The Value of Limited Flexibility in Service Network Designs

Ahmad Baubaid\textsuperscript{1,2}, Natashia Boland\textsuperscript{1}, and Martin Savelsbergh\textsuperscript{1}

\textsuperscript{1}H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
baubaid@gatech.edu, \{natashia.boland, martin.savelsbergh\}@isye.gatech.edu
\textsuperscript{2}Department of Systems Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

May 23, 2019

Abstract

Less-than-truckload carriers rely on the consolidation of freight from multiple shippers to achieve economies of scale. Collected freight is routed through a number of transfer terminals at each of which shipments are grouped together for the next leg of their journeys. We study the service network design problem confronted by these carriers. This problem includes determining (1) the number of services (trailers) to operate between each pair of terminals, and (2) a load plan which specifies the sequence of transfer terminals that freight with a given origin and destination will visit. Traditionally, for every terminal and every ultimate destination, a load plan specifies a unique next terminal.

We introduce the $\mathbf{p}$-alt model, which generalizes traditional load plans by allowing decision-makers to specify a desired number of next terminal options for terminal-destination pairs using a vector $\mathbf{p}$. We compare a number of exact and heuristic approaches for solving a two-stage stochastic variant of the $\mathbf{p}$-alt model. Using this model, we show that by explicitly considering demand uncertainty and by merely allowing up to two next terminal options for terminal-destination pairs in the load plans, carriers can generate substantial cost savings; in the order of 10\% over traditional load plan designs obtained by deterministic models.

1 Introduction

Less-than-truckload (LTL) carriers provide transportation for freight shipments that occupy less than 10\% of trailer capacity. Consequently, these carriers rely on consolidation of freight from multiple shippers to make their operations economically viable. LTL carriers create consolidation opportunities by routing collected freight through a network of consolidation terminals as it is transported between origins and destinations. At each of these terminals, freight on incoming trailers is sorted and then consolidated into outgoing trailers that transport that freight to the next terminal on its journey to its final destination.

At the start of each operating season, LTL carriers must plan how that season’s demand is to be served. This tactical planning process is referred to in the literature as service network design (Crainic 2000, Wieberneit 2008). In particular, for LTL carriers, service network design problems address the following aspects of the carriers’ operations: (1) deciding the number of services (trailers) to operate in the network, their frequencies, and their types/sizes, and (2) deciding the
A load plan which dictates the sequence of terminals that freight with a given origin and destination should follow as part of its journey through the network. The load plan, therefore, is a key part of a carrier’s tactical plan as it dictates how collected freight will be consolidated and routed on a day-to-day basis.

The traditional form of a load plan for an LTL carrier restricts freight movement such that freight arriving at a terminal and destined for some destination terminal is always loaded onto trailers destined for a unique next terminal, regardless of the origin of that freight. Because load plans are devised with respect to predicted OD demands, some carriers recognize the benefit of having more flexible load plans in order to deal with the inherently uncertain demand. In particular, at some terminals, carriers might allow incoming freight destined for destination $d$ to be loaded into trailers destined for two different next terminals. The choice of this additional routing option is usually determined after obtaining the traditional load plan, often based on past experience; these decisions are not integrated into the load plan design process. The additional routing option provides a backup option in case demand on the day is such that the routing option in the traditional load plan for that freight has insufficient capacity. (We present a formal discussion of these types of load plans in Section 2.)

In this paper, we demonstrate the benefit of explicitly designing more flexible load plans, and show that such plans are effective in dealing with demand uncertainty. In fact, we empirically show that these types of load plans with seemingly limited flexibility are as effective in dealing with demand uncertainty as load plans that allow freight with a given destination to be sent to any next terminal. As the latter setup is much more difficult to implement in practice, it is appealing for carriers to have a load plan with limited flexibility that is only marginally more complex to operate than a traditional one, but that still manages to yield the benefits of a much more flexible form of load plan.

To demonstrate these advantages, we formulate and solve a two-stage stochastic Capacitated Multi-commodity Network Design (CMND) problem (Gendron et al. 1999). In the first stage, scheduled capacity is installed in the network (i.e., the number of trailers to operate between each pair of terminals in the network) and an associated load plan is determined. The load plan respects the planner’s desired level of flexibility (where flexibility is represented by the number of next terminals that a terminal can have for a particular destination). To accommodate realized daily demand, we allow the acquisition of additional capacity (i.e., outsourced trailers) in the second stage, if needed. Consequently, both stages of the model feature discrete decisions.

It is impractical to find optimal solutions to this two-stage stochastic CMND model, and, therefore, we employ a Sample Average Approximation (SAA) framework (Kleywegt et al. 2002). In this approach, we randomly sample a small number of demand scenarios, and a sample average is used to approximate the expected objective value in the original stochastic program. The resulting mixed-integer program, which we refer to as the sample problem, is also a CMND problem, and can be solved by deterministic optimization techniques. This process is repeated for a certain number of iterations, each time with a different sample, to obtain candidate first-stage solutions to the original problem. Each of these candidates is then evaluated on a much larger set of scenarios in order to obtain better estimates of the recourse cost, i.e., the cost of acquiring outsourced trailers. Ultimately, the first-stage solution candidate with the lowest estimated expected total cost is selected as the final solution. The sample problem in each SAA iteration is a CMND problem, which is known to be difficult to solve, even if the sample problem contains just a single scenario (Ghamlouche et al. 2003, Crainic et al. 2000, Bai et al. 2014). Therefore, we evaluate and compare
a number of solution approaches (both exact and heuristic) for solving the sample problem. After selecting a solution approach for the sample problems, we then conduct experiments to assess the benefits of limited flexibility load plans.

Our contributions in this paper include the following:

- Introducing and studying a generalization of the traditional load plan design model by including a parameter that allows for controlling the level of flexibility desired in the load plan. We call the resulting model the $p$-alt model.
- Showing that load plans with limited flexibility, i.e., two next destination options, can yield most of the benefits of operating with a fully-flexible load plan when faced with uncertain demand (while adding only marginal operational complexity compared to traditional load plans).
- Demonstrating that by explicitly considering demand uncertainty in load plan design, designs with noticeable cost savings are generated when compared with their deterministic counterparts (which are currently used in practice).
- Comparing the performance of a number of exact and heuristic approaches for solving the sample problems of the $p$-alt model in terms of runtime and solution quality (within the context of a SAA approach).

The remainder of the paper is structured as follows. Section 2 provides a brief introduction to the operations of LTL carriers and load plan designs. Section 3 presents an overview of relevant research. Section 4 formally defines the $p$-alt model for the design of flexible load plans. Section 5 outlines the approaches we use to obtain high-quality solutions. Section 6 discusses the results of an extensive computational study, which explores algorithmic choices and analyzes the benefits of load plan variants. Finally, Section 7 presents our conclusions.

2 LTL Carrier Operations and Load Plans

An LTL carrier’s network is made up of two types of terminals: end-of-line (EOL) terminals, which are the terminals that serve as origins and/or destinations of freight, and breakbulk (BB) terminals, which in addition to being freight origins/destinations, are also consolidation points in the carrier’s network. The set of EOL and BB terminals together comprise the carrier’s line-haul network.

During the day, each terminal dispatches trucks early in the day to deliver and pick up shipments from local customers that are served by that terminal. This daily local operation is known as the city operation. The collected shipments are brought to the terminal when the trucks return late in the day. Typically, there is not enough freight to economically justify dispatching a full trailer from the terminal directly to a destination terminal. For this reason, these shipments are loaded onto trailers that are dispatched to one or more nearby BB terminals to be consolidated with other shipments. After sorting the freight at the BB terminals, some of it is loaded onto trailers that are headed for other BBs, other freight is loaded onto trailers that are headed for nearby EOLs (containing shipments whose final destinations are consignees in the areas served by those EOLs), and some freight is loaded onto a city trailer for local delivery in a city operation. The movement of freight between the terminals of the network is known as the line-haul operation of the carrier, and is the focus of this work.
2.1 Traditional load plans

A load plan is a freight routing plan that specifies the sequence of terminals that each shipment will follow given its origin and destination. Traditionally, in their load plans, LTL carriers impose an anti-arborescence (or in-tree) structure for each destination $d$ in the network (Powell and Koskosidis 1992). In other words, all freight headed for destination $d$ that is currently at some intermediate terminal $i$ in the network is always directed to a single next terminal $j$, regardless of its origin. An example of this can be seen in Figure 1 where the solid lines represent a traditional load plan for destination $d$. Specifically, at terminal $i$, all freight headed for destination $d$ is sent to terminal $j$. Note that this structure also implies that freight with a given OD pair will follow a unique path from origin to destination. Although this type of load plan is the most restrictive of the load plans we discuss, it is the simplest to operate locally at each terminal as workers need only check the final destination of an incoming shipment to determine which outgoing trailer that shipment should be loaded onto.

![Figure 1: Example of a traditional load plan (solid lines) and a limited flexibility load plan (solid and dashed lines together) for a destination $d$ in the line-haul network; Circles represent EOL terminals, squares represent BB terminals, and arrows indicate freight movement)](image)

2.2 Limited-Flexibility load plans

In what we refer to as limited-flexibility load plans, terminals can have up to two routing options for freight with a given destination. In practice, this would mean that there is a primary routing option, as well as an alternative option (or an “alt”) which is used after exhausting the capacity offered on the primary on that day. However, in our work, we do not impose a priority for the primary option. We use the term limited-flexibility to contrast this with a full-flexibility load plan in which freight arriving at a given terminal can go to any next terminal.

If, on a given day, the scheduled trailer capacity available on the primary and the alt is not sufficient, then carriers can add additional outsourced trailers to either one at higher cost. Deals are typically negotiated with independent owner-operators in advance to help the carrier prepare for such eventualities. The dashed lines in Figure 1 represent an example of alts at some terminals for destination $d$ (solid lines now represent the primaries). In particular, all freight headed for
destination $d$ that is currently at terminal $i$ can be sent to either terminal $j$ (on the primary) or terminal $k$ (on the alt) on a given day.

In this paper, we show that limited-flexibility load plans can yield most of the benefits of full-flexibility load plans. As executing full-flexibility load plans involves complex workflows and requires a high-level of automation, limited-flexibility load plans provide an attractive alternative that provides similar benefits while being only marginally more complex to operate compared to traditional load plans.

### 2.3 Motivating example

To motivate the benefits of introducing an alt at some terminals, consider the small example represented in the networks of Figure 2. In this example, the cost of operating a scheduled trailer moving between nodes $i$ and $k$ is $\frac{1}{2}$ (in either direction), while the cost of operating a scheduled trailer between nodes $i$ and $j$ as well as nodes $j$ and $k$ is 1 (also in either direction), and outsourced trailers cost 50% more than scheduled trailers. Furthermore, the underlying network structure is a complete graph. There are 3 commodities, and two possible realizations of demand, occurring with probabilities $p_1$ and $p_2$, described in Table 1.

Table 1: Demand values (in trailer-loads) for examples presented in Figures 2 and 3

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
<th>Realization 1 ($p_1 = 0.6$)</th>
<th>Realization 2 ($p_2 = 0.4$)</th>
<th>Expected demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$j$</td>
<td>2.00</td>
<td>1.00</td>
<td>1.60</td>
</tr>
<tr>
<td>$k$</td>
<td>$i$</td>
<td>1.30</td>
<td>0.30</td>
<td>0.90</td>
</tr>
<tr>
<td>$k$</td>
<td>$j$</td>
<td>1.00</td>
<td>2.70</td>
<td>1.68</td>
</tr>
</tbody>
</table>

Let us first design the service network using the expected demand values with the aim of minimizing the cost of the scheduled trailers. Note that we are using a deterministic model in this case even though we are operating in a stochastic environment. Thus, the existence of outsourced trailers is not explicitly considered – even though they may be required when the uncertain demand realizes. Using the expected demand, we can obtain both a traditional load plan design (which we will denote as Determ-T) as well as a limited-flexibility design (which we will denote as Determ-L). The word ‘design’ here refers to decisions made about the scheduled trailers on an arc, and decisions about the load plan. For this particular instance, it turns out that the model chooses to offer the same scheduled services in both cases, namely two trailers on arc $(i,j)$, one trailer on arc $(k,j)$, and two trailers on arc $(k,i)$. Thus, the total cost of both solutions is 4.5. The difference in these two designs, however, is in the underlying load plan. The load plan in the Determ-T solution stipulates that: freight originating at terminal $i$ and destined for $j$ can only be sent to $j$ directly, freight originating at terminal $k$ and destined for $i$ can only be sent to $i$ directly, and freight originating at terminal $k$ and destined for $j$ can only be sent to $j$ directly. On the other hand the load plan in the Determ-L solution allows for freight originating at a terminal to transit through a second terminal on its way to its final destination.

In Figure 2, we show the solutions of both load plans under the two demand realizations. The solid, dashed, and dotted arcs are used to indicate the flows of commodities $(i,j)$, $(k,i)$, and $(k,j)$,
respectively. Numbers alongside each arc indicate the number of scheduled trailers on that arc. The plus sign is used on some arcs to indicate the need for a number of outsourced trailers. Numbers in between parentheses indicate total freight flows on the arcs. Thus, in the case of the Determ-T solution, we need an outsourced trailer on arc \((k,i)\) to meet the demand in Realization 1 (at an additional cost of 0.75), and an additional trailer on arc \((k,j)\) to meet the demand in Realization 2 (at an additional cost of 1.5). Therefore, the total expected cost of this solution is 5.55. In the case of the Determ-L design, we need an outsourced trailer on arc \((k,i)\) again to meet the demand in Realization 1, but this time we do not need any outsourced trailers to meet the demand in Realization 2 since commodity \((k,j)\) can now be split along the two paths from \(k\) to \(j\). Thus, the total expected cost of this design is 4.95.

![Figure 2: Deterministic solutions; (2a) shows the Determ-T solution in Realization 1; (2b) shows the Determ-L solution in Realization 1; (2c) shows the Determ-T solution in Realization 2; and (2d) shows the Determ-L solution in Realization 2.](image)

We now consider the use of a stochastic model to design the service network with the aim of minimizing the expected cost of operating trailers (both scheduled and outsourced) to service the demand. We only look at the case of allowing for a limited-flexibility load plan (which we denote as Stoch-L) and compare the performance of the Stoch-L solution with the Determ-L solution. For completeness, however, we show the total expected costs for all four cases including the case of a stochastic traditional load plan (i.e. Stoch-T) in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>5.55</td>
<td>5.05</td>
</tr>
<tr>
<td>Limited-flexibility</td>
<td>4.95</td>
<td>4.55</td>
</tr>
</tbody>
</table>

Figure 3 shows the Stoch-L solutions under both realizations. We can immediately observe that the cost of the scheduled trailers in this solution is 3.5, which is the less than the 4.5 of the
Determ-L solution. In other words, the stochastic design does not over-commit to scheduled trailers because it recognizes that the use of outsourced trailers and smarter on-the-day consolidation will be cheaper, on average, in servicing the demand. In Realization 1, we need an outsourced trailer on arc \((k, i)\) (at a cost of 0.75), and we need an outsourced trailer to service the demand in Realization 2 (on arc \((k, j)\) at a cost of 1.5). Therefore, the total expected cost of the Stoch-L solution is 4.55 (lower than the 4.95 of its deterministic counterpart).

![Figure 3: Stochastic solutions; (3a) shows the Stoch-L solution for Realization 1; (3b) shows the Stoch-L solution for Realization 2](image)

Finally, although in practice the word “alt” usually just refers to the alternative next terminal option, for the remainder of this paper, we will use the term “alt” to refer to any next terminal option, be it a primary or an alternative option, for the sake of brevity, and we will make no distinction between the two.

3 Literature Review

The \(p\)-alt problem we present in Section 4 is a form of service network design (SND); a class of problems that has been well-studied in the literature and is applicable to multiple transportation settings. In particular, this SND model integrates LTL load plan requirements into the design process. For an overall review of SND, we refer the reader to Crainic (2000) and Wieberneit (2008). Furthermore, Guastaroba et al. (2016) offers a brief review of the role of intermediate facilities in SND problems. In addition to trucking, this is a class of problems applicable to various transportation settings including maritime transportation, e.g., Christiansen et al. (2004), Agarwal and Ergun (2008), Reinhardt and Pisinger (2012), express shipment, e.g., Barnhart and Schneur (1996), Kim et al. (1999), Grünert and Sebastian (2000), Barnhart et al. (2002), Armacost et al. (2002, 2004), rail, e.g., Newton et al. (1998), Barnhart et al. (2000), Luelli et al. (2011), Zhu et al. (2014), and multi-modal transportation systems, e.g., Crainic and Rousseau (1986), Crainic and Roy (1988), Nozick and Morlok (1997), Yamada et al. (2009), Inghels et al. (2016). Generally, SND problems are formulated as Capacitated Multi-Commodity Network Design (CMND) problems (Gendron
et al. 1999) and are known to be intractable except for relatively small instances (Ghamlouche et al. 2003, Crainic et al. 2000, Bai et al. 2014).

Many exact and heuristic approaches have been proposed in the literature for this class of problems. In terms of exact solution methods, a number of papers study cuts for these problems, e.g., Magnanti and Wong (1984), Barahona (1996), Bienstock et al. (1998), Günlük (1999), Atamtürk (2002), Frangioni and Gendron (2009), Raack et al. (2011), Chouman et al. (2017). Some of the inequalities studied, including the cut inequalities we use in our solution approach (described in Section 5), are inequalities based on cutsets in the underlying graph, and heuristic procedures were proposed to separate for these cutset-based inequalities such as those found in Bienstock et al. (1998), Günlük (1999), Raack et al. (2011), Chouman et al. (2017). In addition to the exact methods, two common heuristic approaches frequently appearing in the literature are local search, e.g., Crainic et al. (2000), Ghamlouche et al. (2003, 2004), Erera et al. (2013), Lindsey et al. (2016), and slope scaling, e.g., Crainic et al. (2004), Jarrah et al. (2009), the latter of which we use in this work. Among the papers proposing heuristics, Crainic et al. (2000) presents a Tabu Search algorithm which searches a neighborhood using column generation and simplex-like pivoting moves in the space of the path flow variables. Ghamlouche et al. (2003) defines a neighborhood consisting of cycles which can be used to reroute the flows of multiple commodities, and then embeds this neighborhood in a Tabu Search algorithm. This was further combined with a path-relinking algorithm in Ghamlouche et al. (2004). On the other hand, slope scaling is a heuristic approach introduced in Kim and Pardalos (1999) that iteratively solves linear approximations of the original problem formulation, adjusting the costs in each iteration, in an attempt to arrive at good feasible solutions. Crainic et al. (2004) introduces a slope scaling algorithm which integrates a Lagrangian perturbation scheme with some metaheuristic elements.

A number of recent papers on SND study how demand uncertainty affects network designs, and develop heuristic approaches for these problems. Specifically, Lium et al. (2009) investigate the effects of demand uncertainty on network designs and the structural differences of these designs compared to their deterministic counterparts. They observe that consolidation is a natural byproduct that arises in designs to hedge against the uncertainty of demand. Crainic et al. (2011) and Hoff et al. (2010) develop metaheuristic solution approaches for the stochastic service network design problem. In search of network flexibility, Bai et al. (2014) propose a stochastic network design model that allows rerouting or rescheduling of vehicles in the second stage of the stochastic program. Finally, Wang and Wallace (2016), Sun et al. (2017), and Wang et al. (2018) study the quality of deterministic solutions and their “upgradeability” to solutions of the stochastic scheduled service network design problem.

Relating to SND for LTL carriers, load plan design for these carriers was first studied in Powell and Sheffi (1983) and later that was followed by Powell (1986), Powell and Sheffi (1989), and Powell and Koskosidis (1992). In these papers, the problem was defined and formulated as a mixed-integer program, and local improvement heuristics were suggested to solve large-scale instances for a large U.S. LTL carrier. These papers were the first to explicitly consider the in-tree load plan structures in the network designs. To model service requirements more accurately, Jarrah et al. (2009) present a service network design formulation defined on a time-expanded network, and a heuristic which combines slope scaling and column-generation (where the columns are the in-tree load plans) to arrive at high-quality solutions for large-scale real-life instances for a large carrier. More recently, Erera et al. (2013) investigate the cost savings generated by varying the load plan by day of week to increase the flexibility of designs, and note that such flexibility generates approximately 6.5%
in savings to the carrier. Moreover, Erera et al. (2013) and Lindsey et al. (2016) develop methods
to improve load plan designs through IP-based local search techniques which they use to generate
significant cost savings for a U.S. carrier.

There is also a number of papers dealing with load planning for time-definite freight delivery
common carriers. Time-definite carriers provide guaranteed door-to-door pickup and delivery
services for small shippers, typically publishing rates, routes and schedules for the general public.
They consolidate shipments and utilize load plans with an in-tree structure similar to that
of LTL carriers. Lin (2001) presents the (deterministic) freight routing problem for these carriers
along with two approaches to solve this problem, a Lagrangian relaxation approach and an implicit
evaluation algorithm with $\epsilon$-optimality (IE-$\epsilon$). Lin (2004) presents a two-stage stochastic load
planning problem for time-definite freight common carriers facing uncertain demand, which was
later expanded into a multi-stage model in Lin and Lin (2007). Their models (in particular, the
in-tree load planning requirement and the use of additional trailers as a recourse) are similar to
the traditional load plan variant of the model we present in this paper. However, their models
contained additional constraints relating to service commitment, facility handling capacity, and
trailer balancing. Furthermore, demand was assumed to be given by a finite discrete distribution,
whereas we work with continuous distributions. Both papers develop heuristics to solve their re-
spective models, and use them to compare costs of deterministic and stochastic solutions for small
instances selected as subsets of the network of a large Taiwanese carrier. They conclude that using
a stochastic model yields solutions with a lower expected operating cost compared to solutions of
deterministic models.

4 Problem Description

We formulate the stochastic $p$-alt problem as a two-stage stochastic CMND problem with integer
variables for service selection and an additional $p$-alt constraint. We use the boldface parameter
$p$ to represent a vector specifying the desired number of alts allowed for each terminal-destination
pair. This is a general form for this parameter as, in practice, the number of alts desired for a
terminal-destination pair may depend, for example, on the size of the terminal and the volume
moving through that terminal to the destination.

In the first stage, the aim is to design a service network with the desired $p$-alt structure. A design
specifies the number of scheduled trailers on each arc in the network, i.e., the installed capacity,
as well as a load plan with the desired alt structure, i.e., the potential sequences of terminals that
freight with a given origin and destination can follow. In the second stage, this design is used to
satisfy realized demand, possibly introducing outsourced trailers obtained at a higher cost. Hence,
the design variables (first stage variables) are independent of demand realizations, whereas the
variables relating to commodity flows and outsourced capacity are realization-dependent.

We define the problem on a carrier’s line-haul network which is comprised of EOL and BB
terminals. Only EOL terminals serve as origins or destinations in this model. This is without loss
of generality, as “EOL” copies of BB terminals can be added to the network if freight also originates
from or can be destined to certain BB terminals. Commodities in this problem represent shipments
that have the same OD pair, and associated with each is a certain quantity representing its demand.
The problem is to minimize the total expected cost which consists of the cost of scheduled trailers,
plus the expected cost of outsourced trailers. We assume here a homogeneous fleet of vehicles in
both stages of the problem, although the model can be easily extended to more general fleets.
Let $G = (\mathcal{N}, \mathcal{A})$ be a digraph that represents the terminal network of the carrier with $\mathcal{N} = \mathcal{B} \cup \mathcal{E}$, where $\mathcal{B}$ and $\mathcal{E}$ represent the sets of BB and EOL terminals, respectively. Define the set of commodities $\mathcal{K}$, as a subset of the set of all possible EOL pairs for which there might be demand, i.e. $\mathcal{K} \subseteq \{(o,d) : o, d \in \mathcal{E}, o \neq d\}$. Let $o_k, d_k \in \mathcal{E}$ denote the origin and ultimate destination for commodity $k$, respectively. For notational convenience, define $\mathcal{D}$ as the set of all EOL terminals that are ultimate destinations for at least one commodity, i.e. $\mathcal{D} = \{d_k : k \in \mathcal{K}\}$, and $\mathcal{K}(d)$ as the set of all commodities with ultimate destination $d$.

Let $p_{id}$ be the number of alts allowed in the load plan design for terminal $i$ and destination $d$. We define $c_{ij}$ as the cost of operating a scheduled trailer on arc $(i,j)$, and $\hat{c}_{ij} > c_{ij}$ as the cost of operating an outsourced trailer operated on that same arc. Define $Q$ to be the uniform trailer capacity (for both scheduled and outsourced trailers). Let $\Omega \subseteq \mathbb{R}^{|\mathcal{K}|}$ represent the set of random commodity demands that are possible for commodities in $\mathcal{K}$ (this set can be either discrete or continuous), with $\omega \in \Omega$ representing a particular realization. Finally, define $q^\omega_k$ as the quantity of commodity $k$ in realization $\omega$.

Our decision variables are as follows:

- $r_{ij}$ = number of scheduled trailers to operate on arc $(i,j) \in \mathcal{A}$,
- $z_{ij}^\omega$ = number of outsourced trailers to operate on arc $(i,j) \in \mathcal{A}$ in realization $\omega$,
- $x_{ijk}^\omega$ = flow on arc $(i,j) \in \mathcal{A}$ for commodity $k \in \mathcal{K}$ in realization $\omega$,
- $y_{ijd} = \begin{cases} 1, & \text{if commodities with destination } d \text{ are allowed to use arc } (i,j) \in \mathcal{A} \text{ as an alt,} \\ 0, & \text{otherwise.} \end{cases}$

### 4.1 Mathematical Formulation

The stochastic $p$-alt model is:

\[
\begin{align*}
\min & \sum_{(i,j) \in \mathcal{A}} c_{ij}r_{ij} + \mathbb{E}_\omega [Q(r, y, \omega)], \\
\text{s.t.} & \sum_{(i,j) \in \delta^+(i)} y_{ijd} = \min\{p_{id}, |\delta^+(i)|\}, & \forall d \in \mathcal{D}, i \in \mathcal{N}, i \neq d, \\
& y_{ijd} \in \{0, 1\}, & \forall d \in \mathcal{D}, (i,j) \in \mathcal{A}, \\
& r_{ij} \in \mathbb{Z}_+, & \forall (i,j) \in \mathcal{A},
\end{align*}
\]

where

\[
Q(r, y, \omega) = \min \{ \sum_{(i,j) \in \mathcal{A}} \hat{c}_{ij}z_{ij}^\omega : (z^\omega, x^\omega) \in \mathcal{P}(r, y, \omega) \}
\]
and

\[ P(r, y, \omega) = \{(z^\omega, x^\omega) : \sum_{k \in K} x^\omega_{ijk} \leq Q(r_{ij} + z^\omega_{ij}), \quad \forall (i, j) \in A, \] \tag{1f}

\[ \sum_{(i,j) \in \delta^+(i)} x^\omega_{ijk} - \sum_{(j,i) \in \delta^-(i)} x^\omega_{jik} = \begin{cases} q^\omega_k, & \text{if } i = o_k, \\ -q^\omega_k, & \text{if } i = d_k, \\ 0, & \text{otherwise}, \end{cases} \quad \forall i \in N, k \in K, \] \tag{1g}

\[ x^\omega_{ijk} \leq q^\omega_k y_{ijd}, \quad \forall d \in D, (i, j) \in A, k \in K(d), \] \tag{1h}

\[ x^\omega_{ijk} \geq 0, \quad \forall (i, j) \in A, k \in K, \] \tag{1i}

\[ z^\omega_{ij} \in \mathbb{Z}_+, \quad \forall (i, j) \in A. \] \tag{1j}

The first stage determines the design of the network, which includes determining the number of scheduled trailers to operate on each arc and choosing up to \( p_{id} \) alts for each terminal-destination pair. The objective function (1a) minimizes the total expected cost which consists of the cost for installing scheduled capacity as well as the cost of acquiring additional capacity. Constraints (1b) are the \( p \)-alt constraints that determine the number of alts allowed. Written in this form, the constraint ensures that each terminal-destination pair is assigned \( p \) alts for each commodity wherever possible, and as many alts as possible otherwise. Although there is no reason to select a value for \( p_{id} \) that is greater than the outdegree of a node \( i \), writing the constraint in this form allows us to conveniently refer to the traditional load plan model as a 1-alt model, and a limited-flexibility load plan as a 2-alt model, where 1 and 2 are the vectors of all ones and all twos, respectively.

In the second stage, we deal with realized demand. Given our first stage design decisions, we attempt to satisfy all demand using the scheduled capacity or, if necessary, with additional outsourced capacity. The objective function (1e) minimizes the cost of installing additional capacity measured in the cost of adding additional trailers. Constraints (1f) ensure that sufficient arc capacities are available to accommodate freight flows. Constraints (1g) are the standard flow balance constraints, and ensure that all demand is met. Constraints (1h) ensure that flows are compatible with the chosen load plan design by only allowing a commodity \( k \) to flow on an arc \((i, j)\) if that arc is chosen as an alt for terminal \( i \) and for destination \( d_k \). Note that we use the disaggregated form of these constraints, as they provide a tighter formulation and performed better in our computational experiments. Although this means many more constraints in our model, for the sizes of instances we consider, this was not a major concern.

It is worth noting that an important feature of our setup is that we only allow outsourced trailers to be operated on arcs that are alts for at least one destination. From a practical perspective, this restriction helps carriers plan in advance for the services that they expect they would need when demand realizes enabling them to negotiate better deals for obtaining extra capacity from independent owner-operators. Although this restriction is not explicitly enforced in the model as a constraint, the combination of the objective function (1e), constraints (1f) and (1h) achieves the desired outcome.

Although most models for finding a standard 1-alt load plan presented in the literature are path-based models, using an arc-based model is far more convenient when seeking to find a 2-alt load plan. Note that whereas a 1-alt structure implies a single path for an OD pair, a 2-alt structure allows many paths for an OD pair (with the exact number depending on the configuration of the alts).
To benchmark the cost savings of $2$-alt designs, we will make use of a similar stochastic model that does not impose any structure on the freight flows. We will refer to this model as the stochastic Infinite-Alt model. Note that for a given network, setting all the values of the vector $p$ to $\max_{i \in N}\{\delta^+(i)\}$ makes the alt constraint redundant. Therefore, the stochastic infinite-alt model represents a relaxation of the stochastic $p$-alt model and serves as a natural benchmark for performance. For the sake of completeness, the stochastic Infinite-Alt (IA) model is as follows:

$$\begin{align*}
\min & \sum_{(i,j) \in A} c_{ij} r_{ij} + E_{\omega}[Q(r, \omega)], \\
\text{s.t.} & (1d),
\end{align*}$$

where

$$Q(r, y, \omega) = \min \sum_{(i,j) \in A} \tilde{c}_{ij} z_{ij}^\omega,$$

$$\text{s.t.} (1f), (1g), (1i), \text{ and } (1j).$$

Note the omission of the $y$ variables in the above model.

5 Model Solution

Given that a two-stage stochastic CMND problem (with integer variables in both stages) is very difficult to solve, we employ Sample Average Approximation (SAA) (Kleywegt et al. 2002) as the overarching solution framework. A SAA method consists of a number of iterations, each requiring the solution of a deterministic sample problem. In our setting, the sample problem is a deterministic CMND problem. Because the sample problem is difficult to solve, we investigate the use of both exact and heuristic approaches.

The rest of this section is structured as follows. First, we describe the SAA framework that we employ. Then, we present the cut inequalities for the CMND problem and discuss the methods we use to separate for these inequalities. Finally, we outline the different exact and heuristic approaches we use to solve the sample problem in each iteration of the SAA framework.

5.1 Sample Average Approximation (SAA)

Given that the set of possible scenarios $\Omega$ is usually very large (and possibly infinite), SAA uses sampling to sidestep this issue. In SAA, the set $\Omega$ is replaced by $S$, a small set of randomly sampled scenarios of size $N << |\Omega|$, and the expectation $E_{\omega}[Q(r, y, \omega)]$ is approximated by the sample average obtained using these $N$ scenarios, $\frac{1}{N} \sum_{\omega \in S} Q(r, y, \omega)$. This results in the following deterministic model which we will refer to as the sample problem:

$$\begin{align*}
\min & \sum_{(i,j) \in A} c_{ij} r_{ij} + \frac{1}{N} \sum_{\omega \in S} \sum_{(i,j) \in A} \tilde{c}_{ij} z_{ij}^\omega, \\
\text{s.t.} & (1b) - (1d), (z^\omega, x^\omega) \in P(r, y, \omega), \forall \omega \in S.
\end{align*}$$

SAA repeats this process $M$ times, using $M$ different $N$-samples, and by solving the sample problem in each one, it obtains a set of $M$ candidate first-stage decisions (or designs), $(\hat{r}^m, \hat{y}^m), m =$
1, \ldots, M$. To get a better estimate of the recourse cost, each of these designs is evaluated on $N' \gg N$ sampled scenarios. The total cost of a design, $(\hat{r}^m, \hat{y}^m)$, can then be approximated by

$$
\sum_{(i,j) \in A} c_{ij} \hat{r}_{ij}^m + E_\omega [Q(\hat{r}^m, \hat{y}^m, \omega)] \approx \sum_{(i,j) \in A} c_{ij} \hat{r}_{ij}^m + \frac{1}{N'} \sum_{n=1}^{N'} Q(\hat{r}^m, \hat{y}^m, \omega^n).
$$

(4)

where $\omega^n, n = 1, \ldots, N'$, is the set of sampled scenarios for evaluation.

For completeness, we include here a description of a generic SAA procedure for a minimization problem (Kleywegt et al. 2002):

1. Select $M$, the number of iterations for the procedure. Select $N$, the sample size for the sample problems. Select $N'$, the sample size for the evaluation subproblems.

2. For $m = 1, \ldots, M$,

2.1 Generate $N$ scenarios of $\Omega$, and solve the resulting sample problem. Let $\hat{x}^m$ be the first-stage optimal solution of the sample problem (not to be confused with the flow variables particular to the $p$-alt problem), and $v_N^m$ be the sample problem’s objective value.

3. Calculate

$$
\bar{v}_N = \frac{1}{M} \sum_{m=1}^{M} v_N^m,
$$

a statistical estimate for a lower bound on the optimal value of the true problem, and its variance

$$
\sigma^2_{\bar{v}_N} = \frac{1}{(M-1)M} \sum_{m=1}^{M} (v_N^m - \bar{v}_N)^2.
$$

4. For any feasible first-stage solution $\hat{x}$, calculate a statistical estimator for an upper bound on the optimal value of the true problem, by generating $N'$ scenarios from $\Omega$ (independently from the $N$ used in the sample problem), then computing

$$
\hat{v}_{N'}(\hat{x}) = \frac{1}{N'} \sum_{n=1}^{N'} G(\hat{x}, \omega^n)
$$

where $G(\hat{x}, \omega^n)$ is the total objective value (first-stage and recourse) obtained by solving the evaluation subproblem for scenario $\omega^n$ and feasible solution $\hat{x}$. The variance of this estimator can be computed as

$$
\sigma^2_{\hat{v}_{N'}(\hat{x})} = \frac{1}{(N'-1)N'} \sum_{n=1}^{N'} (G(\hat{x}, \omega^n) - \hat{v}_{N'}(\hat{x}))^2.
$$

5. Select $\hat{x}^* \in \arg \min_{\hat{x}^m} \{ \hat{v}_{N'}(\hat{x}^m) \}$.

6. Calculate the estimate for the optimality gap

$$
\hat{v}_{N'}(\hat{x}^*) - \bar{v}_N,
$$

and its variance,

$$
\sigma^2_{\hat{v}_{N'}(\hat{x}^*)} + \sigma^2_{\bar{v}_N}.
$$
In some variants of SAA, an additional step is added to check if the optimality gap estimate given by Equation (5) is sufficiently small. If not, the entire procedure can be repeated, with one of $M, N, \text{or } N'$ increased (typically, $N$). For the stochastic $p$-alt model we presented, the pseudocode for the SAA framework we use is outlined in Appendix A.1.

5.2 Cut Inequalities

The cut inequalities (Bienstock et al. 1998) we present here will be used in some of the solution methods for solving the sample problem (discussed in the next subsection). We describe these inequalities in the context of the sample problem of the $p$-alt model given by (3).

These inequalities are generic cuts that can be used for many network flow models. They stipulate that for any given cut in the graph, the capacity crossing the cut should be enough to serve the demand crossing that cut. To adapt these inequalities to our problem setting, let $V$ be any cutset in $G$ and define $d^\omega(V, \bar{V})$ to be the total demand that has to traverse the cut defined by $V$ in scenario $\omega$, i.e., the demand of commodities whose origins and destinations are on different sides of the cut. Then, we write the cut inequalities for scenario $\omega$ as

$$\sum_{(i,j) \in \delta^+(V)} (r_{ij} + z_{ij}^\omega) \geq \frac{d^\omega(V, \bar{V})}{Q}, \quad \forall V \subset \mathcal{N}. \tag{7}$$

Using the integrality of the trailer variables in both stages of the problem, we can obtain a stronger version of this inequality by simply rounding up the right hand side of (7). This yields the set of inequalities:

$$\sum_{(i,j) \in \delta^+(V)} (r_{ij} + z_{ij}^\omega) \geq \left\lceil \frac{d^\omega(V, \bar{V})}{Q} \right\rceil, \quad \forall V \subset \mathcal{N}, \omega \in \mathcal{S}. \tag{8}$$

These inequalities are usually violated by LP relaxation solutions to multi-commodity network flow problems (Bienstock et al. 1998), and therefore, significantly strengthen the formulation. Note, however, that there is an exponential number of such inequalities, namely $\mathcal{O}(N \cdot 2^{|\mathcal{N}|})$. Therefore, enumerating all such inequalities is impractical. Furthermore, the separation problem for these inequalities is known to be NP-Hard (Bienstock et al. 1998), and separation heuristics are needed.

5.3 Solving the Sample Problem

In each iteration of the SAA method, a sample problem given by mixed-integer linear program (3) has to be solved. Although simpler than the original stochastic problem, it is still difficult to solve to optimality in a reasonable amount of time.

In this section, we discuss the approaches we have investigated for solving the sample problem. We have considered exact approaches, in which we retain the integrality of both stages of the sample problem, as well as heuristic approaches, in which we relax the integrality of the second-stage variables. Note that by retaining the integrality of the first-stage variables, we always obtain a feasible design. By relaxing the second-stage variables, we hope to make the problem easier, but still obtain good-quality designs. As we found the evaluation subproblems in the SAA algorithm to be relatively easy to solve, we solve them exactly in both approaches.

In all but one of the approaches we have investigated, we use cut inequalities strengthen the formulation (exact approaches) and/or to mitigate the effect of relaxing the second-stage variables.
(heuristic approaches). The inequalities are added either \textit{statically}, i.e., they are added up front, or \textit{dynamically}, i.e., they are added only when violated (during the solution process).

In the static \textsc{Select} approach, we add only a select few cut inequalities to the model. By exploiting the structure of the instances, we generate a small number of cuts that, hopefully, correspond to important cutsets. By generating the cuts upfront and by keeping the number of cuts small, most of the solution time can be spent exploring the search tree.

In our instances, an EOL terminal is connected to either one or two BB terminals, where an EOL terminal connected to two BB terminals is referred to as a \textit{multi-loading} EOL (a multi-loading EOL will have a primary BB terminal, taken to be the nearest BB terminal, and a secondary BB terminal).

To illustrate the types of cuts we add in this approach, consider the example in Figure 4. In this example, all the EOL terminals that connect to terminals B1 and B2 are shown. Note, in particular, that EOL terminal E1 is a multi-loading EOL since it is connected to two BB terminals, with B2 being its primary BB.

The first types of cuts we consider are the cut inequalities corresponding to EOL cutsets. However, for an EOL terminal that connects to only one BB terminal, the cutset contains only a single arc, and the flow on the arc will be the total demand originating from (or destined for) the EOL terminal. The solver should be able to deduce this immediately from the flow balance constraints (1g), and then use a combination of the capacity constraints (1f) and the integrality of the trailer variables to arrive at the cut inequality corresponding to this cutset. Therefore, for EOLs, we will only add the cut inequalities corresponding to multi-loading EOLs (in both directions). An example can be seen in Figure 4a.

The next type of cuts we consider are the cut inequalities corresponding to cutsets containing a BB terminal along with its EOLs. There are two options here: (1) adding all EOLs connected to that BB in the cutset, (2) adding only EOLs for which that BB terminal is a primary BB. Preliminary experiments on small instances showed that the latter outperformed the former in terms of average solution time for the sample problems. Therefore, we add cut inequalities corresponding to BB terminals along with connected EOLs that have that BB as their primary BB to the model (we add the cut inequality in both directions here as well). An example of this can be seen in Figure 4b.

In the static \textsc{Enum} approach, all cutsets of size less than or equal to a given maximum size, $M^{\text{Enum}}$, are generated and the cut inequalities corresponding to those cutsets as well as to their complements are added to the model. We are careful to only add inequalities corresponding to \textit{relevant} cuts, where relevant cuts are defined as those that separate at least one commodity’s origin from its destination.

In the dynamic \textsc{RandCut} approach, the separation heuristic randomly generates a prespecified number of cutsets, $R^{\text{MaxIter}}$, for each scenario and checks if they are violated. Specifically, the heuristic randomly selects nodes in the graph to be in the cutset with each node having probability $\frac{1}{2}$ of being selected. If the cut inequality corresponding to the resulting cutset or its complement is violated, then it is recorded along with its violation. When $R^{\text{MaxIter}}$ cutsets have been generated, the recorded cutsets for the scenario are sorted in non-increasing order of their violations and the $\gamma^{\text{RC}}$ most violated cut inequalities are added to the model. The pseudocode for \textsc{RandCut} is given in Appendix A.2.

In the dynamic \textsc{RGContraction} approach, a randomized greedy contraction heuristic is used to find violated cut inequalities. The heuristic combines and builds on ideas found in Bienstock.
et al. (1998), Günlük (1999), Raack et al. (2011) and Chouman et al. (2017). The heuristic is based on the observation that cut inequalities have a higher chance of being violated when the arcs in the cut have smaller slacks with respect to constraint (1f) and when they have smaller fractional parts with respect to the right hand side of that constraint, i.e., $(r_{ij} + z_{ij}^\omega) - \lfloor (r_{ij} + z_{ij}^\omega) \rfloor$.

The heuristic iteratively contract arcs of the graph whose slacks and/or fractional parts are large. What remains at the end of this contraction process should be a graph whose arcs have small slacks and/or fractional values, and cuts in that graph are hoped to have an increased chance of violating the cut inequalities. More specifically, arcs in the graph are sorted in non-increasing order of their slack values, and then by non-increasing order of their fractional part values. A GRASP-inspired selection procedure is used to identify the next arc in the sorted list to contract. Namely, we form a Restricted Candidate List (RCL) of a predetermined size, $l$, containing the first $l$ arcs in the sorted list, and we randomly select an arc to contract from that RCL. Our experiments have shown that there is benefit to be gained in using this selection procedure as opposed to always contracting the first arc in the list.

The contraction is terminated whenever the shrunk graph has a certain number of remaining nodes, $N_{\text{final}}$. All cuts on the shrunk graph are enumerated and checked for violation. Cutsets whose inequalities are violated are recorded (along with their violation), and this process is repeated for a pre-specified number of iterations, $\text{RGMaxIter}$, to increase the chance of finding violated cut inequalities. Finally, all collected cutsets are sorted in non-increasing order of their violations, and a pre-specified number of these, $\gamma_{\text{RG}}$, is added to the model. The pseudocode for RGContraction is given in Appendix A.3.

Note that when contracting an arc, $(u,v)$, if for some node $w \neq u,v$ both the arcs $(u,w)$ and $(v,w)$ (or $(w,u)$ and $(w,v)$) are in $\mathcal{G}$, then the slack of the new arc, $(u,w)$ (or $(u,w)$) say, is the sum of the slacks on the arcs $(u,w)$ and $(v,w)$ (or $(w,u)$ and $(w,v)$), and the fractional part of the new arc is the fractional part of the sum of the two fractional parts.

A final heuristic approach provides a different way of mitigating the effects of the relaxation of the second-stage variables, namely using dynamic slope scaling (Kim and Pardalos (1999)). Since

Figure 4: Illustration of selected cuts in the Exact-Select method; squares represent BB terminals and circles represent EOL terminals, and dashed lines indicate the cuts in each figure while the shaded nodes indicate nodes that are in the cutset; (4a) shows the cut corresponding to the multi-loading EOL E1; (4b) shows the cut corresponding to the BB terminal B1 and the EOL terminals for which it is a primary BB (note the exclusion of E1).
relaxing the second-stage variables under-approximates the cost in the objective function, here we compensate for that by introducing cost multipliers that will be iteratively adjusted using dynamic slope scaling. Let $\rho_{ij\omega}^t$ be the cost multiplier for arc $(i, j)$ in iteration $t$ and scenario $\omega$ of the slope scaling algorithm, and solve

$$
\min \sum_{(i,j) \in A} c_{ij}r_{ij} + \frac{1}{N} \sum_{\omega \in S} \sum_{(i,j) \in A} \rho_{ij\omega}^t z_{ij\omega}^\omega
$$

s.t. (1b) – (1d), ($z_\omega^\omega, x_\omega^\omega$) $\in P_{LP}(r, y, \omega)$, $\forall \omega \in S$

where $P_{LP}(r, y, \omega)$ denotes the LP relaxation of $P(r, y, \omega)$.

In each iteration of the slope scaling algorithm, if the stopping criteria are not met, we adjust the cost multipliers of each arc and each scenario as follows:

$$
\rho_{ij\omega}^{t+1} = \begin{cases} 
\frac{\hat{c}_{ij} \lceil \frac{z_{ij\omega}^\omega}{Q} \rceil}{\frac{z_{ij\omega}^\omega}{Q}}, & \text{if } \frac{z_{ij\omega}^\omega}{Q} > 0, \\
\rho_{ij\omega}^t, & \text{otherwise,}
\end{cases}
$$

where $z_{ij\omega}^\omega$ is the value of the continuous relaxation of the $z$ variable for arc $(i, j)$ in scenario $\omega$ from the solution of (9) in iteration $t$ of the slope scaling procedure. The slope scaling portion of the algorithm is terminated when two successive slope scaling iterations yield design solutions that have approximately the same costs.

To initialize slope scaling in the very first iteration of SAA (i.e. when $m = 1$), we set $\rho_{ij\omega}^0 = \frac{c_{ij}}{Q}$, $\forall (i, j), \omega$. However, for subsequent iterations of SAA ($m > 1$), we make use of the multipliers found at the end of the slope scaling procedure in iteration $m - 1$ of SAA. To do this, we can initialize the slope scaling multipliers as a convex combination of $\frac{c_{ij}}{Q}$ (the starting multiplier) and $m - 1$’s final multipliers. The Heuristic-SS algorithm is outlined in Appendix A.4.

In summary, we use the following solution approaches: Exact-Select, Exact-Enum, Exact-RandCut, Exact-RGContraction, Heuristic-Select, Heuristic-Enum, Heuristic-RandCut, Heuristic-RGContraction, and Heuristic-SS.

We note that it is possible to combine static and dynamic generation of cut inequalities, i.e., enumerating cutsets of cardinality less than or equal a certain size up front, but dynamically “extending” some or all of these cutsets to larger ones that might be violated by a current solution. Computational experiments with such a combined approach revealed only minor improvements.

## 6 Computational Study

The primary objectives of this computational study are to:

1. demonstrate the effectiveness of 2-alt load plans in handling demand uncertainty by comparing the expected cost of operating with such load plans with their 1-alt and Infinite-alt counterparts,

2. gain insights into how 1-alt and 2-alt load plans differ,

3. assess the value of using stochastic models instead of deterministic models when solving the $p$-alt model, and
4. select, from among the solution approaches presented in Section 5, an approach that increases our chances of finding close-to-optimal solutions to the stochastic $p$-alt problem to facilitate the previous objectives.

This section is organized as follows. We first describe the instances and parameters used in our experiments in Section 6.1. In Section 6.2, we demonstrate the effectiveness of adding the cut inequalities (8), and compare the performance of the exact approaches. In Section 6.3, we compare the performance of the heuristic approaches. The objective of these two comparisons is to select a solution approach to use in the final set of experiments where we analyze 2-alt designs. Finally, in Section 6.4, we use the chosen solution approach to conduct the main experiment in this section. In this experiment, we demonstrate the effectiveness of 2-alt designs in handling demand uncertainty by comparing their total expected costs with both 1-alt and Infinite-alt designs. The word design is used in this section to refer to a first-stage solution, i.e. decisions relating to the operation of scheduled trailers and the load plan structure. We also share some managerial insights related to 2-alt load plans.

All algorithms were implemented in Python. All experiments were run on an Intel i5 Quad-Core 2.6GHz computer with 8GB of RAM running Fedora 27, with Gurobi 7.5.2 used as the IP solver. In all cases, we set Gurobi’s MIPFocus parameter to focusing more on finding feasible solutions.

6.1 Description of Instances and Parameters

In this study, instances are created using an instance generator which handles generating a graph representing the line-haul network of the carrier, as well as the OD demand information. To create a line-haul network, the generator randomly generates coordinates in the plane corresponding to EOL and BB terminals. Care was taken to ensure that BB terminals were located in more central locations. We allow three types of arcs in the graphs which are classified by the type of terminals they connect:

1. EOL-BB arcs: These are arcs that connect an EOL terminal to a BB terminal and represent movement of freight from an EOL to a BB. For all EOLs, we include in the graphs the arc connecting that EOL to its nearest BB terminal.

   In addition, we check two simple conditions to determine whether or not an EOL should have an additional connection to its second nearest BB terminal, i.e., whether or not an EOL is a multi-loading EOL. Specifically, for an EOL terminal, $E$, with terminals $B_1$ and $B_2$ as its nearest and second nearest BBs, respectively, we check the following conditions: (1) if the distance of the direct connection $E-B_2$ is less than 0.7 times the distance of transiting through the nearest BB terminal, i.e. $E-B_1-B_2$, and (2) if the distance of the direct connection $E-B_2$ is less than 1.5 times the distance to the nearest BB terminal, i.e. $E-B_1$. If both of these conditions are satisfied, then the arc representing the connection $E-B_2$ is added to the graph. Using these conditions, only a small fraction of EOLs become multi-loading EOLs, as is the case in practice.

2. BB-EOL arcs: These are arcs that connect a BB terminal to an EOL terminal, and represent movement of freight from a BB to an EOL. For these arcs, we use the same setup as the one used for EOL-BB arcs described above, i.e., all EOL-BB arcs have BB-EOL counterparts included in the graph.
3. BB-BB arcs: These are arcs that connect a BB terminal to another BB terminal. All such arcs were included in our graphs.

The cost, $c_{ij}$, of an arc $(i, j)$ in the graph was then set to be the Euclidean distance of that connection, and we set $\hat{c}_{ij} = 1.5 c_{ij}$ for all arcs $(i, j)$. Note that direct EOL-EOL connections are not included in our networks. This is justified as these connections rarely happen in practice, and even if they do, one can apply a pre-processing step that takes care of full (or nearly full) truckload shipments between these EOLs, and consider only the remaining partial truckload shipments in the model. An example of one of the networks generated and used in this study can be seen in Figure 5. This network is comprised of 12 EOL and 5 BB terminals. For completeness, we also include the expected demand data in an aggregated format in Table 3. For each EOL terminal, the 'Outgoing' column gives the sum of all the expected demand values (in trailer-loads) for the commodities that have that EOL as their origin, and the 'Destinations' column lists the EOL terminals that those commodities are destined for. Similarly, the 'Incoming' column shows the sum of all the expected demand values (in trailer-loads) for the commodities that have that EOL as their destination, and the 'Origins' column lists the EOL terminals that those commodities originated from.

![Figure 5: Example of LTL line-haul network generated by our instance generator; squares represent BB terminals while circles represent EOL connections](image)

To create demand data, the generator takes as input the desired percentage of EOL pairs that make up the set of commodities $\mathcal{K}$ in the model, i.e., the EOL pairs that have positive demand (recall that only EOL terminals can act as origins/destinations in our setup). It translates the percentage into a number of EOL pairs, rounding up if necessary, and then randomly picks that many EOL pairs to represent the commodities in the model. Thus, all EOL pairs that are not selected as part of the set $\mathcal{K}$ have zero demand in every generated scenario. We assume that the demand for each of these commodities follows a truncated normal distribution with a user-defined upper and lower limit, both taken to be the same for all commodities. Expected demand values for each of the commodities are randomly generated within that interval. Furthermore, by specifying a uniform
Table 3: Aggregated demand data for instance represented in Figure 5

<table>
<thead>
<tr>
<th>EOL</th>
<th>Outgoing Destinations</th>
<th>Incoming</th>
<th>Origins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.21</td>
<td>3,9,10</td>
<td>2.78</td>
</tr>
<tr>
<td>1</td>
<td>1.50</td>
<td>0,3,9</td>
<td>1.46</td>
</tr>
<tr>
<td>2</td>
<td>2.26</td>
<td>0,1,3,6</td>
<td>1.42</td>
</tr>
<tr>
<td>3</td>
<td>3.52</td>
<td>0,2,4,7,9,10,11</td>
<td>0.69</td>
</tr>
<tr>
<td>4</td>
<td>2.62</td>
<td>0,5,6,9</td>
<td>2.78</td>
</tr>
<tr>
<td>5</td>
<td>0.26</td>
<td>8</td>
<td>3.15</td>
</tr>
<tr>
<td>6</td>
<td>2.58</td>
<td>3,4,5,7,8,10</td>
<td>2.44</td>
</tr>
<tr>
<td>7</td>
<td>1.28</td>
<td>1,6</td>
<td>0.85</td>
</tr>
<tr>
<td>8</td>
<td>1.12</td>
<td>2,5,7,9,10</td>
<td>1.29</td>
</tr>
<tr>
<td>9</td>
<td>2.25</td>
<td>3,4,5,6,8</td>
<td>2.49</td>
</tr>
<tr>
<td>10</td>
<td>1.14</td>
<td>0,2,8</td>
<td>1.51</td>
</tr>
<tr>
<td>11</td>
<td>2.58</td>
<td>0,2,4,5</td>
<td>0.47</td>
</tr>
</tbody>
</table>

upper-limit on the standard deviation values, $\sigma$, standard deviations are randomly generated in the range $(0, \sigma)$ for each commodity. The user also supplies the value for the uniform trailer capacity, $Q$.

As for algorithm parameters, we set $M = 10$, $N = 10$, and $N' = 1000$ for the over-arching SAA algorithm and set a time limit of 60 minutes for the solution of a sample problem. (We experimented with other settings as well, e.g., $M = 20$ and a time limit of 30 minutes, but results did not improve.)

For the cut generation algorithms described in Section 5.3, we set the following parameter values:

- **Enum**: We set $M_{\text{Enum}}^E = 4$ (the size of cutsets we enumerate up front).

- **RandCut**: We set $RCMaxIter = 50$ and $\gamma_{\text{RC}} = 20$ for the root node, while we have $RCMaxIter = 15$ and $\gamma_{\text{RC}} = 10$ elsewhere.

- **RGContraction**: We set $RGMaxIter = 30$ and $\gamma_{\text{RG}} = 20$ for the root node, while we have $RGMaxIter = 15$ and $\gamma_{\text{RG}} = 10$ elsewhere. We also set $l = 3$ (the size of the RCL), and $N_{\text{final}}^{\text{RG}} = 3$.

We set $RGMaxIter$ to be lower than $RCMaxIter$ at the root node because its algorithm is much slower in execution compared to RandCut. We also note that in the case of the heuristic methods, the callback functions containing the separation routines are executed after every LP solve in a branch-and-bound node, as well as whenever a feasible solution to the relaxed model was found.

In all the experiments presented in this section, we solve all the evaluation subproblems in the SAA scheme exactly, using RandCut for generating cut inequalities (which was favored for its speed).

Table 4 provides the characteristics for the instances used in this study. The values of $Q$ and the expected demand and standard deviation are all expressed in trailer-loads. The instances are labeled as follows: no. of EOLs - no. of BBs - % OD density - $Q$ - $\sigma$. We note that the network in
Figure 5 corresponds to the networks used in Instances 12-5-35-1-0.2 and 12-5-35-1-0.6. Similarly, the data in Table 3 corresponds to the same two instances (as both instances have the same expected demands).

Table 4: Instance characteristics

<table>
<thead>
<tr>
<th>Instance</th>
<th>Line-haul Network</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No. of EOLs</td>
<td>No. of BBs</td>
<td>No. of arcs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-5-20-1-0.2</td>
<td>12</td>
<td>5</td>
<td>52</td>
<td>20</td>
<td>27</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.2)</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>12</td>
<td>5</td>
<td>52</td>
<td>35</td>
<td>47</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.2)</td>
</tr>
<tr>
<td>14-7-36-1-0.2</td>
<td>14</td>
<td>7</td>
<td>82</td>
<td>35</td>
<td>64</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.2)</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>12</td>
<td>5</td>
<td>52</td>
<td>20</td>
<td>27</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.6)</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>12</td>
<td>5</td>
<td>52</td>
<td>35</td>
<td>47</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.6)</td>
</tr>
<tr>
<td>14-7-36-1-0.6</td>
<td>14</td>
<td>7</td>
<td>82</td>
<td>35</td>
<td>64</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 0.6)</td>
</tr>
</tbody>
</table>

6.2 Comparison of exact approaches

The objective of this experiment is to compare the exact approaches Exact-Select (S), Exact-Enum (E), Exact-RandCut (RC), and Exact-RGContraction (RG). In addition, we include Exact-NoCuts (NC), where we solve the sample problems exactly but without the the cut inequalities, as a benchmark. Because we are primarily after good designs, we set the first stage variables to have a higher branching priority than their second stage counterparts. It is also important to note that the $N$-sample and the $N'$-sample generated for the $i^{th}$ sample problem in SAA were the same across all the different methods we consider.

We analyze the results of solving instances 12-5-20-1-0.2, 12-5-35-1-0.2, and 14-7-35-1-0.2 using both 1-alt and 2-alt models. Although we only consider three instances, because we need to solve $M = 10$ sample problems for each combination of instance and model, this gives us a total of 60 sample problems to base our comparison upon. Each of the sample problems for a particular instance-model-method combination was allotted a computational time of one hour (for a combined total of 10 hours). This time does not include the evaluation phases which, in the worst case, may take up an additional total of 2-3 hours.

In Table 5, we report the following statistics about each instance and each method (each number in the table represents an average over $M = 10$ sample problems).

- **IPGap**: the average percentage optimality gap at termination as reported by the solver for the sample problems.

- **GTB**: the average gap relative to the best overall solution found by any of the five methods measured using the total expected cost (first-stage cost plus recourse cost estimate obtained using the $N' = 1000$ scenarios), calculated as $100 \times \frac{\text{TotalCost} - \text{BestCost}}{\text{BestCost}}$.

- **TFD**: the average time to final design, which for a given sample problem is defined as the time from the start of the solution process until the last observed change in the first-stage cost.
Although the first-stage cost only reflects decisions about scheduled trailers and a design also includes decisions about the load plan (alt) structure, we use this cost as a proxy to check for design changes during the solve of a sample problem.

- No. B&B nodes: the average number of branch-and-bound nodes explored by the solver during the solve of a sample problem.

We also include a column containing the averages across all 1-alt and 2-alt instances (Averages), and we highlight the best values for that column in bold (except for the No. B&B nodes statistic).

From the IPGap values in Table 5, we readily observe that even for such relatively small networks with 17-21 terminals, these sample problems are difficult to solve to optimality. Among all 300 sample problems considered for all six combinations of instances and models, only five were solved to optimality. All of these five sample problems were in the 2-alt version of instance 12-5-20-1-0.2 and all were solved with the help of the cut inequalities. It is well-known that network design problems typically suffer from weak lower bounds which is a major contributor to their difficulty. We also observe that on average, the gaps at termination for the 2-alt models are less than their 1-alt counterparts. The RG method also stands out as the method achieving the best average IPGap on both 1-alt and 2-alt instances which suggests that this approach is an effective separation approach for the cut inequalities.

The best solution quality is consistently found by S on the 1-alt instances (GTB), while RG finds the best solutions, on average, in the 2-alt instances. Ultimately, there is a trade-off here between node exploration in the B&B search tree and spending time at each of those nodes separating for cut inequalities. For instance, by adding only some of the cut inequalities using knowledge of the instances, S purposefully spends less time looking for cut inequalities during the solution process, and is able to use that extra time to explore more of the search tree and find better solutions in case of the 1-alt instances. This is also corroborated by observing that the number of branch-and-bound nodes explored by S are much greater than that of RG and E. Furthermore, E adds the same cut inequalities that S adds (and many more), but this results in less exploration of the search tree, and an inability to match the solution quality found by S. In other words, adding too many cut inequalities can be a hindrance. To visually represent the trade-off between solution time and solution quality, we include in Appendix B plots showing the trade-off between solution quality and solution time (Figure 9).

We also observe a trade-off between a design’s quality and when that design is found in the solution process, represented by the TFD statistic. On average, RG takes the most time to find its final design on both sets of instances, while RC and S are the quickest on the 1-alt and 2-alt instances, respectively. A visual representation of this trade-off can be seen in Figure 10 in Appendix B.

Although there does not seem to be an exact method that clearly and consistently dominates on all metrics, S and RG outperform the other three. While we could choose either of those two as the best exact approach, we ultimately favor S due to its simplicity and its lower TFD values on both sets of instances.

6.3 Comparison of heuristic approaches

The objective of this experiment is to compare the heuristic approaches Heuristic-Select (S), Heuristic-Enum (E), Heuristic-RandCut (RC), Heuristic-RGContraction (RG), and Heuristic-SS (SS). In addition, we include Heuristic-NoCuts (NC), where we solve the relaxed sample problems.
Table 5: Statistics for exact methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Statistic</th>
<th>12-5-20-1-0.2</th>
<th>12-5-35-1-0.2</th>
<th>14-7-35-1-0.2</th>
<th>Averages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td>NoCuts (NC)</td>
<td>GTB (%)</td>
<td>0.00</td>
<td>0.03</td>
<td>0.19</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>IPGap (%)</td>
<td>3.79</td>
<td>2.94</td>
<td>4.67</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>137.43</td>
<td>124.97</td>
<td>707.16</td>
<td>744.46</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>50,318.90</td>
<td>284,927.60</td>
<td>13,841.50</td>
<td>62,945.60</td>
</tr>
<tr>
<td>RGContraction (RG)</td>
<td>GTB (%)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>IPGap (%)</td>
<td>3.37</td>
<td>0.26</td>
<td>4.67</td>
<td>1.20</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>1,066.15</td>
<td>263.05</td>
<td>1,408.66</td>
<td>1,518.35</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>4,518.00</td>
<td>7,353.90</td>
<td>3,894.60</td>
<td>6,078.60</td>
</tr>
<tr>
<td>RandCut (RC)</td>
<td>GTB (%)</td>
<td>0.00</td>
<td>0.08</td>
<td>0.09</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>IPGap (%)</td>
<td>3.11</td>
<td>0.30</td>
<td>4.34</td>
<td>1.60</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>303.67</td>
<td>140.84</td>
<td>535.58</td>
<td>873.63</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>34,670.40</td>
<td>173,433.00</td>
<td>10,956.80</td>
<td>53,883.10</td>
</tr>
<tr>
<td>Enum (E)</td>
<td>GTB (%)</td>
<td>0.54</td>
<td>0.19</td>
<td>0.20</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>IPGap (%)</td>
<td>4.33</td>
<td>3.18</td>
<td>4.61</td>
<td>1.96</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>346.77</td>
<td>125.34</td>
<td>1,109.39</td>
<td>796.11</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>24,612.80</td>
<td>203,683.10</td>
<td>7,126.20</td>
<td>66,492.90</td>
</tr>
<tr>
<td>Select (S)</td>
<td>GTB (%)</td>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>IPGap (%)</td>
<td>3.40</td>
<td>2.10</td>
<td>4.49</td>
<td>1.67</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>390.07</td>
<td>121.63</td>
<td>786.52</td>
<td>451.99</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>45,629.30</td>
<td>361,577.50</td>
<td>12,256.70</td>
<td>136,932.60</td>
</tr>
</tbody>
</table>
but without making use of the cut inequalities, as a benchmark. Again, both the \(N\)-sample and the \(N'\)-samples generated for the \(i^{th}\) sample problem were the same across all the different heuristic methods.

For NC, RC, E, and RG, we set a time limit of an hour for solving a sample problems. For SS, to solve a sample problem, we may have to perform several iterations of slope scaling. In each of these iterations, we solve a MIP given by (9), referred to as SSMIP, with a certain set of multipliers before adjusting the multipliers again using (10) until a stopping criterion is met. For this reason, we implement a timing scheme for SS that takes this into consideration. Specifically, this scheme consists of two components:

1. The total time limit for solving the SSMIPs of a single sample problem, set to one hour.

   We observed that the first few SSMIPs solve in little time, but subsequent SSMIPs are generally more difficult to solve, thereby causing SS to exhaust the total time limit having performed only a few slope scaling updates. However, we also observed that in most cases a high-quality (or optimal) solution to each SSMIP is found early in the solution process, and that most of the time spent on the harder SSMIPs is likely spent proving optimality. Therefore, we implemented an individual time limit for each SSMIP.

2. An individual time limit for each SSMIP in a single sample problem solve, determined dynamically.

   Experiments with SS on smaller instances suggested that the average number of slope scaling iterations taken by a sample problem was around five. Therefore, we allocate the one hour among the individual SSMIPs as follows: the first SSMIP is allowed 12 minutes, the time taken by the first and second SSMIPs combined is not to exceed 24 minutes, and so on. Therefore, if a SSMIP of a sample problem finishes before exhausting its individual time limit, the remaining time is carried over to the next SSMIP for that sample problem.

   Additionally, in Heuristic-SS, to solve sample problem \(m > 1\) in the SAA process, we initialize the slope scaling multipliers for iteration \(m\) using:

   \[
   \rho_{ij\omega}^m = 0.7 \rho_{ij\omega}^{m-1} + 0.3 \hat{c}_{ij} Q \forall i, j, \omega, \text{ where } \rho_{ij\omega}^{m-1} \text{ represents the final multipliers at the final SSMIP of sample problem } m - 1 \text{ for arc } (i, j) \text{ and scenario } \omega.
   \]

   Again, we analyze the results of solving instances 12-5-20-1-0.2, 12-5-35-1-0.2, and 14-7-35-1-0.2 using both 1-alt and 2-alt models. In Table 6, we show the results of these runs. In addition to GTB and TFD which were previously defined, we define the statistic RCP as the average (over 10 sample problems) percentage of the expected total cost that is comprised by the recourse cost obtained from the evaluation phase with 1,000 scenarios. In the Averages column in the table, we highlight the best values for GTB and TFD in bold. Furthermore, we refer the reader to Figures 9 and 10 in Appendix B for plots of the expected total cost vs. solution time, expected total cost vs. TFD, and a more in-depth discussion of the results.

   We readily observe from Table 6 that SS consistently finds the best solution in all six instances. The next best, in terms of solution quality, is RG, while the other three approaches find solutions that are much worse compared to SS and RG. While RC is the fastest in finding a design, the quality is not much better than not adding any cut inequalities. An effective separation routine is critical for the heuristics that use the cut inequalities, as we need to increase the likelihood of cutting off solutions found by the solver to the relaxed problem by finding violated cut inequalities. We see that RG does a much better job than RC generating effective cuts and mitigating the effects of relaxing the second-stage variables.
<table>
<thead>
<tr>
<th>Method</th>
<th>Statistic</th>
<th>Instances</th>
<th></th>
<th>Averages</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>12-5-20-1-0.2</td>
<td>12-5-35-1-0.2</td>
<td>14-7-35-1-0.2</td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
<td>2-alt</td>
<td>1-alt</td>
</tr>
<tr>
<td>NoCuts (NC)</td>
<td>GTB (%)</td>
<td>17.24</td>
<td>10.69</td>
<td>27.17</td>
<td>11.67</td>
<td>18.31</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>5.66</td>
<td>6.36</td>
<td>21.32</td>
<td>50.16</td>
<td>1531.12</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>43.90</td>
<td>34.60</td>
<td>49.50</td>
<td>36.80</td>
<td>40.70</td>
</tr>
<tr>
<td>RGContraction (RG)</td>
<td>GTB (%)</td>
<td>3.13</td>
<td>0.97</td>
<td>6.25</td>
<td>0.57</td>
<td>9.84</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>183.04</td>
<td>61.50</td>
<td>800.57</td>
<td>247.55</td>
<td>1879.59</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>14.40</td>
<td>6.20</td>
<td>15.70</td>
<td>6.10</td>
<td>25.50</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>13.11</td>
<td>8.49</td>
<td>53.05</td>
<td>9.17</td>
<td>247.05</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>39.00</td>
<td>31.90</td>
<td>46.80</td>
<td>37.60</td>
<td>40.90</td>
</tr>
<tr>
<td>Enum (E)</td>
<td>GTB (%)</td>
<td>12.57</td>
<td>6.95</td>
<td>14.80</td>
<td>5.29</td>
<td>16.78</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>46.72</td>
<td>15.03</td>
<td>154.13</td>
<td>68.64</td>
<td>1419.65</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>32.57</td>
<td>24.98</td>
<td>32.07</td>
<td>19.94</td>
<td>36.18</td>
</tr>
<tr>
<td>Select (S)</td>
<td>GTB (%)</td>
<td>16.86</td>
<td>9.73</td>
<td>15.28</td>
<td>8.98</td>
<td>17.57</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>25.52</td>
<td>10.20</td>
<td>115.72</td>
<td>22.86</td>
<td>999.41</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>40.07</td>
<td>33.14</td>
<td>37.58</td>
<td>31.47</td>
<td>39.22</td>
</tr>
<tr>
<td>Slope Scaling (SS)</td>
<td>GTB (%)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>1865.30</td>
<td>480.33</td>
<td>2552.28</td>
<td>1513.04</td>
<td>3175.66</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>6.70</td>
<td>3.90</td>
<td>4.10</td>
<td>3.00</td>
<td>5.70</td>
</tr>
</tbody>
</table>
Looking at the RCP values in Table 6, we can also make the observation that methods that find solutions with higher quality (e.g. SS) exhibit a lower RCP value, i.e., the percentage of the expected total cost that is made up by the recourse cost is very low. This is natural as relaxing the sample problems leads to an underestimation of the recourse cost in those problems, thereby favoring meeting more demand than usual using the second-stage additional capacity rather than the first-stage capacity, which, in turn, results in low-quality first-stage decisions. This observation is consistent with what we observed in solutions obtained via the exact methods (in which RCP ranges from 3% to 7%).

Because SS consistently finds the design with the best quality, we select SS as the best among the heuristic approaches.

6.4 Analysis of 2-alt designs

The primary objective of our experiments is to analyze 2-alt designs and highlight their potential. We accomplish this by comparing 2-alt designs to their 1-alt and Infinite-alt counterparts using a number of different metrics. A secondary objective of our experiments is to demonstrate the value of obtaining a design by using stochastic models rather than deterministic ones.

For the next experiment, and to facilitate making direct comparisons between designs, we generate 1,000 scenarios up front to represent all possible realizations of demand that can occur, and use these 1,000 scenarios exclusively for all the runs of the experiment. In particular, the \( N = 10 \) scenarios for the sample problems are now randomly sampled from these 1,000 scenarios, and all resulting designs are evaluated on the same 1,000 scenarios. Furthermore, we favored Exact-Select for this experiment over Heuristic-SS due its slight edge in terms of finding better designs within the time limit (Figure 9 in Appendix B). The results reported are based on the solutions to our six instances (see Table 4 for instances characteristics).

6.4.1 The benefits of 2-alt designs

In this section, we compare 1-alt, 2-alt, and Infinite-alt designs on a number of metrics to demonstrate the benefits that can be realized by adopting 2-alt designs. For each model and instance, we compute the following metrics associated with the chosen design:

1. **Cost**: This is the primary metric that we use to compare designs. The costs reported here for a design are the total expected costs, i.e. the cost of the first-stage decisions plus the expected recourse cost when evaluated over the 1,000 scenarios.

2. **Consolidation metrics**: LTL carriers are consolidation carriers, and hence, improving the consolidation of commodities/shipments naturally results in cost savings. We use consolidation metrics, therefore, to shed some light on where cost savings come from. As it is difficult to define a single consolidation metric that accurately captures the level of consolidation provided by a design, proxies are used (see Lium et al. (2009) and Wang et al. (2018) for similar approaches). Proxies generally fall in one of two categories: (1) metrics describing the level of *multipath usage* and (2) metrics describing the level of *path sharing*. Generally speaking, it is expected that if more commodities have multiple paths to get from their origins to their destinations, and if more commodities share arcs in their paths from their origins to their destinations, the higher the likelihood that freight is indeed consolidated on the services of the network. We use the following three metrics:
(a) **Commodities Consolidated:** For each design, we report the average number of commodities that are consolidated on an arc (taken across all 1,000 scenarios). This is done by computing, for each scenario, the number of commodities whose flow variables are positive on a particular arc, then averaging those values over the set of 1,000 scenarios, and then again over the set of all arcs in the network.

(b) **OD Paths:** For each design, we report the average number of paths in the load plan between origins and destinations. This is done by computing, for each commodity, the number of paths in the load plan that can be used to get from its origin to its destination. Since alt variables have no cost, we are careful here to only include in the load plans the alts selected by the solver if at least one commodity in a load plan uses that arc. Then, we average these numbers over the set of all commodities.

(c) **Utilization:** For each design, we report here the average percent utilization of trailers taken over all 1,000 scenarios. For each scenario $\omega$, we calculate its average percent utilization as follows:

$$u_{\text{scenario}} = 100 \left( \frac{1}{|A_{\text{used}}|} \sum_{(i,j) \in A_{\text{used}}} \frac{\sum_{k \in K} x_{ijk}^\omega}{r_{ij} + z_{ij}^\omega} \right),$$  

where $A_{\text{used}}$ here represents that set of arcs on which at least one trailer (regardless of its type) is operated. These values are then averaged over all 1,000 scenarios.

3. **% Alt Subset:** We use this metric specifically to compare 1-alt and 2-alt load plans. This metric is defined as the average percentage of arcs in the 1-alt load plan that are also part of the 2-alt load plan. We compute this percentage for each destination in the network, and then average the percentages across all destinations to arrive at the overall average percentage.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Model</th>
<th>1-alt</th>
<th>2-alt</th>
<th>Infinite-alt</th>
<th>Savings (%)</th>
<th>Gap closed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-5-20-1-0.2</td>
<td>849.47</td>
<td>798.97</td>
<td>797.37</td>
<td>5.94</td>
<td>96.91</td>
<td></td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>879.43</td>
<td>823.48</td>
<td>822.52</td>
<td>6.36</td>
<td>98.31</td>
<td></td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>1243.57</td>
<td>1174.60</td>
<td>1172.78</td>
<td>5.55</td>
<td>97.42</td>
<td></td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>1290.88</td>
<td>1215.10</td>
<td>1208.78</td>
<td>5.87</td>
<td>92.31</td>
<td></td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>1745.59</td>
<td>1605.16</td>
<td>1593.53</td>
<td>8.04</td>
<td>92.35</td>
<td></td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>1814.37</td>
<td>1658.49</td>
<td>1651.27</td>
<td>8.59</td>
<td>95.57</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>6.73</strong></td>
<td><strong>95.48</strong></td>
</tr>
</tbody>
</table>

Table 7 shows the total expected costs for all three models and all six instances. We use the Infinite-alt model here as a benchmark since it completely relaxes the restrictive load plan requirements. The “Savings” column represents the percentage savings obtained from using a 2-alt design over a 1-alt design. The “Gap closed” column shows the percentage of the gap between the 1-alt and Infinite-alt costs that the 2-alt model is able to close. Specifically, we calculate the gap
closed as follows:

$$100 \left( \frac{C^1 - C^2}{C^{\text{Inf}}} \right)$$

where $C^1$, $C^2$, and $C^{\text{Inf}}$ are the total expected cost values for the 1-alt, 2-alt, and Infinite-alt model, respectively.

Furthermore, in Table 8, we show the design costs for all pairs of instances and models. We also report the Design Cost Percentage (DCP), defined as the percentage of the total expected cost that is made up of the design cost.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Design Costs</th>
<th>DCP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
</tr>
<tr>
<td>12-5-20-1-0.2</td>
<td>800.31</td>
<td>764.54</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>800.31</td>
<td>761.92</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>1219.36</td>
<td>1137.07</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>1200.12</td>
<td>1120.38</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>1668.25</td>
<td>1505.16</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>1688.53</td>
<td>1483.78</td>
</tr>
<tr>
<td>Average</td>
<td>1229.48</td>
<td>1128.81</td>
</tr>
</tbody>
</table>

The first observation we make is that, for these instances, adopting a 2-alt design results in cost savings of almost 7%. In fact, for the largest two instances, the savings are more than 8%. This demonstrates the inherent restrictiveness of the 1-alt load plans, and that injecting a little bit of additional flexibility by adding a second alt is enough to generate substantial cost savings. For carriers, reducing their costs by 6-7% is a significant amount. For instance, for a large US LTL carrier, cost savings in the order of 6-7% translate roughly into $300,000 per week in savings (Lindsey et al. 2016). Furthermore, we note that all instances with higher variability (instances whose identifiers end in 0.6) exhibit higher savings than their low variability counterparts. This suggests that adopting 2-alt load plans in more uncertain environments is highly beneficial, and supports what these carriers are actually doing in practice by adding a second alt to some terminals and for some destinations. Of course, in practice, these carriers often solve deterministic models instead of stochastic ones to obtain 1-alt designs, but we still show that the practice of adding alts to cope with demand variations is, at the very least, a good idea.

A natural question to ask then is: “How much would the carrier save if it were to add a third alt to its load plans, then a fourth one, and so on?” We answer this by benchmarking the 2-alt models against the Infinite-alt models and observing the Gap closed column in Table 7. On average, adopting a 2-alt design closes 95.48% of the cost gap between the 1-alt and Infinite-alt designs which represent the two extremes. In other words, just adding a second alt to a traditional load plan yields cost savings that are very close to allowing all alts. While this result might be an artifact of the sizes and structure of the instances we use, we still think the benefits would exist if larger networks are studied (even though the percentage of the cost gap closed may be different from what we found). In fact, our findings are also consistent with what was observed in Erera et al. (2013) where the authors studied unrestricted load plans (these correspond to our
Infinite-alt models) for real-life instances and gathered statistics about how many outbound arcs were used for a terminal-destination pair. They report that the vast majority (roughly 90%) of terminal-destination pairs use only 1-2 outbound arcs, which is exactly what our 2-alt model captures.

This is an example of a situation where allowing a system some limited flexibility gives rise to benefits that are close to allowing full flexibility in the system. Another, famous example, is the “long-chain” concept presented in Jordan and Graves (1995) in the context of manufacturing flexibility. These surprising benefits of limited flexibility come as good news for carriers, since operating a full-flexibility load plan involves complex workflows and requires a high-level of automation. Carriers prefer traditional load plans for their operational simplicity, and what the results of these experiments show is that with only two alts allowed for each terminal and ultimate destination, the carrier can reap most of the benefits of full flexibility with only a fraction of the complexity.

When comparing the design costs and DCP values of the different models in Table 8, it appears that, on average, the 2-alt and Infinite-alt designs are comparable, while the 1-alt designs have higher design costs and DCP values. Therefore, we observe that 1-alt designs tend to invest more in scheduled trailers, whereas 2-alt and Infinite-alt designs rely more on outsourced trailers. This is, at least in part, due to the increase in consolidation options in the case of 2-alt load plans. This observation can potentially be exploited in designing a heuristic for the 2-alt model.

To shed some light on where these cost savings come from, we report in Table 9, the consolidation metrics described above for all three models and all six instances. Because these statistics can potentially be distorted by the presence of a large number of alternative optimal solutions, we implement a hierarchical multi-objective scheme in each of the evaluation subproblems to try and reduce the impact that this might have. Specifically, we initially solve the evaluation subproblem as described above and obtain its optimal total cost, $z^*$, and a vector of flow variables, $\mathbf{x}$. Next, using the vector $\mathbf{x}$ and for each commodity, we define a subgraph consisting of all the arcs that had positive flow for that commodity in the initial solve, and use that to define path-flow variables for all paths in this subgraph that connect its origin to its destination. We also define binary variables that indicate if this path is used or not. Then, using these variables, we resolve the model but now seeking a solution that minimizes the total number of paths that are used by all commodities subject to maintaining the value of $z^*$ found in the initial solve.

On average, the 2-alt designs consolidate a higher number of commodities and have a higher
utilization than the 1-alt designs, and the values are closer to those of the Infinite-alt designs than to those of the 1-alt designs. Another interesting observation is that the number of OD paths that a commodity can follow from origin to destination in the 2-alt case is an order of magnitude smaller than that of the Infinite-alt case. While we expect the number of paths in the 2-alt case to be more than one, the increase is small with only 2-3 more paths for a commodity on average. Furthermore, we note that instances with higher variability allow for the use of more OD paths per commodity in both the 2-alt and Infinite-alt model case, and therefore, the usage of multiple paths for each commodity appears to be a natural way to protect against demand volatility. A concrete example of this difference can be seen in the example shown in Figure 6, where we show the load plans for both the 1-alt and 2-alt models for destination terminal EOL05 in the instance of Figure 5. In this example, only terminals EOL04, EOL06, EOL08, EOL09, and EOL11 have freight that is destined for EOL05, and for the purpose of this discussion, we will refer to commodities by their origin terminals. In Figure 6, we observe the increase in the number of OD paths available for each of the commodities in the 2-alt load plan compared to its 1-alt counterpart. Whereas the 1-alt load plan only allows a single path for each OD pair, the 2-alt load plan takes advantage of the additional routing options to provide some additional flexibility. This allows for freight to be “shuffled around” the network to avoid congested links in certain demand realizations, thereby utilizing existing capacity that may be available elsewhere. For instance, if the scheduled capacity on the link (BB00, BB03) is used up in its entirety on a certain day (could even be used by other commodities not destined for EOL05), then freight EOL04 and EOL06 can be routed along the links (BB00, BB02) followed by (BB02, BB03), and, therefore, bypass the congested link.

![Figure 6: Example of load plan structures for the destination terminal EOL05 in Instance 12-5-35-1-0.2; the solid arcs represent the 1-alt load plan, and the dashed arcs together with the solid arcs represent the 2-alt load plan](image)

Finally, in Table 10, we compare the percentages of arcs in the 1-alt chosen design (averaged
Table 10: Percentage of arcs of 1-alt load plan that are in 2-alt load plan

<table>
<thead>
<tr>
<th>Instance</th>
<th>% Alt Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-5-20-1-0.2</td>
<td>98.57</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>95.83</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>92.15</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>97.48</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>89.90</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>100.00</td>
</tr>
<tr>
<td>Average</td>
<td>95.66</td>
</tr>
</tbody>
</table>

over all destinations) with those of their 2-alt counterparts, % Alt Subset. On average, a high percentage of the arcs that were chosen by the 1-alt model are chosen again as one of the two alts in the 2-alt model (approx. 96%). For example, the load plan for EOL05 shown in Figure 6 shows a case where all the 1-alt load plan arcs also appear in the 2-alt load plan for that terminal.

Since there may well be multiple optimal solutions with different choices for the alt variables, these percentages are not definitive (and could even be higher). However, this suggests that (good) 1-alt designs have the potential to be reasonable starting points and can be extended to good 2-alt designs. This information, combined with the observations above, can be used to design heuristics for the 2-alt model.

Overall, these results indicate that 2-alt designs are an attractive option when it comes to designing more flexible load plans. These designs capture most of the cost/consolidation benefits of the fully-flexible Infinite-alt load plans without the added complexity associated with operating such designs. Furthermore, we presented a number of observations that offer some managerial insights into why 2-alt designs perform so well, and how these designs harness these benefits.

6.4.2 The value of stochastic models

In this section, we present results that compare the use of a deterministic p-alt model (with expected demand values as input) to the use of a stochastic p-alt model. In the motivating example presented in Section 2.3, we presented a small example of an instance where solving a stochastic version of both the 1-alt and the 2-alt models provides better solutions than their deterministic equivalents. In the next experiment, we will solve a deterministic model using the expected demand for both the 1-alt and 2-alt models. Solving such a model is equivalent to solving a single sample problem (3) with a single scenario corresponding to the expected demand values. We again use the Exact-Select algorithm to solve this model, and we limit the solver to an hour of computational time. The design obtained is evaluated on the same 1,000 scenarios that were used to evaluate the stochastic model, and a total expected cost is calculated.

In Figure 7, we compare the total expected costs of the 1-alt deterministic and stochastic designs. On average, across the six instances, the stochastic 1-alt model yields a 3.6% savings over the deterministic model. The savings are more pronounced on the larger instances which yield a considerable 5-7% savings. In Figure 8, we compare the total expected costs of the 2-alt deterministic and stochastic designs. On average, across the six instances, the stochastic 2-alt model yields a 1.4% savings over the deterministic model - much less than that of the 1-alt model. This
suggests that, at least on these instances, a 2-alt deterministic model might be a good starting point for a heuristic for the stochastic 2-alt model, and may even be used in lieu of solving a stochastic program as it provided good solutions compared to the best solutions found using the stochastic model. We note, however, that despite the deterministic model containing a single scenario, we were only able to solve instances 1-4 to optimality within the one hour time limit. The computational time limit was exhausted by the two largest instances, and the optimality gap reported at termination for those two instances was 4-5%.

Therefore, these instances show that carriers can generate considerable savings by moving from solving a deterministic 1-alt model to a stochastic one, and then even more savings by allowing for two alts and solving a 2-alt stochastic model (for a combined savings of about 10%). We note that most of the models in the literature that have studied 1-alt (traditional) load plans for LTL carriers have been deterministic, so our findings here show the value of explicitly considering uncertainty in designing load plans for these carriers. The results also indicate that there may be potential for deterministic 2-alt models to be reasonable heuristics for stochastic models, or a good starting point for one.

7 Conclusion

We have introduced the \( p \)-alt model, a service network design model that enables LTL carriers to extend their traditional load plans by allowing for additional flexibility when faced with uncertain demands. While traditional load plans allow for only one option (or “alt”) for routing freight for each terminal-destination pair in the network, the \( p \)-alt model allows for up to \( p_{id} \) options for each pair of terminal \( i \) and destination \( d \). We have investigated a number of exact and heuristic approaches for solving the two-stage stochastic \( p \)-alt problem within the context of a Sample Average Approximation framework. The Exact-Select method has been chosen to carry out the final analysis...
of alts where we have studied the benefits of adding a single additional alt for terminal-destination pairs in the network, and compared the benefits of that limited form of flexibility with the benefits of allowing all options for each terminal-destination pair.

On the instances used, we have shown that adopting a design generated by a stochastic $2$-alt model yields 6-7% savings compared to its $1$-alt counterpart. Not only are the savings considerable, but just adding a single additional option at terminal-destination pairs closes about 95% of the cost gap between the $1$-alt model and the Infinite-alt model (which completely relaxes the load plan requirements). That is, the $2$-alt load plan designs offer benefits that are close to that of the Infinite-alt model, while maintaining a good degree of the operational simplicity that is characteristic of the $1$-alt load plan. These cost savings are generated by the added consolidation opportunities created by the presence of these second alts (indicated by an increase in the average number of commodities consolidated per arc, an increase in the number of OD paths per commodity, and an increased utilization rate).

We also show the value of using a stochastic $p$-alt model compared to a deterministic one. Our findings here indicate that adopting a stochastic $1$-alt model yields average savings in the order of 3-4% over a deterministic one. As many carriers currently adopt deterministic $1$-alt models, switching to stochastic $2$-alt models, therefore, yields a combined savings in the order of about 10%.

There are many future research directions which can be pursued to extend this work, primarily those related to solution approaches for the $p$-alt model. While we investigated a number of heuristic approaches to solve the stochastic $p$-alt model, we retained the integrality of the first-stage variables in doing so. Due to the difficulty of solving the stochastic $p$-alt model, heuristics that tackle the stochastic program directly are worth exploring. The managerial insights we share in this work can be used as a starting point for developing such heuristics. Another potential research direction is studying the polyhedral structure of the $p$-alt problem. Different formulations of this problem can be compared against one another, and cuts that take into consideration the inherent structure of the problem may be identified.

**Acknowledgement**

Ahmad Baubaid would like to acknowledge financial support from the King Fahd University of Petroleum & Minerals.

**References**


A Psuedocodes

A.1 SAA Algorithm for p-alt Problem

\begin{algorithm}
\caption{Sample Average Approximation Framework}
\textbf{Input:} $M, N, N'$. \Comment{Specifying algorithm parameters.}
\begin{algorithmic}[1]
\For{$m = 1, \ldots, M$}
\State Generate $S \subseteq \Omega$, a random demand $N$-sample for sample problem $m$.
\State Generate (independently) demand sample of size $N' \gg N$, for evaluation.
\State Solve sample problem $m$ given by (3) to obtain a first-stage design for iteration $m$,
\quad ($\hat{r}_m, \hat{y}_m$).
\EndFor
\For{$n = 1, \ldots, N'$}
\Comment{Evaluation phase.}
\State Solve second-stage subproblem (1e) given ($\hat{r}_m, \hat{y}_m$) to compute $Q(\hat{r}_m, \hat{y}_m, \omega^n)$.
\EndFor
\State Calculate $\hat{v}_{N'}(\hat{r}_m, \hat{y}_m) = \sum_{(i,j) \in A} c_{ij}\hat{r}_{ij} + \frac{1}{N'} \sum_{n=1}^{N'} Q(\hat{r}_m, \hat{y}_m, \omega^n)$, the approximate expected total cost for design ($\hat{r}_m, \hat{y}_m$).
\EndFor
\State Select $(\hat{r}^*, \hat{y}^*) \in \operatorname{arg\,min}_{m \in \{1, \ldots, M\}} \{\hat{v}_{N'}(\hat{r}_m, \hat{y}_m)\}$.
\State Compute the optimality gap estimate for $(\hat{r}^*, \hat{y}^*)$ and its variance using Equations (5) and (6).
\State \textbf{return} $(\hat{r}^*, \hat{y}^*)$, optimality gap estimate, and optimality gap variance estimate.
\end{algorithmic}
\end{algorithm}
A.2 RandCut Separation Heuristic

Algorithm 2 RandCut

\textbf{Input:} $G, r, z, RCMaxIter, \gamma^{RC}$. \hspace{1cm} $\triangleright$ Specifying algorithm parameters.

1: \hspace{0.5cm} \textbf{for} $\omega \in S$ \textbf{do} [\hspace{0.5cm} $L$ is the list of cutsets for which cut inequality is violated.]
2: \hspace{1cm} $L \leftarrow \emptyset$.
3: \hspace{1cm} \textbf{for} $t = 1, \ldots, RCMaxIter$ \textbf{do}
4: \hspace{1.5cm} $V \leftarrow \emptyset$.
5: \hspace{1.5cm} \textbf{for} $v \in \bar{V}$ \textbf{do}
6: \hspace{2cm} Choose a random byte $b \in \{0, 1\}$.
7: \hspace{2cm} \textbf{if} $b = 1$ \textbf{then}
8: \hspace{2.5cm} $V \leftarrow V \cup v$.
9: \hspace{2cm} \textbf{end if}
10: \hspace{1.5cm} \textbf{end for}
11: \hspace{1cm} \textbf{if} Equation (8) is violated for $V$ \textbf{then}
12: \hspace{1.5cm} $L \leftarrow L \cup V$.
13: \hspace{1.5cm} \textbf{end if}
14: \hspace{1cm} \textbf{if} Equation (8) is violated for $\bar{V}$ \textbf{then}
15: \hspace{1.5cm} $L \leftarrow L \cup \bar{V}$.
16: \hspace{1.5cm} \textbf{end if}
17: \hspace{1cm} \textbf{end for}
18: \hspace{0.5cm} Sort cutsets in $L$ in non-increasing order according to violation of Equation (8), and add the inequalities corresponding to the first $\gamma^{RC}$ cutsets to the model.

19: \hspace{0.5cm} \textbf{end for}
Algorithm 3 RGContraction

Input: \( G, r, \bar{x}, \bar{z}, RGM\text{MaxIter}, l, \gamma^{RG} \).  
▷ Specifying algorithm parameters.
1: for \( \omega \in S \) do
2: \( L \leftarrow \emptyset \).  
▷ \( L \) is the list of cutsets for which cut inequality is violated.
3: for \( (i, j) \in G \) do
4: Compute slack for \( (i, j) \) with respect to constraint (1f) using \( r, \bar{x} \) and \( \bar{z} \).
5: Compute fractional parts for arcs given by \( (r_{ij} + z_{ij}^\omega) - \lfloor (r_{ij} + z_{ij}^\omega) \rfloor \).
6: end for
7: for \( t = 1, \ldots, RGM\text{MaxIter} \) do
8: \( ArcList \leftarrow \) Sort arcs by their slacks in non-increasing order, then by their fractional parts in non-increasing order.
9: \( G' \leftarrow G \).  
▷ \( G' = (N', A') \) represents the contracted graph.
10: while \( ArcList \neq \emptyset \) and \( |N'| > N'_{final} \) do
11: \( RCL \leftarrow \) Select first \( l \) arcs in \( ArcList \).
12: \( (u, v) \leftarrow \) Select an arc randomly from \( RCL \).
13: Contract arc \((u, v)\) in \( G' \) while updating the slacks and fractional parts as described above.
14: Update \( ArcList \).
15: end while
16: \( V' \leftarrow \) Enumerate cutsets on \( G' \).
17: for \( V \in V' \) do
18: if Equation (8) is violated for \( V \) then
19: \( L \leftarrow L \cup V \).
20: end if
21: end for
22: end for
23: Sort cutsets in \( L \) in non-increasing order according to violation of Equation (8), and add the inequalities corresponding to the first \( \gamma^{RG} \) cutsets to the model.
24: end for
A.4 Heuristic-SS Algorithm

Algorithm 4 Heuristic-SS

1: $SSIterate \leftarrow True.$
2: Initialize multipliers: $\rho_{i,j,\omega}^0 = \frac{c_{ij}}{d} \quad \forall (i,j) \in G, \omega \in S.$
3: $t \leftarrow 0$ \hspace{1cm} $\triangleright$ Initialize iteration counter.
4: \hspace{1cm} while $SSIterate$ do
5: \hspace{2cm} Solve (9) with multipliers $\rho_{i,j,\omega}^t$ to obtain a feasible integer set of first-stage variables
6: \hspace{2cm} for iteration $t$ of slope scaling, $(r^t, y^t).$
7: \hspace{2cm} Update $SSIterate$ by checking stopping criteria.
8: \hspace{2cm} $\rho_{i,j,\omega}^{t+1} \leftarrow$ Adjust multipliers using (10).
9: \hspace{2cm} $t \leftarrow t + 1$ \hspace{1cm} $\triangleright$ Updating iteration counter.
10: \hspace{1cm} end while
11: $(\hat{r}^m, \hat{y}^m) \leftarrow (r^{t-1}, y^{t-1}).$
12: return $(\hat{r}^m, \hat{y}^m).$

B Comparison and Analysis of Solution Approaches

In this appendix, we present a more in-depth comparison of the exact and heuristics approaches discussed in Section 5. Specifically, we present a visual representations of the trade-off between solution quality and solution time, as well as the one between solution quality and the time to final design (TFD).

In Figure 9, we plot for each instance and each method the average total cost estimate taken over the $M = 10$ SAA iterations against the average time of an SAA iteration. We define the total cost estimate of a sample problem as the first-stage cost of that sample problem plus the recourse cost estimate obtained by evaluating the first-stage design using $N' = 1000$ scenarios. The time of an SAA iteration is defined as the time to solve the sample problem plus the total evaluation time of all the $N' = 1000$ scenarios for that sample problem. Each row of plots in the figure corresponds to a single instance, with the two plots in that row representing the different alt structures considered (1-alt and 2-alt).

In terms of exact approaches, we point out that in Figure 9 that there is very little variability in terms of these approaches in both solution time and solution quality. Due to the sizes of the instances considered and the fact that almost all the sample problems time out at the hour mark, this is not a surprising observation. In the case of Figure 9b, for instance, the variability in time is a result of two of the approaches, namely Exact-RandCut and Exact-RGContraction, solving some of their sample problems to optimality before the hour mark, whereas the other three exhausted the time limit without being able to do so. This suggests that the two separation heuristics RandCut and RGContraction are effective separation heuristics for the cut inequalities, and have helped in the solution process.

Figure 10 shows that there is slightly more variability in TFD among the five exact approaches. Generally, Exact-RGContraction is the slowest to settle on a final design, and this rarely comes with an increase in design quality. In addition, Exact-Select seems to have a relatively low TFD while obtaining designs of good quality (except perhaps in Figure 10f).

For the heuristic approaches, the first thing to note from Figure 9 is that there is much more vari-
Figure 9: Total Expected Cost vs. Average Time per SAA Iteration
Figure 10: Total Expected Cost vs. TFD
ability in terms of solution quality and runtime among the different heuristic approaches compared to their exact counterparts. Moreover the average runtime of the heuristic approaches indicates that these heuristics can be faster than the exact approaches while still finding solutions of reasonable quality. In fact, on these instances, some heuristic approaches (e.g. Heuristic-SS) are comparable in solution quality to the exact approaches.

To no surprise, we see that relaxing the integrality requirements for the second-stage variables without any method of compensating for that relaxation (NC) yields solutions that are usually much worse in quality compared to the exact approaches. In all six cases, Heuristic-NC is either the worst heuristic method in terms of solution quality or a very close second, and the difference in quality compared to the Heuristic-RG, Heuristic-E, Heuristic-S, and Heuristic-SS is significant. Furthermore, the algorithm Heuristic-RC doesn’t seem to offer much improvement on the quality of the solution obtained by the Heuristic-NC approach. On the other hand, both of these algorithms are generally the among the fastest to run (especially compared to Heuristic-RG and Heuristic-SS). However, on the largest instance they lose their edge in terms of speed. This is partially explained by noting that in both settings of the smaller two instances, the sample problems were easy to solve to optimality, thereby allowing Heuristic-NC and Heuristic-RC to terminate before exhausting the time limit. In the 14-7-35-1-0.2 instance, the sample problems are generally a lot more difficult to solve in the hour time limit, and even Heuristic-NC and Heuristic-RC exhaust this limit (as is the case with the other four) causing all six methods to have more comparable solution times for the sample problems. The differences seen in Figures 9e and 9f are due to time taken by the evaluation subproblems. In Figure 9f we see a curious case in which Heuristic-E and Heuristic-S take a much longer time compared to the other four. Specifically, the evaluation subproblems for those two approaches took roughly 4.5 times the total time taken by the evaluation phase of Heuristic-NC. An interesting thing to note is that Heuristic-SS is the approach that consistently finds the best solution among heuristic methods. Furthermore, this quality does not always necessarily come at the expense of more computational time; SS is competitive in terms of average time in Figures 9a–9d, but is actually the fastest heuristic method on the final two figures corresponding to the largest instance.

In Figure 10, we again notice reasonable variability in TFD between the heuristic methods. The interesting comparison is between Heuristic-RG and Heuristic-SS. Despite Heuristic-SS consistently finding the best quality solution, it takes longer to reach this final design. On the other hand, Heuristic-RG achieves a much smaller TFD value compared to Heuristic-SS, sometimes sacrificing a little bit of solution quality in the process. The biggest drop-off in solution quality occurs in the 1-alt version of Instance 14-7-35-1-0.2 (Figure 10e).