathematical Analysis and Applications*, 457(2), 1675–1695. https://doi.org/10.1016/j.jmaa.2016.09.031

Thai, J., Hariss, R., & Bayen, A. (2015). A multi-convex approach to latency inference and control in traffic equilibria from sparse data. In *2015 American Control Conference (ACC)* (pp. 689–695). https://doi.org/10.1109/ACC.2015.7170815

Todini, E. (1999). An operational decision support system for flood risk mapping, forecasting and management. *Urban Water*, 1(2), 131–143. https://doi.org/10.1016/S1462-0758(00)00010-8

US Department of Commerce. (1964). *Bureau of public roads: Traffic assignment manual*.
Appendix A. Data analysis component: inverse optimization

A.1. Proof of $d^w - Nx^w = 0$

**Lemma 1.** For the following complementarity problem:

\[
\begin{align*}
0 &\leq c(x) + NT^w x^w \preceq 0, \forall w \in \mathcal{W} \quad (A.1a) \\
0 &\leq d^w - Nx^w \preceq y^w \geq 0, \forall w \in \mathcal{W} \quad (A.1b)
\end{align*}
\]

\[d^w - Nx^w = 0 \text{ when there is a solution for } (A.1) \text{ and when we assume that } c(x) \text{ is greater than 0 for all } x \geq 0.\]

**Proof.** This proof is adapted from a proof seen in (Ban, 2005). Assume that the $c(x)$ function is greater than 0 for all $x \geq 0$. Also assume for the sake of contradiction that

\[0 < d^w - \left( \sum_{l \in G^d} x^w_{l(i)} - \sum_{j \in G^d} x^w_{j(i)} \right) \quad (A.2)\]

for some destination $w \in \mathcal{W}$ and for some $i \in \mathcal{N}$. \(\sum_{l \in G^d} x^w_{l(i)}\) represents the inflow at node $i$, and \(\sum_{j \in G^d} x^w_{j(i)}\) represents the outflow at node $i$. We know $y^w_i = 0$ by complementarity in $(A.1b)$. We can rearrange the inequality in $(A.2)$ to say:

\[0 \leq \sum_{l \in G^d} x^w_{l(i)} < d^w + \sum_{j \in G^d} x^w_{j(i)} \quad (A.3)\]

for some $w \in \mathcal{W}$ (representing the final destination) and for some $i \in \mathcal{N}$. We have that the $\sum_{l \in G^d} x^w_{l(i)}$ term is greater than or equal to 0 because we know all $x^w \geq 0$. The inequality in $(A.3)$ produces three difference cases:

**Case 1:** $d^w = 0$. This means at least one link in the $\sum_{l \in G^d} x^w_{l(i)}$ sum must be positive because $0 < \sum_{l \in G^d} x^w_{l(i)} + \sum_{j \in G^d} x^w_{j(i)}$. Therefore, for such a link $(i, j)$, $x^w_{j(i)} > 0$ forces the following equality:

\[c_{l(i)}(x) + y^w_i - y^w_j = 0 \quad (A.4)\]

We know $y^w_i = 0$, so we have $c_{l(i)}(x) + y^w_j > 0$. Since $y^w_j \geq 0$, both components must be zero but that contradicts the assumption that $c(x)$ is greater than 0 for all $x \geq 0$, thereby contradicting $(A.2)$. 

**Case 2:** $d^w$ is negative. This means at least one $x^w_{j(i)} > 0$ and contradiction follows as in Case 1.

**Case 3:** $d^w$ 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 $k \neq i$:

\[\sum_{l \in G^d} x^w_{l(k)} - \sum_{j \in G^d} x^w_{j(k)} = d^w_k \quad (A.5)\]

for the cases of $d^w_k = 0$ or when $d^w_k$ is negative, based on Cases 1 and 2.

- **Sub-Case A:** $\sum_{l \in G^d} x^w_{l(i)} > 0$. In this case, we arrive at the same contradictions we arrived at for the previous two cases.
- **Sub-Case B:** $\sum_{l \in G^d} x^w_{l(i)} = 0$ and $\sum_{l \in G^d} x^w_{j(i)} = 0$. In this case, we know there is some $d^w_k$ that is negative, which by $(A.5)$ means $\sum_{l \in G^d} x^w_{l(k)} > 0$. For any connecting nodes $q$ between node $k$ and node $i$, $\sum_{l \in G^d} x^w_{l(q)} = \sum_{l \in G^d} x^w_{j(q)} > 0$. Therefore, for node $i$, the inflow sum $\sum_{l \in G^d} x^w_{l(i)}$ 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:** $\sum_{l \in G^d} x^w_{l(i)} = 0$ and $\sum_{l \in G^d} x^w_{j(i)} > 0$ but less than $d^w_k$. For any $k$ nodes in which $d^w_k$ is negative, we know the absolute sum over these $k$ nodes is equal to $d^w_k$ when $i = w$. Therefore, as in Sub-Case B, there are outflows at these $k$ nodes due to the relationship $(A.5)$. There is also conservation of flow at the $q$ nodes in which $d^w_q = 0$. 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 $d^w_k$ sum. Therefore, for node $i$, the inflow sum $\sum_{l \in G^d} x^w_{l(i)}$ must be equal to $d^w_i$. This contradicts our assumption that the inflow would be less than $d^w_i$. Overall, we arrive at the contradiction because flow would be pushed toward the $i$ destination in order to satisfy the relationships established by $(A.5)$.

Consequently, we have shown that a solution to the traffic equilibrium problem will result in the $d^w - Nx^w = 0$ if we assume the $c(x)$ function is greater than 0 for all $x \geq 0$. ■

A.2. Explanation of forming the inverse model from Bertsimas et al. (Bertsimas et al., 2015)

To form the inverse optimization mathematical program from Bertsimas et al. (Bertsimas et al., 2015) for
our traffic equilibrium problem, we return to the VI formulation of the problem and notice the structure

\[ c(x^*)^T x \geq c(x^*)^T x' - \epsilon, \forall x \in \mathcal{F} \]  

(A.6a)

\[ \mathcal{F} = \{ x : x \in \mathbb{R}^n \text{s.t.} N x \leq d \} \]  

(A.6b)

Note, the \( \mathcal{F} \) set is written slightly differently than how it was initially introduced in Equation (2) in Section 3.1 in order to mirror the complementarity problem (A.1) 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.6a) inequality into a minimization problem

\[ \min_{x \in \mathcal{F}} c(x^*)^T x \]  

(A.7)

in which \( x^* \) is fixed, and this minimization problem forms the tightest upper bound on the right hand side of (A.6a) since we are choosing the \( x \) to minimize the left hand side of (A.6a) (Bertsimas et al., 2015; Facchinei & Pang, 2007). Because (A.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 (Murty, 1983; Winston, 1994). The dual of this problem is:

\[ \max_y (-d)^T y \]  

(A.8a)

\[ -N^T y \leq c(x^*) \]  

(A.8b)

\[ y \geq 0 \]  

(A.8c)

As in Bertsimas et al. (Bertsimas et al., 2015), 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.6a) are:

\[ c(x^*)^T x' + (d)^T y' \leq \epsilon \]  

(A.9a)

\[ -N^T y' \leq c(x^*) \]  

(A.9b)

\[ y' \geq 0 \]  

(A.9c)

which include the equating of the dual and primal objectives (A.9a) as well as the dual feasibility constraints (A.9b-A.9c). Using these conditions, we can then form an optimization problem including each data point \( x^i \) representing the flow on the network such that:

There is one OD pair for each instance \( x^i \).

There is the same node-arc incidence matrix \( N \) for each \( x^i \).

The optimization problem becomes for \( J \) data points, parameters \( \theta \in \Theta \) with \( \Theta \) as a convex subset of \( \mathbb{R}^\mathcal{E} \) (\( \mathcal{E} \) representing a number of parameters), \( y^i \in \mathbb{R}^{|\mathcal{N}|} \), and \( \epsilon \in \mathbb{R}^j \):

\[ \min_{\theta \in \Theta, y^i} \| \epsilon \|^2 \]  

(A.10a)

\[ -(N)^T y^i \leq c(x^i; \theta), \quad j = 1, \ldots, J, \]  

(A.10b)

\[ y^i \geq 0, \quad j = 1, \ldots, J, \]  

(A.10c)

\[ c(x^i; \theta)^T x^i + (d)^T y^i \leq \epsilon^i, \quad j = 1, \ldots, J, \]  

(A.10d)

The set \( \Theta \) is determined by the lower and upper bounds on the parameter values found in Table 1.

### B. Normative model: two-stage stochastic model

#### B.1. Two-stage stochastic model: parameter details

We adopt the notation from Fan and Liu (Fan & Liu, 2010), with the exception of the parameters in the \( h_a(u_a) \) function which, although inspired by (Fan & Liu, 2010), 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
- \( x_a^{k,t} \): the flow on arc \( a \) that is destined for the \( k \)th destination in scenario \( s \). The vector \( x^k \in \mathbb{R}^m \) denotes the flow on all arcs that is headed for the \( k \)th destination in scenario \( s \). (units = thousands of vehicles)
- \( f_s^a \): the total flow on arc \( a \) in scenario \( s \), and \( f^a \) as the vector containing all of the \( f_s^a \) 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 (W. F. Programme, 2019). (units = proportion of necessary resources needed to fully insure the arc)
- \( W \): the node-link adjacency matrix. We use the definition from (Marcotte & Patriksson, 2007)’s work of this matrix which states \( W \in \{ -1, 0, 1 \}^{|\mathcal{N}| \times |\mathcal{A}|} \) 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.
- \( q^k \in \mathbb{R}^n \): designates the amount of flow originating at each node that is headed to destination \( k \). We based our construction of the \( q^k \) vectors upon the set-up from (Marcotte & Patriksson, 2007)’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)
- \( h^a(u_a) \): the capacity of an arc \( a \) given first stage decision \( u_a \) under scenario \( s \):

\[ h^a(u_a) = \begin{cases} \text{cap}_a & \text{if } a \notin \mathcal{A} \\ \text{cap}_a - m_a(1 - u_a) & \text{if } a \in \mathcal{A} \end{cases} \]  

(B.1)

| Table C.1. Flow errors for BPR functions on the two networks. |
|---------------------------------------------------------------|
| 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                                 |
| 0.0002                    | 8.62e-05                                 |
| 0.0007                    | 0.0004                                   |
| 0.0001                    | 4.02e-05                                 |
| 6.39e-05                  | 4.43e-05                                 |
| 0.0001                    | 2.66e-05                                 |
| 7.82e-05                  | 6.06e-05                                 |
with $c_p$ representing capacity of the arc without it being affected by a disaster, $m_s^a$ representing the amount of damage done to arc $a$ in scenario $s$ if not protected, and $A$ represents the set of arc vulnerable to the disaster. Note that $m_s^a$ could be 0 in certain scenarios. (units = thousands of vehicles)

- $t_d(F^i)$ represents the time per vehicle along arc $a$ (Lu et al., 2016, 2018) as a function of the flows $F^i$ in scenario $s$. We explore multiple different forms for $t_d$.
- $k_{i,k}^s$ as “the minimum time from node $i$ to destination $k$” in scenario $s$ (Fan & Liu, 2010). (units = travel time)
- $d_{i,k}^s$ as the vector of extra variables that acts as a buffer for any flow that cannot be properly apportioned. (units = thousands of vehicles).

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 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 1 that the maximum value of $\phi_a$ and $\beta_a$ 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

Figure C.1. Experiment IIIA results: Nguyen & Dupuis network with linear cost. The parameter differences refer to the $\phi$ differences.

Figure C.2. Experiment IVA results: Nguyen & Dupuis network with BPR. The parameter differences refer to the $\alpha$ differences.

Figure C.3. Experiment A timing results (minutes).
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 $M_{ik}^a$ is 90$(m)(2)$, with $m$ as the number of arcs.

### C. Results

#### C.1. Flow error under IO $\alpha$

Allen et al. (Allen et al., 2022) define a flow error metric to evaluate whether or not an IO parameterization is valid for their application. Since the $\alpha$ values (for the BPR functions) imputed through IO are different from the original $\alpha$ 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 C.1 below.

From the small magnitude of these values, we see that the $\alpha$ recovered by the IO model produce flow values that are very close to the flow values produced by the original $\alpha$ values.

#### C.2. Median/min-max tables and Nguyen & Dupuis boxplots

When examining the medians as a percentage of the budget for Experiments IA-III A in Tables C.2 and C.3, the O-I O metric medians are quite small compared to the U-I O & 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 C.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 C.2, 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 $\alpha$ 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.

### C.3. Run time results

The following box-plots illustrate the run time data for Experiments IA-IVA and for both values of $\epsilon$, which is the value of the $g^{(b)}$ 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 min and, for $\epsilon = 0.001$, most of the trials of the experiments were below 500 min. 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.

### D. 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 (Hart et al., 2011, Hart et al., 2017)
  - pysp 5.7.1 (Watson et al., 2012)
  - networkx 2.5 (Schult & Swart, 2008)
  - pandas 1.1.3 (McKinney, 2010)
  - numpy 1.19.2 (Oliphant, 2007)
  - scipy 1.5.2 (Virtanen et al., 2020)
  - matplotlib 3.3.2 (Hunter, 2007)
  - scikit-learn 0.23.2 (Pedregosa et al., 2011)
- **Solvers**
  - gurobi Version 9.1.1 (Gurobi Optimization, 2021)
  - ipopt (Wächter & Biegler, 2006)
  - MATLAB 9.8.0.147392 (R2020a) Update 4 (MATLAB., 2020)
  - GAMS (GAMS Development Corporation, 2021) with PATH solver (Dirkse & Ferris, 1995; Ferris & Munson., 2000) (PATH website (Ferris & Munson., 2020))
  - Wolfram Alpha (Wolfram—Alpha, 2021)
- **Data**
  - Nguyen & Dupuis Network (Nguyen & Dupuis, 1984)
- **Important Sites with Example Code for pysp Implementation in Scripts**
  - https://projects.coin-or/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

---

### Table C.2. Medians, medians as percentage of $l = 6$ budget, and ranges for experiments, $\epsilon = 0.01$.

| Experiment | Med (Min, Max) | Experiment | Med (Min, Max) | Experiment | Med (Min, Max) | Experiment | Med (Min, Max) |
|------------|----------------|------------|----------------|------------|----------------|------------|----------------|
| O-I O      | 0.0 0.0%       | U-I O      | 0.006 0.01%    | U-O        | 0.007 0.05%    | U-IO       | 0.004 0.01%    |
| U-I O      | 0.3456 5.76%   | U-O        | 0.0446 0.74%   | U-O        | 0.0270 4.51%   | U-O        | 0.2729 4.55%   |

### Table C.3. Medians, medians as percentage of $l = 6$ budget, and ranges for experiments A, $\epsilon = 0.001$.

| Experiment | Med (Min, Max) | Experiment | Med (Min, Max) | Experiment | Med (Min, Max) | Experiment | Med (Min, Max) |
|------------|----------------|------------|----------------|------------|----------------|------------|----------------|
| O-I O      | 0.0 0.0%       | U-I O      | 0.006 0.01%    | U-O        | 0.0012 0.02%   | U-O        | 0.2699 4.5%    |
| U-I O      | 0.3134 5.22%   | U-O        | 0.0472 0.79%   | U-O        | 0.2733 4.56%   | U-O        | 0.0005 0.16%   |
| U-O        | 0.3134 5.22%   | U-O        | 0.007, 0.0232  | U-O        | 0.1446, 0.3511 | U-O        | 0.0075 0.13%   |