Strengthening Dual Bounds for Multicommodity Capacitated Network Design with Unsplittable Flow Constraints

Lacy M. Greening[a,*], Santanu S. Dey[b], Alan L. Erera[b]

[a]*School of Computing and Augmented Intelligence, Arizona State University, Tempe, Arizona, United States*
[b]*H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, United States*

Abstract

Multicommodity capacitated network design (MCND) models can be used to optimize the consolidation of shipments within e-commerce fulfillment networks. In practice, fulfillment networks require that shipments with the same origin and destination follow the same transfer path. This unsplittable flow requirement complicates the MCND problem, requiring integer programming (IP) formulations with binary variables replacing continuous flow variables. To enhance the solvability of instances of this variant of the MCND problem for large-scale logistics networks, this work focuses on strengthening dual bounds. We investigate the polyhedra of arc-set relaxations, and we introduce two new classes of valid inequalities that can be implemented within solution approaches. We develop one approach that dynamically adds valid inequalities to the root node of a reformulation of the MCND IP with additional valid metric inequalities. We demonstrate the effectiveness of our ideas with a comprehensive computational study using path-based fulfillment instances, constructed from data provided by a large U.S.-based e-commerce company, and the well-known arc-based Canad instances. Experiments show that our best solution approach for a practical path-based model reduces the IP gap by an average of 26.5% and 22.5% for the two largest instance groups, compared to solving the reformulation alone, demonstrating its effectiveness in improving the dual bound. In addition, experiments using only the arc-based relaxation of the model highlight the strength of our new valid inequalities when compared only to the linear programming relaxation (LPR), resulting in an average 90% reduction in the IP gap.

Keywords: Network design, fulfillment logistics, valid inequalities, dual bounds

1. Introduction

The rapid growth of e-commerce has led companies to seek improved fulfillment logistics systems to reduce the costs of shipping customer orders and meeting on-time delivery promises. Optimization approaches for the design of such networks have subsequently received attention [36, 25, 18, 31, 29, 7, 34]. A core optimization modeling approach used in this application is the

*Corresponding author: lacy.greening@asu.edu

multicommodity capacitated network design (MCND) problem, which seeks to optimize the transportation of shipments through a minimum-cost transshipment network [2, 27, 3]. Shipments are grouped into *commodities*, defined as origin-destination pairs with specific demand volumes, and the design model prescribes one or more paths for the flow of each commodity. The selected paths and assigned shipment flows then require integer multiples of a capacity unit (e.g., truck trailers) to be installed on directed arcs, incurring a fixed cost for each installed unit and possibly a variable cost based on demand volume transported. Costs are shared among commodities that use the same arc, and this modeling approach identifies the best ways to consolidate shipments and improve the fill rate of dispatched trailers. Recent studies show that these approaches can reduce fulfillment network costs by over 30%. However, achieving these results for the practical networks operated by industry relies on solving large-scale mixed-integer programming models involving thousands of binary and integer variables, as well as tens of thousands of constraints [30].

In real-world fulfillment networks, there are additional dispatch and network design constraints specific to each organization beyond those typically found in the standard MCND model. A key distinction between practical applications and the standard MCND model is that in practice all shipments of the same commodity must follow the same path of arcs from origin to destination within the network. That is, MCND models for fulfillment networks have unsplittable (or non-bifurcated) flow constraints [2, 33, 29, 32, 13]. This requirement simplifies the logistics process and reduces errors, since outbound loading destinations remain consistent for shipments of the same commodity. However, it can make realistically-sized instances even more challenging to solve to optimality because it requires replacing continuous flow variables with binaries. Numerous works focus on developing effective metaheuristic approaches to identify strong primal solutions for these so-called single-path MCND problems [18, 4]; however, assessing the quality of solutions is typically difficult for realistically-sized instances due to weak dual bounds, a common issue in network design problems caused by their weak linear programming (LP) relaxations. In this paper, we aim to enhance the solvability of this specific integer programming (IP) variant of the MCND problem by strengthening the dual bounds.

Of the works studying the unsplittable flow MCND variant, several aim to improve the solvability of this hard class of problems by focusing on the polyhedra of arc-set relaxations to identify valid inequalities and related separation algorithms [2, 13, 12, 35, 6]. The authors of [12] introduce the c-strong inequalities and characterize their necessary and sufficient conditions to be facet-defining. The authors of [2] extend this work by showing that the separation problem of c-strong inequalities is $\mathcal{NP}$-hard and that it is sufficient to consider only the subspace of fractional variables. The authors also introduce the k-split c-strong and lifted knapsack cover inequalities, both of which include c-strong inequalities as a special case with k-split c-strong inequalities often resulting in greater improvements. More recently, the authors of [13] develop a generalized exact separation algorithm for the unsplittable flow arc-set polyhedron, allowing for an arbitrary number of facilities and capacities (i.e., installing multiple unit types with varying capacities, instead of integer

multiples of one type). Motivated by these studies, we will also study the arc-set polyhedra to identify two new classes of valid inequalities, in addition to identifying an implementation of metric inequalities [3, 8, 17] which improve weak dual bounds.

Thus, to improve the solvability of practical instances, this work seeks approaches that strengthen dual bounds for logistics and transportation network design problems. Specifically, we:

- reformulate the MCND model to utilize a multiple-choice binary variable scheme to select arc capacities, as opposed to integer capacity variables, and verify this reformulation enables solvers to find stronger lower bounds;
- strengthen the LP relaxation of this reformulation by introducing two new classes of valid inequalities for the unsplittable MCND problem (applicable to both arc- and path-based formulations);
- identify an implementation of metric inequalities, valid for both the IP and LP MCND models, that improves the bounding procedures of commercial solvers by helping them select better cutting planes implied by the integer hull of these metric inequalities; and
- conduct an extensive computational study on two sets of instances – one for path-based formulations and the other for arc-based formulations – to demonstrate the value of our proposed reformulation, valid inequalities, and solver-aiding constraints.

The remainder of this paper is organized as follows. In Section 2, we define and model the MCND problem using two path-based formulations, one which utilizes integer variables to model arc capacities (INT) and another which utilizes a multiple-choice selection of binary variables to model arc capacities (BIN). In Section 3, we study a structured relaxation of the BIN formulation and introduce two new classes of valid inequalities. In Section 4, we describe a set of metric inequalities which aid solvers in generating and selecting better cutting planes. Finally, in Section 5, we report the results of computational experiments performed on instances constructed from e-commerce fulfillment network and demand data, as well as on a well-known set of MCND instances for arc-based formulations.

2. Formulations

Let $G = (\mathcal{N}, \mathcal{A})$ define a transshipment network, where $\mathcal{N}$ represents a set of locations (nodes) in the network and $\mathcal{A}$ represents a set of directed arcs connecting pairs of locations. Shipment demand is modeled using a set $\mathcal{K}$ of commodities, where each commodity $k \in \mathcal{K}$ has an origin $o(k) \in \mathcal{N}$, destination $d(k) \in \mathcal{N}$, and demand volume d_k that must be shipped from origin to destination along a single path (or sequence of adjoined arcs) through the network. As the number of paths between pairs of locations can be exponential, we define a set of paths $\mathcal{P}_k$ that each commodity $k \in \mathcal{K}$ can use. Let f_a be the fixed cost of installing one unit of capacity on arc $a \in \mathcal{A}$, c_p be the variable cost per unit of demand when using path $p \in \cup_{k \in \mathcal{K}} \mathcal{P}_k$, and q_a denote the unit capacity of arc $a \in \mathcal{A}$. The MCND problem is to determine a minimum-cost allocation of transportation capacity on network arcs to ensure that commodity demand requirements are satisfied.

2.1. Integer Capacity Variables

Let binary variables x_p indicate if commodity $k \in \mathcal{K}$ uses path $p \in \mathcal{P}_k$ or not. Let integer variables τ_a represent the number of capacity units installed on arc $a \in \mathcal{A}$. The path-based unsplittable MCND problem with integer capacity variables (referred to as INT) is formulated as follows:

$$\min_{x,\tau} \quad \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} c_p d_k x_p + \sum_{a\in\mathcal{A}} f_a \tau_a \tag{1a}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}_k} x_p = 1, \qquad \forall\, k \in \mathcal{K}, \tag{1b}$$

$$\sum_{k\in\mathcal{K}} \sum_{\{p\in\mathcal{P}_k | p\ni a\}} d_k x_p \le q_a \tau_a, \quad \forall\, a \in \mathcal{A}, \tag{1c}$$

$$x_p \in \{0,1\}, \qquad \forall\, p \in \mathcal{P}_k,\ \forall\, k \in \mathcal{K}, \tag{1d}$$

$$\tau_a \in \mathbb{Z}_{\ge 0}, \qquad \forall\, a \in \mathcal{A}. \tag{1e}$$

The objective is to minimize the sum of fixed and variable costs. Constraints (1b) ensure that one path is selected for each commodity. Constraints (1c) ensure that the required number of capacity units are installed to transport the flow volume assigned to each arc as defined by selected paths.

2.1.1. Binary Capacity Variables

Previous studies [27, 21, 22, 10] have shown that by reformulating flow and/or capacity variables to use the so-called *multiple-choice* selection, where an expanded set of binary selector variables replace either an integer capacity and/or binary flow variable, one can define potentially stronger linking constraints and other valid inequalities to produce higher-quality LP relaxations. We also elect to use a multiple-choice reformulation of INT (1), and later use this reformulation to define a new class of valid inequalities in Section 3.

In our multiple-choice reformulation, we replace integer capacity variables τ_a for arcs $a \in \mathcal{A}$ with a set of binary variables y_{at} that select an integer capacity t from a set $\mathcal{T}_a$ containing allowable non-zero capacity values. We define $\mathcal{T}_a = \{1, \ldots, T_a^{\max}\}$ for each arc $a \in \mathcal{A}$, where $T_a^{\max}$ is the maximum number of capacity units that may be necessary on arc $a \in \mathcal{A}$ and is defined as $T_a^{\max} = \left\lceil \frac{\sum_{k\in\mathcal{K}} \delta_a^k d_k}{q_a} \right\rceil$ with $\delta_a^k = 1$ if the path set $\mathcal{P}_k$ for commodity $k \in \mathcal{K}$ contains a path that includes arc $a \in \mathcal{A}$ (i.e., $|\{p \in \mathcal{P}_k | p \ni a\}| \ge 1$) and $\delta_a^k = 0$ otherwise. Thus, we reformulate the integer capacity variables by making the following replacement:

$$\tau_a = \sum_{t\in\mathcal{T}_a} t y_{at}, \quad \forall\, a \in \mathcal{A}. \tag{2}$$

The path-based unsplittable MCND problem with multiple-choice binary capacity variables (re-

ferred to as BIN) is formulated as follows:

$$\min_{x,y} \quad \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} c_p d_k x_p + \sum_{a\in\mathcal{A}} f_a \sum_{t\in\mathcal{T}_a} t y_{at} \tag{3a}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}_k} x_p = 1, \qquad \forall\, k \in \mathcal{K}, \tag{3b}$$

$$\sum_{k\in\mathcal{K}} \sum_{\{p\in\mathcal{P}_k | p\ni a\}} d_k x_p \le q_a \sum_{t\in\mathcal{T}_a} t y_{at}, \qquad \forall\, a \in \mathcal{A}, \tag{3c}$$

$$\sum_{t\in\mathcal{T}_a} y_{at} \le 1, \qquad \forall\, a \in \mathcal{A}, \tag{3d}$$

$$x_p \in \{0,1\}, \qquad \forall\, p \in \mathcal{P}, \tag{3e}$$

$$y_{at} \in \{0,1\}, \qquad \forall\, t \in \mathcal{T}_a,\ \forall\, a \in \mathcal{A}. \tag{3f}$$

The objective (3a) and constraints (3b) and (3c) function the same as the objective (1a) and constraints (1b) and (1c), respectively. Constraints (3d) ensure the model selects at most one integer capacity to install on arc $a \in \mathcal{A}$.

While reformulation (3) is larger (with $\sum_{a\in\mathcal{A}}(|\mathcal{T}_a|-1)$ more variables and $|\mathcal{A}|$ more constraints) than (1), we will show in Section 5.2 that a commercial optimization solver is able to produce stronger lower bounds when solving (3). However, it will also be shown that even small instances of these problems cannot be solved within the provided runtime and larger instances continue to have weak lower bounds despite the reformulation. Thus, in the next section, we will study a structured relaxation of the MCND problem to identify a new class of valid inequalities which can produce strong LP relaxations, and ultimately help solvers identify stronger lower bounds.

3. Valid Inequalities

It is well known that MCND formulations may produce weak LP relaxations (referred to as LP(INT) and LP(BIN) for models (1) and (3), respectively) for many practical instances due to the variable upper bound capacity constraints (1c) and (3c). We provide an illustration of a root cause of this phenomenon in Figure 1 by plotting the cost $f_a\tau_a$ of an arc a in INT and LP(INT) given the volume assigned to flow on that arc, $\sum_{k\in\mathcal{K}} \sum_{\{p\in\mathcal{P}_k | p\ni a\}} d_k x_p$. We see that the cost function for INT is a step function, whereas LP(INT) has a linear cost function with a slope of $\frac{f_a}{q_a}$ (due to continuous τ_a variables causing constraints (1c) of LP(INT) to hold at equality). Thus, LP(INT) minimizes cost by sending commodities along arcs (similarly, paths composed of a sequence of arcs) with the cheapest marginal cost per unit of demand sent. When the volume sent along an arc constitutes a small proportion of the integer capacity, LP(INT) is a weak approximation of INT. Note the same holds for BIN and LP(BIN).

A common way to strengthen the MCND LP relaxation is to add the following disaggregated

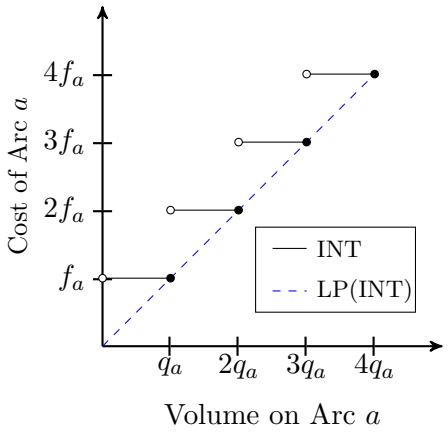

Figure 1: Comparing the cost of volume flowing arc a for INT and LP(INT).

capacity linking constraints to LP(INT) and LP(BIN), respectively:

$$t_{ak}^{\min} \sum_{\{p \in \mathcal{P}_k | p \ni a\}} x_p \leq \tau_a, \quad \forall\, a \in \mathcal{A},\ \forall\, k \in \mathcal{K}, \tag{4}$$

$$\sum_{\{p \in \mathcal{P}_k | p \ni a\}} x_p \leq \sum_{t=t_{ak}^{\min}}^{T_a^{\max}} y_{at}, \quad \forall\, a \in \mathcal{A},\ \forall\, k \in \mathcal{K}, \tag{5}$$

where $t_{ak}^{\min}$ is the minimum number of capacity units required to transport commodity $k \in \mathcal{K}$ on arc $a \in \mathcal{A}$ and is defined as $t_{ak}^{\min} = \left\lceil \frac{d_k}{q_a} \right\rceil$. If $t_{ak}^{\min} = 1$ for all $a \in \mathcal{A}$ and $k \in \mathcal{K}$ (i.e., no individual commodity demand exceeds an arc's single-unit capacity), the solutions of LP(INT) and LP(BIN) with the addition of disaggregated capacity linking constraints (4) and (5), respectively, are equivalent. However, if this is not true and $t_{a'k'}^{\min} > 1$ for some $a' \in \mathcal{A}$ and $k' \in \mathcal{K}$, then LP(BIN) with constraints (5) can produce a marginally stronger relaxation solution than LP(INT) with constraints (4). See Appendix A for an example.

Potentially stronger valid inequalities can be generated from simple structured relaxations over more complicated sets, like those focused on the unsplittable flow arc-set polyhedron (i.e., the convex hull of solutions to capacity constraints of the form (1c) or (3c)) [2, 3, 13, 8, 12, 35]. Thus, in the following section, we investigate a simple structured relaxation of BIN (3) in which we also focus on the capacity-related constraints, including the single-capacity selection constraints (3d). We will show through examples, and later demonstrate through a computational study, that the two new classes of valid inequalities developed can produce stronger LP relaxations as compared to when adding constraints (5) to LP(BIN).

3.1. Valid Inequalities from Relaxation

For a given arc $a \in \mathcal{A}$, we consider the following set:

$$S := \left\{ (x, y) \in \{0,1\}^{|\mathcal{P}_a|} \times \{0,1\}^{|\mathcal{T}_a|} \,|\, \sum_{p \in \mathcal{P}_a} d_p x_p \leq q_a \sum_{t \in \mathcal{T}_a} t y_t; \sum_{t \in \mathcal{T}_a} y_t \leq 1 \right\}, \tag{6}$$

where $\mathcal{P}_a = \{p \in \cup_{k\in\mathcal{K}}\mathcal{P}_k | p \ni a\}$, $d_p \in \mathbb{Z}_+$ for $p \in \mathcal{P}_a$ and $q_a \in \mathbb{Z}_+$. Note that a path $p \in \mathcal{P}_k$ is unique for commodity k; thus, to simplify notation, we use d_p in place of d_k for $p \in \mathcal{P}_k$.

We first analyze the convex hull solutions of (6), from which we deduce the general form of the constraints defining the set.

Proposition 1. *There exists non-negative matrices G, H such that*

$$\text{conv}(S) = \left\{ (x,y) \in [0,\ 1]^{|\mathcal{P}_a|} \times [0,\ 1]^{|\mathcal{T}_a|} \;\middle|\; Gx \le Hy; \sum_{t\in\mathcal{T}_a} y_t \le 1 \right\}.$$

Proof. Observe that if $y_t = 0$ for all $t \in \mathcal{T}_a$, then $x = 0$. Otherwise for $t \in \mathcal{T}_a$, let:

$$S_t := \left\{ x \in \{0,1\}^{|\mathcal{P}_a|} \;\middle|\; \sum_{p\in\mathcal{P}_a} d_p x_p \le q_a t \right\},$$

and let

$$\text{conv}(S_t) = \left\{ u \ge 0 \;\middle|\; s_a^t u \le r^t \right\}.$$

Then it is clear that,

$$S = (\mathbf{0}, \mathbf{0}) \cup \bigcup_{t\in\mathcal{T}_a} (S_t, e_t).$$

Therefore, we obtain that

$$\begin{aligned} \text{conv}(S) &= \text{proj}_{x,y}(T), \text{ where} \\ T &= \left\{ (x, y, x^1, x^2, \ldots, x^{T_a^{\max}}) \;\middle|\; x^t \ge 0, s_a^t x^t - r^t y_t \le 0, t \in \mathcal{T}_a; x - \sum_{t\in\mathcal{T}_a} x^t = 0; \sum_{t\in\mathcal{T}_a} y_t \le 1 \right\}. \end{aligned}$$

We may apply Fourier-Motzkin elimination to T to project out the variables $x^1, \ldots, x^{T_a^{\max}}$. Since these variables only appear in the constraints $x^t \ge 0, s_a^t x^t - r^t y_t \le 0, t \in \mathcal{T}_a; x - \sum_{t\in\mathcal{T}_a} x^t = 0$, where the right-hand-side is 0 for all the constraints, we arrive at the fact that (after Fourier-Motzkin elimination)

$$\text{conv}(S) = \left\{ (x,y) \in [0,\ 1]^{|\mathcal{P}_a|} \times [0,\ 1]^{|\mathcal{T}_a|} \;\middle|\; \widetilde{G}x \le \widetilde{H}y; \sum_{t\in\mathcal{T}_a} y_t \le 1 \right\}.$$

It remains to prove that $\widetilde{G}$ and $\widetilde{H}$ are non-negative.

We will first consider $\widetilde{G}$. Suppose that $\widetilde{G}_{v,w} < 0$ for some (v, w). We claim that if we replace $\widetilde{G}_{v,w}$ with 0, the resulting inequality:

$$\sum_{p\in\mathcal{P}_a\setminus\{w\}} \widetilde{G}_{v,p} x_p \le \widetilde{H}_v y \tag{7}$$

is still valid for S. This will prove the result, since replacing $\widetilde{G}_{v,w}$ by 0 results in a stronger inequality that dominates the original inequality (since $x_w \ge 0$). Suppose $(\hat{x}, \hat{y}) \in S$ and this point does not

satisfy (7). First note that $\hat{x}_w = 1$, since if $\hat{x}_w = 0$, then $\sum_{p\in\mathcal{P}_a\setminus\{w\}} \widetilde{G}_{v,p}\hat{x}_p = \sum_{p\in\mathcal{P}_a} \widetilde{G}_{v,p}\hat{x}_p$. However, now observe that $(\tilde{x}, \hat{y}) \in S$, where:

$$\tilde{x}_j = \begin{cases} \hat{x}_p & p \neq w \\ 0 & p = w \end{cases}.$$

Therefore, $\widetilde{H}_v\hat{y} \geq \sum_{p\in\mathcal{P}_a} \widetilde{G}_{v,p}\tilde{x}_p = \sum_{p\in\mathcal{P}_a\setminus\{w\}} \widetilde{G}_{v,p}\tilde{x}_p = \sum_{p\in\mathcal{P}_a\setminus\{w\}} \widetilde{G}_{v,p}\hat{x}_p$. Thus, $(\hat{x}, \hat{y})$ satisfies (7), a contradiction.

We now verify that $\tilde{H}$ is non-negative. Note that $(\mathbf{0}, e_p) \in S$ for all $p \in \mathcal{P}_a$. Therefore, if $H_{v,k} < 0$, we have that the point $(\mathbf{0}, e_k)$ will not satisfy the inequality $\sum_{p\in\mathcal{P}_a} \tilde{G}_{v,p}x_p \leq \tilde{H}_v y$. This concludes the proof. □

It is important to note that the general form of the constraints lacks a constant term in $Gx \leq Hy$, unlike the inequality introduced in [13]. The reasons for this are twofold: (i) arcs do not have pre-existing capacities, which would introduce a constant term to the right-hand side of (3c), and (ii) our reformulation imposes that each arc has a uniquely defined capacity value such that only one $y_t > 0$ for all $t \in \mathcal{T}_a$. If either condition were not true, a constant term would be required. In the following sections, we describe new valid inequalities for S that are motivated by Proposition 1.

3.1.1. Single-Arc Commodity Packing Constraints

Since G is a non-negative matrix, we first explore a subset of constraints which involve only $\{0, 1\}$ coefficients for the path variables x. In this case, since both x and y are binary variables, each element in H is bounded above by $|\mathcal{P}_a|$, as the left-hand side cannot exceed this value and at most one y variable on the right-hand side can be nonzero.

Observe that for a given a left-hand-side binary vector g corresponding to the x vector, we can explicitly determine h_t, the coefficient for y_t, by counting the maximum number of *commodities* with at least one path selected in g (i.e., x variable with a coefficient of 1 in g) that can be transported using (or *packed* into) t capacity units. Note we emphasize the number of commodities and not the number of paths, as commodities can have more than one path in $\mathcal{P}_a$ but should not be counted more than once. We state this result formally next.

Proposition 2. *Consider a non-zero binary vector g. The inequality*

$$\sum_{p\in\mathcal{P}_a} g_p x_p \leq \sum_{t\in\mathcal{T}_a} \alpha_t y_t, \tag{8}$$

is a valid inequality for S if

$$\alpha_t = \max\left\{ \sum_{p\in\mathcal{Z}} x_p \;\middle|\; \sum_{p\in\mathcal{Z}} d_p x_p \leq t q_a, x_p \in \{0,1\} \right\}, \tag{9}$$

where $\mathcal{Z} = \{p \in \mathcal{P}_a \,|\, g_p = 1\}$.

Proof. Clearly (8) is satisfied by $(0,0)$. Now consider any non-zero solution $(\hat{x},\hat{y}) \in S$ with $\hat{y}_t = 1$ for $t = \hat{t}$ and $\hat{y}_t = 0$ for $t \neq \hat{t}$. Then, by the definition of S, we have that $\sum_{p\in\mathcal{P}_a} \hat{x}_p d_p \leq \hat{t} q_a$. Therefore, we obtain that $\sum_{p\in\mathcal{Z}} \hat{x}_p d_p \leq \sum_{p\in\mathcal{P}_a} \hat{x}_p d_p \leq \hat{t} q_a$. Now by the definition of α_t in (9), we have that $\alpha_{\hat{t}} \geq \sum_{p\in\mathcal{Z}} \hat{x}_p$. In other words, we obtain that

$$\sum_{p\in\mathcal{P}_a} g_p \hat{x}_p = \sum_{p\in\mathcal{Z}} \hat{x}_p \leq \alpha_{\hat{t}} = \sum_{t\in\mathcal{T}_a} \alpha_t \hat{y}_t,$$

that is (8) is satisfied by $(\hat{x},\hat{y})$. □

Using Proposition 2, we can define the following *single-arc commodity packing* (SAC-Pack) constraints:

$$\sum_{p\in\mathcal{Z}} x_p \leq \sum_{t\in\mathcal{T}_a} \alpha_t y_t, \quad \forall\, a \in \mathcal{A}, \tag{10}$$

where α_t is an integer coefficient equal to the maximum number of selected commodities (those with at least one path $p \in \mathcal{Z}$) that can be transported by $t \in \mathcal{T}_a$ capacity units. In a sense, constraints (10) are a "smart re-aggregation" of linking constraints (5) for a set of commodities with at least one path $p \in \mathcal{Z}$, and can at times even dominate them.

Example 1. *Consider the following example where we demonstrate how SAC-Pack constraints* (10) *can dominate disaggregated linking constraints* (5).

Assume $\mathcal{P}_a = \mathcal{Z} = \{1,2\}$. *Let the solution of LP(BIN) be* $x_1 = 0.5$ *and* $x_2 = 0.4$ *and let* $d_1 = 60$ *and* $d_2 = 70$ *with* $q_a = 100$. *We will also assume that* $T_a^{\max} = 2$ *for clarity of exposition. Constraints* (3c) *set the capacity of arc* a *as* $y_{a1} = 0.58$ *units (similarly,* $y_{a1} + 2y_{a2} = 0.58$*). To strengthen LP(BIN), we add the following constraints* (10)*:*

$$x_1 + x_2 \leq y_{a1} + 2y_{a2}.$$

This increases the installed capacity to 0.9*, whereas if we only add the following constraints* (5)*:*

$$x_1 \leq y_{a1} + y_{a2},$$
$$x_2 \leq y_{a1} + y_{a2},$$

$y_{a1} = 0.58$ *satisfies both constraints, and the solution of LP(BIN) does not change.*

We note that the process of determining the coefficients given by (9) can be done efficiently by sorting the selected commodities in order of non-decreasing demand values, adding demands until the capacity is exceeded, and setting the coefficient to the number of demands added minus one (as the last demand cannot be transported). We next formally define a separation routine to identify violated constraints (10).

Separation. Given the set S described in (6) and a point (x^*, y^*), there exists a SAC-Pack constraint (10) violated by (x^*, y^*) if and only if there exists a set $\mathcal{Z} \subseteq \mathcal{P}_a$ such that $\sum_{p\in\mathcal{Z}} x^*_p - \sum_{t\in\mathcal{T}_a} \alpha_t y^*_{at} > 0$. Because we can determine the α values in closed form (see Proposition 2), we can explicitly check for such a violation by formulating the problem as an IP with the objective $\max_{\mathcal{Z}\subseteq\mathcal{P}_a}\{\sum_{p\in\mathcal{P}_a} z_p x^*_p - \sum_{t\in\mathcal{T}_a} \alpha_t y^*_{at}\}$, where z_p is a binary variable indicating whether path p is selected to be in $\mathcal{Z}$, and α_t is an integer variable representing the number of selected commodities that can be transported by $t \in \mathcal{T}_a$ capacity units. If the objective value is greater than 0, we add the violated constraint to the model.

Table 1: SAC-Pack separation problem variable definitions.

Variable	**Description**
$z_p \in \{0,1\}$	Indicate whether path $p \in \mathcal{P}_a$ is selected for set $\mathcal{Z} \subseteq \mathcal{P}_a$.
$\alpha_t \in \mathbb{Z}_{\geq 0}$	Coefficient of y_t representing the number of selected commodities that can be transported by $t \in \mathcal{T}_a$ capacity units.
$s_{kt} \in \{0,1\}$	Indicate commodity $k \in \mathcal{K}_a$ is in the maximum subset that can be transported by $t \in \mathcal{T}_a$ capacity units.
$u_t \in \{0,1\}$	Indicate if all selected commodities can be transported by $t \in \mathcal{T}_a$ capacity units (deactivates constraints).
$v_t \in \{0,1\}$	Indicate if not all selected commodities can be transported by $t \in \mathcal{T}_a$ capacity units (activates constraints).
$w_{kt} \in \{0,1\}$	Indicate if a selected commodity $k \in \mathcal{K}_a$ has at least one path selected but not in the subset of commodities that can be transported by $t \in \mathcal{T}_a$ capacity units.

Let the set $\mathcal{K}_a \subseteq \mathcal{K}$ be defined as the set of commodities with at least one path containing arc a (or $\{k \in \mathcal{K} || \{p \in \mathcal{P}_k | p \ni a\}| \geq 1\}$) in order of non-decreasing demands. We define the decision variables for the separation problem in Table 1. We can now formulate the separation problem as:

$$\max \quad \sum_{p\in\mathcal{P}_a} z_p x^*_p - \sum_{t\in\mathcal{T}_a} \alpha_t y^*_{at} \tag{11a}$$

$$\text{s.t.} \quad \alpha_t \geq \sum_{k\in\mathcal{K}_a} s_{kt} - 1 + u_t, \qquad \forall\, t \in \mathcal{T}_a, \tag{11b}$$

$$s_{kt} \leq \sum_{\{p\in\mathcal{P}_k | p\ni a\}} z_p, \qquad \forall\, t \in \mathcal{T}_a, \forall\, k \in \mathcal{K}_a, \tag{11c}$$

$$t(1-u_t) + \epsilon \leq \sum_{k\in\mathcal{K}_a} \frac{d_k}{q_a} s_{kt}, \qquad \forall\, t \in \mathcal{T}_a, \tag{11d}$$

$$z_p - s_{kt} \leq w_{kt}, \qquad \forall\, t \in \mathcal{T}_a, \forall\, k \in \mathcal{K}_a, \forall\, p \in \{\mathcal{P}_a \cap \mathcal{P}_k\}, \tag{11e}$$

$$\sum_{k\in\mathcal{K}_a} w_{kt} \leq |\mathcal{K}_a| v_t, \qquad \forall\, t \in \mathcal{T}_a, \tag{11f}$$

$$u_t \leq 1 - v_t, \qquad \forall\, t \in \mathcal{T}_a, \tag{11g}$$

$$|\mathcal{K}_a \setminus \{k \leq k_1\}|\,(s_{k_1 t} + 1 - w_{k_1 t}) \geq \sum_{\{k_2\in\mathcal{K}_a | k_2 > k_1\}} s_{k_2 t}, \qquad \forall\, t \in \mathcal{T}_a, \forall\, k_1 \in \mathcal{K}_a, \tag{11h}$$

$$z_p \in \{0,1\}, \qquad \forall\, p \in \mathcal{P}_a, \tag{11i}$$

$$u_t \in \{0,1\},\ v_t \in \{0,1\},\ \alpha_t \in \mathbb{Z}_{\geq 0}, \qquad \forall\, t \in \mathcal{T}_a, \tag{11j}$$

$$s_{kt} \in \{0,1\},\ w_{kt} \in \{0,1\}, \qquad \forall\, k_1 \in \mathcal{K}_a, \forall\, t \in \mathcal{T}_a. \tag{11k}$$

Constraints (11b) capture the value of each α_t coefficient by summing the commodities selected for the subset. The 1 is subtracted because the s_{kt} variables are collected until the total demand exceeds t capacity units; thus, the last commodity added should be removed to make it feasible. However, if the total volume of the commodities is less than t capacity units, all commodities should be counted; thus, we add the binary variable u_t which is equal to 1 when this is the case. In the case where all commodities have $s_{kt} = 1$ but the total volume exceeds t, u_t can and still will equal 0 because the objective minimizes the α_t coefficients. Constraints (11c) enforce that commodities cannot be selected for the maximum subset that fits in t capacity units if none of its paths were selected. Constraints (11d) determine the number of commodities plus 1 that can be transported by t capacity units. The constraints are turned off if the total commodity volume is less than t capacity units. The ϵ parameter ensures that the right-hand side does not equal t, as we later subtract 1, assuming that the volume is exceeded. Thus, we note that for any $\epsilon > 0$, the generated inequality is valid, with larger values of ϵ possibly making the resulting inequality weaker. In our computational study, we set $\epsilon = 0.001$, which is sufficiently small to avoid weakening the constraints because $\epsilon < \min_{k \in \mathcal{K}_a}\{d_k/q_a\}$. Constraints (11e) determine if at least one path for commodity k was selected but is not included in the subset of commodities that can be transported by t capacity units. Note that here we use $\{\mathcal{P}_a \cap \mathcal{P}_k\}$ to represent the set of paths for commodity $k \in \mathcal{K}_a$ that contain arc a. Constraints (11f) set $v_t = 1$ if any $w_{kt} = 1$ as set by constraints (11e). Constraints (11g) ensure that $u_t = 0$ if the size of the subset of commodities that can be transported by t capacity units, plus 1, is less than the total number of commodities whose paths were selected. Finally, constraints (11h) are precedence constraints for each capacity t that ensure commodities are selected in order of non-decreasing demands, if selected at all.

To reduce the size of the separation problem, one can consider only the paths and capacity variables with non-zero values in (x^*, y^*). The coefficients for variables $y_t^* = 0$ can be calculated once the paths are selected for the constraint as previously described. Another way to reduce the problem size is to aggregate paths for the same commodity. That is, let z_k be a binary variable that indicates if all non-zero paths in $\mathcal{P}_a$ for commodity $k \in \mathcal{K}_a$ are selected for the set $\mathcal{Z}$. The objective changes to $\sum_{k \in \mathcal{K}_a} z_k \sum_{p \in \{\mathcal{P}_a \cap \mathcal{P}_k\}} x_p^* - \sum_{t \in \mathcal{T}_a} \alpha_t y_{at}^*$, the number of variables is reduced by $|\mathcal{P}_a| - |\mathcal{K}_a|$, and the number of constraints is reduced by $|\{\mathcal{P}_a \cap \mathcal{P}_k\}| - 1$ for each $k \in \mathcal{K}_a$.

3.1.2. Generalized Single-Arc Commodity Packing Constraints

We now investigate the case in which the elements of G (or coefficients of the x variables) are in the list $\{0, 1, \ldots, B\}$. To motivate the use of such more general inequalities, consider the following example.

Example 2. *In this example, we demonstrate how constraints (10) can be strengthened to cut off solutions in some directions if we allow x variables to have coefficients in the list $\{0, 1, \ldots, B\}$.*

Let $q_a = 100$ and $\mathcal{Z} = \{1, 2, 3, 4\}$ with $d_1 = 30$, $d_2 = 30$, $d_3 = 30$, and $d_4 = 60$, where each

$p \in \mathcal{Z}$ is for a unique commodity. Assuming $T_a^{\max} = 2$, constraints (10) would be:

$$x_1 + x_2 + x_3 + x_4 \leq 3y_{a1} + 4y_{a2},$$

as the three smaller commodities can be transported by one unit of capacity, but all four would require two units. If x variables had coefficients in the list $\{0, 1, \ldots, B\}$, we could strengthen the previous inequality for the single-unit capacity case (i.e., when $y_{a1} > 0$) by increasing the coefficient for commodity 4, as follows:

$$x_1 + x_2 + x_3 + 2x_4 \leq 3y_{a1} + 5y_{a2},$$

as it can only be transported with one other commodity if only one unit of capacity is installed. Notice that we also increased the coefficient for y_{a2} by 1; thus, when two units of capacity are used, the constraints are equivalent in that they allow for all four demands to be transported with two units of capacity.

To demonstrate how some solutions may be cut off, let the solution of LP(BIN) be $x_1 = x_2 = x_3 = x_4 = 0.5$, setting $y_{a1} = 0.75$. This solution remains feasible with constraints (10), as the left-hand side is 2 and the right-hand side is 2.25. However, if you increase the coefficient of x_4 to 2, this solution is no longer feasible, as the left-hand side is now 2.5. Thus, y_{a1} is increased to 0.83.

We define the *generalized single-arc commodity packing* (Gen-SAC-Pack) constraints as follows:

$$\sum_{p \in \mathcal{Z}} \theta_p x_p \leq \sum_{t \in \mathcal{T}_a} \alpha_t y_t, \quad \forall\, a \in \mathcal{A}, \tag{12}$$

where θ_p are non-negative integer coefficients for paths $p \in \mathcal{Z}$ bounded above by B.

Post-processing SAC-Pack constraints. Motivated by the previous example, a simple method to obtain constraints of the form (12) is to post-process the SAC-Pack constraints found using (11). The idea is to identify paths (or commodities) which cannot be transported with $\alpha_t - 1$ other commodities for $t = \min\{t \in \mathcal{T}_a \,|\, y_t > 0\}$. Recall the example above, the largest commodity demand (represented by x_4) could not be transported with two other commodities using only one unit of capacity; thus, its coefficient was increased by one. This idea can also be extended to more than one path with an increased coefficient. We will demonstrate in Section 5 that this simple approach is able to further strengthen LP(BIN) with constraints (10) with a marginal increase in runtime.

Separation. To obtain potentially stronger cuts, we use the following separation approach to define new Gen-Sack-Pack constraints of the form (12).

We are given the set S described in (6) and a point (x^*, y^*) that we want to separate from the convex hull of S. We obtain separating inequalities using the following row-generation approach [11].

1. Let $\{x^j, y^j\}_{j=1}^r$ be a subset of points in S.
2. Solve:

$$\begin{aligned} \text{SEPVAL} := \quad \max_{\theta,\alpha} \quad & \theta^\top x^* - \alpha^\top y^* \\ \text{s.t.} \quad & \theta^\top x^j - \alpha^\top y^j \leq 0, \ \forall\, j \in [r], \\ & \alpha_t \leq \alpha_{t+1}, \ \forall\, t \in \mathcal{T}_a \setminus \{T_a^{\max}\}, \\ & \theta_p \leq B, \ \forall\, p \in \mathcal{P}_a, \\ & \theta \in \mathbb{Z}_{\geq 0}, \ \alpha \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{13}$$

   If SEPVAL ≤ 0, then STOP as there is no separating inequality. Else, let (θ^*, α^*) be an optimal solution of (13).
3. Solve:

$$\begin{aligned} \text{FEASVAL} := \quad \max_{x,y} \quad & \theta^{*\top} x - \alpha^{*\top} y \\ \text{s.t.} \quad & (x, y) \in S. \end{aligned} \tag{14}$$

   If FEASVAL $= 0$[1], then STOP as $\theta^{*\top} x \leq \alpha^{*\top} y$ is a separating inequality. Else, let $(\hat{x}, \hat{y})$ be an optimal solution of (14). Let $r \leftarrow r + 1$, let $(x^r, y^r) \leftarrow (\hat{x}, \hat{y})$ and return to STEP 2 above.

Note that parameter B in (13) above is an artificial upper bound. We will demonstrate in Section 5 that using $B \leq 3$ gives high-quality cuts in comparison to larger upper bounds. See Appendix B for the dynamic programming approach we use in our computational experiments to decrease the runtime of solving (14). Additionally, as in the separation problem defined in Section 3.1.1, paths for the same commodity can be aggregated into a single variable to help further reduce required compute time. While the previously-defined row-generation approach with $B = 1$ can also be used to identify SAC-Pack constraints (10), it requires more compute time as compared to solving model (11). Thus, we only employ this separation method when $B > 1$.

We also note that while we focus on the *path-based*, unsplittable MCND problem in this work, the previous valid inequalities (10) and (12) and their separation routines can also be applied to *arc-based*, unsplittable MCND problems. In this case, the set $\mathcal{Z}$ would consist of the commodity-specific binary arc-flow variables. We demonstrate this later in Section 5 by generating valid inequalities (10) and (12) for arc-based, unsplittable MCND models using the Canad instances [26].

4. Helper Metric Inequalities

In this section, we outline an approach to generate a set of helper metric inequalities. We use the term 'helper' because even though the metric inequalities we generate are redundant when added back to the IP formulation, they can help commercial solvers generate and select better cutting planes, producing better bounds, as we will demonstrate in Section 5 with the Gurobi solver[2].

[1] Note FEASVAL ≥ 0 since $(0, 0) \in S$

[2] To show that the benefits are not specific to one commercial solver, we also provide limited results for two other commercial solvers in Appendix C

Metric inequalities are valid inequalities for the multicommodity network flow polytope [3, 8, 17]. While metric inequalities can be used to generate user cutting planes, we observed through computational testing that also adding valid metric inequalities (identified from an LP-feasible capacity vector) directly to the BIN IP led to greater bound improvements when using commercial solvers. This is because commercial solvers are able to generate strong cutting planes implied by the integer hull of these metric inequalities. Metric inequalities are thus "good candidate" aggregations of the original constraints of the problem [9].

To generate metric inequalities, we use a feasible capacity vector $\bar{q}$ defined by the optimal solution $(\bar{x}, \bar{y})$ to LP(BIN) (or $\bar{q}_a = q_a \sum_{t \in \mathcal{T}_a} t\bar{y}_{at}, \ \forall\, a \in \mathcal{A}$). We then use the standard linear programming dual representation of the multicommodity flow polytope, later described in detail in (15), to generate individual metric inequalities. Bounding any extreme ray of the dual creates an associated inequality. Note that this dual is formulated using the arc-based relaxation of the original multicommodity flow problem (i.e., use commodity-specific arc-flow variables as opposed to path-flow variables). Since the arc-based formulation is a relaxation, the identified metric inequalities remain valid for the path-based formulation. Importantly, the capacity vector $\bar{q}$ used to generate the metric inequalities is feasible for the more restricted path-based formulation by definition. Moreover, the resulting metric inequalities are directly applicable to the path-based formulation, as the arc-flow variables are projected out given $\bar{q}$ and the inequalities are defined using only arc-capacity variables and commodity demands transiting those arcs.

We further reduce the problem size by redefining the origin-destination-based commodity set to be either origin based, where all commodities with the same origin node are redefined to be a single commodity, or destination based, where all commodities with the same destination are redefined to be a single commodity. Note these are feasible aggregations because the demand flow in the LP is splittable.

We now outline the steps to generate a single helper metric inequality and integral metric inequality (i.e., user cut) when commodities are origin-based.

Origin-based commodity approach. Let all commodities originating from the same origin $o(k) = o$ be consolidated into one commodity $o \in \mathcal{O}$, where the set $\mathcal{O}$ represents the redefined commodity set. We define the path set for commodity $o \in \mathcal{O}$ as $\mathcal{P}_o = \{\cup_{\{k \in \mathcal{K} | o(k) = o\}} \mathcal{P}_k\}$. Commodity demand d_k for $k \in \mathcal{K}$ is relabeled as ψ^o_d when originating at $o(k) = o$ and destined for $d(k) = d$, and included in the total demand emanating from $o(k) = o$, or $\psi^o_o = \sum_{\{k \in \mathcal{K} | o(k) = o\}} -d_k$. Note $\psi^o_i = 0$ when $i \neq o(k), d(k)$, for all $\{k \in \mathcal{K} | o(k) = o\}$.

Given the optimal solution $(\bar{x}, \bar{y})$ to the (path-based) LP(BIN), set capacities to the values $\bar{q}_a = q_a \sum_{t \in \mathcal{T}_a} t\bar{y}_{at}$ for each $a \in \mathcal{A}$. To formulate the dual of the LP relaxation of the *arc-based* MCND problem, let dual variables u and v be associated to the arc flow conservation constraints[3]. and variable upper bound capacity constraints (i.e., arc-based versions of (3c)), respectively. Also

[3] See Appendix D

note that because one flow conservation constraint for each commodity $o \in \mathcal{O}$ is redundant, we can assume $u_o^o = 0$ for each $o \in \mathcal{O}$. We now formulate the dual as follows:

$$\max \quad \sum_{i \in \mathcal{N}} \sum_{o \in \mathcal{O}_i} \psi_i^o u_i^o - \sum_{a \in \mathcal{A}} \bar{q}_a v_a \tag{15a}$$

$$\text{s.t.} \quad u_j^o - u_i^o \leq v_{ij}, \forall\, o \in \mathcal{O}_{ij},\ \forall\ i,j \in \mathcal{A}, \tag{15b}$$

$$u_o^o = 0, \qquad \forall\, o \in \mathcal{O}, \tag{15c}$$

$$v_a \in \mathbb{R}_{\geq 0}, \qquad \forall\, a \in \mathcal{A}, \tag{15d}$$

$$u_i^o \in \mathbb{R}, \qquad \forall\, o \in \mathcal{O}_i,\ \forall\ i \in \mathcal{N}. \tag{15e}$$

The set $\mathcal{O}_{ij}$ represents the subset of commodities that have one or more paths that use arc $(i,j) \in \mathcal{A}$ (i.e., $\mathcal{O}_{ij} = \{o \in \mathcal{O} | (i,j) \in \mathcal{P}_o\}$) and, similarly, the set $\mathcal{O}_i$ represents the subset of commodities that have one or more paths that include node $i \in \mathcal{N}$ (i.e., $\mathcal{O}_i = \{o \in \mathcal{O} | i \in \mathcal{P}_o\}$). As previously mentioned, this is a relaxed version of the original path-based model; this is because path set $\mathcal{P}_o = \{\cup_{\{k \in \mathcal{K} | o(k) = o\}} \mathcal{P}_k\}$ is used to define set $\mathcal{O}_{ij}$, meaning that commodity k with $o(k) = o$ may transit any arc in $\mathcal{P}_o$, including arcs not in their original path set $\mathcal{P}_k$.

In an iterative fashion, we optimize (15) with $v_a = 1$ for each arc transporting commodity demands (i.e., the set of arcs $\{a \in \mathcal{A} | \sum_{t \in \mathcal{T}_a} \bar{y}_{at} > 0\}$). In each iteration, if the optimal objective value is 0, let $(\bar{v}, \bar{u})$ be an optimal solution to (15) and collect the following helper metric inequality:

$$\sum_{a \in \mathcal{A}} \bar{v}_a q_a \sum_{t \in \mathcal{T}_a} t y_{at} \geq \sum_{i \in \mathcal{N}} \sum_{o \in \mathcal{O}_i} \psi_i^o \bar{u}_i^o \tag{16}$$

and *integral metric inequality* generated from constraints (16):

$$\sum_{a \in \mathcal{A}} \bar{v}_a \left\lceil \frac{q_a}{q_{a'}} \right\rceil \sum_{t \in \mathcal{T}_a} t y_{at} \geq \sum_{i \in \mathcal{N}} \sum_{o \in \mathcal{O}_i} \left\lceil \frac{\psi_i^o}{q_{a'}} \right\rceil \bar{u}_i^o, \tag{17}$$

where $q_{a'} = \min_{\{a \in \mathcal{A} | \bar{v}_a = 1\}} \{q_a\}$. Identical metric inequalities may be generated for two different arcs; thus, only unique inequalities are collected.

While constraints (16) are always valid for and thus do not strengthen LP(BIN), one method to find potentially better aggregations of the original constraints is to generate metric inequalities using a point $(\bar{x}, \bar{y})$ from a strengthened version of LP(BIN) (e.g., LP(BIN) with constraints (10)). We will use this approach later in Section 5.

4.1. Lagrangian Selection

As discussed in the previous section, the metric inequalities generated correspond to solutions of LP(BIN). In this section, we discuss a method that searches for solutions to use when generating the metric inequalities. We adapt a method from [24, 5], which was designed to select good Gomory mixed integer cuts. The key idea is the following: Instead of adding multiple rounds of cuts which can have numerical issues [16, 23], moving the cuts to the objective function, so that we can re-solve the linear programming relaxation with an updated objective function, leads to finding a new

feasible point of the linear programming relaxation, allowing us to generate more metric inequalities iteratively.

When collecting helper metric inequalities (16) to add directly to BIN, we add the integral metric inequalities (17) to LP(BIN) using a Lagrangian relaxation approach. That is, we move these constraints to the objective of LP(BIN) with their Lagrange multiplier before generating another set of helper metric inequalities. See Algorithm 1 for pseudocode to generate and collect a set of helper and integral metric inequalities.

Algorithm 1: Lagrangian relaxation for metric inequality generation

Input: LP and solution to LP $(\bar{x}, \bar{y})$, commodity demands ψ

Result: Helper constraint set $\mathcal{M}^H$, updated LP (and solution)

$\bar{q}_a \leftarrow q_a \sum_{t \in \mathcal{T}_a} t\bar{y}_{at} \; \forall \; a \in \mathcal{A}$;
$\mathcal{M}^H \leftarrow \emptyset$;
$\mathcal{M}^I \leftarrow \emptyset$;
for $\{a \in \mathcal{A} | \sum_{t \in \mathcal{T}_a} \bar{y}_{at} > 0\}$ **do**
    Solve (15) with $\bar{q}$ and $v_a = 1$;
    if *optimal objective in STEP 5* $= 0$ **then**
        $(\bar{v}, \bar{u}) \leftarrow$ optimal solution of STEP 5;
        if ((16) *with* $(\bar{v}, \bar{u})) \notin \mathcal{M}^H$ **then**
            $\mathcal{M}^H \leftarrow \mathcal{M}^H \cup \{$(16) with $(\bar{v}, \bar{u})\}$;
            $\mathcal{M}^I \leftarrow \mathcal{M}^I \cup \{$(17) with $(\bar{v}, \bar{u})\}$;
end
LP$' \leftarrow$ Add constraint set $\mathcal{M}^I$ to constraint set of LP;
$(x^*, y^*) \leftarrow$ optimal solution of LP$'$;
$\mathcal{D}^I \leftarrow$ dual values for constraints $\mathcal{M}^I$ in optimal solution of LP$'$;
LP$'' \leftarrow$ Add constraint set $\mathcal{M}^I$ to objective of LP with dual multipliers $\mathcal{D}^I$;
return $\mathcal{M}^H$, LP$''$ (and (x^*, y^*))

5. Computational Study

Computational experiments are performed on two sets of instances to assess the impact of the previously discussed results on path- and arc-based models. We generate fulfillment instances using the demand data and network topology of an e-commerce company to evaluate path-based problems and use the publicly-available Canad instances [26] to evaluate arc-based problems. The objectives of our computational study are to (i) assess the value of reformulating INT using the multiple-choice binary capacity variables (2), (ii) measure the improvement to LP(BIN) (and the arc-based version) when applying the SAC-Pack constraints (10) and the Gen-SAC-Pack constraints (12), and (iii) identify a bound-improving strategy (e.g., helper metric inequalities, SAC-Pack constraints, etc.) for BIN that leads to a significant improvement to the lower bound found by a commercial solver within the provided runtime.

The optimization models and separation routines are coded in Python 3.10 and solved using Gurobi 11.0 with default settings for the IP solver, unless noted otherwise. We use a Linux computing cluster, consisting of nodes using 24-core dual Intel Xeon Gold 6226 CPUs @ 2.7 GHz with 192GB of RAM each, to run all experiments. Reported times are in hours, unless otherwise noted.

5.1. Instance Generation

We use two sets of instances for this study: e-commerce fulfillment instances and the Canad instances [26]. We utilize the e-commerce fulfillment instances to assess all previously described path-based methodologies and results, as well as a newly defined bound-improving strategy. In contrast, the Canad instances are only used to evaluate the effectiveness of the SAC-Pack and Gen-SAC-Pack constraints in strengthening the LP relaxations for *arc-based* models. Thus, all experiments are conducted on the e-commerce fulfillment instances, unless otherwise noted.

E-commerce fulfillment instances. We generate synthetic instances constructed from historical order fulfillment and middle-mile network data provided by an e-commerce company. We create three instance groups, as defined by the number of vendors $\mathcal{N}_v \subset \mathcal{N}$, fulfillment centers (FCs) $\mathcal{N}_f \subset \mathcal{N}$, and destinations $\mathcal{N}_d \subset \mathcal{N}$. In each instance, the set of vendors $\mathcal{N}_v$ represents externally-owned locations from which the company fulfills orders, the set of FCs $\mathcal{N}_f$ represents the internal locations within the fulfillment network from which the e-commerce company stores products and fulfills orders, and the set of destinations $\mathcal{N}_d$ represents last-mile distribution center locations that complete the last-mile of the delivery process to the final customer. Fulfillment centers act as both commodity origins and transfer centers; they can be used for intermediate shipment transfer along a path, whereas locations in $\mathcal{N}_v$ and $\mathcal{N}_d$ cannot. Thus, the total number of commodity origins for a given instance is $|\mathcal{N}_v| + |\mathcal{N}_f|$ and total number of arcs is at most $(|\mathcal{N}_v| + |\mathcal{N}_f|)|\mathcal{N}_d| + |\mathcal{N}_f|(|\mathcal{N}_v| + |\mathcal{N}_f| - 1)$.

For each group, we generate 5 randomized instances. Locations $\mathcal{N}_v$, $\mathcal{N}_f$, and $\mathcal{N}_d$ are sampled from larger sets of vendor, FC, and destination locations, respectively, for each instance. A set of potential commodities is initialized with all vendor-destination pairs. We then randomly remove a subset of commodities to better reflect the sparsity of real-world fulfillment networks, resulting in an average of 60% of vendor-destination pairs in the commodity set. A set of commodities originating at FC locations is also created. To do so, we create 5 commodities for each destination in $\mathcal{N}_d$ by assuming that the 5 nearest FCs send orders to d. For context, this construction is designed to mimic the practice of using multiple FCs to stock the same items and then to sometimes require shipment of items from further FCs due to inventory stock availabilty. Thus, the number of FC-originating commodities is always $5|\mathcal{N}_d|$ for each instance. Commodity demands are sampled from empirical demand distributions again generated from company shipment volume data.

We generate a set of arcs $\mathcal{A}$, consisting of both direct arcs $\mathcal{A}_d$ and consolidation arcs $\mathcal{A}_c$. Consolidation arcs connect all vendors to FCs, FCs to other FCs, and FCs to destinations, while direct arcs connect each vendor to destination if a commodity for that pair exists. To generate a path set $\mathcal{P}_k$ for commodity $k \in \mathcal{K}$, we start by including the direct path (equivalent to an arc)

connecting $o(k)$ to $d(k)$, with a length denoted as l_a. We then add the following four paths: (1) the shortest distance two-arc path using a single transfer FC, (2) the two-arc path transferring at the FC closest to the origin, (3) the two-arc path transferring at the FC closest to the destination, and (4) a three-arc path transferring at the FCs in (2) and (3), if they are not the same. We then include all network paths with a total length $\sum_{a' \in p} l_{a'}$ of $1.1 l_a$ or less and containing a maximum of three arcs. Duplicate paths in each set $\mathcal{P}_k$ are removed.

Consolidation arcs transport commodity flow using a truckload freight mode with a single capacity $q_a = 12000$ and cost dependent on the arc length l_a, whereas direct arcs can ship commodities using both truckload and less-than-truckload (LTL) freight modes. We utilize the same capacity and cost structure as [29] for direct arcs, as shown in Table 2, where we use l_a to denote the length of arc $a \in \mathcal{A}_d$ as measured in miles and v_a to denote the volume flowing on arc $a \in \mathcal{A}_d$. We pre-process all direct arc costs and incorporate them into the variable path cost c_p. This pre-processing step enables us to reduce the problem size by eliminating direct arcs from the set of arcs in the MCND models.

Table 2: Freight mode costs per shipment on direct arc $a \in \mathcal{A}_d$.

Freight Mode	Volume (lbs)	Cost
Truckload	$0 < v_a \leq 12,000$	$750 + 1.27 l_a$
LTL_1	$0 < v_a \leq 2,000$	$0.05(750 + 1.27 l_a) + v_a(0.234 + 0.0004 l_a)$
LTL_2	$2,000 < v_a \leq 2,700$	$0.05(750 + 1.27 l_a) + 2000(0.234 + 0.0004 l_a)$
LTL_3	$2,700 < v_a \leq 4,000$	$0.8 v_a(0.234 + 0.0004 l_a)$

In Table 3, we provide the following instance characteristics: group number, number of vendors, FCs, and destinations, instance number, number of commodities, total volume across all commodities, and number of arcs and paths.

Canad instances. To assess the strength of our new valid inequalities when using an arc-based formulation, we use a set of multicommodity network flow Canad instances [26] used in several papers [33, 20, 19, 28, 14, 15, 1]. In the previously mentioned papers, the instances are used to benchmark solution approaches for the MCND problem with splittable demand; however, they remain feasible with unsplittable demands. Thus, following [32], we use the set identified as "C" to study the unsplittable MCND problem. This set consists of 31 instances with the naming scheme #nodes-#arcs-#commodities-cost attribute-capacity attribute. There are two cost attributes denoted as F for a high fixed-to-variable cost ratio and V otherwise. There are two capacity attributes denoted as T for tightly-capacitated and L for loosely-capacitated arcs relative to total demand. The instances range in size from 20-30 nodes, 230-700 arcs, and 40-400 commodities.

Unlike the e-commerce fulfillment instances, all nodes can act as transfer nodes, and a single node can be an origin for one commodity and a destination for another; thus, there is no notion of node sets like $\mathcal{N}_v$, $\mathcal{N}_f$, or $\mathcal{N}_d$. Furthermore, the Canad instances were developed for the case where arc-capacity binary variables indicate whether an arc, with its predefined capacity q_a, is turned on,

Table 3: Instance characteristics.

Group	Vend	FC	Dest	Inst	Comm	Total Vol (lbs)	Arcs $\lvert\mathcal{A}\rvert$	Arcs $\lvert\mathcal{A}_c\rvert$	Paths $\lvert\mathcal{P}\rvert$
				1	252	641,443	365	188	972
				2	264	782,451	382	193	996
1	20	5	15	3	263	741,524	380	192	989
				4	262	747,582	380	193	1,093
				5	254	606,122	368	189	1,012
				1	3,040	2,177,671	4,113	1,348	18,639
				2	3,577	2,461,797	4,677	1,375	20,006
2	90	9	55	3	3,071	2,167,907	4,166	1,370	20,082
				4	2,841	1,995,813	3,930	1,364	17,570
				5	2,958	2,075,331	4,048	1,365	16,931
				1	4,233	2,313,167	5,698	1,790	29,727
				2	3,812	2,133,729	5,264	1,777	28,295
3	105	10	65	3	4,402	2,432,839	5,856	1,779	28,966
				4	4,375	2,427,937	5,837	1,787	30,009
				5	4,325	2,475,692	5,787	1,787	30,441

as opposed to installing integer multiples of q_a. Thus, for our arc-based experiments, we use the same formulation as the authors of [32] to allow for a direct comparison (see Appendix E for the arc-based, fixed-capacity formulation).

Lastly, while the e-commerce fulfillment Group 1 instances are comparable in size to the Canad instances, Groups 2 and 3 are significantly larger than those commonly used to evaluate MCND methodologies. We chose these larger instances to better evaluate the potential of embedding our developed methodology into more sophisticated solution approaches for addressing real-world e-commerce fulfillment network problems.

5.2. Comparison of Formulations

We begin our computational experiments by demonstrating how reformulating INT (1) with binary multiple-choice capacity variables (2) can improve the lower bound solution found by a commercial solver. To accomplish this, we optimize both models, INT (1) and BIN (3), for 3 hours using the best bound focus setting. In Table 4, we report the number of capacity variables, capacity constraints, IP lower bounds, IP gaps to the best objective found, and the improvements to the IP gaps. All values are the average across the 5 instances of each group. The best objective used to calculate the IP gap for each instance is obtained by optimizing both models for 12 hours with default settings and selecting the minimum objective value of the two. We measure the improvement (Impr) by calculating the percent decrease of the IP gaps.

We first observe that even with an increase in model size ($\sum_{a\in\mathcal{A}_c}(T_a^{\max}-1)$ more variables and $|\mathcal{A}|$ more constraints), the commercial solver finds stronger lower bounds, on average, for

Table 4: Comparing INT (1) and BIN (3) MCND formulations.

Group	Model Type	Cap Vars	Cap Constrs	IP LB	IP Gap	Impr
1	INT	191.0	191.0	189,577	0.9%	-
	BIN	394.8	382.0	190,268	0.6%	38.5%
2	INT	1,364.4	1,364.4	658,504	6.2%	-
	BIN	2,182.8	2,728.8	663,859	5.4%	12.4%
3	INT	1,784.0	1,784.0	747,035	8.3%	-
	BIN	2,788.8	3,568.0	748,815	8.1%	2.6%

BIN compared to INT. When analyzing the detailed results, we observe that BIN consistently outperforms in terms of best bound across *every* instance in each group.

We next observe that the largest improvement occurs for Group 1 instances, which can be attributed to their higher proportion of arcs with $T_a^{\max} > 1$, as evidenced by the increase in capacity variables. This confirms that the BIN model is especially effective in networks with a significant proportion of arcs potentially requiring more than one truckload.

Lastly, among the 5 instances in Group 1, a state-of-the-art commercial solver is only able to achieve optimality within 3 hours for the first instance, confirming the difficulty of these problems.

5.3. Improvements to LP Relaxation

In this section, we evaluate the strength of the valid inequalities introduced in Section 3, comparing them to existing methods.

E-commerce Fulfillment Instances. We begin by assessing the performance of the SAC-Pack and Gen-SAC-Pack constraints in strengthening the LP relaxation (LPR) solutions for e-commerce fulfillment instances using the Group 1 instances. Specifically, we compare the objective values of LP(BIN) strengthened by disaggregated linking constraints (5), integral metric inequalities (17), k-split c-strong inequalities with k set to a maximum value of 10 (applied using the same method described in the computational study of [2]), and our new valid inequalities (10) and (12). In each case, we continue generating additional cuts until no further improvement of the LP relaxation is observed. In Table 6, we present the results for a set of configurations (Conf) defined in Table 5. Specifically, Table 6 reports the number of cuts added and the improvement over LP(BIN) without valid inequalities (denoted as configuration (a)) for each instance in Group 1. It also includes the average number of cuts added, the average improvement, and the average IP gap across all instances. We list the configurations in decreasing order of average IP gap, where IP gap is defined as:

$$\frac{(\text{best IP objective} - \text{LPR})}{\text{best IP objective}}. \tag{18}$$

We define the improvement (Impr) of the strengthened LP relaxation solution (LPR) over the base LP relaxation solution (LP(BIN), configuration (a)), which is also equivalent to the improvement

of the IP gap, as:

$$\frac{(\text{LPR} - \text{LP(BIN)})}{(\text{best IP objective} - \text{LP(BIN)})}. \tag{19}$$

Table 5: Definitions of the valid inequality configurations reported. Configurations (b) through (k) add the listed inequalities to (a).

Conf	Definition
(a)	LP(BIN)
(b)	Integral metric inequalities (17)
(c)	Disaggregated linking constraints (5)
(d)	k-split c-strong inequalities for $k \leq 10$
(e)	(b), (c), and (d)
(f)	SAC-Pack constraints (10)
(g)	(f) plus post-processing
(h)	(g) plus Gen-SAC-Pack constraints (12) with $B \leq 3$
(i)	(g) plus Gen-SAC-Pack constraints (12) with $B \leq 10$
(j)	(b) plus (h)
(k)	(b) plus (i)

In Table 6, we first observe that the addition of integral metric inequalities alone (denoted as (b)) does not substantially improve the IP gap, and notably results in a worse IP gap than by simply adding the disaggregate linking constraints (5) (denoted as (c)). However, when combined with SAC-Pack, post-processed SAC-Pack, and Gen-SAC-Pack constraints, the inclusion of the integral metric inequalities can further strengthen the LP relaxation (see configurations (j) and (k)); whereas, we found in our initial experiments that combining the disaggregated linking constraints (c) with either (h) or (i) did not. We next observe that the k-split c-strong inequalities produce stronger relaxations than (c), as expected, but alone are significantly outperformed by the SAC-Pack constraints (e), which also add fewer cuts on average. We next combine integral metric inequalities, disaggregated linking constraints, and k-split c-strong inequalities (denoted as (e)), and find that this combination is also outperformed by SAC-Pack constraints and generates almost double the number of cuts, on average. As we continue down the table, we see that with each addition of cut type or looser upper bound (i.e., B increased from 3 to 10), the LP relaxation continues to improve and we ultimately achieve an IP gap of 4.9%, decreasing the IP Gap of LP(BIN) by 85.1%. We note also that the configuration that takes the longest to generate all cuts is (k) with an average time of 187s, whereas the next longest is (j) requiring only 93s on average. We will use these findings in Section 5.4 when designing a strategy to find stronger lower bounds for BIN.

Canad Instances. As previously noted, the two new classes of valid inequalities, SAC-Pack and Gen-SAC-Pack, can also be generated for arc-based formulations. To assess their effectiveness in strengthening the LP relaxation of such formulations, we compare the strengthened LP relaxation solutions for the arc-based Canad instances after adding disaggregated capacity linking constraints, k-split c-strong inequalities with k set to a maximum value of 10, and/or our new valid inequalities

Table 6: Valid inequalities (defined in Table 5) applied to the Group 1 instances.

Conf	1		2		3		4		5		Average		
	Cuts	Impr	Cuts	Impr	Cuts	Impr	Cuts	Impr	Cuts	Impr	Cuts	IP Gap	Impr
(a)	0	0%	0	0%	0	0%	0	0%	0	0%	0	24.8%	0%
(b)	817	17.3%	554	15.5%	474	17.0%	875	14.4%	636	21.2%	671	20.6%	17.1%
(c)	1,319	70.8%	1,350	65.2%	1,327	60.7%	1,470	62.5%	1,366	67.4%	1,366	8.6%	65.3%
(d)	2,056	69.8%	2,321	70.2%	2,382	65.1%	2,155	64.9%	1,985	66.0%	2,180	8.2%	67.2%
(e)	3,853	76.5%	4,079	75.9%	3,978	71.3%	4,415	74.1%	3,940	74.6%	4,053	6.3%	74.4%
(f)	1,793	79.5%	1,852	79.9%	2,020	73.9%	2,333	75.0%	2,456	77.0%	2,091	5.7%	77.1%
(g)	2,611	79.6%	2,707	80.5%	2,739	74.3%	3,068	75.3%	3,378	77.4%	2,901	5.6%	77.4%
(h)	2,940	81.0%	2,673	81.9%	3,092	75.9%	3,323	76.7%	3,332	78.9%	3,072	5.2%	78.9%
(i)	3,796	82.1%	3,181	82.8%	3,429	77.0%	4,551	77.5%	3,410	79.5%	3,673	5.0%	79.8%
(j)	3,709	81.8%	3,231	83.1%	3,658	77.4%	4,244	78.8%	3,816	79.6%	3,732	4.9%	80.2%
(k)	3,844	82.9%	3,841	84.3%	4,312	78.7%	4,693	79.5%	4,003	80.2%	4,139	4.7%	81.1%

(10) and (12). We use the same nomenclature as defined in Table 5. In Table 7, we provide a summary of results for the Canad instances. Specifically, for each configuration we test, we provide the average values across all 31 instances for the LP relaxation solution (LPR), number of cuts added, cut generation time in seconds, IP gap, and the improvement of the IP gap. We also include the number of instances where the optimal objective is achieved after adding the cuts (# Opt). We use Gurobi with default settings and set the solve time limit to 2 hours to obtain the best IP objective for each instance, which we then use to calculate the IP gap defined by (18). Of the 31 instances, Gurobi finds the optimal objective for 17 instances. Although Gurobi cannot solve all instances to optimality within 2 hours, the average IP gap is only 0.96% with an average objective of 194, 527. For the LP relaxation solutions and IP gaps for individual instances, see Appendix F.

Table 7: Summarized results for the Canad instances.

Conf	LPR	Cuts	Cut Time (s)	IP Gap	Impr	# Opt
(a)	163,189	0	0	16.1%	0.0%	0
(c)	185,747	93,245	18	4.5%	72.0%	0
(d)	188,162	1,705	161	3.3%	79.7%	2
(f)	191,107	3,467	125	1.8%	89.1%	2
(i)	191,211	11,710	1,019	1.7%	89.4%	2
(f),(d)	191,507	4,091	246	1.6%	90.4%	3

The results in Table 7 confirm that our new valid inequalities also significantly strengthen arc-based formulations. Our first observation is that adding only the disaggregated capacity linking constraints (c) leads to an improvement of over 70% for the base LP, but this improvement comes at the cost of adding an average of over 90,000 extra constraints. We next observe that while the k-split c-strong inequalities (d) achieve an 80% improvement with only a fraction of the cuts compared to (c), the SAC-Pack constraints (f) achieve an even greater improvement of nearly 90%, reducing the IP gap of (d) by almost half. Notably, although the total number of SAC-Pack constraints generated is twice that of the k-split c-strong inequalities, the SAC-Pack constraints,

on average, require less time to generate.

We next observe that adding Gen-SAC-Pack constraints to SAC-Pack constraints (denoted as (i)) does not appear necessary for these instances and problem definition (i.e., without integer multiples of capacities on arcs). This conclusion is based on the marginal improvement achieved compared to the additional cuts and, more critically, the time required. We next test the configuration where we first generate all SAC-Pack constraints and then generate k-split c-strong inequalities (denoted as (f),(d)). This configuration leads to a marginally improved IP gap but significantly decreases the cut generation time, as compared to (i). However, in general, it seems that the SAC-Pack constraints (f) achieve the right balance of cuts added and time required for the improvements gained.

We additionally compare configurations (f) and (f),(d) to the final bound found in [32] (reported in their Table 8). This final bound is found by dynamically adding disaggregated capacity linking constraints and cover inequalities to their extended LP formulation (through user cuts) at the root node and then executing their branch-and-price procedure for 30 minutes. When comparing the bounds achieved by configurations (f) and (f),(d), we find they outperform the final bounds in [32] for 21 and 29 instances (out of 31), respectively (see Table F.11 in Appendix F for the detailed instance results). While this is a bit of an unfair comparison (due to the advances in modern day commercial solvers, which help in the separation routines of SAC-Pack inequalities, compared to those available at the time [32] was written), we report this finding purely to demonstrate the potential improvements our new valid inequalities could lead to in such branch-and-price procedures.

5.4. Evaluation of Bound-Improving Strategy

Using the findings of the two previous sections regarding BIN and LP(BIN), we devise and report a strategy which results in improved lower bounds found by a commercial solver for the path-based MCND problem. We name this strategy, simply, 'xh-Better', where x represents the runtime limit in hours to generate all constraints and optimize the model with a commercial solver. We similarly call the base (default) model BIN (3) 'xh-BIN'. In this section, all models are solved with a best bound focus.

Better strategy and implementation. The strategy we will test against xh-BIN incorporates helper metric inequalities with Lagrangian selection, SAC-Pack constraints, post-processed SAC-Pack constraints, and Gen-SAC-Pack constraints with an upper bound of $B \leq 3$. We select this strategy because we find that it achieves the right balance between the improvements gained and the time it takes to generate constraints, allowing sufficient solve time. We use the following two-phase implementation:

1. Generate helper metric inequalities (16) to add to the BIN formulation prior to optimization (via solver) using the following procedure:

(a) Generate and collect helper metric inequalities using the optimal solution of LP(BIN), as defined in Algorithm 1. The returned LP is referred to as $\text{LP(BIN)}_{\text{MI}}$.

(b) Using the optimal solution of $\text{LP(BIN)}_{\text{MI}}$, generate and collect SAC-Pack constraints (10) found using the separation IP (11) and Gen-SAC-Pack constraints (12) found by post-processing the generated SAC-Pack constraints for each arc with assigned volume. Add these valid inequalities to $\text{LP(BIN)}_{\text{MI}}$. We refer to this LP as $\text{LP(BIN)}_{\text{MI+VI}}$

(c) Generate and collect helper metric inequalities using the optimal solution of $\text{LP(BIN)}_{\text{MI+VI}}$, as defined in Algorithm 1.

(d) Add helper metric inequalities generated and collected in STEPS (a) and (c) to BIN.

2. Add the following user cuts within the solver via callback routine, only at the root node: (i) SAC-Pack constraints (10), (ii) Gen-SAC-Pack constraints (12) generated by post-processing the SAC-Pack constraints generated in (i), and (iii) Gen-SAC-Pack constraints (12) with $B \leq 3$ generated using the separation routine outlined in Section 3.1.2 (with dynamic programming approach outlined in Appendix B).

In Table 8, we compare the results of 3h-BIN and 3h-Better for Group 1. Impressively, we observe that with 3h-Better, we can now solve 4 out of 5 instances, with an average IP gap of 0.1%, whereas with 3h-BIN, we can only solve the first instance, resulting in an average IP gap of 0.6%. Furthermore, the total runtime (sum of constraint generation and solve time) is an hour or less for each of the four instances solved using 3h-Better, with a substantial improvement (a 94% decrease) for instance 1 compared to 3h-BIN. Although we are still unable to solve instance 4 within 3 hours, we decrease the IP gap by 74% to 0.4%. We note that an average of 175 helper metric inequalities are added to the 3h-Better models and an average of 30 user cuts are employed by the solver in its bounding procedure.

Table 8: Group 1 results for 3h-BIN and 3h-Better.

Inst	3h-BIN		3h-Better		IP Gap	Time
	IP Gap	Runtime	IP Gap	Runtime	Impr	Impr
1	0.0%	2.1	0.0%	0.1	0%	94%
2	0.4%	3.0	0.0%	0.5	100%	82%
3	0.5%	3.0	0.0%	0.6	100%	79%
4	1.6%	3.0	0.4%	3.0	74%	0%
5	0.4%	3.0	0.0%	1.0	100%	67%

For Groups 2 and 3, we report the results for 3h-BIN, 12h-BIN, and 3h-Better in Table 9. For 3h-Better, the difference between 3 hours and the solve time is the time it takes to generate all constraints prior to optimization. The most significant observation is that 3h-Better (with limited solve time) produces stronger bounds than those found when solving BIN for 12 hours, with an average IP gap improvement of 26.5% and 22.5% for Groups 2 and 3, respectively. Thus, we find that, even for larger instances, utilizing the Better strategy results in significantly stronger lower bounds than those found by solving BIN for extended solve times.

Table 9: Groups 2 and 3 results for 12h-Base and 3h-Better.

Group	Inst	3h-BIN	12h-BIN	3hr-Better		3h-BIN	12h-BIN
		IP Gap	IP Gap	Solve Time	IP Gap	Impr	Impr
2	1	5.2%	5.1%	2.5	3.9%	25.3%	24.3%
	2	5.2%	5.1%	2.3	4.2%	19.5%	18.2%
	3	5.6%	5.5%	2.2	3.6%	35.3%	33.8%
	4	5.9%	5.7%	2.4	3.9%	33.4%	32.1%
	5	5.3%	5.1%	2.5	3.9%	25.9%	24.2%
3	1	8.1%	7.1%	1.5	5.5%	32.9%	22.7%
	2	7.2%	6.8%	1.5	4.9%	32.3%	28.8%
	3	7.7%	6.7%	1.5	5.1%	33.0%	23.2%
	4	9.7%	8.7%	1.6	7.1%	26.6%	18.4%
	5	7.5%	6.8%	1.0	5.5%	27.1%	19.2%

6. Conclusions

In this paper, we presented a reformulation of the multicommodity capacitated network design (MCND) problem which redefines the integer capacity variables as a multiple-choice selection of binary variables. We studied a structured relaxation of this formulation and showed that the convex hull solutions can be described using inequalities of a certain form. Using this result, we defined two new classes of valid inequalities, namely single-arc commodity packing (SAC-Pack) and generalized single-arc commodity packing (Gen-SAC-Pack) constraints, for the MCND problem with unsplittable flow and developed separation routines for each. We described metric inequalities, as well as a Lagrangian relaxation implementation approach to collect these inequalities, and later demonstrated how they help solvers identify stronger cutting planes to improve bounds. Finally, we presented computational results conducted on path-based instances derived from the historical demand data and network topology of a large e-commerce company, as well as on the arc-based Canad instances. We found that solvers can find better lower bounds for the reformulation using multiple-choice binary capacity variables (BIN) compared to when solving the integer capacity formulation (INT). Our computational study also showed the strength of the newly defined SAC-Pack and Gen-SAC-Pack constraints in comparison to existing methods for both path- and arc-based models. Our findings ultimately indicated that employing a strategy to solve BIN models, which combines the use of the new valid inequalities and aforementioned helper metric inequalities in a smart way, can substantially improve the bounds found by commercial solvers. Although we have defined and tested a strategy which results in stronger bounds compared to the base model, the components (e.g., SAC-Pack constraints) can be included in alternative, custom bounding procedures (e.g., branch-and-price algorithms) dependent on the problem.

Appendix A. Disaggregated Capacity Constraints Example

Consider the following example where the solution of LP(BIN) with constraints (5) produces a stronger solution compared to LP(INT) with constraints (5).

Assume we have 2 commodities $k = 1$ and $k = 2$ that can both traverse arc a using paths $p = 1$ and $p = 2$, respectively. Let demand $d_1 = 5$, demand $d_2 = 105$, and $q_a = 100$. Thus, $t^{\min}_{a1} = 1$, whereas $t^{\min}_{a2} = 2$. Let the LP relaxation solution be $x_1 = 0.75$ and $x_2 = 0.5$. The aggregrated linking constraints (1c) (and, similarly, (3c) using (2)) state:

$$5x_1 + 105x_2 \leq 100\tau_a.$$

Thus, the installed capacity on arc a is 0.5625 units for LP(INT) and LP(BIN).

To strengthen LP(INT), we can add the following constraints (4):

$$x_1 \leq \tau_a,$$
$$2x_2 \leq \tau_a.$$

Because $x_2 = 0.5$, the installed capacity increases to 1 unit (or $\tau_a = 1$).

To strengthen LP(BIN), we can add the following constraints (5):

$$x_1 \leq y_{a1} + y_{a2},$$
$$x_2 \leq y_{a2}.$$

This would set $y_1 = 0.25$ and $y_2 = 0.5$ (because y_1 has a lower penalty cost), which is a total installed capacity of 1.25 units.

Thus, linking constraints (5) added to LP(BIN) can produce a stronger relaxation solution compared to LP(INT) with linking constraints (4) if there exists a subset of commodities with demands that exceed an arc's single-unit capacity.

Appendix B. Separation Using Dynamic Programming.

We would like to improve the runtime of solving (14).

Observation 1 (Decomposition)**.** *Let*

$$S^t := \left\{ x \mid \sum_{p \in \mathcal{P}_a} a_p x_p \leq q_a t \right\} \ \forall\, t \in \mathcal{T}_a,$$

then we can solve:

$$\begin{aligned} \text{FEASVAL}^t := \quad & \max_{x,y} \quad \theta^{*\top} x - \alpha^*_t \\ & \text{s.t.} \quad (x, y) \in S^t. \end{aligned} \tag{B.1}$$

Note that:

- $\text{FEASVAL} = \max\{\max_t\{\text{FEASVAL}^t\}, 0\}$
- *If* $\text{FEASVAL} \neq 0$*, then* $\hat{t} \in \text{argmax}_t\{\text{FEASVAL}^t\}$ *and* $x^{\hat{t}} \in \text{argmax}_x S^{\hat{t}}$*, then* $(x^{\hat{t}}, e_{\hat{t}})$ *is an optimal solution of (14). If* $\text{FEASVAL} = 0$*, then* $(0, 0)$ *is an optimal solution of (14).*

Appendix B.1. Improving DP running time.

Observe now that (B.1) is a knapsack problem. Moreover, we know that $\|\theta\|_\infty \leq B$, as we artificially constrained this in (13). We can exploit this feature in the following fashion.

Let us consider a general knapsack of the following form:

$$\begin{aligned} \text{OPT} := \quad & \max \quad \textstyle\sum_{j=1}^n \theta_j x_j \\ & \text{s.t.} \quad \textstyle\sum_{j=1}^n a_j x_j \leq q_a, \\ & \qquad x \in \{0,1\}^n. \end{aligned} \tag{B.2}$$

We "guess" an optimal value of (B.2) is γ. Consider the following min-knapsack:

$$\begin{aligned} \text{VAL}(\gamma) := \quad & \min \quad \textstyle\sum_{j=1}^n a_j x_j \\ & \text{s.t.} \quad \textstyle\sum_{j=1}^n \theta_j x_j \geq \gamma, \\ & \qquad x \in \{0,1\}^n. \end{aligned} \tag{B.3}$$

DP running-time. The key fact is that since θ's are small, any guessed value of γ is not too large; in particular $\gamma \leq n \cdot \|\theta\|_\infty$. So the DP to solve (B.3) takes $\mathcal{O}(n \cdot n\|\theta\|_\infty) = \mathcal{O}(n^2 B)$ time, which could be significantly smaller time than solving (B.2) which takes $\mathcal{O}(n \cdot q_a)$ time since q_a can be super large and B is quite small.

To solve (B.2) using (B.3), we begin with an observation.

Observation 2. *Observe that:*

- *If we guessed too low, i.e,* $\gamma \leq \text{OPT}$*, then* $\text{VAL}(\gamma) \leq q_a$*.*
- *If we guessed too high, i.e,* $\gamma > \text{OPT}$*, then* $\text{VAL}(\gamma) > q_a$*.*

We can use Observation (2) to design a bisection search as follows:

1. (Pre-process) Let $\text{LB} = 0$ and $\text{UB} = \sum_{j=1}^n \theta_j$. Solve (B.2) with $\gamma = \text{UB}$. If $\text{VAL}(\text{UB}) \leq q_a$, STOP, as we have solved the problem and the optimal solution of (B.3) is an optimal solution of (B.2). Else:
2. Let $\gamma = \left\lceil \frac{1}{2} \cdot (\text{LB} + \text{UB}) \right\rceil$. Solve (B.2).
3. If $\text{VAL}(\gamma) > q_a$, then set $\text{UB} \leftarrow \gamma$. If $\text{VAL}(\gamma) \leq q_a$, then set $\text{LB} \leftarrow \gamma$. If $\text{LB} = \text{UB}$, STOP (and the optimal solution of (B.3) is an optimal solution of (B.2)), else GOTO STEP 2.

Overall running time. The number of iterations of bisection search is known to be $\mathcal{O}\left(\log(\sum_{j=1}^n \theta_j)\right)$. Combining everything (accounting for the running-time of solving (B.3)), we get a running-time of $\mathcal{O}(n^2 B \cdot \log(nB))$ which can be much better than $\mathcal{O}(n \cdot q_a)$.

Appendix C. Solver Performances with Helper Metric Inequalities

To assess whether the benefits of helper metric inequalities are specific to Gurobi, we also utilize CPLEX and SCIP to solve Group 6 instances of (3) (3h-BIN) and (3) plus two rounds of helper metric inequalities added (3h-MI2; as described in Section 4 and Algorithm 1). We provide a runtime of 3 hours and solve the models using default settings for all solvers. In Table C.10, we report the solvers and versions tested, the resulting average IP Gaps calculated using the best found objective, and the average improvement to the IP gap for each solver after adding the helper metric inequalities to BIN. Also note that while the runtime for each setup was 3 hours, the time to generate the helper metric inequalities took an average of 1.5 hours, meaning the 3h-MI2 models had an average solve time limit of 1.5 hours.

Table C.10: Comparison of solver performance using Group 6 instances.

Solver	Version	IP Gap		IP Gap
		3h-BIN	3h-MI2	Impr
Gurobi	11.0.0	8.3%	6.0%	27.1%
CPLEX	22.1.1.0	9.3%	5.9%	36.4%
SCIP	9.0.0	8.7%	7.6%	12.9%

Consistent with the results in Section 5.4, the addition of helper metric inequalities results in all three solvers finding stronger lower bounds through improved cutting-plane generation, with CPLEX seemingly benefiting the most.

Appendix D. Arc-based Flow Conservation Constraints

Let $x_{ij}^o \in \{0,1\}$ equal 1 if commodity $o \in \mathcal{O}$ is transported via arc $(i,j) \in \mathcal{A}$, and 0 otherwise. The arc-based flow conservation constraints are formulated as follows:

$$\sum_{(i,j)\in\delta^+(i)} x_{ij}^o - \sum_{(j,i)\in\delta^-(i)} x_{ji}^o = \begin{cases} \psi_o^o, & i = o, \\ 0, & i \neq o, d, \\ \psi_d^o, & i = d, \end{cases} \qquad \forall\ i \in \mathcal{N}^o,\ \forall\ o \in \mathcal{O}, \tag{D.1}$$

where $\delta^+(i)$ and $\delta^-(i)$ refer to the sets of arcs emanating from and ending at node i, respectively, and set $\mathcal{N}^o$ represents the nodes contained in path set $\mathcal{P}_o$ for commodity $o \in \mathcal{O}$.

Appendix E. Arc-Based Formulation When Solving Canad Instances

Similar to the problem definition in Section 2, let c_{ij} represent the variable cost of transporting one unit of demand on arc $(i,j) \in \mathcal{A}$, f_{ij} represent the fixed cost of activating (i.e., installing one unit of capacity) on arc $(i,j) \in \mathcal{A}$, let q_{ij} represent the capacity of arc $(i,j) \in \mathcal{A}$, and let d_k represent the demand of commodity $k \in \mathcal{K}$ with origin $o(k)$ and destination $d(k)$. Let $x_{ij}^k \in \{0,1\}$ be a binary variable indicating if commodity $k \in \mathcal{K}$ is transported via arc $(i,j) \in \mathcal{A}$ or not. Let

$y_{ij} \in \{0,1\}$ be a binary variable indicating if arc $(i,j) \in \mathcal{A}$ is activated to transport commodity demands or not. The arc-based unsplittable MCND problem is:

$$\min_{x,y} \quad \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij} d_k x_{ij}^k + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} \tag{E.1a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} x_{ij}^k - \sum_{(j,i)\in\delta^-(i)} x_{ji}^k = \begin{cases} 1, & i = o(k), \\ 0, & i \neq o(k), d(k), \\ -1, & i = d(k), \end{cases} \quad \forall\, i \in \mathcal{N},\ \forall\, k \in \mathcal{K}, \tag{E.1b}$$

$$\sum_{k\in\mathcal{K}} d_k x_{ij}^k \leq q_{ij} y_{ij}, \qquad \forall\, (i,j) \in \mathcal{A}, \tag{E.1c}$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall\, (i,j) \in \mathcal{A},\ \forall\, k \in \mathcal{K}, \tag{E.1d}$$

$$y_{ij} \in \{0,1\}, \qquad \forall\, (i,j) \in \mathcal{A}, \tag{E.1e}$$

where $\delta^+(i)$ and $\delta^-(i)$ refer to the sets of arcs emanating from and ending at node $i \in \mathcal{N}$, respectively. Constraints (E.1b) ensure flow balance. Constraints (E.1c) activate arc $(i,j) \in \mathcal{A}$ and ensure its capacity is not exceeded.

When noted, we use the following disaggregated capacity linking constraints strengthen the LP relaxation:

$$x_{ij}^k \leq y_{ij}, \qquad \forall\, k \in \mathcal{K},\ \forall\, (i,j) \in \mathcal{A}.$$

Appendix F. Detailed Canad Instance Results

In Table F.11, we provide the individual Canad instance results that were summarized in Table 7. Results in bold indicate the optimal objective was achieved for that configuration.

References

[1] Akhavan Kazemzadeh, M. R., Bektaş, T., Crainic, T. G., Frangioni, A., Gendron, B., and Gorgone, E. (2022). Node-based lagrangian relaxations for multicommodity capacitated fixed-charge network design. *Discrete Applied Mathematics*, 308:255–275. Combinatorial Optimization ISCO 2018.

[2] Atamtürk, A. and Rajan, D. (2002). On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming*, 92:315–333.

[3] Atamturk, A. and Gunluk, O. (2021). Multicommodity multifacility network design. In Crainic, T. G., Gendreau, M., and Gendron, B., editors, *Network Design with Applications to Transportation and Logistics*, chapter 5, pages 141–166. Springer.

[4] Bakir, I., Erera, A. L., and Savelsbergh, M. W. P. (2021). Motor carrier service network design. In Crainic, T. G., Gendreau, M., and Gendron, B., editors, *Network Design with Applications to Transportation and Logistics*, chapter 14, pages 427–467. Springer.

Table F.11: Arc-based results for the Canad instances. Objectives for (f) and (f),(d) are marked with an asterisk if the LPR objective was greater than the final bound found in [32].

Instance	Obj	(a)		(c)		(d)		(f)		(i)		(f),(d)	
		LPR	IP Gap	LPR	IP Gap	LPR	IP Gap	LPR	IP Gap	LPR	IP Gap	LPR	IP Gap
20-230-40-V-L	423,933	378,623	10.69%	422,853	**0.25%**	**423,933**	**0.00%**	**423,933***	**0.00%**	**423,933**	**0.00%**	**423,933***	**0.00%**
20-230-40-V-T	398,870	359,175	9.95%	368,494	7.62%	378,573	5.09%	397,606*	0.32%	397,653	0.31%	397,826*	0.26%
20-230-40-F-T	668,699	584,306	12.62%	633,466	5.27%	642,157	3.97%	667,972*	**0.11%**	667,972	0.11%	**668,699***	**0.00%**
20-230-200-V-L	93,893	68,805	26.72%	90,934	3.15%	90,934	3.15%	90,517	3.60%	90,622	3.48%	90,945	3.14%
20-230-200-F-L	138,084	98,343	28.78%	131,951	4.44%	131,951	4.44%	130,981	5.14%	130,920	5.19%	131,965*	4.43%
20-230-200-V-T	98,610	73,976	24.98%	95,461	3.19%	95,521	3.13%	94,742*	3.92%	94,718	3.95%	95,553*	3.10%
20-230-200-F-T	137,731	100,728	26.87%	131,499	4.53%	131,513	4.51%	131,011*	4.88%	131,467	4.55%	131,559*	4.48%
20-300-40-V-L	429,335	386,462	9.99%	427,029	**0.54%**	**429,335**	**0.00%**	**429,335***	**0.00%**	**429,335**	**0.00%**	**429,335***	**0.00%**
20-300-40-F-L	597,059	492,690	17.48%	575,255	3.65%	584,975	2.02%	589,451*	1.27%	589,451	1.27%	589,537*	1.26%
20-300-40-V-T	501,766	448,170	10.68%	460,930	8.14%	479,890	4.36%	500,047*	0.34%	500,047	0.34%	500,301*	0.29%
20-300-40-F-T	643,395	546,666	15.03%	596,667	7.26%	614,656	4.47%	642,043*	0.21%	642,043	0.21%	642,043*	0.21%
20-300-200-V-L	75,763	58,934	22.21%	73,107	3.51%	73,136	3.47%	72,817	3.89%	73,030	3.61%	73,226*	3.35%
20-300-200-F-L	117,983	87,743	25.63%	110,795	6.09%	110,834	6.06%	109,987	6.78%	110,005	6.76%	110,860*	6.04%
20-300-200-V-T	76,508	61,482	19.64%	73,956	3.34%	74,051	3.21%	73,959*	3.33%	74,056	3.20%	74,111*	3.13%
20-300-200-F-T	109,632	84,733	22.71%	103,568	5.53%	103,618	5.49%	103,177*	5.89%	103,414	5.67%	103,748*	5.37%
30-520-100-V-L	54,387	44,118	18.88%	53,023	2.51%	53,246	2.10%	53,279*	2.04%	53,350	1.91%	53,415*	1.79%
30-520-100-F-L	96,277	67,560	29.83%	90,174	6.34%	90,541	5.96%	90,401*	6.10%	90,550	5.95%	90,839*	5.65%
30-520-100-V-T	53,812	46,551	13.49%	51,239	4.78%	51,840	3.67%	53,034*	1.45%	53,031	1.45%	53,091*	1.34%
30-520-100-F-T	99,195	76,710	22.67%	93,999	5.24%	94,847	4.38%	95,232*	3.99%	95,330	3.90%	95,744*	3.48%
30-520-400-V-L	114,484	96,615	15.61%	111,512	2.60%	111,576	2.54%	111,165	2.90%	111,287	2.79%	111,613*	2.51%
30-520-400-F-L	150,618	122,680	18.55%	146,471	2.75%	146,699	2.60%	146,039*	3.04%	146,333	2.84%	146,828*	2.52%
30-520-400-V-T	116,332	100,649	13.48%	113,973	2.03%	114,163	1.86%	113,804	2.17%	114,027	1.98%	114,213	1.82%
30-520-400-F-T	155,982	126,357	18.99%	149,725	4.01%	149,928	3.88%	149,479*	4.17%	149,732	4.01%	149,996*	3.84%
30-700-100-V-L	47,883	39,055	18.44%	47,292	1.23%	47,478	0.85%	47,441*	0.92%	47,441	0.92%	47,528*	0.74%
30-700-100-F-L	60,384	45,709	24.30%	58,155	3.69%	58,342	3.38%	58,308	3.44%	58,313	3.43%	58,568*	3.01%
30-700-100-V-T	47,670	40,992	14.01%	45,078	5.44%	45,822	3.88%	46,629*	2.18%	46,764	1.90%	46,894*	1.63%
30-700-100-F-T	56,686	44,810	20.95%	53,660	5.34%	54,485	3.88%	55,059*	2.87%	55,270	2.50%	55,157*	2.70%
30-700-400-V-L	98,635	79,056	19.85%	96,545	2.12%	96,565	2.10%	96,038	2.63%	96,246	2.42%	96,578*	2.09%
30-700-400-F-L	136,783	106,564	22.09%	129,807	5.10%	130,707	4.44%	130,101	4.89%	130,167	4.84%	130,782*	4.39%
30-700-400-V-T	96,756	81,217	16.06%	94,006	2.84%	94,105	2.74%	93,670	3.19%	93,786	3.07%	94,164*	2.68%
30-700-400-F-T	132,617	109,364	17.53%	127,545	3.82%	127,611	3.77%	127,075*	4.18%	127,243	4.05%	127,670*	3.73%
Average	194,508	163,189	0.0%	185,747	0.0%	188,162	0.0%	191,107	0.0%	191,211	0.0%	191,507	0.0%

[5] Becu, B., Dey, S. S., Qiu, F., and Xavier, A. S. (2024). Approximating the Gomory mixed-integer cut closure using historical data. *arXiv preprint arXiv:2411.15090*.

[6] Benhamiche, A., Mahjoub, A. R., Perrot, N., and Uchoa, E. (2016). Unsplittable non-additive capacitated network design using set functions polyhedra. *Computers & Operations Research*, 66:105–115.

[7] Benidis, K., Paschos, G., Gross, M., and Iosifidis, G. (2023). Middle-mile optimization for next-day delivery. *arXiv preprint arXiv:2310.18388*.

[8] Bienstock, D., Chopra, S., Günlük, O., and Tsai, C.-Y. (1998). Minimum cost capacity installation for multicommodity network flows. *Mathematical programming*, 81:177–199.

[9] Bodur, M., Del Pia, A., Dey, S. S., Molinaro, M., and Pokutta, S. (2018). Aggregation-based cutting-planes for packing and covering integer programs. *Mathematical Programming*, 171:331–359.

[10] Bonami, P. and Margot, F. (2014). Cut generation through binarization. In *Integer Programming and Combinatorial Optimization: 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings 17*, pages 174–185. Springer.

[11] Boyd, E. A. (1994). Fenchel cutting planes for integer programs. *Operations Research*, 42(1):53–64.

[12] Brockmüller, B., Günlück, O., and Wolsey, L. A. (1996). Designing private line networks-polyhedral analysis and computation. Technical report, Université catholique de Louvain, Center for Operations Research and ....

[13] Chen, L., Chen, W.-K., Yang, M.-M., and Dai, Y.-H. (2021). An exact separation algorithm for unsplittable flow capacitated network design arc-set polyhedron. *Journal of Global Optimization*, 81:659–689.

[14] Chouman, M. and Crainic, T. G. (2015). Cutting-plane matheuristic for service network design with design-balanced requirements. *Transportation Science*, 49(1):99–113.

[15] Chouman, M., Crainic, T. G., and Gendron, B. (2017). Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 51(2):650–667.

[16] Cornuéjols, G., Margot, F., and Nannicini, G. (2013). On the safety of gomory cut generators. *Mathematical Programming Computation*, 5:345–395.

[17] Costa, A. M., Cordeau, J.-F., and Gendron, B. (2009). Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392.

[18] Crainic, T. and Gendreau, M. (2021). Heuristics and metaheuristics for fixed-charge network design. In Crainic, T. G., Gendreau, M., and Gendron, B., editors, *Network Design with Applications to Transportation and Logistics*, chapter 14, pages 427–467. Springer.

[19] Crainic, T. G., Frangioni, A., and Gendron, B. (2001). Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):73–99.

[20] Crainic, T. G., Gendreau, M., and Farvolden, J. M. (2000). A simplex-based tabu search method for capacitated network design. *INFORMS journal on Computing*, 12(3):223–236.

[21] Croxton, K. L., Gendron, B., and Magnanti, T. L. (2003). Models and methods for merge-in-transit operations. *Transportation science*, 37(1):1–22.

[22] Croxton, K. L., Gendron, B., and Magnanti, T. L. (2007). Variable disaggregation in network flow problems with piecewise linear costs. *Operations research*, 55(1):146–157.

[23] Dey, S. S. and Molinaro, M. (2018). Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170:237–266.

[24] Fischetti, M. and Salvagnin, D. (2011). A relax-and-cut framework for gomory mixed-integer cuts. *Mathematical Programming Computation*, 3:79–102.

[25] Fontaine, P., Crainic, T. G., Jabali, O., and Rei, W. (2021). Scheduled service network design with resource management for two-tier multimodal city logistics. *European Journal of Operational Research*, 294(2):558–570.

[26] Frangioni, A. (2012). Multicommodity flow problems. Retrieved April 2024, from https://commalab.di.unipi.it/datasets/mmcf/#Canad.

[27] Frangioni, A. and Gendron, B. (2009). 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229–1241.

[28] Ghamlouche, I., Crainic, T. G., and Gendreau, M. (2003). Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations research*, 51(4):655–667.

[29] Greening, L. M., Dahan, M., and Erera, A. L. (2023). Lead-time-constrained middle-mile consolidation network design with fixed origins and destinations. *Transportation Research Part B: Methodological*, 174:102782.

[30] Greening, L. M., Park, J., Dahan, M., Erera, A. L., and Montreuil, B. (2024). Integrating order-to-delivery time sensitivity in e-commerce middle-mile consolidation network design. *optimization-online preprint*.

[31] Hewitt, M. (2022). The flexible scheduled service network design problem. *Transportation Science*, 56(4):1000–1021.

[32] Hewitt, M., Nemhauser, G., and Savelsbergh, M. W. (2013). Branch-and-price guided search for integer programs with an application to the multicommodity fixed-charge network flow problem. *INFORMS Journal on Computing*, 25(2):302–316.

[33] Hewitt, M., Nemhauser, G. L., and Savelsbergh, M. (2010). Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS journal on computing*, 22(2):314–325.

[34] Satici, O. and Dayarian, I. (2024). Tactical and operational planning of express intra-city package services. *Omega*, 122:102940.

[35] van Hoesel, S. P., Koster, A. M., van de Leensel, R. L., and Savelsbergh, M. W. (2002). Polyhedral results for the edge capacity polytope. *Mathematical Programming*, 92:335–358.

[36] Wang, Z., Qi, M., Cheng, C., and Zhang, C. (2019). A hybrid algorithm for large-scale service network design considering a heterogeneous fleet. *European Journal of Operational Research*, 276(2):483–494.