Constraint Generation for Two-Stage Robust Network Flow Problem

To appear in INFORMS Journal on Optimization

David Simchi-Levi
Department of Civil and Environmental Engineering, Institute for Data, Systems and Society, Operations Research Center, Massachusetts Institute of Technology, dslevi@mit.edu

He Wang
School of Industrial and Systems Engineering, Georgia Institute of Technology, he.wang@isye.gatech.edu

Yehua Wei
Carroll School of Management, Boston College, weiyl@bc.edu

In this paper, we propose new constraint generation algorithms for solving the two-stage robust minimum cost flow problem, a problem that arises from various applications such as transportation and logistics. In order to develop efficient algorithms under general polyhedral uncertainty set, we repeatedly exploit the network-flow structure to reformulate the two-stage robust minimum cost flow problem as a single-stage optimization problem. The reformulation gives rise to a natural constraint generation (CG) algorithm, and more importantly, leads to a method for solving the separation problem using a pair of mixed integer linear programs (MILPs). We then propose another algorithm by combining our MILP-based method with the column-and-constraint generation (C&CG) framework of Zeng and Zhao (2013). We establish convergence guarantees for both CG and C&CG algorithms. In computational experiments, we show that both algorithms are effective at solving two-stage robust minimum cost flow problems with hundreds of nodes.

Key words: robust optimization, two-stage adaptive models, network flows, constraint generation

1. Introduction

In real-world applications, the exact model parameters in optimization problems are often not known to the decision maker until some initial decisions have been made. For example, a retail chain may need to determine product storage levels at its warehouses when demand is still unknown. After demand is realized through orders placed by individual stores, the company can ship products from warehouses to stores. As another example, a manufacturer may need to invest in equipment and labor before production season starts, while product demand is still unpredictable at this stage. Once the production season approaches, the manufacturer observes product orders and schedules its production. In both examples, the firms decisions can be naturally modeled as a two-stage (adaptive) optimization problem.

In a two-stage adaptive optimization problem, there are two sets of decision variables: the here-and-now variables that are decided in the first stage before uncertainties are realized, as well as

the wait-and-see variables that are decided in the second stage after the exact model parameters are observed. The uncertainties are typically modeled through one of the two approaches. The traditional approach quantifies uncertainties using stochastic distribution, which leads to two-stage stochastic optimization models (Birge and Louveaux 2011). Alternatively, the uncertainties can be modeled using the worst-case scenario in an uncertainty set (Ben-Tal et al. 2009), which leads to two-stage robust optimization models, and is the focus of this paper.

Over the last two decades, two-stage robust optimization models have drawn much research interest, as they are often more tractable compared to their stochastic counterparts. We refer to recent surveys by Gabrel et al. (2014b) and Gorissen et al. (2015) for a detailed review of two-stage robust optimization problems. Some applications of two-stage robust optimization include supply chain contracts (Ben-Tal et al. 2005), empty resource repositioning (Erera et al. 2009), emergency planning for disasters (Ben-Tal et al. 2011, Simchi-Levi et al. 2017), unit commitment in electric power network (Bertsimas et al. 2013a), vehicle routing problems (Agra et al. 2013), location-transportation problems (Ardestani-Jaafari and Delage 2017, Gabrel et al. 2014a), supply chain risk mitigation (Simchi-Levi et al. 2018) and defense against bioterrorism (Simchi-Levi et al. 2017). Interestingly, a significant portion of these examples (Erera et al. 2009, Ben-Tal et al. 2011, Simchi-Levi et al. 2017, 2018, Ardestani-Jaafari and Delage 2017, Gabrel et al. 2014a) possess the network-flow structure in the second stage decision problems. This motivates us to study general two-stage robust network flow problems.

Literature Review. Bertsimas and Sim (2003) were among the first to study static robust optimization problems for network flows. Atamtürk and Zhang (2007) studied the two-stage robust network flow problem and showed that while the problem is NP-hard in general, there exists polynomial-time algorithms when the networks have certain special structures. Bertsimas et al. (2013b) studied maximum flow problems in both robust and two-stage settings when arcs in the network can fail. The authors provided hardness results and proposed an approximate solution method using a path-based formulation. A key difference between this paper and Atamtürk and Zhang (2007) and Bertsimas et al. (2013b) is that we focus on developing exact algorithms for the two-stage minimum cost flow problem with *general* network structure and *general* polyhedral uncertainty set.

Two-stage robust optimization problems are usually much harder to compute compared to single-stage robust optimization problems. Ben-Tal et al. (2004) studied two-stage robust linear programs, and showed that they are NP-hard in general. The current computation methods for two-stage robust optimization can be classified into two categories: those that provide approximate solutions and those that provide exact solutions.

Approximate Solutions: Ben-Tal et al. (2004) proposed a tractable approximation solution called Affinely Adjustable Robust Counterpart (AARC), which also referred to as the affine decision rule or the linear decision rule. For some special cases, AARC is proven to be optimal (Bertsimas et al. 2010, Iancu et al. 2013, Ardestani-Jaafari and Delage 2016, Simchi-Levi et al. 2017); for general cases, however, AARC can only approximate the optimal policy, and its approximation gap and performance are studied by Kuhn et al. (2011), Bertsimas and Goyal (2012), Gorissen et al. (2015), Bertsimas and de Ruiter (2016). Another approximate solution technique for two-stage robust optimization is finite adaptability, or k-adaptability, which is studied recently by Bertsimas and Caramanis (2010), Hanasusanto et al. (2015), Buchheim and Kurtz (2017), among others.

Exact Solutions: Researchers have also proposed using constraint generation (CG) to find exact solutions for problems that fall into the two-stage robust optimization framework (Gorissen and Den Hertog 2013, Bertsimas et al. 2013a, Zeng and Zhao 2013, Billionnet et al. 2014, Gabrel et al. 2014a, Ayoub and Poss 2016). Zeng and Zhao (2013) proposed a column-and-constraint generation (C&CG) algorithm, and showed it is usually much faster than regular CG algorithm in numerical experiment. For both CG and C&CG methods, a key challenge is solving the *separation problem* at each iteration. Thiele et al. (2010) considered a mixed integer linear program (MILP) reformulation of the separation problem when the uncertainty set has special structure. Zeng and Zhao (2013) proposed an MILP reformulation of the separation problem for general polyhedral uncertainty sets. The reformulation in Zeng and Zhao (2013) is based on the KKT condition of the second-stage problem and requires the two-stage problem to have *relatively complete recourse*, i.e., the second-stage problem is feasible for any first-stage decision and any parameter in the uncertainty set. Ayoub and Poss (2016) proposed an MILP reformulation of the separation problem without assuming relatively complete recourse generalizing the algorithm of Zeng and Zhao (2013). Moreover, they present a different formulation tailored for 0-1 uncertainty sets that is significantly faster than the algorithm based on KKT conditions. Recently, Georghiou et al. (2016) proposed a robust dual dynamic programming (RDDP) scheme for multi-stage robust optimization problems. The RDDP scheme can be used with or without relatively complete recourse. With relatively complete recourse, the RDDP scheme solves the separation problem using the method in Zeng and Zhao (2013); without relatively complete recourse, the RDDP scheme solves relaxations to the separation problem iteratively.

Main Contribution. Our paper proposes a method to find exact solutions to two-stage robust minimum cost flow (TRMCF) problems. Because TRMCF problems are in general NP-hard, we look for easy-to-implement algorithms that take advantage of the powerful state-of-the-art mixed integer linear program (MILP) solvers. A key idea of our approach is that when we apply constraint generation to TRMCF, we can take advantage of the network structure to reformulate the

separation problems as MILPs. Our method works for general polyhedral uncertainty set and does not require the two-stage problem to have relatively complete recourse. Comparing to the existing methods for solving the separation problem for general two-stage robust optimization (Zeng and Zhao 2013, Ayoub and Poss 2016), our MILP formulations are more compact and do not involve the big-M method. We demonstrate that this new MILP reformulation can be applied to both CG and C&CG-based methods to obtain exact algorithms for the TRMCF problem. In numerical experiments, we show that the algorithms are potentially viable to real-world applications such as location-transportation and production postponement problems. In addition, we reformulate the TRMCF problem using network flow structure, and derive an theoretical upper-bound on the number of iterations for CG and C&CG algorithms to find the optimal solution.

Notation. Throughout the paper, we write vectors and matrices in bold font, sets in script font, and scalars in normal font. All inequality signs represent entry-wise inequalities. We use $\mathbf{I}$ to denote the identity matrix of appropriate dimension. The vectors $\mathbf{0}$ and $\mathbf{1}$ represent vectors with all zeros and ones, respectively. The symbol "$\circ$" denotes entry-wise product of two vectors or two matrices with the same dimension. The symbol "$\vee$" denotes the join operator $a \vee b = \max\{a, b\}$, and $\max\{a, 0\}$ is denoted by a^+. We use $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$ to denote the set of reals, integers, and nonnegative integers respectively.

2. The Model

We define the two-stage robust network flow problem over a directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with nodes in set $\mathcal{N}$ and arcs in set $\mathcal{A}$. The number of nodes in the network is denoted by $n := |\mathcal{N}|$ and the number of arcs is denoted by $m := |\mathcal{A}|$. We use $\mathbf{N} \in \mathbb{R}^{n \times m}$ to denote the node-arc incidence matrix of the network: for each arc $(i, j) \in \mathcal{A}$, column $\mathbf{N}_{ij}$ has a "+1" in the ith row, a "−1" in the jth row; the rest of its entries are zero.

In the first stage, the decision maker chooses variables $\mathbf{x} \in \mathcal{X}$ in anticipation of uncertain outcomes. The variables $\mathbf{x}$ are often called first-stage decisions or "here-and-now" decisions. The set $\mathcal{X}$ represents all the constraints on the first-stage decisions $\mathbf{x}$. In all applications that we will discuss in the paper, $\mathcal{X}$ represents a set of linear constraints. However, the algorithms we develop in this paper also allows general constraint set $\mathcal{X}$ as long as an oracle can solve the optimization problem of the following form: $\max_{\mathbf{x} \in \mathcal{X}}\{\mathbf{c}^T\mathbf{x}, \text{ s.t. } \mathbf{A}\mathbf{x} \le \mathbf{b}\}$.

Uncertainties in the model are represented by a vector of uncertain parameters $\boldsymbol{\zeta} \in \mathcal{U} \subset \mathbb{R}^r$. We make the following assumption on the uncertainty set $\mathcal{U}$.

Assumption 1. *The set $\mathcal{U}$ is a polytope defined by $\mathcal{U} = \{\boldsymbol{\zeta} \in \mathbb{R}^r : \mathbf{D}\boldsymbol{\zeta} \le \mathbf{d}\}$. That is, the uncertainty set is defined by a set of linear inequalities and is bounded.*

We note that polytope uncertainty sets are widely used in the robust optimization literature, and special cases include popular choices such as box uncertainty sets and budget uncertainty sets (Bertsimas and Sim 2004). As we will show in the next section, Assumption 1 enables us to solve the model using mixed integer linear programs (MILPs). Although it is possible to consider other forms of uncertainty sets with nonlinear constraints, our solution method would then involve solving mixed integer nonlinear programs. For example, if ellipsoidal uncertainty sets are used, our proposed method requires solving a sequence of mixed integer conic quadratic programs (MICQPs). Although there are several open-source solvers for MICQPs, the dimensions of solvable MICQP by current solvers are much smaller compared to MILPs (Bonami et al. 2012). Therefore, we focus on polytope uncertainty sets in the paper.

In the second stage, the uncertainty parameters $\boldsymbol{\zeta}$ are realized. The realized value of $\boldsymbol{\zeta}$ and the first-stage decisions $\mathbf{x}$ together define a minimum cost flow problem. For the network $\mathcal{G}$, we assume that the arc costs are known in advance, but supply and demand on the nodes and capacity of the arcs are uncertain. More specifically, each node $i \in \mathcal{G}$ is associated with a function $b_i(\mathbf{x}, \boldsymbol{\zeta})$, indicating that it is a supply node if $b_i(\mathbf{x}, \boldsymbol{\zeta}) > 0$, demand node if $b_i(\mathbf{x}, \boldsymbol{\zeta}) < 0$; and transhipment node if $b_i(\mathbf{x}, \boldsymbol{\zeta}) = 0$.

We assume that the net flow balance condition $\sum_{i \in \mathcal{N}} b_i(\mathbf{x}, \boldsymbol{\zeta}) = 0$ always holds for any $\mathbf{x} \in \mathcal{X}, \boldsymbol{\zeta} \in \mathcal{U}$ so the network flow problem is well-defined. Moreover, each arc $(i, j) \in \mathcal{A}$ is associated with a constant cost f_{ij} per unit of flow and flow capacity $u_{ij}(\mathbf{x}, \boldsymbol{\zeta})$. Without of loss of generality, we assume the lower bounds of flows on all arcs are zero.[1] For brevity, we often write the supply/demand and capacity coefficients in the vector form, $\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}), \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta})$, and we assume they satisfy the following assumption.

ASSUMPTION 2. *We assume* $\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}), \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta})$ *are affine mappings in* $\mathbf{x}$ *and* $\boldsymbol{\zeta}$*. Furthermore,* $\mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}) \geq 0$ *for all* $\mathbf{x} \in \mathcal{X}, \boldsymbol{\zeta} \in \mathcal{U}$*.*

In sum, the second-stage decision $\mathbf{y}$ is defined by the following minimum cost flow problem:

$$\begin{aligned} \Pi(\mathbf{x}, \boldsymbol{\zeta}) = \min_{\mathbf{y}} \quad & \mathbf{f}^T \mathbf{y} \\ \text{s.t.} \quad & \mathbf{N}\mathbf{y} = \mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}) \\ & \mathbf{0} \leq \mathbf{y} \leq \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}). \end{aligned} \tag{1}$$

The first set of constraints in (1) represents flow balance constraints (recall that $\mathbf{N}$ is the node-arc incidence matrix of the network), and the second set of constraints represents arc capacity

[1] In general, suppose $l_{ij}(\mathbf{x}, \boldsymbol{\zeta})$ is the lower bound of flow on arc (i, j), we can reduce the problem to one with zero lower bound by defining a new flow variable $y'_{ij} = y_{ij} - l_{ij}(\mathbf{x}, \boldsymbol{\zeta})$ and a new upper bound $u'_{ij}(\mathbf{x}, \boldsymbol{\zeta}) = u_{ij}(\mathbf{x}, \boldsymbol{\zeta}) - l_{ij}(\mathbf{x}, \boldsymbol{\zeta})$.

constraints. Note that the second-stage decision $\mathbf{y}$, also known as "wait-and-see" decisions, should be treated as a function of $\mathbf{x}$ and $\boldsymbol{\zeta}$; but we write $\mathbf{y}$ in (1) instead of $\mathbf{y}(\mathbf{x}, \boldsymbol{\zeta})$ to simplify notation.

Throughout the paper, we also make the following assumption on the cost vector $\mathbf{f}$ of the network flow problem.

ASSUMPTION 3. *The values* $\mathbf{f}$ *are nonnegative integers.*

In general, if the values of $\mathbf{f}$ are rational numbers, one can transform them into integers by rescaling the objective function. Also, if f_{ij} is negative for some arc $(i,j) \in \mathcal{A}$, we can study an equivalent problem using arc (j,i) with cost $-f_{ij}$ instead of arc (i,j), plus appropriate modifications to the flow capacity.

In a two-stage adaptive robust optimization framework, uncertain parameters $\boldsymbol{\zeta}$ are chosen adversarially from the uncertainty set after the first-stage decisions $\mathbf{x}$ but before the second-stage decisions $\mathbf{y}$. Therefore, using the second stage funcion $\Pi(\mathbf{x}, \boldsymbol{\zeta})$ defined by (1), the *two-stage robust minimum cost flow* (TRMCF) problem is formulated as

$$\begin{aligned} \min_{\mathbf{x},\delta} \quad & \mathbf{c}^T\mathbf{x} + \delta \\ \text{s.t.} \quad & \Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta, \quad \forall \boldsymbol{\zeta} \in \mathcal{U}, \\ & \mathbf{x} \in \mathcal{X}. \end{aligned} \tag{2}$$

As we discussed in §1, the TRMCF model defined above can be applied to a wide range of problem domains. We also note that the TRMCF problem includes the two-stage robust network problem studied in Atamtürk and Zhang (2007) as a special case.

For each $\mathbf{x} \in \mathcal{X}$, if the second stage problem (1) is feasible for every $\boldsymbol{\zeta} \in \mathcal{U}$, we say that $\mathbf{x}$ is *feasible* for TRMCF. If there is some $\boldsymbol{\zeta} \in \mathcal{U}$ such that no feasible second stage solution exists, we say that $\mathbf{x}$ is *infeasible* for TRMCF; we use the convention that the optimal value of an infeasible minimization problem is infinity. When the second-stage problem is feasible for any $\mathbf{x} \in \mathcal{X}$ and $\boldsymbol{\zeta} \in \mathcal{U}$, we say that problem (2) satisfies the *relatively complete recourse* condition.

Our proposed method does not require the assumption of relatively complete recourse. This is important, because in some applications, there is no cost associated with the second-stage problem; the decision maker only needs to find a feasible solution to the network flow problem defined in the second stage. Problems without second stage cost are known as a *feasible flow problem*, and they are also NP-hard in general (Atamtürk and Zhang 2007). A typical application of the feasible flow problem is in scheduling, where finding a feasible schedule is often equivalent to finding a feasible flow in some network (Ahuja et al. 1993). Problems with no second-stage cost correspond to a special case of TRMCF with $\mathbf{f} = \mathbf{0}$, which we call the *two-stage robust feasible flow* problem.

3. Computational Algorithms

In this section, we develop two algorithms for solving the TRMCF problem. In §3.1, we first develop some useful properties of the TRMCF problem, and reformulate the problem into a *single-stage* optimization problem with linear constraints. Then, based on this reformulation, we propose a constraint generation algorithm for the TRMCF problem in §3.2, and develop a subroutine for solving the separation problem at each iteration with at most two mixed integer linear programs (MILPs). In §3.3, we propose a column-and-constraint generation algorithm for the TRMCF problem by combining the subroutine developed in §3.2 with the column-and-constraint generation framework from Zeng and Zhao (2013).

3.1. TRMCF Reformulation

Our reformulation of the TRMCF problem exploits the network flow structure of $\Pi(\mathbf{x},\boldsymbol{\zeta})$ defined in (1). To reformulate TRMCF, our goal is to rewrite constraint

$$\Pi(\mathbf{x},\boldsymbol{\zeta}) \leq \delta, \quad \forall \boldsymbol{\zeta} \in \mathcal{U},$$

as a finite set of linear constraints with respect to $\mathbf{x}$, $\boldsymbol{\zeta}$ and δ. To this end, we consider two sets of conditions. The first set of conditions ensure the feasibility of $\mathbf{x}$ for the TRMCF problem, which states that $\Pi(\mathbf{x},\boldsymbol{\zeta})$ is finite for all $\boldsymbol{\zeta} \in \mathcal{U}$; the second set of conditions ensure that given $\mathbf{x}$ and δ, the inequality $\Pi(\mathbf{x},\boldsymbol{\zeta}) \leq \delta$ is feasible for all $\boldsymbol{\zeta} \in \mathcal{U}$.

We first describe the set of constraints which ensures feasibility of $\mathbf{x}$. Using the duality theorem for the feasible flow problem, one can show that $\Pi(\mathbf{x},\boldsymbol{\zeta})$ is finite if and only if

$$\sum_{i\in\mathcal{S}} b_i(\mathbf{x},\boldsymbol{\zeta}) \leq \sum_{(i,j)\in\mathcal{A}: i\in\mathcal{S}, j\in\mathcal{N}/\mathcal{S}} u_{ij}(\mathbf{x},\boldsymbol{\zeta}), \quad \forall \mathcal{S}\subset\mathcal{N}. \tag{3}$$

Equation (3) has the following interpretation: if we consider any cut $(\mathcal{S},\mathcal{N}/\mathcal{S})$ of the network, the left-hand side is the net outflow from all nodes in $\mathcal{S}$, and the right-hand side is the capacity of all arcs in the cut-set determined by the cut $(\mathcal{S},\mathcal{N}/\mathcal{S})$. Clearly, if there exists a feasible flow, the outflow must be upper bounded by the capacity of the cut-set, which proves the "only-if" part of (3). The strong duality of feasible flow also establishes the "if" part. The proof of (3) is standard and thus omitted; a proof can be found in Ahuja et al. (1993). Using (3), we provide the following conditions that are equivalent to the feasibility of $\mathbf{x}$.

Proposition 1. *For any $\mathbf{x}\in\mathcal{X}$, $\mathbf{x}$ is feasible for TRMCF if and only if the optimization problem below has an optimal objective value equal to zero.*

$$\begin{aligned} \max_{\mathbf{p},\mathbf{q},\boldsymbol{\zeta}} \quad & \mathbf{b}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{q} \\ s.t. \quad & \mathbf{N}^T\mathbf{p} \leq \mathbf{q}, \mathbf{p}\in\{0,1\}^n, \mathbf{q}\in\{0,1\}^m, \boldsymbol{\zeta}\in\mathcal{U}. \end{aligned} \tag{4}$$

Furthermore, for each $\mathbf{p} \in \{0,1\}^n$ *and some fixed* $\mathbf{x}_0 \in \mathcal{X}$*, let*

$$\tilde{\boldsymbol{\zeta}}^{\mathbf{p}} = \arg\max_{\boldsymbol{\zeta} \in \mathcal{U}} \mathbf{b}(\mathbf{x}_0, \boldsymbol{\zeta})^T \mathbf{p} - \mathbf{u}(\mathbf{x}_0, \boldsymbol{\zeta})^T (\mathbf{0} \vee \mathbf{N}^T \mathbf{p}), \tag{5}$$

then $\mathbf{x}$ *is feasible for TRMCF if and only if*

$$\mathbf{b}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T (\mathbf{0} \vee \mathbf{N}^T \mathbf{p}) \leq 0, \quad \forall \mathbf{p} \in \{0,1\}^n. \tag{6}$$

REMARK 1. Because $b_i(\cdot)$ and $u_{ij}(\cdot)$ are separable in $\mathbf{x}$ and $\boldsymbol{\zeta}$, the solution of the optimization problem (5), denoted by $\tilde{\boldsymbol{\zeta}}^{\mathbf{p}}$, is independent to the choice of $\mathbf{x}_0$.

Proof of Proposition 1. By the strong duality theorem of the feasible flow problem, we know that $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \infty$ holds for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if Equation (3) is satisfied for all $\boldsymbol{\zeta} \in \mathcal{U}$ and $\mathcal{S} \subset \mathcal{N}$. For each $\mathbf{p} \in \{0,1\}^n$ and $(i,j) \in \mathcal{A}$, we have $(\mathbf{N}^T \mathbf{p})_{ij} = p_i - p_j$. Equation (3) can be rewritten as

$$\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta})^T \mathbf{q} \leq 0, \forall \mathbf{p} \in \{0,1\}^n, \mathbf{q} \in \{0,1\}^m, \mathbf{N}^T \mathbf{p} \leq \mathbf{q}.$$

Therefore, $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \infty$ holds for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if problem defined by (4) has nonpositive optimal objective value. Moreover, the optimal objective of (4) is also at least zero because $\mathbf{p}$ and $\mathbf{q}$ both equal to $\mathbf{0}$ is a feasible solution of (4). Thus, $\mathbf{x}$ is feasible if and only if the the optimal value of (4) is zero.

Recall that $\mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}) \geq \mathbf{0}$ for any pair of $(\mathbf{x}, \boldsymbol{\zeta})$ in our TRMCF formulation. This implies that the coefficient associated with the variable $\mathbf{q}$ in the dual objective function is nonpostive $(-\mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}))$, so an optimal solution would assign the smallest possible value to $\mathbf{q}$. Thus, for each fixed $\mathbf{p}$, the objective of (4) is maximized by setting $\mathbf{q} = \mathbf{0} \vee \mathbf{N}^T \mathbf{p}$, and $\boldsymbol{\zeta} = \tilde{\boldsymbol{\zeta}}^{\mathbf{p}}$ (defined in (5)) for any $\mathbf{x}$. This implies that $\mathbf{x}$ is feasible for TRMCF if and only if Equation (6) holds. □

Next, we reformulate the TRMCF model under the assumption that $\mathbf{x}$ is feasible. The dual problem of network flow formulation (1) is given by

$$\begin{aligned} \Pi(\mathbf{x}, \boldsymbol{\zeta}) = \max_{\mathbf{p}, \mathbf{q}} \quad & \mathbf{b}(\mathbf{x}, \boldsymbol{\zeta})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta})^T \mathbf{q} \\ \text{s.t.} \quad & \mathbf{N}^T \mathbf{p} - \mathbf{I}\mathbf{q} \leq \mathbf{f} \\ & \mathbf{p} \in \mathbb{R}^n, \mathbf{q} \in \mathbb{R}^m_+. \end{aligned} \tag{7}$$

It is well known that if $\mathbf{N}$ is the node-arc incidence matrix of a directed network, the matrix $[\mathbf{N}^T - \mathbf{I}; \mathbf{O} - \mathbf{I}]$ associated with the constraints in (7) is totally unimodular (see Ahuja et al. 1993). Since $\mathbf{f}$ are integers by Assumption 3, we can restrict $\mathbf{p}, \mathbf{q}$ to be integers in the dual problem (7). We can further restrict $\mathbf{p}$ to be nonnegative since shifting all variables $\mathbf{p}$ by a constant does not change the objective function and constraints.

Next, note that given an optimal flow $\mathbf{y}$ in (1), the dual variables $\mathbf{p}$ are interpreted as node potentials for the *residual network* of $\mathcal{G}$ with respect to the optimal second-stage flow $\mathbf{y}$ (see Ahuja et al. 1993). That is, the value $p_i - p_j$ is the shortest path distance from node i to node j in the residual network defined by the flow $\mathbf{y}$. Therefore, when $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \infty$, we can upper-bound the dual variables by $\mathbf{p}$ by $\sum_{(i,j)\in\mathcal{A}} f_{ij}$ and obtain an equivalent dual problem.

In the rest of the paper, when $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \infty$, we let constant $p_i^{\max}$ be an upper bound of the node potential p_i that does not change the dual of $\Pi(\mathbf{x}, \boldsymbol{\zeta})$. As mentioned, we can set $p_i^{\max} = \sum_{(i,j)\in\mathcal{A}} f_{ij}$, and in special cases such as bipartite networks, we can often find tighter bound for $p_i^{\max}$ (see examples in §4). Next, we provide the conditions where $\Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta$ for all $\boldsymbol{\zeta} \in \mathcal{U}$, when $\mathbf{x}$ is feasible.

PROPOSITION 2. *Suppose $\mathbf{x} \in \mathcal{X}$ is a feasible first stage solution. Then $\Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta$ for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if the optimization problem below has an objective value less than or equal to δ.*

$$\begin{aligned} \max_{\mathbf{p},\mathbf{q},\boldsymbol{\zeta}} \quad & \mathbf{b}(\mathbf{x}^k, \boldsymbol{\zeta})^T \mathbf{p} - \mathbf{u}(\mathbf{x}^k, \boldsymbol{\zeta})^T \mathbf{q} \\ s.t. \quad & \mathbf{N}^T \mathbf{p} - \mathbf{q} \le \mathbf{f}, \mathbf{p} \le \mathbf{p}^{\max}, \mathbf{p} \in \mathbb{N}^n, \mathbf{q} \in \mathbb{N}^m, \boldsymbol{\zeta} \in \mathcal{U}. \end{aligned} \tag{8}$$

Furthermore, for all vectors $\mathbf{p} \in \mathbb{N}^n$ with $\mathbf{p} \le \mathbf{p}^{\max}$ and some $\mathbf{x}_0 \in \mathcal{X}$, let

$$\boldsymbol{\zeta}^{\mathbf{p}} = \arg\max_{\boldsymbol{\zeta}\in\mathcal{U}} \mathbf{b}(\mathbf{x}_0, \boldsymbol{\zeta})^T \mathbf{p} - \mathbf{u}(\mathbf{x}_0, \boldsymbol{\zeta})^T (\mathbf{N}^T \mathbf{p} - \mathbf{f})^+. \tag{9}$$

Then, $\Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta$ for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if

$$\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}^{\mathbf{p}})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}^{\mathbf{p}})^T (\mathbf{N}^T \mathbf{p} - \mathbf{f})^+ \le \delta, \ \forall \mathbf{p} \in \mathbb{N}^n, \mathbf{p} \le \mathbf{p}^{\max}. \tag{10}$$

Proof of Proposition 2. From the strong duality theorem, totally unimodularity of the minimum cost flow problem, and the fact that $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \infty$, we know that $\max_{\boldsymbol{\zeta}\in\mathcal{U}} \Pi(\mathbf{x}, \boldsymbol{\zeta})$ can be reformulated as (8). This implies that $\Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta$ for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if the optimal objective of (8) is less than or equal to δ. Also, for each fixed $\mathbf{p}$, the objective of (8) is maximized if we pick $\mathbf{q} = (\mathbf{N}^T \mathbf{p} - \mathbf{f})^+$, and $\boldsymbol{\zeta} = \boldsymbol{\zeta}^{\mathbf{p}}$. Therefore, $\Pi(\mathbf{x}, \boldsymbol{\zeta}) \le \delta$ for all $\boldsymbol{\zeta} \in \mathcal{U}$ if and only if Equation (10) holds. □

Combining Proposition 1 and Proposition 2, we obtain the following reformulation for the TRMCF problem.

THEOREM 1. *The TRMCF problem can be formulated as*

$$\min_{\mathbf{x}\in\mathcal{X},\delta} \quad \mathbf{c}^T \mathbf{x} + \delta \tag{11a}$$

$$s.t. \quad \mathbf{b}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T (\mathbf{0} \vee \mathbf{N}^T \mathbf{p}) \le 0, \quad \forall \mathbf{p} \in \{0,1\}^n, \tag{11b}$$

$$\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}^{\mathbf{p}})^T \mathbf{p} - \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}^{\mathbf{p}})^T (\mathbf{N}^T \mathbf{p} - \mathbf{f})^+ \le \delta, \quad \forall \mathbf{p} \in \mathbb{N}^n, \mathbf{p} \le \mathbf{p}^{\max}, \tag{11c}$$

where $\tilde{\boldsymbol{\zeta}}^{\mathbf{p}}$ and $\boldsymbol{\zeta}^{\mathbf{p}}$ are defined by Equations (5) and (9).

Proof of Theorem 1. First, the TRMCF problem can be reformulated as

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\delta} \quad & \mathbf{c}^T\mathbf{x}+\delta \\ \text{s.t.} \quad & \Pi(\mathbf{x},\boldsymbol{\zeta})\le\delta, \quad \forall\boldsymbol{\zeta}\in\mathcal{U}. \end{aligned}$$

By Proposition 1 and Proposition 2, $\Pi(\mathbf{x},\boldsymbol{\zeta})\le\delta$ for all $\boldsymbol{\zeta}\in\mathcal{U}$ if and only if Equations (6) and (10) are satisfied, and this completes the proof. □

Theorem 1 shows that the TRMCF problem can be reformulated as an optimization problem with just the first stage variables $\mathbf{x}$, an additional variable δ, and at most $(K+1)^n+2^n$ *linear* constraints, where $K=\max_{i\in\mathcal{N}} p_i^{\max}$, which is upper-bounded by $\sum_{(i,j)\in\mathcal{A}} f_{ij}$. An interesting property of formulation (11) is that the number of additional constraints depends on the structure of the second stage problem, instead of the number of vertices in the uncertainty set which typically occurs in formulations for two-stage robust linear problems. As a consequence, when n is small, one may enumerate all of the constraints in (11b)–(11c) to solve optimization problem (11) directly. In the next subsection, we will also use the formulation in Theorem 1 to develop a constraint generation algorithm for the case when n is large.

Finally, we remark that if the second stage cost is zero ($\mathbf{f}=\mathbf{0}$) or if the TRMCF problem has relative complete recourse (any $\mathbf{x}\in\mathcal{X}$ is feasible), then we can obtain a more compact formulation compared to (11). Note that when $\mathbf{f}=\mathbf{0}$, the TRMCF is reduced to the two-stage robust feasible flow problem.

COROLLARY 1. *1) If $\mathbf{f}=\mathbf{0}$, then TRMCF can be formulated as*

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}} \quad & \mathbf{c}^T\mathbf{x} \\ \textit{s.t.} \quad & \mathbf{b}(\mathbf{x},\tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T\mathbf{p}-\mathbf{u}(\mathbf{x},\tilde{\boldsymbol{\zeta}}^{\mathbf{p}})^T(\mathbf{0}\vee\mathbf{N}^T\mathbf{p})\le 0, \quad \forall\mathbf{p}\in\{0,1\}^n, \end{aligned} \tag{12}$$

where $\tilde{\boldsymbol{\zeta}}^{\mathbf{p}}$ is defined by Equation (5).

2) If $\Pi(\mathbf{x},\boldsymbol{\zeta})<\infty$ for any $\mathbf{x}\in\mathcal{X}$ and $\boldsymbol{\zeta}\in\mathcal{U}$, then TRMCF can be formulated as

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\delta} \quad & \mathbf{c}^T\mathbf{x}+\delta \\ \textit{s.t.} \quad & \mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^{\mathbf{p}})^T\mathbf{p}-\mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^{\mathbf{p}})^T(\mathbf{N}^T\mathbf{p}-\mathbf{f})^+\le\delta, \quad \forall\mathbf{p}\in\mathbb{N}^n,\mathbf{p}\le\mathbf{p}^{\max}, \end{aligned} \tag{13}$$

where $\boldsymbol{\zeta}^{\mathbf{p}}$ is defined by Equation (9).

Proof. If $\mathbf{f}=\mathbf{0}$, then for any feasible $\mathbf{x}$, the objective value of the TRMCF problem is equal to $\mathbf{c}^T\mathbf{x}$. By Proposition 1, $\mathbf{x}$ is feasible if and only if Equation (6) are satisfied, implying that TRMCF can be formulated as (12).

If $\Pi(\mathbf{x},\boldsymbol{\zeta})<\infty$ for any $\mathbf{x}\in\mathcal{X}$ and any $\boldsymbol{\zeta}\in\mathcal{U}$, then $\mathbf{x}$ is always feasible. By Proposition 2, $\Pi(\mathbf{x},\boldsymbol{\zeta})<\delta$ for all $\boldsymbol{\zeta}\in\mathcal{U}$ if and only if Equation (10) are satisfied, implying that TRMCF can be formulated as (13). □.

3.2. Constraint Generation Procedure

In this section, we provide a constraint generation procedure for solving the TRMCF problem when the number of constraints in formulation (11) is huge. The basic idea of the constraint generation algorithm is straightforward. The algorithm starts by solving (11) without any constraints. At each iteration, the constraint algorithm solves a *master problem* to obtain a pair of solutions $(\mathbf{x}^*, \delta^*)$. Then, the algorithm solves a subproblem known as the *separation problem*, which either returns a certificate that $(\mathbf{x}^*, \delta^*)$ is optimal, or a violating constraint in the form of (11b) or (11c). If $(\mathbf{x}^*, \delta^*)$ is optimal, the algorithm terminates; and if not, a violating constraint is added to the master problem, and the algorithm continues to next iteration.

The main difficulty of implementing the constraint generation algorithm lies in the step of solving the separation problem. We solve the separation problem using the reformulation from §3.1. By Proposition 1, the feasibility of $\mathbf{x}$ can be checked by solving the bilinear program of form (4); additionally, by Proposition 2, given $\mathbf{x}$ is feasible, the condition $\Pi(\mathbf{x}, \boldsymbol{\zeta}) < \delta$ can be checked by solving the bilinear program of form (8). Therefore, the separation problem is solved if we can solve the two bilinear programs (4) and (8). Algorithm 1 gives the complete pseudo code of the constraint generation algorithm.

Algorithm 1 Constraint Generation Algorithm for the TRMCF problem.

Initialize with $k_1 \leftarrow 0$, $k_2 \leftarrow 0$.

loop

1. Solve a relaxation problem with constraints generated from previous iterations:

$$\min_{\mathbf{x}\in\mathcal{X},\delta} \quad \mathbf{c}^T\mathbf{x} + \delta \tag{14a}$$

$$\text{s.t.} \quad \mathbf{b}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^\kappa)^T \tilde{\mathbf{p}}^\kappa - \mathbf{u}(\mathbf{x}, \tilde{\boldsymbol{\zeta}}^\kappa)^T \tilde{\mathbf{q}}^\kappa \le 0, \quad \forall 1 \le \kappa \le k_1, \tag{14b}$$

$$\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}^\kappa)^T \mathbf{p}^\kappa - \mathbf{u}(\mathbf{x}, \boldsymbol{\zeta}^\kappa)^T \mathbf{q}^\kappa \le \delta, \quad \forall 1 \le \kappa \le k_2, \tag{14c}$$

Let $(\mathbf{x}^*, \delta^*)$ be the optimal solution.

2a. Solve optimization problem (4) with $\mathbf{x} = \mathbf{x}^*$ via the MILP reformulation in Proposition 3.

if the objective of (4) is strictly positive **then**

Obtain the optimal solution of (4), and let it be $(\tilde{\mathbf{p}}^{k_1+1}, \tilde{\mathbf{q}}^{k_1+1}, \tilde{\boldsymbol{\zeta}}^{k_1+1})$.

Add constraint of form (14b) using $(\tilde{\mathbf{p}}^{k_1+1}, \tilde{\mathbf{q}}^{k_1+1}, \tilde{\boldsymbol{\zeta}}^{k_1+1})$.

Set $k_1 \leftarrow k_1 + 1$. Go back to Step 1.

2b. Solve the optimization problem (8) with $\mathbf{x} = \mathbf{x}^*$ via the MILP reformulation in Proposition 4.

if the objective of (8) is strictly greater than δ^* **then**

Obtain the optimal solution of (8), and let it be $(\mathbf{p}^{k_2+1}, \mathbf{q}^{k_2+1}, \boldsymbol{\zeta}^{k_2+1})$.

Add constraint of form (14c) using $(\mathbf{p}^{k_2+1}, \mathbf{q}^{k_2+1}, \boldsymbol{\zeta}^{k_2+1})$.

Set $k_2 \leftarrow k_2 + 1$. Go back to Step 1.

else

Stop. Return $\mathbf{x}^*$ as the optimal solution.

The bilinear programs (4) and (8) are non-convex. As result, they typically cannot be solved to optimality with guarantee using off-the-shelf solvers. Moreover, it is known that both (4) and (8) are NP-hard (Atamtürk and Zhang 2007, Simchi-Levi and Wei 2015). Thus, our approach is to reformulate (4) and (8) to a pair of mixed integer linear programs (MILPs). While the MILP formulation does not guarantee polynomial runtime, we observe through numerical experiment in §4 that the MILP formulation usually can be solved efficiently using off-the-shelf MILP solvers.

For deriving MILP formulations, we use the following notations. Since $\mathbf{b}(\mathbf{x},\boldsymbol{\zeta})$ and $\mathbf{u}(\mathbf{x},\boldsymbol{\zeta})$ are affine functions of $\mathbf{x}$ and $\boldsymbol{\zeta}$ (Assumption 2), we write them explicitly as

$$\mathbf{b}(\mathbf{x},\boldsymbol{\zeta}) = \boldsymbol{\beta}(\mathbf{x}) + \mathbf{B}\boldsymbol{\zeta}, \quad \mathbf{u}(\mathbf{x},\boldsymbol{\zeta}) = \boldsymbol{\upsilon}(\mathbf{x}) + \mathbf{U}\boldsymbol{\zeta},$$

where $\boldsymbol{\beta}(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^n, \boldsymbol{\upsilon}(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^m$ are affine mappings of $\mathbf{x}$, and $\mathbf{B} \in \mathbb{R}^{n\times\ell}, \mathbf{U} \in \mathbb{R}^{m\times\ell}$ are fixed matrices. Since the uncertainty set $\mathcal{U}$ (a polytope) is bounded, we can find vectors $\mathbf{b}_{\min}, \mathbf{b}_{\max}, \mathbf{u}_{\min}, \mathbf{u}_{\max}$ such that

$$\mathbf{b}_{\min} \le \mathbf{B}\boldsymbol{\zeta} \le \mathbf{b}_{\max}, \quad \mathbf{u}_{\min} \le \mathbf{U}\boldsymbol{\zeta} \le \mathbf{u}_{\max}, \quad \forall \boldsymbol{\zeta} \in \mathcal{U}.$$

Now, we derive an MILP reformulation for the bilinear program in the form of (4) by applying the classical McCormick envelopes (McCormick 1976).

PROPOSITION 3. *The optimization problem (4) can be reformulated as*

$$\begin{aligned} \max \quad & \mathbf{1}^T\mathbf{z} + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \mathbf{1}^T\mathbf{w} - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ s.t. \quad & \mathbf{z} \le \mathbf{b}_{\max} \circ \mathbf{p}, \quad \mathbf{z} \le \mathbf{B}\boldsymbol{\zeta} - \mathbf{b}_{\min} \circ (\mathbf{1}-\mathbf{p}) \\ & \mathbf{w} \ge \mathbf{u}_{\min} \circ \mathbf{q}, \quad \mathbf{w} \ge \mathbf{U}\boldsymbol{\zeta} - \mathbf{u}_{\max} \circ (\mathbf{1}-\mathbf{q}) \\ & \mathbf{N}^T\mathbf{p} \le \mathbf{q} \\ & \mathbf{p} \in \{0,1\}^n, \mathbf{q} \in \{0,1\}^m, \boldsymbol{\zeta} \in \mathcal{U}. \end{aligned} \tag{15}$$

Proof. The first two sets of constraints in (15) are known as overestimates of $\mathbf{z}$ and underestimates of $\mathbf{w}$ in McCormick envelops. It is easily verified that $\mathbf{z} \le (\mathbf{B}\boldsymbol{\zeta}) \circ \mathbf{p}$ and $\mathbf{w} \ge (\mathbf{U}\boldsymbol{\zeta}) \circ \mathbf{q}$ are implied by the first two constraints (recall that we use symbol "$\circ$" to denote entry-wise product); moreover, the inequality signs hold as equality in the optimal solution since the coefficients of $\mathbf{z}$ and $\mathbf{w}$ in the objective function are positive and negative, respectively. Therefore, the objective function of (15) is equal to

$$\begin{aligned} & \mathbf{1}^T\mathbf{z} + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \mathbf{1}^T\mathbf{w} - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ =& \mathbf{1}^T\left[(\mathbf{B}\boldsymbol{\zeta}) \circ \mathbf{p}\right] + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \mathbf{1}^T\left[(\mathbf{U}\boldsymbol{\zeta}) \circ \mathbf{q}\right] - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ =& (\mathbf{B}\boldsymbol{\zeta})^T\mathbf{p} + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - (\mathbf{U}\boldsymbol{\zeta})^T\mathbf{q} - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ =& \mathbf{b}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{q}, \end{aligned}$$

and the last line is the objective function of optimization problem (4). □

We next develop an MILP reformulation for (8). Unlike formulation (4), the integer variables in (8) are not binary, so we cannot directly apply the McCormick envelops to linearize the bilinear terms in the objective function. To resolve this issue, we consider a reformulation of (8) where the integer variables $\mathbf{p}$, $\mathbf{q}$, are replaced with their binary representations (Gupte et al. 2013). To derive the binary representation, we use the fact that $\mathbf{p}$ is upper-bounded by $\mathbf{p}^{\max}$; and because the objective of (8) for fixed $\mathbf{p}$ is maximized if we pick $\mathbf{q} = (\mathbf{N}^T\mathbf{p} - \mathbf{f})^+$, if we let $q_{ij}^{\max} = p_i^{\max} - f_{ij}$, then $\mathbf{q}$ can be upper-bounded by $\mathbf{q}^{\max}$ without loss of optimality. Therefore, we can replace p_i and q_{ij} by a set of binary variables p_i^k and q_{ij}^ℓ such that

$$p_i = \sum_{k=0}^{P} 2^k p_i^k, \quad q_{ij} = \sum_{\ell=0}^{Q} 2^\ell q_{ij}^\ell,$$

where $P = \max_{i\in\mathcal{N}}\{\lceil \log_2(p_i^{\max}+1)\rceil - 1\}$ and $Q = \max_{(i,j)\in\mathcal{A}}\{\lceil \log_2(q_{ij}^{\max}+1)\rceil - 1\}$. With the binary representations, we apply the McCormick envelops to reformulate (8).

PROPOSITION 4. *The optimization problem (8) can be formulated as*

$$\begin{aligned}
\delta(\mathbf{x}) = \max \quad & \sum_{k=0}^{P} 2^k \left(\mathbf{1}^T\mathbf{z}^k + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p}^k\right) - \sum_{\ell=0}^{Q} 2^\ell \left(\mathbf{1}^T\mathbf{w}^\ell - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q}^\ell\right) \\
s.t. \quad & \mathbf{z}^k \le \mathbf{b}_{\max}\circ\mathbf{p}^k, \quad \mathbf{z}^k \le \mathbf{B}\boldsymbol{\zeta} - \mathbf{b}_{\min}\circ(\mathbf{1}-\mathbf{p}^k), \quad \forall k \\
& \mathbf{w}^\ell \ge \mathbf{u}_{\min}\circ\mathbf{q}^\ell, \quad \mathbf{w}^\ell \ge \mathbf{U}\boldsymbol{\zeta} - \mathbf{u}_{\max}\circ(\mathbf{1}-\mathbf{q}^\ell), \quad \forall \ell \\
& \mathbf{N}^T\left(\sum_{k=0}^{P} 2^k\mathbf{p}^k\right) - \sum_{\ell=0}^{Q} 2^\ell\mathbf{q}^\ell \le \mathbf{f} \\
& \forall k : \mathbf{p}^k \in \{0,1\}^n, \forall \ell : \mathbf{q}^\ell \in \{0,1\}^m, \boldsymbol{\zeta}\in\mathcal{U}.
\end{aligned} \tag{16}$$

Proof. The first two sets of constraints in (16) (McCormick envelops) ensure $\mathbf{z}^k \le (\mathbf{B}\boldsymbol{\zeta})\circ\mathbf{p}^k$ and $\mathbf{w}^\ell \ge (\mathbf{U}\boldsymbol{\zeta})\circ\mathbf{q}^\ell$. Because the coefficients for $\mathbf{z}^k$ are positive and the coefficients for $\mathbf{w}^\ell$ are negative in the objective function, if $\mathbf{z}^k$ and $\mathbf{w}^l$ are optimal, then $\mathbf{z}^k = (\mathbf{B}\boldsymbol{\zeta})\circ\mathbf{p}^k$ and $\mathbf{w}^\ell = (\mathbf{U}\boldsymbol{\zeta})\circ\mathbf{q}^\ell$. Therefore, letting $\mathbf{p} = \sum_{k=0}^{P} 2^k\mathbf{p}^k$, $\mathbf{q} = \sum_{\ell=0}^{Q} 2^\ell\mathbf{q}^\ell$, the objective of (16) is equal to

$$\begin{aligned}
& \sum_{k=0}^{P} 2^k \left(\mathbf{1}^T\mathbf{z}^k + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p}^k\right) - \sum_{\ell=0}^{Q} 2^\ell \left(\mathbf{1}^T\mathbf{w}^\ell - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q}^\ell\right) \\
=& \sum_{k=0}^{P} 2^k \left(\mathbf{1}^T\left[(\mathbf{B}\boldsymbol{\zeta})\circ\mathbf{p}^k\right] + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p}^k\right) - \sum_{\ell=0}^{Q} 2^\ell \left(\mathbf{1}^T\left[(\mathbf{U}\boldsymbol{\zeta})\circ\mathbf{q}^\ell\right] - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q}^\ell\right) \\
=& \sum_{k=0}^{P} 2^k\mathbf{b}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{p}^k - \sum_{\ell=0}^{Q} 2^\ell\mathbf{u}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{q}^\ell \\
=& \mathbf{b}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{q},
\end{aligned}$$

and the last line is the objective of (8). The third constraint of (16) holds if and only if $\mathbf{N}^T\mathbf{p} - \mathbf{I}\mathbf{q} \le \mathbf{f}$. So problem (16) is equivalent to the problem (8). □

It is important to note that when $(\mathbf{p}^\kappa, \mathbf{q}^\kappa, \zeta^\kappa)$ is an optimal solution for (4) or (8), the constraint generated by $(\mathbf{p}^\kappa, \mathbf{q}^\kappa, \zeta^\kappa)$ in Algorithm 1 corresponds to exactly one constraint defined by Equation (11b) or (11c), respectively. As a result, it is guaranteed that Algorithm 1 terminates in at most T iterations, where T is the number of linear constraints in the reformulation (11). Recall from §3.1 that T is upper-bounded by $(K+1)^n + 2^n$, where K only depends on the structure of the second stage problem. Interestingly, this is in contrast to the standard constraint generation or cutting plane algorithms proposed for general two-stage robust optimization problems (Thiele et al. 2010, Zeng and Zhao 2013), where the number of constraints generated typically depends on the structure of uncertainty sets. This observation is mainly of theoretical interest, as numerical examples suggest that the actual number of constraints generated is typically much smaller than T.

A few comments are in order for implementing Algorithm 1. First, there are several advantages of using the MILP reformulations instead of using the bilinear program formulations stated in Algorithm 1 directly. Over the past few decades, there has been vast improvement in MILP solution techniques, so solving (15) and (16) allows us to take advantage of the progress made in the state-of-the-art MILP solvers. Moreover, at each iteration, we can use the MILP solution from the previous iteration for warm start, therefore significantly reducing the computational time required by MILP solvers. In our computational experiment, we use the Gurobi Mixed Integer Solver (7.0.2) on a standard laptop, which is able to solve the separation problem for networks with hundreds of nodes consistently in a few seconds.

Second, notice that in order to generate a constraint in Step 2a (or 2b) of Algorithm 1, it is not necessary to solve the optimization problem to optimality. Instead, the algorithm can add a cut whenever the MILP solver finds a feasible solution $(\mathbf{p}, \mathbf{q}, \boldsymbol{\zeta})$ such that the objective function is strictly larger than zero (or δ). Using the solution $(\mathbf{p}, \mathbf{q}, \boldsymbol{\zeta})$, we can generate a valid constraint in each iteration, while saving computation time by stopping the MILP solver early. After a feasible solution $(\mathbf{p}, \mathbf{q}, \boldsymbol{\zeta})$ is found by the MILP solver, we can use it as an initial point to solve the bilinear program form of the separation problem to local optimal using an *alternating iterative method*. Namely, we fix either $(\mathbf{p}, \mathbf{q})$ or $\boldsymbol{\zeta}$ while optimizing over the other, in order to strengthen the constraint generated. Because the alternating iterative method only involves linear optimization, it can be performed very fast. In our implementation of this method, the last iteration is always an optimization over $\boldsymbol{\zeta}$ with fixed $(\mathbf{p}, \mathbf{q})$, so it ensures that Algorithm 1 always generate a constraint of form (11b) or (11c) in each iteration.

Third, when the uncertainty set lies in a unit hypercube, i.e., $\mathcal{U} \subset [0,1]^r$, there is an alternative MILP reformulation of the separation problem. This reformulation is sometimes stronger than the reformulations in Propositions 3 and 4, and is provably stronger for box uncertainty set, i.e., $\mathcal{U} =$

$[0,1]^r$. To define this formulation, we introduce continuous decision variables $\mathbf{Z} \in \mathbb{R}^{n\times r}, \mathbf{W} \in \mathbb{R}^{m\times r}$. Let $\mathbf{1}^n \in \mathbb{R}^n, \mathbf{1}^m \in \mathbb{R}^m$, and $\mathbf{1}^r \in \mathbb{R}^r$ be vectors with all elements equal to 1. Below is an MILP reformulation for the optimization problem in Proposition 3. (The problem in Proposition 4 can be reformulated using the same approach and is omitted here.)

PROPOSITION 5. *If $\mathcal{U} \subset [0,1]^r$, the optimization problem (4) can be reformulated as*

$$\begin{aligned} \max \quad & \mathrm{tr}(\mathbf{B}^T\mathbf{Z}) + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \mathrm{tr}(\mathbf{U}^T\mathbf{W}) - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ s.t. \quad & \mathbf{Z} \le \mathbf{p}\mathbf{1}_r^T, \quad \mathbf{Z} \le \mathbf{1}_n\boldsymbol{\zeta}^T, \quad \mathbf{Z} \ge (\mathbf{1}_n - \mathbf{p})\mathbf{1}_r^T - \mathbf{1}_n\boldsymbol{\zeta}^T, \quad \mathbf{Z} \ge 0 \\ & \mathbf{W} \le \mathbf{q}\mathbf{1}_r^T, \quad \mathbf{W} \le \mathbf{1}_m\boldsymbol{\zeta}^T, \quad \mathbf{W} \ge (\mathbf{1}_m - \mathbf{q})\mathbf{1}_r^T - \mathbf{1}_m\boldsymbol{\zeta}^T, \quad \mathbf{W} \ge 0 \\ & \mathbf{N}^T\mathbf{p} \le \mathbf{q} \\ & \mathbf{p} \in \{0,1\}^n, \mathbf{q} \in \{0,1\}^m, \boldsymbol{\zeta} \in \mathcal{U}. \end{aligned} \tag{17}$$

Moreover, when $\mathcal{U} = [0,1]^r$, the optimal value of the LP relaxation of (17) is no greater than that of the LP relaxation of (15).

Proof. In (17), since $\mathbf{p}$ and $\mathbf{q}$ are binary variables and $\mathbf{0} \le \boldsymbol{\zeta} \le \mathbf{1}$, applying the McCormick envelopes (McCormick 1976), the first two sets of constraints are equivalent to $\mathbf{Z} = \mathbf{p}\boldsymbol{\zeta}^T, \mathbf{W} = \mathbf{q}\boldsymbol{\zeta}^T$. Therefore, the objective function of (17) is equal to

$$\begin{aligned} & \mathrm{tr}(\mathbf{B}^T\mathbf{Z}) + \boldsymbol{\beta}(\mathbf{x}^*)^T\mathbf{p} - \mathrm{tr}(\mathbf{U}^T\mathbf{W}) - \boldsymbol{\upsilon}(\mathbf{x}^*)^T\mathbf{q} \\ =& \mathrm{tr}(\mathbf{B}^T\mathbf{p}\boldsymbol{\zeta}^T) + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \mathrm{tr}(\mathbf{U}^T\mathbf{q}\boldsymbol{\zeta}^T) - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ =& \boldsymbol{\zeta}^T\mathbf{B}^T\mathbf{p} + \boldsymbol{\beta}(\mathbf{x})^T\mathbf{p} - \boldsymbol{\zeta}^T\mathbf{U}^T\mathbf{q} - \boldsymbol{\upsilon}(\mathbf{x})^T\mathbf{q} \\ =& \mathbf{b}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta})^T\mathbf{q}, \end{aligned}$$

which is exactly the objective function of (4).

To prove the second part of the proposition, given a feasible solution to the LP relaxation of (17), define $\tilde{\mathbf{z}} := \mathrm{diag}(\mathbf{B}\mathbf{Z}^T), \tilde{\mathbf{w}} := \mathrm{diag}(\mathbf{U}\mathbf{W}^T)$ ($\mathrm{diag}(\cdot)$ here is the vector containing the diagonal elements of a matrix). Let $\mathbf{B}^+ = \mathbf{B} \vee \mathbf{0}$ and $\mathbf{B}^- = (-\mathbf{B}) \vee \mathbf{0}$. Since $\mathbf{Z} \le \mathbf{p}\mathbf{1}_r^T, \mathbf{Z} \ge 0$, we have

$$\tilde{\mathbf{z}} = \mathrm{diag}(\mathbf{B}^+\mathbf{Z}^T) - \mathrm{diag}(\mathbf{B}^-\mathbf{Z}^T) \le \mathrm{diag}(\mathbf{B}^+ \cdot \mathbf{1}_r\mathbf{p}^T) - \mathbf{0} = \mathrm{diag}(\mathbf{B}^+ \cdot \mathbf{1}_r\mathbf{p}^T) = \mathbf{b}_{\max} \circ \mathbf{p},$$

where the last step follows from $\mathbf{b}_{\max} := \max_{\boldsymbol{\zeta}\in[0,1]^r} \mathbf{B}\boldsymbol{\zeta} = \mathbf{B}^+\mathbf{1}_r$. Since $\mathbf{Z} \le \mathbf{1}_n\boldsymbol{\zeta}^T, \mathbf{Z} \ge (\mathbf{1}_n - \mathbf{p})\mathbf{1}_r^T - \mathbf{1}_n\boldsymbol{\zeta}^T$, we have

$$\begin{aligned} \tilde{\mathbf{z}} =& \mathrm{diag}(\mathbf{B}^+\mathbf{Z}^T) - \mathrm{diag}(\mathbf{B}^-\mathbf{Z}^T) \le \mathrm{diag}(\mathbf{B}^+ \cdot \boldsymbol{\zeta}\mathbf{1}_n^T) - \mathrm{diag}(\mathbf{B}^- \cdot \boldsymbol{\zeta}\mathbf{1}_n^T) - \mathrm{diag}(\mathbf{B}^- \cdot \mathbf{1}_r(\mathbf{1}_n - \mathbf{p})^T) \\ =& \mathrm{diag}(\mathbf{B} \cdot \boldsymbol{\zeta}\mathbf{1}_n^T) - \mathrm{diag}(\mathbf{B}^- \cdot \mathbf{1}_r(\mathbf{1}_n - \mathbf{p})^T) = \mathbf{B}\boldsymbol{\zeta} - \mathbf{b}_{\min} \circ (\mathbf{1} - \mathbf{p}), \end{aligned}$$

where the last step follows from $\mathbf{b}_{\min} := \min_{\boldsymbol{\zeta}\in[0,1]^r} \mathbf{B}\boldsymbol{\zeta} = -\mathbf{B}^-\mathbf{1}_r$. Similarly, one can prove that $\tilde{\mathbf{w}} \ge \mathbf{u}_{\min} \circ \mathbf{q}$, $\tilde{\mathbf{w}} \ge \mathbf{U}\boldsymbol{\zeta} - \mathbf{u}_{\max} \circ (\mathbf{1} - \mathbf{q})$. Therefore, $(\tilde{\mathbf{z}}, \tilde{\mathbf{w}})$ is feasible for the LP relaxation of problem (15). Finally, $\mathrm{tr}(\mathbf{B}^T\mathbf{Z}) = \mathrm{tr}(\mathbf{B}\mathbf{Z}^T) = \mathbf{1}^T\tilde{\mathbf{z}}$, $\mathrm{tr}(\mathbf{U}^T\mathbf{W}) = \mathrm{tr}(\mathbf{U}\mathbf{W}^T) = \mathbf{1}^T\tilde{\mathbf{w}}$, so the optimal value of the LP relaxation of (17) is no greater than that of the LP relaxation of (15). □

3.3. Column-and-Constraint Generation Algorithm

The efficiency of Algorithm 1 critically depends on the number of constraints generated until the algorithm terminates. An approach commonly used in the two-stage robust optimization literature to reduce the number of constraint generation iterations is the column-and-constraint generation procedure introduced by Zeng and Zhao (2013). In this section, we present another algorithm for the TRMCF problem. This second algorithm incorporates the MILP reformulations developed in §3.2 into the column-and-constraint generation procedure of Zeng and Zhao (2013).

To motivate the column-and-constraint generation procedure, recall that the TRMCF model can be formulated as

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\delta} \quad & \mathbf{c}^T\mathbf{x}+\delta \\ \text{s.t.} \quad & \Pi(\mathbf{x},\boldsymbol{\zeta})\le\delta, \quad \forall\boldsymbol{\zeta}\in\mathcal{U}. \end{aligned}$$

In the previous constraint generation algorithm (Algorithm 1), at each iteration, we add a linear constraint of the form

$$\mathbf{b}(\mathbf{x},\tilde{\boldsymbol{\zeta}}^{\kappa})^T\tilde{\mathbf{p}}^{\kappa}-\mathbf{u}(\mathbf{x},\tilde{\boldsymbol{\zeta}}^{\kappa})^T\tilde{\mathbf{q}}^{\kappa}\le 0, \quad \forall 1\le\kappa\le k_1, \tag{18a}$$

$$\text{or} \quad \mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^{\kappa})^T\mathbf{p}^{\kappa}-\mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^{\kappa})^T\mathbf{q}^{\kappa}\le\delta, \quad \forall 1\le\kappa\le k_2, \tag{18b}$$

if $\tilde{\boldsymbol{\zeta}}^{\kappa}$ or $\boldsymbol{\zeta}^{\kappa}$ creates a violating constraint with respect to the current optimal solution of the master problem, i.e., $\Pi(\mathbf{x}^*,\tilde{\boldsymbol{\zeta}}^{\kappa})>\delta^*$ or $\Pi(\mathbf{x}^*,\boldsymbol{\zeta}^{\kappa})>\delta^*$. Note that as the algorithm proceeds, the number of constraints increases, but the set of decision variables remains unchanged.

Alternatively, in the *column-and-constraint generation* procedure, instead of adding one linear constraint in form (18a) or (18b) during each iteration, once we find some $\boldsymbol{\zeta}^{\kappa}$ such that $\Pi(\mathbf{x}^*,\boldsymbol{\zeta}^{\kappa})>\delta^*$, we add a *nonlinear* constraint of form

$$\Pi(\mathbf{x},\boldsymbol{\zeta}^{\kappa})\le\delta.$$

One can convert $\Pi(\mathbf{x},\boldsymbol{\zeta}^{\kappa})\le\delta$ into a set of linear constraints. One approach to accomplish this is to add recourse decision variables $\mathbf{y}^{\kappa}$ associated with each uncertainty parameter $\boldsymbol{\zeta}^{\kappa}$, and add the following set of linear constraints:

$$\mathbf{f}^T\mathbf{y}^{\kappa}\le\delta, \tag{19a}$$

$$\mathbf{N}\mathbf{y}^{\kappa}=\mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^{\kappa}), \tag{19b}$$

$$\mathbf{0}\le\mathbf{y}^{\kappa}\le\mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^{\kappa}). \tag{19c}$$

Note that the left-hand side of (19a) is the primal objective function $\Pi(\mathbf{x},\boldsymbol{\zeta}^{\kappa})$, while (19b) and (19c) correspond to the primal feasibility condition of $\Pi(\mathbf{x},\boldsymbol{\zeta}^{\kappa})$. Thus, Inequalities (19a)–(19c) are equivalent to the condition $\Pi(\mathbf{x},\boldsymbol{\zeta}^{\kappa})\le\delta$.

The separation problem for the column-and-constraint generation procedure requires checking whether the optimization problem $\max_{\boldsymbol{\zeta}\in\mathcal{U}} \Pi(\mathbf{x}^*, \boldsymbol{\zeta}^\kappa)$ has an optimal value greater than δ^*. Note that this separation problem is *the same* separation problem solved in Algorithm 1. Therefore, as in §3.2, we can convert $\max_{\boldsymbol{\zeta}\in\mathcal{U}} \Pi(\mathbf{x}^*, \boldsymbol{\zeta}^\kappa)$ into two bilinear programs. Moreover, Propositions 3 and 4 show that the two bilinear programs can be reformulated as MILPs and solved using state-of-the-art MILP solvers. Having described the intuition behind the column-and-constraint generation procedure for the TRMCF problem, we formally define it in Algorithm 2.

We note that Zeng and Zhao (2013) have proposed an MILP formulation for solving the separation problem of general two-stage robust linear programs with the additional *relatively complete recourse* assumption. Ayoub and Poss (2016) extended the approach in Zeng and Zhao (2013) to relax the relatively complete recourse assumption. For general polyhedral uncertainty sets, both Zeng and Zhao (2013) and Ayoub and Poss (2016) uses the KKT conditions and their method can be applied if the second-stage problem is a general LP. In contrast, our MILP reformulation of the separation problem exploits the network flow structure of the second-stage problem, and thus is more compact (with fewer variables and constraints) compared to methods in Zeng and Zhao (2013) and Ayoub and Poss (2016) for solving the separation problem with general polyhedral uncertainty sets.

In Algorithm 2, formulation (20) is equivalent to

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\delta} \quad & \mathbf{c}^T\mathbf{x} + \delta \\ \text{s.t.} \quad & \Pi(\mathbf{x}, \boldsymbol{\zeta}^\kappa) \leq \delta, \forall 1 \leq \kappa \leq k. \end{aligned} \tag{21}$$

Thus, formulation (20) is a relaxation of the original TRMCF problem by partially enumerating a small subset of scenarios in the uncertainty set $\mathcal{U}$. Compared to formulation (14) from Algorithm 1 — which is another relaxation of the TRMCF problem — the relaxation (20) includes both $\mathbf{x}$ and $\mathbf{y}$ variables, while (14) has only first-stage variables and fewer constraints. Moreover, the number of $\mathbf{y}$ variables in (20) increases as the number of iterations k goes up. Therefore, with the same number of iterations, Step 1 of Algorithm 2 usually requires more computation time than Step 1 of Algorithm 1. However, the benefit of introducing $\mathbf{y}$ variables in (2) is that at each iteration, it generates much tighter constraints (i.e., cuts) to the relaxation of the TRMCF problem. In other words, suppose $(\mathbf{x}^*, \delta^*)$ is part of the optimal solution we obtained from Step 1 of Algorithm 2 and there exists $\boldsymbol{\zeta}^* \in \mathcal{U}$ such that $\Pi(\mathbf{x}^*, \boldsymbol{\zeta}^*) > \delta^*$. Then, the constraints generated by Algorithm 2 using $\boldsymbol{\zeta}^*$ are stronger than the constraint generated by Algorithm 1 (Step 2a or Step 2b) using the same $\boldsymbol{\zeta}^*$. This observation is formalized in the next proposition.

Algorithm 2 Column-and-Constraint Generation Algorithm for the TRMCF problem.

Initialize with $k \leftarrow 0$.

loop

1. Solve a relaxation of the TRMCF problem with k scenarios:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X},\mathbf{y},\delta} \quad & \mathbf{c}^T\mathbf{x}+\delta \\ \text{s.t.} \quad & \mathbf{f}^T\mathbf{y}^\kappa \le \delta, \quad \forall \kappa = 1,\dots,k \\ & \mathbf{N}\mathbf{y}^\kappa = \mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^\kappa), \quad \forall \kappa = 1,\dots,k \\ & \mathbf{0} \le \mathbf{y}^\kappa \le \mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^\kappa), \quad \forall \kappa = 1,\dots,k. \end{aligned} \tag{20}$$

Let $(\mathbf{x}^*,\delta^*)$ be the optimal solution.

2a. Solve optimization problem (4) with $\mathbf{x}=\mathbf{x}^*$ via the MILP reformulation in Proposition 3.

if the objective of (4) is strictly positive **then**

Let $\boldsymbol{\zeta}^{k+1}$ be the value of $\boldsymbol{\zeta}$ in the optimal solution of (4). Go to Step 3.

2b. Solve the optimization problem (8) with $\mathbf{x}=\mathbf{x}^*$ via the MILP reformulation in Proposition 4.

if the objective of (8) is strictly greater than δ^* **then**

Let $\boldsymbol{\zeta}^{k+1}$ be the value of $\boldsymbol{\zeta}$ in the optimal solution of (8). Go to Step 3.

else

Stop. Return $\mathbf{x}^*$ as an optimal solution.

3. Store $\boldsymbol{\zeta}^{k+1}$; create decision variables $\mathbf{y}^{k+1}\in\mathbb{R}^m$, and add constraints (19a)–(19c) using $\boldsymbol{\zeta}^{k+1}$ and $\mathbf{y}^{k+1}$. Set $k\leftarrow k+1$. Go to Step 1.

PROPOSITION 6. *Given any $\boldsymbol{\zeta}^*\in\mathcal{U}$, for any $\mathbf{x}\in\mathcal{X}$ and δ, the condition $\Pi(\mathbf{x},\boldsymbol{\zeta}^*)\le\delta$ implies the following:*

$$\mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^*)^T\tilde{\mathbf{p}} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^*)^T\tilde{\mathbf{q}} \le 0, \text{ for all } (\tilde{\mathbf{p}},\tilde{\mathbf{q}}) \text{ feasible for (4)}, \tag{22a}$$

$$\mathbf{b}(\mathbf{x},\boldsymbol{\zeta}^*)^T\mathbf{p} - \mathbf{u}(\mathbf{x},\boldsymbol{\zeta}^*)^T\mathbf{q} \le \delta, \text{ for all } (\mathbf{p},\mathbf{q}) \text{ feasible for (8)}. \tag{22b}$$

Proof. If $\mathbf{x}$ and δ satisfies $\Pi(\mathbf{x},\boldsymbol{\zeta}^*)\le\delta$, then we obviously have $\Pi(\mathbf{x},\boldsymbol{\zeta}^*)<\infty$, which by the strong duality of the feasible flow problem (see Equation (3)), implies that $(\mathbf{x},\delta)$ satisfies Inequalities (22a). Also, by taking the dual of the network problem representing $\Pi(\mathbf{x},\boldsymbol{\zeta}^*)$ (see Equation (7)), we have that $\Pi(\mathbf{x},\boldsymbol{\zeta}^*)\le\delta$ implies $(\mathbf{x},\delta)$ satisfies Inequalities (22b). □.

By Proposition 6, each iteration of the column-and-constraint procedure is effectively adding *exponentially many* constraints in the form of (22a)–(22b); in contrast, the constraint generation procedure in §3.2 adds only a single constraint at each iteration that maximizes the left-hand-side of (22a) and (22b). Therefore, the column-and-constraint generation algorithm usually terminate in much fewer iterations than the constraint generation algorithm.

In §3.2, we argued that Algorithm 1 terminates within T steps, where T is the number of linear constraints in the reformulation given by Theorem 1. It turns out that Algorithm 2 has a slightly

stronger theoretical termination guarantee: the maximum number of iterations is bounded by the number of extreme points of the uncertainty set, or T, whichever is smaller.

COROLLARY 2. *The number of iterations of Algorithm 2 is bounded by*

$$\min\{|\text{ext}(\mathcal{U})|, T\},$$

where $\text{ext}(\mathcal{U})$ *is the set of extreme points of uncertainty set* $\mathcal{U}$*, and* T *is the number of linear constraints in formulation* (11) *from Theorem 1.*

Proof. Let k denote the number of iterations of Algorithm 2. First, we show that $k \leq |\text{ext}(\mathcal{U})|$. For each iteration $\kappa = 1, \ldots, k$, scenario $\boldsymbol{\zeta}^\kappa$ is either an optimal solution of (4) or (8). In both cases, $\boldsymbol{\zeta}^\kappa$ is an extreme point of $\mathcal{U}$. By Step 1 of Algorithm 2, the same $\boldsymbol{\zeta}^\kappa$ cannot be generated more than once. Therefore, we have $k \leq |\text{ext}(\mathcal{U})|$.

Next, we show that $k \leq T$. For each iteration $\kappa = 1, \ldots, k$, if scenario $\boldsymbol{\zeta}^\kappa$ is generated from Step 2a, then by Proposition 6, $\Pi(\mathbf{x}, \boldsymbol{\zeta}^\kappa) \leq \delta$ includes at least one constraint of form (11b); and if the scenario $\boldsymbol{\zeta}^\kappa$ is generated from Step 2b, $\Pi(\mathbf{x}, \boldsymbol{\zeta}^\kappa) \leq \delta$ includes at least one constraint of form (11c). Because the same constraint cannot be generated more than once, we have $k \leq T$, the number of constraints of form (11b) or (11c). This complete the proof. □

To see the bound from Corollary 2 is indeed an improvement over either $|\text{ext}(\mathcal{U})|$ or T, consider

$$\mathcal{U}_1 = \{\boldsymbol{\zeta} \in \mathbb{R}^n : \sum_{i=1}^{n} (\zeta_i - k)^+ \leq \frac{(n-k+1)(n-k)}{2}, \forall 1 \leq k \leq n\},$$

$$\text{and } \mathcal{U}_2 = \{\boldsymbol{\zeta} \in \mathbb{R}^n : \sum_{i=1}^{n} \zeta_i = (n-1)l + u, l \leq \zeta_i \leq u, \forall 1 \leq i \leq n\}.$$

First, observe that $\boldsymbol{\zeta}^* = [1, 2, \ldots, n]^T$ is an extreme point in $\mathcal{U}_1$. Because $\mathcal{U}_1$ is defined symmetrically, if we permute the entries in $\boldsymbol{\zeta}^*$ by any permutation, we get a different extreme point in $\mathcal{U}_1$, as the entires in $\boldsymbol{\zeta}^*$ are unique. Therefore, $\mathcal{U}_1$ has at least $n!$ extreme points, and since T grows exponentially with a constant base, we have $|\text{ext}(\mathcal{U}_1)| > T$ when n is large enough. Next, observe that $\text{ext}(\mathcal{U}_2)$ is consisted of all vectors with $n-1$ entries equal to u and one entry equal to l. Therefore, we have that $|\text{ext}(\mathcal{U}_2)| = n$ and hence $|\text{ext}(\mathcal{U}_2)| < T$ for large n. Finally, we note that Corollary 2 is mainly of theoretical interest. In numerical studies, we observe that Algorithm 2 typically converges in several iterations, much smaller than the value of $\min\{|\text{ext}(\mathcal{U})|, T\}$.

We conclude §3.3 by stating that in our numerical examples, the column-and-constraint generation (C&CG) algorithm usually converges faster than the constraint generation (CG) algorithm, confirming the intuition behind Proposition 6 and Corollary 2. However, the C&CG algorithm does significantly increase the complexity of the master problem (the optimization problem solved in Step 1 of Algorithm 1 and 2). While our numerical results consistently show that C&CG is the

superior method, when the master problem is difficult to solve, it is possible that C&CG algorithm spends much longer time each iteration and therefore result in an inferior performance compared to the CG algorithm.

4. Numerical Examples and Applications

In this section, we test the constraint generation (CG) algorithm and the column-and-constraint generation (C&CG) algorithm using numerical examples.

4.1. Joint Inventory-Transportation Problem

We consider a classical problem in logistics with joint inventory and transportation decisions. The problem can be naturally formulated into the two-stage robust optimization framework, and several variants of this problem have been studied in the robust optimization literature by Atamtürk and Zhang (2007), Gabrel et al. (2014a), Ardestani-Jaafari and Delage (2017), Bertsimas and de Ruiter (2016).

Suppose a firm needs to distribute a product through a network with multiple supply nodes ($i = 1, \ldots, N_s$) and multiple demand nodes ($j = 1, \ldots, N_d$). In the first stage, the firm determines the storage level (x_i) at each supply node, without actual demand realization. We assume the demand realization at all demand nodes is given by $\mathbf{d}(\boldsymbol{\zeta}) = \bar{\mathbf{d}} + \boldsymbol{\sigma} \circ \boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ belongs to an uncertainty set of the following form:

$$\mathcal{U} = \{\boldsymbol{\zeta} \in \mathbb{R}^{N_d} : -\mathbf{1} \leq \boldsymbol{\zeta} \leq \mathbf{1}, |\boldsymbol{\zeta}|_1 \leq \beta, \mathbf{1}^T \boldsymbol{\zeta} \leq \gamma\}.$$

Here, the parameter $\boldsymbol{\sigma}$, together with $-\mathbf{1} \leq \boldsymbol{\zeta} \leq \mathbf{1}$, limits the demand variation of each individual demand node. The constraints $|\boldsymbol{\zeta}|_1 \leq \beta$ and $\mathbf{1}^T \boldsymbol{\zeta} \leq \gamma$ bound the aggregate absolute demand variation and total demand across all the demand nodes. These two constraints are motivated by the central limit theorem (CLT) approach from Bandi and Bertsimas (2012): if variables $\zeta_j \, (j = 1, \cdots, n)$ are sampled independently in $[-1, 1]$ (and hence $|\zeta_j| \, (j = 1, \cdots, n)$ are sampled independently in $[0, 1]$ as well), one can choose appropriate values for β and γ using the CLT such that the constraints hold with high probability for large n. In our numerical example, we assume that ζ_j follows uniform distribution and choose the bounds at two standard deviations. Thus, we set

$$\beta = nE[|\zeta_j|] + 2\sqrt{n \cdot Var(|\zeta_j|)} = n/2 + \sqrt{n/3}, \quad \gamma = nE[\zeta_j] + 2\sqrt{n \cdot Var(\zeta_j)} = 2\sqrt{n/3}.$$

Note that for $n \geq 1$, we have $\beta > \gamma$, so the constraint $\mathbf{1}^T \boldsymbol{\zeta} \leq \gamma$ is not redundant. We choose to bound both the aggregate absolute demand variation and total demand because this provides a less conservative robust optimization model for the true demand distribution close to independent.

In the special case with $\gamma = +\infty$, $\mathcal{U}$ becomes a budget uncertainty set in the form of Bertsimas and Sim (2004). Furthermore, when β is an integer, $\mathcal{U}$ is a 0-1 polytope. In that case, it is known

that the separation problem can be solved by enumerating all extreme points of $\mathcal{U}$ or by applying the MILP reformulation (see Thiele et al. 2010, Zeng and Zhao 2013, Ayoub and Poss 2016). However, in the general case with $\gamma < \beta$, where both γ and β taking fractional values, $\mathcal{U}$ is not a 0-1 polytope. Furthermore, it is not known whether $\mathcal{U}$ admits a compact 0-1 polytope representation. Therefore, in the algorithms we test below, we treat $\mathcal{U}$ as a general polyhedral set in our computational algorithms.

We assume that inventory stored at supply node i in the first stage incurs a unit holding cost of c_i. In the second stage, parameter $\boldsymbol{\zeta}$ (and demand $\mathbf{d}(\boldsymbol{\zeta})$) is realized, and the firm must transport items from the supply nodes to the demand nodes through a bipartite network with arcs $\mathcal{A}$. A flow from supply node i to demand node j has a unit transportation cost of f_{ij}. The problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}\geq 0} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \min_{\mathbf{y}\geq 0} \quad & \sum_{i=1}^{N_s} c_i x_i + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{A}} y_{ij} \leq x_i, \quad \forall i = 1,\ldots,N_s \\ & \sum_{i:(i,j)\in\mathcal{A}} y_{ij} = d_j(\boldsymbol{\zeta}), \quad \forall j = 1,\ldots,N_d. \end{aligned} \tag{23}$$

The first set of constraints enforces inventory constraints at supply nodes, and the second set of constraints specifies that all demand should be met from the shipments.

To show that (23) is a TRMCF problem, we add a dummy demand node indexed by 0, and then add arcs from all supply nodes to it with zero transportation cost. We assume the demand at location 0 is given by

$$d_0(\mathbf{x},\boldsymbol{\zeta}) = \sum_{i=1}^{N_s} x_i - \sum_{j=1}^{N_d} d_j(\boldsymbol{\zeta}).$$

Then we can equivalently formulate (23) as

$$\begin{aligned} \min_{\mathbf{x}\geq 0} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \min_{\mathbf{y}\geq 0} \quad & \sum_{i=1}^{N_s} c_i x_i + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in\mathcal{A}} y_{ij} = x_i, \quad \forall i = 1,\ldots,N_s \\ & -\sum_{i:(i,j)\in\mathcal{A}} y_{ij} = -d_j(\boldsymbol{\zeta}), \quad \forall j = 1,\ldots,N_d, \\ & -\sum_{i:(i,0)\in\mathcal{A}} y_{i0} = -d_0(\mathbf{x},\boldsymbol{\zeta}). \end{aligned} \tag{24}$$

Formulation (24) is a TRMCF problem in the form of Equation (2) where demand $b(\mathbf{x},\boldsymbol{\zeta})$ is given by the right-hand side of (24) and the arc capacities are $u(\mathbf{x},\boldsymbol{\zeta}) = +\infty$. Note that problem (24)

does not have relatively complete recourse. For example, the second-stage problem is infeasible if $d_0(\mathbf{x}, \boldsymbol{\zeta}) < 0$ for some $\boldsymbol{\zeta} \in \mathcal{U}$.

We compare five algorithms for this problem:

1. Full-LP: directly solving the reformulation of (23) by Theorem 1, which is a linear program in this example;
2. CG: constraint generation Algorithm 1 presented in §3.2;
3. C&CG: constraint-and-column generation Algorithm 2 presented in §3.3.

In addition, Ayoub and Poss (2016) have recently proposed an MILP approach to solve the separation problem of two-stage robust optimization with general polyhedral uncertainty set. Appendix 5 shows how their method can be applied to the TRMCF problem. Thus, we also test the following two algorithms by replacing Step 2 in Algorithms 1 and 2 with the MILP formulation in Ayoub and Poss (2016):

4. CG-KKT: constraint generation algorithm by Ayoub and Poss (2016), which reformulates the separation problem using the KKT conditions;
5. C&CG-KKT: constraint-and-column generation algorithm by Ayoub and Poss (2016), which reformulates the separation problem using the KKT conditions.

We first test a setting without second-stage transportation costs. We generate four groups of bipartite networks with numbers of nodes ranging from 20 to 500. Each group includes 100 random problem instances. We assume the number of warehouses and number of demand locations are equal, and each demand location can be served by a random number of warehouses which is uniformly distributed between 1 and 10. The inventory holding cost at each warehouse is uniformly distributed between 1 and 10.

The tests were performed using the Gurobi 7.0.2 solver on a 2.8GHz Intel CPU. During each constraint or column-and-constraint generation iteration, instead of solving the MILP formulation to optimality, we stop whenever the solver finds a violating constraint at a prespecified level.[2] In Table 1, we show the average CPU time of different algorithms. By Corollary 1, the linear programming reformulation of this special case has exactly 2^n constraints, where n is the number of nodes in the network. In this example, because the network is bipartite and there is no uncertainty associated with supply nodes (warehouses), we can further reduce the number of constraints to $2^{N_d} = 2^{n/2}$. We notice that for networks with small size ($n = 20$), the speed of Full-LP is comparable to CG and C&CG. However, when the network size grows, Full-LP quickly becomes impractical

[2] For example, when we solve the separation problem by solving the MILP in Proposition 4, we stop the optimization procedure whenever we find a solution with objective θ such that $\theta - \delta > 0$ and $\theta - \delta$ is within x fraction of $\theta^* - \delta$ where θ^* is the optimal objective of the MILP. While any value of $x \geq 0$ will ensure the correctness of our algorithm, we choose $x = 0.2$ in our experiments. Intuitively, bigger x terminates the MILP solver earlier when a cut is found, while smaller x let the MILP solver work longer to find a better cut.

to solve since the number of constraints grows exponentially fast. Full-LP is not able to solve for networks with more than 100 nodes. CG is able to solve network with hundreds of nodes. C&CG is extremely efficient, and is able to solve a logistics network with 500 nodes under 3 seconds.

The CG and C&CG algorithms by Ayoub and Poss (2016) (CG-KKT and C&CG-KKT), on the other hand, require significantly longer CPU time than our CG and C&CG algorithms and even the Full-LP algorithm. When the number of nodes is greater than 50, the method by Ayoub and Poss (2016) does not terminate in six hours for a single test instance, so we are not able to report average CPU time for $n \geq 50$. When $n = 20$, the method by Ayoub and Poss (2016) is about 10,000 times slower than our CG and C&CG algorithms. The large runtime for the algorithm for Ayoub and Poss (2016) may due to the fact that the algorithm solves an MILP with a large number of binary variables and uses big-M reformulation. We note that Ayoub and Poss (2016) also proposed a significantly more effective algorithm for 0-1 uncertainty sets, but the uncertainty sets in our experiments are not 0-1. Also, while our algorithms exploit the network flow structure of the second-stage problem, the method of Ayoub and Poss (2016) is more general, and can work with any second-stage problem that is an LP.

Table 1 also lists a few additional statistics about the various algorithms we tested. "Number of iterations" gives the average number of iterations in CG, C&CG, CG-KKT, and C&CG-KKT before the optimal solution is found. "MIP/LP ratio" measures the average ratio of the optimal value of an MILP reformulation of the separation problem and its LP relaxation; since the value of the MILP is positive (otherwise the algorithm terminates), the ratio is between 0 and 1, and a ratio close to 1 means the LP relaxation is tight. "% of time on SP" shows the percentage of the time that CG and C&CG algorithms spend on solving separation problems, i.e., step 2 in Algorithms 1 and 2.

In all cases, we observe that compared to the theoretical guarantees, only a small number of iterations is required before the algorithms terminate, which explains why they are faster than Full-LP. In particular, we observe that C&CG requires very few iterations, which is consistent with the observation by Zeng and Zhao (2013). The CG and C&CG algorithms by Ayoub and Poss (2016) require roughly the same number of iterations as our method, but solving each separation problem is slow: the algorithms spend more than 99.9% of the time on solving separation problems. A possible explanation is that the MILP reformulation by Ayoub and Poss (2016) of the separation problem uses the big-M method and is thus inefficient; evidently, the ratio of the optimal value of the MILP reformulation and its LP relaxation is close to 0. Finally, note that the master problem in CG and C&CG is an LP. Thus, even C&CG adds more variables to the master problem and spends a larger portion of time solving them, the additional variables do not significantly affect

Table 1 Efficiency of Different Algorithms without Transportation Cost

	Number of nodes (n)	20	50	100	200	500
Full-LP	CPU time (sec)	0.12	149	—	—	—
	Constraints ($2^{n/2}$)	1.0×10^{3}	1.0×10^{6}	1.1×10^{15}	1.3×10^{30}	1.8×10^{75}
CG	CPU time (sec)	0.03	0.34	1.98	14.0	516
	Number of iter.	11.3	47.2	123	365	1472
	MIP/LP ratio	0.837	0.781	0.853	0.853	0.823
	% of time on SP	75.0%	83.9%	85.6%	87.6%	90.2%
C&CG	CPU time (sec)	**0.02**	**0.13**	**0.28**	**0.44**	**2.70**
	Number of iter.	4.4	8.5	8.9	8.4	8.3
	MIP/LP ratio	0.656	0.622	0.789	0.858	0.878
	% of time on SP	68.7%	78.9%	81.5%	79.4%	84.1%
CG-KKT	CPU time (sec)	324.5	—	—	—	—
	Number of iter.	9.8	—	—	—	—
	MIP/LP ratio	0.004	—	—	—	—
	% of time on SP	99.9%	—	—	—	—
C&CG-KKT	CPU time (sec)	266.7	—	—	—	—
	Number of Iter.	5.4	—	—	—	—
	MIP/LP ratio	0.003	—	—	—	—
	% of time on SP	99.9%	—	—	—	—

the total computational time, which explains why C&CG is much faster than CG. In Section 4.2, we further investigate the efficiency of C&CG when the master problem contains integer variables.

We then test the algorithms for settings with second-stage transportation cost. We generate four groups of bipartite networks with numbers of nodes increasing from 10 to 200, where each group include 100 random problem instances. The network instances used are the same as before. In addition, we assume each arc has an integral transportation cost randomly distributed between 1 and 8. By Theorem 1, the LP reformulation has no more than $2^{n/2}+8^{n/2}$ constraints. (Because the network is bipartite, we can choose $p^{\max}=8$ in (11c).) We list the CPU time of different algorithms in Table 2. We observe that Full-LP is not able to solve for networks with more than 10 nodes. CG and C&CG are able to solve networks with 100 nodes, and again C&CG is much faster than CG and terminates in much fewer iterations. Similar to the results in Table 1, the method of Ayoub and Poss (2016) is slow and is not able to solve networks with more than 20 nodes. The MILP reformulation of the separation problem has a nearly zero MIP/LP ratio, and the algorithms spend more than 99.9% of the time on solving separation problems.

4.2. Joint Location-Transportation Problem

In this example, we extend the previous lot-sizing and transportation setting by including facility location decisions. For each supply node $i=1,\dots,N_s$, let $z_i\in\{0,1\}$ be a binary variable indicating if a facility is open at location i. If the facility at location i is open, we assume a fixed cost g_i is incurred, and the facility has capacity K_i. The other model assumptions remain the same as in the

Table 2 Efficiency of Different Algorithms with Transportation Cost

	Number of nodes (n)	10	20	50	100
Full-LP	CPU time (sec)	3.34	—	—	—
	Total constraints	3.3×10^4	1.1×10^9	3.8×10^{22}	1.4×10^{45}
CG	CPU time (sec)	0.05	0.35	22.3	480
	Number of iter.	5.8	13.9	64.9	213
	MIP/LP ratio	0.763	0.590	0.437	0.438
	% of time on SP	92.6%	97.5%	99.8%	99.9%
C&CG	CPU time (sec)	**0.04**	**0.17**	**16.1**	**66.7**
	Number of iter.	2.8	4.6	12.2	25.5
	MIP/LP ratio	0.627	0.463	0.272	0.243
	% of time on SP	90.6%	95.3%	98.9%	99.6%
CG-KKT	CPU time (sec)	1.02	321.3	—	—
	Number of iter.	7.4	16	—	—
	MIP/LP ratio	0.001	0.000	—	—
	% of time on SP	98.8%	99.9%	—	—
C&CG-KKT	CPU time (sec)	0.95	224.8	—	—
	Number of iter.	3.2	7.5	—	—
	MIP/LP ratio	0.000	0.000	—	—
	% of time on SP	98.6%	99.9%	—	—

previous lot-sizing/ transportation example. Then, the two-stage robust location-transportation problem is formulated as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z}}\max_{\boldsymbol{\zeta}}\min_{\mathbf{y}} \quad & \sum_{i=1}^{N_s} g_i z_i + \sum_{i=1}^{N_s} c_i x_i + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} \\
\text{s.t.} \quad & x_i \le K_i z_i, \quad \forall i = 1,\dots,N_s \\
& \sum_{j:(i,j)\in\mathcal{A}} y_{ij} \le x_i, \quad \forall i = 1,\dots,N_s \\
& \sum_{i:(i,j)\in\mathcal{A}} y_{ij} = d_j(\boldsymbol{\zeta}), \quad \forall j = 1,\dots,N_d \\
& \mathbf{x}\ge 0,\ \mathbf{z}\in\{0,1\}^{N_s},\ \boldsymbol{\zeta}\in\mathcal{U},\ \mathbf{y}\ge 0.
\end{aligned}
\tag{25}
$$

Similar to §4.1, we test CG (Algorithm 1), C&CG (Algorithm 2), CG-KKT and C&CG-KKT for this problem. We randomly generate four groups of networks with numbers of nodes ranging from 10 to 100. We use the same demand uncertainty sets and cost parameters defined in §4.1. We assume that the fixed cost is $g_i = 5$ and capacity is $K_i = 10$.

Table 3 lists the CPU time, number of iterations, MIP/LP ratio, and percentage of time spent on separation problems for different algorithms (see the definition of these statistics in §4.1). Compared with Table 2, we find that adding first-stage binary variables does not significantly affect the computation time of CG and C&CG when $n \le 50$. This is because even if the master problem is now an MILP, the number of iterations (and thus the size of the master problem) is small when $n \le 50$. When $n = 100$, the computation time of CG becomes significantly worse. The reason is that

CG needs to add about 500 constraints to the master problem. The master problem, now having binary variables, becomes harder and harder to solve as more constraints are added.

Surprisingly, the C&CG algorithm remains efficient for $n = 100$, even if the algorithm adds more constraints and variables to the master problem per iteration. We suspect that the reason is that C&CG terminates after only 22 iterations, while CG terminates after 581 iterations on average. Therefore, the size of the master problem of C&CG is still smaller than that of CG, even if it adds more variables and constraints per iteration. Finally, similar to §4.1, CG-KKT and C&CG-KKT cannot solve networks with more than 20 nodes.

Table 3 Efficiency of Different Algorithms with Location Decisions

	Number of nodes (n)	10	20	50	100
	Total constraints	3.3×10^4	1.1×10^9	3.8×10^{22}	1.4×10^{45}
CG	CPU time (sec)	0.07	0.46	32.8	4849
	Number of Iter.	5.4	13.9	91.8	581
	MIP/LP ratio	0.834	0.696	0.578	0.579
	% of time on SP	77.1%	80.6%	61.2%	8.6%
C&CG	CPU time (sec)	**0.05**	**0.22**	**7.8**	**71.7**
	Number of Iter.	2.6	3.9	11.3	21.9
	MIP/LP ratio	0.687	0.513	0.313	0.307
	% of time on SP	78.0%	78.4%	66.7%	50.8%
CG-KKT	CPU time (sec)	4.19	—	—	—
	Number of Iter.	6.1	—	—	—
	MIP/LP ratio	0.004	—	—	—
	% of time on SP	98.4%	—	—	—
C&CG-KKT	CPU time (sec)	1.74	—	—	—
	Number of Iter.	2.4	—	—	—
	MIP/LP ratio	0.005	—	—	—
	% of time on SP	98.7%	—	—	—

4.3. Production Postponement

We next perform numerical experiment on a production postponement application described in Chou et al. (2014), which is based on the Sports Obermeyer Ltd. case by Hammond and Raman (1994). In this application, Sports Obermeyer Ltd. sells ten different styles of parkas. The company is required to order a portion of its parkas before demand is realized, while the rest of the order can be postponed after demand is revealed. Each style j has its own price, and its demand follows normal distribution with mean μ_j and standard deviation σ_j that is truncated at zero. The price, mean demand, and standard deviation of demand for different styles are listed in Table 4. Demand is lost if there is not enough parkas produced, and lost sales cost of each parka style j is equal to 24% of its price.

In the first stage, Sports Obermeyer Ltd. needs to produce a total of C units of parkas. In the second stage, after the demand $\mathbf{d}(\boldsymbol{\zeta})$ (defined as $\mathbf{d}(\boldsymbol{\zeta}) = \boldsymbol{\mu} + \boldsymbol{\zeta} \circ \boldsymbol{\sigma}$) is realized, the firm produces the remaining parkas from its ten facilities. We use $\mathbf{x}$ and $\mathbf{y}$ to denote the first and second-stage

Table 4 Sports Obermeyer Data

Demand and price information for different styles										
	Gail	Isis	Entice	Assault	Tri	Electra	Stephanie	Seduced	Anita	Daphane
ID (j)	1	2	3	4	5	6	7	8	9	10
Mean (μ_j)	1017	1042	1358	2525	1100	2150	1113	4017	3296	2383
Std.v. (σ_j)	194	323	248	340	381	404	524	556	1047	697
Price (π_j)	110	99	80	90	123	173	133	73	93	148

production quantities, respectively. For $1 \le i \le 10$, facility i has (second-stage) capacity c_i. The production capability of facilities is represented by set $\mathcal{F}$, where $(i,j) \in \mathcal{F}$ implies that facility i is capable of producing style j. In sum, the decisions of Sports Obermeyer Ltd. can be modeled as the following two-stage robust optimization problem:

$$\begin{aligned} \min_{\mathbf{x}\ge\mathbf{0}} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \min_{\mathbf{y},\mathbf{l}\ge\mathbf{0}} \quad & \sum_{j=1}^{10} u_j l_j \\ \text{s.t.} \quad & \sum_{i:\,(i,j)\in\mathcal{F}} y_{ij} + l_j \ge d_j(\zeta) - x_j, \quad \forall 1 \le j \le 10, \\ & \sum_{j:\,(i,j)\in\mathcal{F}} y_{ij} \le c_i, \quad \forall 1 \le i \le 10, \\ & \sum_{j=1}^{10} x_j = C. \end{aligned} \tag{26}$$

The objective of (26) is to minimize worst-case lost sales, where u_j is the lost sales cost. The first set of constraints specifies that demand not met is lost. The second set of constraints specifies the production capacity at the second stage. The third constraint represents the production capacity at the first stage. Similar to §4.1, we assume the demand realization at all demand nodes is given by $\mathbf{d}(\boldsymbol{\zeta}) = \bar{\mathbf{d}} + \boldsymbol{\sigma} \circ \boldsymbol{\zeta}$, and uncertainty set $\mathcal{U}$ to be all $\boldsymbol{\zeta}$ satisfying

$$\mathcal{U} = \{\boldsymbol{\zeta} : \sum_{j=1}^{10} |\zeta_j| \le \beta, \sum_{j=1}^{10} \zeta_j \le \gamma, -1 \le \zeta_j \le 1, \forall 1 \le j \le 10\}.$$

Finally, like §4.1, β and γ are chosen based on the central limit theorem.

The second-stage minimization problem in (26) can be converted into a minimum cost flow problem. Therefore, we can convert (26) into a TRMCF problem, and solve it using algorithms developed in §3. Also, it is easy to check that any first-stage decision $\mathbf{x} \ge \mathbf{0}$ is feasible for (26), implying that problem (26) has relatively complete recourse. As a result, by Corollary 1, the reformulation of TRMCF for problem (26) is simpler than the general TRMCF model. Finally, we can also take advantage of the specific structures in (26) to show that the dual of the second stage network flow problem can be rewritten as an integer program with just integer variables $\{p_j\}_{j=1}^{10}$, and constraints $p_j \le u_j$ for $1 \le j \le 10$. This implies that the reformulation of TRMCF have at most linear $\Pi_{j=1}^{10}(u_j + 1)$ constraints.

Like §4.1 and 4.2, we test the performance of the constraint generation (CG) algorithm and the constraint-and-column generation (C&CG) algorithm for this postponement problem. We randomly generate test instances as follows. We generate $\mathcal{F}$ by starting with the set $\{(i,i)|1 \le i \le 10\}$, implying facility i can always produce parka style i. Then for any i,j with $i \neq j$, we add arc (i,j) with probability 0.1. The additional arcs represent the flexibility of facilities for producing different styles. The first-stage capacity parameter C is drawn uniformly at random from $[0.7\mu_T, 0.9\mu_T]$. where $\mu_T = \sum_{1\le j\le 10} \mu_j$. The second-stage capacity is assumed to be 20% of the expected demand of style i, i.e., $c_i = 0.2\mu_i$. Finally, because our algorithms require the coefficient of the objective function (u_j) to be integers, we rescale and round lost sales cost u_j in increments of U by rounding u_j to the nearest integer to $0.24 \cdot \pi_j/U$ for $U \in \{10, 4, 1, 0.1\}$. Note that a bigger U implies that u_j is smaller after rounding. Therefore the total number of constraints in formulation (11) decreases as we increase the rounding unit U, which improves our computational time. However, picking a big rounding unit U decreases the precision of our estimation to the true lost sales cost.

In Table 5, we list the CPU time of different methods under different rounding unit. Larger rounding units correspond to smaller MILP reformulations of the separation problem, as we deal with smaller binary expansions. For the Full-LP algorithm, we list the number of constraints in the LP formulation in the table; however, as most of instances contains too many constraints, it is impractical to apply the Full-LP algorithm for this problem. Like §4.1 and 4.2, we observe that C&CG is always faster than CG and requires less iterations across different rounding units. Since problem (26) has relatively complete recourse, we were also able to apply the method in Zeng and Zhao (2013) for solving two-stage robust optimization with general polyhedral sets. The results are reported as CG-KKT and C&CG-KKT in Table 5. Notice that the method in Zeng and Zhao (2013) does not require the second-stage cost to be integral, so there is only one column of results. We observe that the method by Zeng and Zhao (2013) is slower than our algorithms, but the computation time is of the same order of magnitude. This is in contrast to the results for the location-transportation examples in §4.1 and 4.2, where the method by Ayoub and Poss (2016), a generalization of the method by Zeng and Zhao (2013), is slower than our algorithms by several orders of magnitude. We suspect that the reason is that the networks in this postponement application have only 20 nodes; moreover, the networks are sparse, with an average of 20 edges per network. This means that the MILP reformulation in Zeng and Zhao (2013) and the MILP reformulation in Theorem 1 has similar size.

5. Conclusion

In this paper, we study two-stage robust network flow problems. This class of problems can model a wide range of tasks where decisions are made in two stages and the second-stage decision can

Table 5 Efficiency of Different Solution Methods for the Postponement Problem ($n = 10$)

	Rounding To (U)	10	4	1	0.1
Full-LP	Number of Const.	38800	$\sim 5.6 \times 10^8$	$\sim 2.2 \times 10^{14}$	$\sim 1.5 \times 10^{24}$
CG	Avg CPU Time(sec)	2.66	12.51	42.81	91.76
	Number of Iter.	29.28	39.57	40.38	42.95
C&CG	Avg CPU Time(sec)	1.59	7.52	28.67	43.38
	Number of Iter.	11.84	13.28	13.18	13.15
CG-KKT	Avg CPU Time(sec)			63.35	
	Number of Iter.			34.34	
C&CG-KKT	Avg CPU Time(sec)			57.81	
	Number of Iter.			13.40	

be formulated as network flows. To solve the two-stage robust network flow problem, first, we propose a constraint generation (CG) algorithm, where the constraint generation subroutine involves solving a bilinear separation problem. We reformulate the bilinear program as a pair of MILPs using the network flow structure, so the constraint generation subroutine can be performed using general MILP solvers. Our MILP reformulation technique allows the uncertainty set of the problem to be an arbitrary polytope, and does not require the problem to satisfy the relatively complete recourse assumption. Moreover, we propose a second algorithm that combines our MILP formulation technique with the column-and-constraint generation framework. The column-and-constraint generation (C&CG) algorithm reduces the number of constraint generation iterations, at a cost of increasing the complexity of the master problem. Finally, we test both CG and C&CG algorithms using numerical examples from lot-sizing/transportation and production postponement applications, and find that both algorithms are effective algorithms for solving large instances.

Appendix

A. An alternative MILP reformulation of the separation problem using KKT conditions

Ayoub and Poss (2016) proposed an MILP reformulation of the separation problem for two-stage robust optimization. The reformulation applies to general second-stage problems under general polyhedral uncertainty, without assuming relatively complete recourse. The key idea of Ayoub and Poss (2016) is to formulate the separation problem as a bilevel optimization problem and then use big-M method to linearize complementary slackness conditions (referred to as KKT conditions in Zeng and Zhao (2013)). Similar ideas have appeared in Mattia (2013), Zeng and Zhao (2013). We present the method by Ayoub and Poss (2016) below in the context of robust network flow problems.

Let $(\tilde{\mathbf{x}}, \tilde{\delta})$ be a first-stage solution to the TRMCF problem (2). By definition, the solution is feasible if and only if $\Pi(\tilde{\mathbf{x}}, \boldsymbol{\zeta}) \leq \tilde{\delta}, \forall \boldsymbol{\zeta} \in \mathcal{U}$. Equivalently, by Eq (1), $(\tilde{\mathbf{x}}, \tilde{\delta})$ is feasible when the optimization problem below has an optimal objective value equal to 0.

$$\max_{\boldsymbol{\zeta} \in \mathcal{U}} \min_{\mathbf{y}} \quad 0$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{f}^T\mathbf{y} \le \tilde{\delta} \\ & \mathbf{N}\mathbf{y} = \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta}) \\ & \mathbf{0} \le \mathbf{y} \le \mathbf{u}(\tilde{\mathbf{x}}, \boldsymbol{\zeta}). \end{aligned}$$

By strong duality, we can reformulate the problem above as

$$\begin{aligned} \max_{\boldsymbol{\zeta}\in\mathcal{U}} \max_{\mathbf{p},\mathbf{q}} \quad & \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})^T\mathbf{q} - \tilde{\delta} r \\ \text{s.t.} \quad & \mathbf{N}^T\mathbf{p} - \mathbf{I}\mathbf{q} - r\mathbf{f} \le 0 \\ & \mathbf{p} \in \mathbb{R}^n, \mathbf{q} \in \mathbb{R}^m_+, r \ge 0. \end{aligned}$$

Because $\mathbf{N}$ is the node-arc incidence matrix and $\mathbf{1}_n^T\mathbf{b}(\mathbf{x}, \boldsymbol{\zeta}) = 0$ by flow balance condition, we can assume $\mathbf{p} \ge 0$ without changing optimal value of the objective. Moreover, given any solution $(\mathbf{p}, \mathbf{q}, r)$, we can rescale all the decision variables by the same positive constant factor without changing the sign of the objective function. Therefore, $(\tilde{\mathbf{x}}, \tilde{\delta})$ is feasible for the first-stage problem if and only if the optimization problem below has a nonpositive optimal value (i.e., $g(\tilde{\mathbf{x}}, \tilde{\delta}) \le 0$).

$$\begin{aligned} g(\tilde{\mathbf{x}}, \tilde{\delta}) = \max_{\boldsymbol{\zeta}\in\mathcal{U}} \max_{\mathbf{p},\mathbf{q}} \quad & \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})^T\mathbf{p} - \mathbf{u}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})^T\mathbf{q} - \tilde{\delta} r \\ \text{s.t.} \quad & \mathbf{N}^T\mathbf{p} - \mathbf{I}\mathbf{q} - r\mathbf{f} \le 0 && \text{(27a)} \\ & \mathbf{p}^T\mathbf{1}_n + \mathbf{q}^T\mathbf{1}_m + r \le 1 && \text{(27b)} \\ & \mathbf{p} \in \mathbb{R}^n_+, \mathbf{q} \in \mathbb{R}^m_+, r \ge 0. \end{aligned}$$

By strong duality again, we have

$$\begin{aligned} g(\tilde{\mathbf{x}}, \tilde{\delta}) = \max_{\boldsymbol{\zeta}\in\mathcal{U}} \min_{\mathbf{y}\ge\mathbf{0}, z\ge 0} \quad & z \\ \text{s.t.} \quad & -\mathbf{f}^T\mathbf{y} + z \ge -\tilde{\delta} && \text{(28a)} \\ & \mathbf{N}\mathbf{y} + z\mathbf{1}_n \ge \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta}) && \text{(28b)} \\ & -\mathbf{y} + z\mathbf{1}_m \ge -\mathbf{u}(\tilde{\mathbf{x}}, \boldsymbol{\zeta}). && \text{(28c)} \end{aligned}$$

This is a bilevel optimization problem. Note that the lower-level problem always has a finite optimal value since the feasible set (27a)–(27b) is bounded and nonempty ($\mathbf{p} = \mathbf{0}, \mathbf{q} = \mathbf{0}, r = 0$ is a feasible solution). Therefore, we can rewrite the lower-level problem by KKT conditions as

$$\begin{aligned} g(\tilde{\mathbf{x}}, \tilde{\delta}) = \max_{\boldsymbol{\zeta},\mathbf{y},z,\mathbf{p},\mathbf{q},r} \quad & z && \text{(29a)} \\ \text{s.t.} \quad & \text{(27a)} - \text{(28c)} \\ & \left[\mathbf{N}\mathbf{y} + z\mathbf{1}_n - \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})\right]^T \mathbf{p} = 0 && \text{(29b)} \\ & \left[-\mathbf{y} + z\mathbf{1}_m + \mathbf{u}(\tilde{\mathbf{x}}, \boldsymbol{\zeta})\right]^T \mathbf{q} = 0 && \text{(29c)} \\ & \left[-\mathbf{f}^T\mathbf{y} + z + \tilde{\delta}\right] r = 0 && \text{(29d)} \\ & \left[\mathbf{N}^T\mathbf{p} - \mathbf{I}\mathbf{q} - r\mathbf{f}\right]^T \mathbf{y} = 0 && \text{(29e)} \\ & \left[\mathbf{p}^T\mathbf{1}_n + \mathbf{q}^T\mathbf{1}_m + r - 1\right]^T z = 0 && \text{(29f)} \\ & \boldsymbol{\zeta} \in \mathcal{U}, \mathbf{p} \in \mathbb{R}^n_+, \mathbf{q} \in \mathbb{R}^m_+, r \ge 0, \mathbf{y} \in \mathbb{R}^m_+, z \ge 0. \end{aligned}$$

Equations (27a)–(28c) guarantee primal and dual feasibility, and the constraints (29b)–(29f) guarantee the KKT conditions. The KKT conditions are special ordered sets of type 1 (SOS-1). They can also be represented as linear constraints using binary variables and big-M method. For example, if we introduce a vector of binary variables $\boldsymbol{\mu}$ for constraint (29b), then it can be reformulated as

$$\mathbf{N}\mathbf{y} + z\mathbf{1}_n - \mathbf{b}(\tilde{\mathbf{x}}, \boldsymbol{\zeta}) \leq M\boldsymbol{\mu}, \quad \mathbf{p} \leq M(\mathbf{1}_n - \boldsymbol{\mu}), \quad \boldsymbol{\mu} \in \{0,1\}^n.$$

The other constraints can be reformulated similarly. Therefore, the bilevel problem (29) can be solved as an MILP, and the first-stage variables $(\tilde{\mathbf{x}}, \tilde{\delta})$ is feasible if and only if $g(\tilde{\mathbf{x}}, \tilde{\delta}) \leq 0$.

References

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L. M., Poss, M., and Requejo, C. (2013). The robust vehicle routing problem with time windows. *Computers & Operations Research*, 40(3):856–866.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice hall.

Ardestani-Jaafari, A. and Delage, E. (2016). Robust optimization of sums of piecewise linear functions with application to inventory problems. *Operations research*, 64(2):474–494.

Ardestani-Jaafari, A. and Delage, E. (2017). The value of flexibility in robust location–transportation problems. *Transportation Science*, 52(1):189–209.

Atamtürk, A. and Zhang, M. (2007). Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673.

Ayoub, J. and Poss, M. (2016). Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239.

Bandi, C. and Bertsimas, D. (2012). Tractable stochastic analysis in high dimensions via robust optimization. *Mathematical Programming*, 134(1):23–70.

Ben-Tal, A., Do Chung, B., Mandala, S. R., and Yao, T. (2011). Robust optimization for emergency logistics planning: Risk mitigation in humanitarian relief supply chains. *Transportation Research Part B: Methodological*, 45(8):1177–1189.

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

Ben-Tal, A., Golany, B., Nemirovski, A., and Vial, J.-P. (2005). Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing & Service Operations Management*, 7(3):248–271.

Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

Bertsimas, D. and Caramanis, C. (2010). Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766.

Bertsimas, D. and de Ruiter, F. J. (2016). Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds. *INFORMS Journal on Computing*, 28(3):500–511.

Bertsimas, D. and Goyal, V. (2012). On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical Programming*, 134(2):491–531.

Bertsimas, D., Iancu, D. A., and Parrilo, P. A. (2010). Optimality of affine policies in multistage robust optimization. *Mathematics of Operations Research*, 35(2):363–394.

Bertsimas, D., Litvinov, E., Sun, X. A., Zhao, J., and Zheng, T. (2013a). Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems*, 28(1):52–63.

Bertsimas, D., Nasrabadi, E., and Stiller, S. (2013b). Robust and adaptive network flows. *Operations Research*, 61(5):1218–1242.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Billionnet, A., Costa, M.-C., and Poirion, P.-L. (2014). 2-stage robust milp with continuous recourse variables. *Discrete Applied Mathematics*, 170:21–32.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bonami, P., Kilinç, M., and Linderoth, J. (2012). Algorithms and software for convex mixed integer nonlinear programs. In *Mixed Integer Nonlinear Programming*, pages 1–39. Springer.

Buchheim, C. and Kurtz, J. (2017). Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1-2):1–23.

Chou, M. C., Chua, G. A., and Zheng, H. (2014). On the performance of sparse process structures in partial postponement production systems. *Operations Research*, 62(2):348–365.

Erera, A. L., Morales, J. C., and Savelsbergh, M. (2009). Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483.

Gabrel, V., Lacroix, M., Murat, C., and Remli, N. (2014a). Robust location transportation problems under uncertain demands. *Discrete Applied Mathematics*, 164:100–111.

Gabrel, V., Murat, C., and Thiele, A. (2014b). Recent advances in robust optimization: An overview. *European Journal of Operational Research*, 235(3):471–483.

Georghiou, A., Tsoukalas, A., and Wiesemann, W. (2016). Robust dual dynamic programming. Working paper (available on Optimization Online).

Gorissen, B. L. and Den Hertog, D. (2013). Robust counterparts of inequalities containing sums of maxima of linear functions. *European Journal of Operational Research*, 227(1):30–43.

Gorissen, B. L., Yanıkoğlu, İ., and den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, 53:124–137.

Gupte, A., Ahmed, S., Cheon, M. S., and Dey, S. (2013). Solving mixed integer bilinear problems using milp formulations. *SIAM Journal on Optimization*, 23(2):721–744.

Hammond, J. and Raman, A. (1994). Sport Obermeyer, Ltd. Harvard Business School Case 9-695-022.

Hanasusanto, G. A., Kuhn, D., and Wiesemann, W. (2015). K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891.

Iancu, D. A., Sharma, M., and Sviridenko, M. (2013). Supermodularity and affine policies in dynamic robust optimization. *Operations Research*, 61(4):941–956.

Kuhn, D., Wiesemann, W., and Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209.

Mattia, S. (2013). The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54(3):619–643.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1):147–175.

Simchi-Levi, D., Trichakis, N., and Zhang, P. Y. (2017). Designing response supply chain against bioattacks. *working paper*.

Simchi-Levi, D., Wang, H., and Wei, Y. (2018). Increasing supply chain robustness through process flexibility and inventory. *Production and Operations Management, Forthcoming*.

Simchi-Levi, D. and Wei, Y. (2015). Worst-case analysis of process flexibility designs. *Operations Research*, 63(1):166–185.

Thiele, A., Terry, T., and Epelman, M. (2010). Robust linear optimization with recourse. University of Michigan, IOE Technical Report TR09-01.

Zeng, B. and Zhao, L. (2013). Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461.