The Value of Limited Flexibility in Stochastic Load Planning and Service Network Design

Ahmad Baubaid$^{1,2}$, Natashia Boland$^1$, and Martin Savelsbergh$^1$

$^1$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

$^2$Department of Systems Engineering, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

March 4, 2019

Abstract

Less-than-truckload (LTL) 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 in each tactical operating season. This problem includes determining (1) the number of services (trailers) to operate between each pair of terminals, and (2) a load plan which dictates the sequence of transfer terminals that freight with a given origin and destination will follow. Traditionally, load plans stipulate that for every terminal-destination pair in the network, there exists only one next terminal option for routing freight at that terminal.

In this paper, we introduce the $p$-alt model that generalizes traditional load plans by allowing decision-makers to specify the desired number of next terminal options for every terminal-destination pair using a vector $p$. We compare a number of exact and heuristic approaches for solving the two-stage stochastic version of the $p$-alt model. Then, we use this model to 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 savings in cost in the order of 10% over traditional load plan designs obtained by deterministic models. Moreover, the cost savings are comparable to the ones yielded by adopting load plans that allow for any next terminal to be a routing option for terminal-destination pairs.
1 Introduction

Less-than-truckload (LTL) transportation 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 then consolidated onto outgoing trailers that transport that freight to its next stop in the network.

At the start of each operating season, these carriers must make many tactical decisions to plan for how that season’s demand is to be served. This tactical planning process is referred to in the literature as service network design [1, 2]. In particular, for LTL carriers, service network design problems address the following aspects of carriers’ operations: (1) deciding the number of services (trailers) to operate in the network, their frequencies, and their types/sizes, and (2) deciding the 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 $d$ is always loaded onto trailers destined for a unique next terminal, regardless of the origin of that freight. Because these load plans are devised with respect to predicted OD demands, some carriers recognize the benefit of having more flexible load plans to deal with the inherently uncertain demand. In particular, at some terminals, carriers might allow incoming freight destined for destination $d$ to be loaded onto trailers destined for two different next terminals. The choice of this additional routing option is determined heuristically as a subsequent step after obtaining the traditional load plan, and therefore, such decisions regarding additional flexibility are not integrated into the load plan design process. In practice, this additional routing option acts as a backup option if demand on the day is such that the routing option in the traditional load plan for that freight is congested. We present a formal discussion of these types of load plans in Section 2.

In this paper, we verify the benefits of explicitly designing more flexible load plans, and demonstrate that these plans are very 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 these carriers to have a load plan with limited flexibility that is only marginally more complex to operate than a traditional one but still manages to yield the benefits of a much more flexible form of load plan.

To demonstrate these advantages, we formulate a two-stage stochastic Capacitated Multi-
commodity Network Design (CMND) problem [3]. In the first stage, scheduled capacity is installed in the network by determining the number of trailers to operate between each pair of terminals, and the associated load plan is determined. The latter is accomplished by adding a parameter in the model which allows the planner to specify the level of flexibility desired in the load plan (where the flexibility is represented by the number of next terminals that a terminal can have for a particular destination). However, to meet realized daily demand, we allow the option of acquiring outsourced trailers in the second stage, if needed. Consequently, both stages of the model feature integer variables.

Due to the nature of the stochastic program we present, it is impractical to find optimal solutions for this model. Hence, to solve the stochastic CMND model, we employ a Sample Average Approximation (SAA) [4] framework. 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. This 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. Ultimately, the first-stage solution candidate with the lowest estimated expected total cost is selected as the final solution. Being CMND problems, the sample problems we obtain in each SAA iteration are known to be very difficult to solve (even if the sample problems contained just a single scenario) [5, 6, 7]. So, we compare a number of approaches (both exact and heuristic) for this problem to be used within the SAA framework to solve the resulting deterministic sample problem in each iteration. We also test and compare two different parameter settings for the SAA framework. After selecting the parameters for the SAA framework, and 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.

• We empirically show that load plans with limited flexibility 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).

• Showing 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).

• 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.

• 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 presents the model we will use to investigate the different variants of load plans we wish to compare. Section 5 outlines the solution approaches we used in an attempt to obtain solutions of good quality. Section 6 contains a computational study in which we discuss our selection of the final solution approach used, as well as compare the 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 cross-docking terminals: end-of-line (EOL) terminals, which are the terminals that serve only 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 (around 8 or 9 A.M.) to simultaneously 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 at the end of the day by 7 or 8 P.M.. 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. Typically, carriers have scheduled trailer movements between each pair of terminals in the line-haul network that they operate regardless of the day-to-day demand realizations.

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 [8]. 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
head for destination $d$ is sent to terminal $j$. Note that this structure also implies that freight with a given OD pair will follow a single 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.

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 be 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. In some situations, carriers designate more than one alt for terminal-destination pairs, but only one of those alts is used on a given day. 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 still 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 there would be significant technology costs associated with
implementing a full-flexibility load plan, limited-flexibility load plans are an attractive alternative that provide 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 realization of 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>
<thead>
<tr>
<th>Demand Realizations</th>
<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></td>
<td>(i)</td>
<td>(j)</td>
<td>2.00</td>
<td>1.00</td>
<td>1.60</td>
</tr>
<tr>
<td></td>
<td>(k)</td>
<td>(i)</td>
<td>1.30</td>
<td>0.30</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<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 offered 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 offered 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.

![Diagram](https://via.placeholder.com/150)

**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

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.

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 offering 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).

![Stochastic solutions diagram]

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

Table 2: Total expected cost for all cases considered in the motivating example

<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>

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 [1] and [2]. Furthermore, [9] 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., [10, 11, 12]), express shipment (e.g., [13, 14, 15, 16, 17, 18]), rail (e.g., [19, 20, 21, 22]), and multi-modal transportation systems.
(e.g., [23, 24, 25, 26, 27]). Generally, SND problems are formulated as Capacitated Multi-Commodity Network Design (CMND) problems [3] and are also known to be intractable except for relatively small instances [5, 6, 7].

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., [28, 29, 30, 31, 32, 33, 34, 35]). 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 [30, 31, 34, 35]. In addition to the exact methods, two common heuristic approaches frequently appearing in the literature are local search (e.g., [6, 5, 36, 37, 38]) and slope scaling (e.g., [39, 40]), the latter of which we use in this work. Among the papers proposing heuristics, [6] 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. [5] defines a new neighborhood consisting of cycles which can be used to reroute the flows of multiple commodities, and then embeds this in a Tabu Search algorithm. This was further combined with a path-relinking algorithm in [36]. On the other hand, slope scaling is a heuristic approach introduced in [41] 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. [39] introduces a slope scaling algorithm which integrates a Lagrangean perturbation scheme with some metaheuristic elements.

A number of recent papers on SND have studied how demand uncertainty affects network designs, and developed heuristic approaches for these problems. Specifically, [42] investigated the effects of demand uncertainty on network designs and the structural differences of these designs compared to their deterministic counterparts. They observed that consolidation is a natural byproduct that arises in designs to hedge against the uncertainty of demand. [43] and [44] develop metaheuristic solution approaches for the stochastic service network design problem. In search of network flexibility, [7] proposes a stochastic network design model that allows rerouting or rescheduling of vehicles in the second stage of the stochastic program. Finally, [45], [46], and [47] 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 [48] and later that was followed by [49], [50], and [8]. 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, [40] presents 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, [37] investigates 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,
[37] and [38] 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. [51] presents the (deterministic) freight routing problem for these carriers along with two approaches to solve this problem, a Lagrangian relaxation approach and an implicit enumeration algorithm with $\epsilon$-optimality (IE-$\epsilon$). [52] 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 [53]. 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 respective 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 concluded 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-valued variables for service selection, with the addition of a $p$-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 more general form for this parameter as, in practice, the number of alts desired for a terminal-destination pairs 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 network satisfying the desired $p$-alt structure. A design includes specifying the number of scheduled trailers on each arc in the network (i.e. installing capacity), as well as determining the associated load plan with the desired alt structure. This design is then used to satisfy the demand realized in the second stage, possibly through the use of outsourced trailers obtained at a higher cost. Hence, our design variables are independent of demand realizations, whereas the variables relating to the outsourced capacity and the commodity flows 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 at 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 $G = (N, A)$ be a digraph that represents the terminal network of the carrier with $N = B \cup E$, where $B$ and $E$ represent the sets of BB and EOL terminals, respectively. Define the set of commodities $K$, as a subset of the set of all possible EOL pairs for which there might be demand, i.e. $K \subseteq \{(o, d) : o, d \in E, o \neq d\}$. Let $o_k, d_k \in E$ denote the origin and ultimate destination for commodity $k$, respectively. For notational convenience, define $D$ as the set of all EOL terminals that are ultimate destinations for at least one commodity, i.e. $D = \{d_k : k \in K\}$, and $K(d)$ 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 similarly, $\hat{c}_{ij} (> c_{ij})$ as the cost of outsourcing a trailer operated on that same arc. Define $Q$ to be the uniform trailer capacity (for both scheduled and outsourced trailers). Let $\Omega \subseteq \mathbb{R}^{[|K|]}$ represent the set of random commodity demands that are possible for commodities in $K$ (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 $(i, j) \in A$, 
- $z^\omega_{ij}$ = number of outsourced trailers to operate on arc $(i, j) \in A$ in realization $\omega$, 
- $x^\omega_{ijk}$ = flow on arc $(i, j) \in A$ for commodity $k \in K$ in realization $\omega$, 
- $y_{ijd} = \begin{cases} 1, & \text{if commodities with destination } d \text{ are allowed to use arc (} i, j \text{) as an alt}, \\ 0, & \text{otherwise.} \end{cases}$

### 4.1 Mathematical Formulation

The stochastic $p$-alt model is therefore:

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

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

(1e)

and

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

\[
\sum_{(i,j) \in \delta^+(i)} x_{ijk}^\omega - \sum_{(j,i) \in \delta^-(i)} x_{ijk}^\omega = \begin{cases} 
q_k^\omega, & \text{if } i = o_k, \\
-q_k^\omega, & \text{if } i = d_k, \\
0, & \text{otherwise},
\end{cases} \quad \forall i \in \mathcal{N}, k \in \mathcal{K}, (1g)
\]

\[
x_{ijk}^\omega \leq q_k^\omega y_{ijd}, \quad \forall d \in \mathcal{D}, (i, j) \in \mathcal{A}, k \in \mathcal{K}(d), (1h)
\]

\[
x_{ijk}^\omega \geq 0, \quad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}, (1i)
\]

\[
z_{ij}^\omega \in \mathbb{Z}_+, \quad \forall (i, j) \in \mathcal{A}. (1j)
\]

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 \(p_{id}\) 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 as well as any additional capacity that might be required. The objective function (1e) minimizes the cost of installing additional capacity measured in the cost of adding additional trailers. Constraints (1f) ensure that the total arc capacities cannot be exceeded. 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 exhibited a tighter formulation 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 explic-
itly enforced in the model as a constraint, the combination of the objective function (1e), constraints (1f) and (1h) achieve the desired outcome.

Although most load planning models in the literature utilize path-based models to model freight flows when a standard 1-alt load plan is required, using an arc-based formulation is much easier to deal with when enforcing a 2-alt structure, whereas accomplishing this desired result in a path-based formulation might be cumbersome. Note that although a 1-alt structure implies a single path for every OD pair, a 2-alt structure can have any number of paths for every OD pair 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:

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

where:

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

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

Note the omission of the \( y \) variables in the above model.

5 Model Solution

Ideally, to conduct an computational study of the effects of an additional alt in facing demand uncertainty, the solutions used for the comparison should be optimal solutions. However, it is well-known that even deterministic CMND problems are very difficult to solve with modern commercial solvers except for small instances [5, 6, 7]. Consequently, finding optimal solutions to a two-stage mixed-integer stochastic program with potentially infinitely many scenarios and integer variables in both stages is hopeless even for relatively small instances. For this reason, we employ a Sample Average Approximation (SAA) [4] algorithm which serves as the overarching framework for our solution methodology. The SAA method consists of a number of iterations, each of which yields a deterministic sample problem which is also a CMND problem. Because these sample problems are still difficult to solve to optimality for the sizes of instances we consider, we compare solving these problems with both exact methods as well as two types of heuristic approaches:
1. Exact: solving the problem as is using a commercial solver with the help of the cut inequalities (described in Section 5.2.1). The cut inequalities are added in one of two ways:

   (a) Adding cut inequalities during the solution process using one of the separation algorithms described in Section 5.2.2 in a Branch & Cut algorithm, which yields the algorithm variants **Exact-RandCut**, **Exact-REnum**, and **Exact-RGContraction**.

   (b) Adding the cuts corresponding to select cutsets to the solver a priori (described in Section 5.3) which yields the algorithm **Exact-Select**.

2. Heuristic: relies on relaxing the integrality of the second stage variables, then we mitigate the effects of the relaxation in one of two ways:

   (a) Applying slope-scaling to the second-stage variables which yields the **Heuristic-SS** algorithm.

   (b) Adding cut inequalities during the solution process using one of the separation algorithms described in Section 5.2.2, which yields the algorithms **Heuristic-RandCut**, **Heuristic-REnum**, and **Heuristic-RGContraction**.

The rest of this section is structured as follows: we first describe the general SAA framework that we use. Next, we present the cut inequalities that we will make use of throughout the remainder of this paper, and discuss methods we used to separate for them in both the exact and heuristic approaches. Finally, we outline the different approaches we use as part of the SAA framework to solve the sample problem in each iteration of the procedure.

### 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 $E[Q(r, y, \omega)]$ 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:

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

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, in our case), $\hat{r}^m, \hat{y}^m, m = 1, \ldots, M$. 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

\[
\sum_{(i,j) \in A} c_{ij} \hat{r}_{ij} + \mathbb{E}_\omega [Q(\hat{r}^m, \hat{y}^m, \omega)] \approx \sum_{(i,j) \in A} c_{ij} \hat{r}_{ij} + \frac{1}{N'} \sum_{n=1}^{N'} Q(\hat{r}^m, \hat{y}^m, \omega^n). \tag{4}
\]

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) [4]:

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 \( v^m_N \) be the sample problem’s objective value.

3. Calculate

\[
\bar{v}_N = \frac{1}{M} \sum_{m=1}^{M} v^m_N,
\]

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

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

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 \( \hat{x}^* \in \arg \min_{m \in \{1, \ldots, M\}} \{ \hat{v}_{N'}(\hat{x}^m) \} \).
6. Calculate the estimate for the optimality gap
\[ \hat{v}_{N'}(\hat{x}^*) - \bar{v}_N, \tag{5} \]
and its variance,
\[ \sigma^2_{\hat{v}_{N'}(\hat{x}^*)} + \sigma^2_{\bar{v}_N}. \tag{6} \]

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 \( M, N, \) or \( N' \) increased (typically, \( N \)). For the stochastic \( p \)-alt model we presented, the SAA framework we use is outlined in Algorithm 1.

Algorithm 1 Sample Average Approximation Framework

Input: \( M, N, N' \).

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{for} \( m = 1, \ldots, M \) \textbf{do}
\State Generate \( S \subset \Omega \), 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 \( m \), \( (\hat{r}^m, \hat{y}^m) \).
\State \textbf{for} \( n = 1, \ldots, N' \) \textbf{do} \Comment{Evaluation phase.}
\State Solve second-stage subproblem (1e) given \( (\hat{r}^m, \hat{y}^m) \) to compute \( Q(\hat{r}^m, \hat{y}^m, \omega^n) \).
\State \textbf{end for}
\State Calculate \( \hat{v}_{N'}(\hat{r}^m, \hat{y}^m) = \sum_{(i,j) \in A} c_{ij} \hat{r}^m_{ij} + \frac{1}{N'} \sum_{n=1}^{N'} Q(\hat{r}^m, \hat{y}^m, \omega^n) \), the approximate expected total cost for design \( (\hat{r}^m, \hat{y}^m) \).
\State \textbf{end for}
\State Select \( (\hat{r}^*, \hat{y}^*) \in \arg\min_{m \in \{1, \ldots, M\}} \{ \hat{v}_{N'}(\hat{r}^m, \hat{y}^m) \} \).
\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}

5.2 Cut Inequalities

In this subsection, we introduce the cut inequalities [30], which we will use in some of the solution methods introduced in the next subsection. We also discuss some separation heuristics for these inequalities. These separation heuristics are used in the exact and heuristic solution approaches that we use for the sample problems. We describe these inequalities in the context of the sample problem of the \( p \)-alt model given by model (3).

5.2.1 Description of Cut Inequalities

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

These inequalities are usually violated by LP relaxation solutions to multi-commodity net-
work flow problems [30], and therefore, significantly strengthen the formulation. Note, how-
ever, that there is an exponential number of such inequalities, namely \( \mathcal{O}(N \cdot 2^{kN}) \). Conse-
quently, enumerating and using all possible inequalities becomes impractical as the size of
the problem grows. Furthermore, the separation problem for these inequalities is known to
be NP-Hard [30], and therefore, separation heuristics are needed.

5.2.2 Separation Heuristics for Cut Inequalities

We make use of three main separation heuristics for the cut inequalities. The separa-
tion heuristics work by generating a cut in the graph (or multiple cuts), from which the
cut inequalities are then generated. These heuristics are: (1) Randomly generating a cut
(RandCut) (2) Enumeration of cuts corresponding to cutsets up to a certain size up front,
then extending those cutsets by randomly adding nodes to the cutset (REnum), and (3)
Random Greedy Contraction (RGContraction), all of which are described below.

RandCut In this approach, we iterate over the scenarios, and randomly generate cutsets
from the graph for a pre-specified number of iterations, \( RCMaxIter \). For each scenario and
each iteration, we randomly select nodes in the graph to be in the cutset (each node has
a probability of selection equal to \( \frac{1}{2} \)). If the cut inequality corresponding to that cutset or
its complement is violated, then we record that violating cutset along with its magnitude
of violation. Finally, after \( RCMaxIter \) iterations, the collected violating cutsets for that
scenario are sorted in non-increasing order of their violations, and a pre-specified number,
\( \gamma_{RC} \), of the most violating inequalities is added to the model before moving on to the next
scenario. The pseudocode for RandCut is given in Algorithm 2.

REnum In this approach, we enumerate all cutsets up to a certain size, and pass the
inequalities corresponding to those cutsets to the solver \textit{a priori}. During the solution process,
we also use a randomized algorithm to extend some cutsets in an attempt to separate for violated inequalities corresponding to cutsets larger than the ones we have enumerated.

Specifically, we enumerate all the inequalities for all scenarios and all cutsets in the graph up to a certain cutset size, $M$. The inequalities corresponding to those cutsets as well as their complements are added to the solver a priori, thereby covering cutsets of sizes in the range $[1, M]$ and the range $[|\mathcal{N}| - M, M]$. We are careful to only add inequalities corresponding to relevant cuts, where relevant cuts are defined as those that separate at least one commodity’s origin from its destination, and where the cut arcs cross from the set containing the origin to the set containing the destination.

During the solution process, we carry out the extension phase for each scenario. The purpose of this phase is to find additional violated inequalities corresponding to cutsets whose size is in the range $(M, |\mathcal{N}| - M)$. We select a number of the cutsets of size $M$, and use a randomized algorithm to “extend” those cutsets to obtain larger supersets of them whose size is within that range. In particular, for each cutset $V$ of size $M$, we first calculate the following measure of violation:

$$
\alpha(V) = \max \left\{ \frac{d^\omega(V, \bar{V})}{Q} - \sum_{(i,j) \in \delta^+(V)} (r_{ij} + z^\omega_{ij}), \quad \frac{d^\omega(\bar{V}, V)}{Q} - \sum_{(i,j) \in \delta^+(V)} (r_{ij} + z^\omega_{ij}) \right\},
$$

(9)

where the values $r_{ij}$ and $z^\omega_{ij}$ represent the values of these variables at the current stage in the
optimization process. Note that this measure is defined as the maximum of the difference of the right and left hand side values of the cut inequality in either the cutset $V$ or its complement, and can be negative.

Cutsets of size $M$ are then sorted based on $\alpha(V)$ in non-increasing order, and we then select 5% of these cutsets to randomly extend. To extend a cutset, we randomly select nodes in $\bar{V}$ to add to $V$ (each node in $\bar{V}$ has probability $\frac{1}{2}$ to be selected), obtaining a new cutset $V_{new}$. If the size of $V_{new}$ is in the range $(M, |N| - M)$, the violations of $V_{new}$ and its complement are checked, and violating cutsets are recorded along with their violations. This is repeated for a pre-specified number of iterations, $REMaxIter$ before moving on to another cutset of size $M$. Finally, all collected violating cutsets are sorted in non-increasing order of their violations, and a pre-specified number of these, $\gamma^{RE}$, is added to the model. The pseudocode for the extension phase of REnum is given in Algorithm 3.

**Algorithm 3 REnum (Extension Phase)**

**Input:** $S, G, M, r, z, REMaxIter, M, \gamma^{RE}$. $\triangleright$ $M$ is a set that includes all cutsets of size $M$.

1: for $\omega \in S$ do
2: \hspace{1em} $\mathcal{L} \leftarrow \emptyset$. $\triangleright \mathcal{L}$ is the list of cutsets for which cut inequality is violated.
3: \hspace{1em} for $V \in M$ do
4: \hspace{2em} Compute $\alpha(V)$ using $r$ and $z$.
5: \hspace{1em} end for
6: \hspace{1em} $ExtensionCandidates \leftarrow$ Select the top 5% of cutsets in $M$ with the largest $\alpha(V)$.
7: \hspace{1em} for $V \in ExtensionCandidates$ do
8: \hspace{2em} for $t = 1, 2, \ldots, REMaxIter$ do
9: \hspace{3em} $V_{new} \leftarrow V$.
10: \hspace{3em} for $v \in \bar{V}$ do
11: \hspace{4em} Choose a random byte $b \in \{0, 1\}$.
12: \hspace{4em} if $b = 1$ then
13: \hspace{5em} $V_{new} \leftarrow V_{new} \cup v$.
14: \hspace{4em} end if
15: \hspace{3em} end for
16: \hspace{3em} if $|V_{new}| < |N| - M$ then
17: \hspace{4em} if Equation (8) is violated for $V_{new}$ then
18: \hspace{5em} $\mathcal{L} \leftarrow \mathcal{L} \cup V_{new}$.
19: \hspace{4em} end if
20: \hspace{3em} end if
21: \hspace{3em} end for
22: \hspace{1em} end for
23: \hspace{1em} Sort cutsets in $\mathcal{L}$ in non-increasing order according to violation of Equation (8), and add the inequalities corresponding to the first $\gamma^{RE}$ cutsets to the model.
24: end for
RGContraction  This approach uses a Randomized Greedy Contraction algorithm to find violated cut inequalities. This approach combines and builds on ideas found in [30, 31, 34, 35]. Cut inequalities have a higher chance of being violated when arcs in that cut have smaller slacks with respect to constraint (1f), as well as smaller fractional parts with respect to the right hand of that constraint, defined as \((r_{ij} + z_{ij}^\omega) - \lfloor(r_{ij} + z_{ij}^\omega)\rfloor\). The procedure described next is applied for each scenario in the sample problem.

The idea of this procedure is to 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. To accomplish this, during the solution process, arcs in the graph are sorted in non-increasing order of their slack values, and then by non-increasing order of fractional part values. We use a GRASP-inspired selection procedure to select the next arc to contract in the sorted list; 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 showed 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, \(N_{\text{final}}\), and all cuts on the shrunk graph are enumerated and checked for violations. Cutsets whose inequalities are violated are recorded (along with their violations), and this process is repeated for a pre-specified number of iterations, \(RGMaxIter\), 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, \(\gamma_{RG}\), is added to the model.

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 \((v,w)\)), and the fractional part of the new arc is the fractional part of the sum of the two fractional parts.

The pseudocode for RGContraction is given in Algorithm 4.

5.3 Solving the Sample Problem

In each iteration of the SAA method, we obtain a sample problem given by the mixed-integer linear program (3). Although this resulting MIP is undoubtedly simpler than the original stochastic problem, this is still a CMND problem (with the addition of the \(p\)-alt constraint), and hence is difficult to solve to optimality in reasonable time, even for relatively small sample sizes, \(N\). For this reason, our goal is to try to obtain high-quality solutions to the sample problems in a limited computational time. To that end, we compare the performances of both exact and heuristic approaches.
Algorithm 4 RGContraction

Input: $G, \mathcal{L}, x, z, RGMaxIter, l, \gamma_{\text{RG}}$. \hfill \triangleright \text{Specifying algorithm parameters.}

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

Three of the exact approaches are standard Branch & Cut algorithms each with one of the separation heuristics: Exact-REnum, Exact-RandCut and Exact-RGContraction. The fourth, Exact-Select will be described below. Furthermore, we propose two types of heuristics for this problem, both of which involve relaxing the integrality of the second-stage variables, thereby making the sample problems easier to tackle. Each heuristic uses a different technique to mitigate the effects of this relaxation. The first approach uses the cut inequalities described in Section 5.2.1 and their separation heuristics. This approach, therefore, consists of three algorithms but, in general, we will refer to this approach as Heuristic-Cut. The other approach is a dynamic slope scaling technique ([41]), which is an iterative approach to heuristically adjust the costs of the second stage variables in the sample problem in an attempt to get the recourse cost term in the sample problem to evaluate closer to the true recourse cost estimation from the evaluation subproblems (true here meaning that the evaluation subproblems in SAA have integer second-stage variables). We will refer to this approach as Heuristic-SS.
Note that by retaining the integrality of the first-stage variables, we obtain a feasible design consisting of trailer and alt variables at the end of the process. Because it is the design decisions that we are really after, the hope is that by relaxing the second-stage variables whose values are of lesser importance to us (except for purposes of evaluating the cost), we make the problem a bit easier to tackle, and still obtain good-quality designs.

As we found the evaluation subproblems in the SAA algorithm to be relatively easy to solve for the sizes of problems we consider, we solve the evaluation subproblem exactly in both approaches without any relaxation.

Since the algorithms Exact-RandCut, Exact-REnum, and Exact-RGContraction are standard Branch & Cut procedures, we focus this section on the Exact-Select algorithm and the two types of heuristic approaches (Heuristic-Cut and Heuristic-SS), and we describe each in turn.

5.3.1 Exact-Select

In this approach, we only add a select few of the cut inequalities to the solver a priori, and do not add any other inequalities during the solution process. The aim here was to exploit the structure of the instances, and add only a small number of cuts corresponding to important cutsets in an attempt to strengthen the formulation, all while allowing for the exploration of more Branch-and-Bound nodes as opposed to spending a lot of time looking for violated inequalities in the search tree.

In our instances, we only allow EOL terminals to connect directly to BBs in either direction, i.e. we do not allow EOL-to-EOL connections. In addition, although most EOL terminals connect to only one BB terminal, a few EOLs are allowed to connect to two BB terminals. We call these EOL terminals connecting to two BB terminals multi-loading EOLs. In this case, these multi-loading EOLs 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 method, 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 singleton EOL cutsets. However, for EOL terminals that connect to only one BB terminal, these cuts contain only a single arc, and the flow on these arcs will be the total demand originating from (or destined for) that EOL node. 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 cut. Therefore, for singleton EOLs, we will only add the cut inequalities corresponding to the singleton multi-loading EOLs (in both directions). An example of this can be seen in
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. Finally, with the addition of these cuts, the model is then solved exactly and without generating any more cut inequalities.

5.3.2 Heuristic-Cut

In this approach, we relax the integrality of the second-stage variables, but we add the cut inequalities (8) to the model. These inequalities are added to the solver during the solution process via callbacks, and using one of the separation heuristics described in Section 5.2.2. Thus, we have three Heuristic-Cut approaches: Heuristic-RandCut, Heuristic-REnum, and Heuristic-RGContraction. Because these inequalities are not valid for the relaxed model, they are added to the model as lazy constraints. We compare the performance of the different variants of Heuristic-Cut in Section 6.3.
5.3.3 Heuristic-SS

In this approach, we again relax the integrality of the second-stage variables in the sample problem. Relaxing these variables under-approximates the cost in the objective function, and to compensate for that, we introduce cost multipliers that will be iteratively adjusted using dynamic slope scaling. Let $\rho_{ij}^t$ be the cost multiplier for arc $(i,j)$ in iteration $t$ and scenario $\omega$ of the slope scaling algorithm, and solve

$$
\begin{align*}
\min & \sum_{(i,j) \in A} c_{ij} r_{ij} + \frac{1}{N} \sum_{\omega \in S} \sum_{(i,j) \in A} \rho_{ij}^t z_{ij}^\omega \\
\text{s.t.} & \ (1b) - (1d), \ (z_{ij}^\omega, x_{ij}^\omega) \in P_{LP}(r, y, \omega), \ \forall \omega \in S
\end{align*}
$$

(10)

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:

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

(11)

where $z_{ij}^\omega$ is the value of the continuous relaxation of the $z$ variable for arc $(i,j)$ in scenario $\omega$ from the solution of (10) 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 $\rho_{ij}^0 = \frac{\hat{c}_{ij}}{Q} \ \forall (i,j), \ \omega$. 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 $\frac{\hat{c}_{ij}}{Q}$ (the starting multiplier) and iteration $m - 1$’s final multipliers. The full Heuristic-SS algorithm – which replaces line 4 of Algorithm 1 – is outlined in Algorithm 5.

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,
**Algorithm 5** Heuristic-SS

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

3. Assess the value of using stochastic models instead of deterministic ones 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 (if not optimal) solutions to the stochastic $p$-alt problem to facilitate the first two 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: Exact-RandCut, Exact-RGContraction, Exact-REnum, and Exact-Select. In Section 6.3, we compare the performance of the heuristic approaches: Heuristic-RandCut, Heuristic-RGContraction, Heuristic-REnum, and Heuristic-SS. The objective of these two comparisons is to select one solution approach of each type to use in the next two sets of experiments. In Section 6.4, using the two chosen approaches, we study the effects of increasing the number of SAA iterations but reducing the computational time allowed for each one on finding better solutions. Finally, in Section 6.5, 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 by comparing 1-alt and 2-alt designs.

All algorithms described here 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 encourage Gurobi to focus 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 (this involved manual adjustment of the locations of some terminals). We allow three types of arcs in our 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 B1 and B2 as its nearest and second nearest BBs, respectively, we check the following conditions: (1) if the distance of the direct connection E-B2 is less than 0.7 times the distance of transiting through the nearest BB terminal, i.e., E-B1-B2, and (2) if the distance of the direct connection E-B2 is less than 1.5 times the distance to the nearest BB terminal, i.e., E-B1. If both of these conditions are satisfied, then the arc representing the connection E-B2 is added to the graph. We tested these conditions and their parameters to ensure that they give rise to sensible network structures, and that 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 an BB to an EOL. For these arcs, we use the same setup mentioned 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 \( \hat{c}_{ij} = 1.5 \ c_{ij} \ \forall (i,j) \). Note that direct EOL-EOL connections are not allowed 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 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

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 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 initially set $M = 10$, $N = 10$, and $N' = 1000$ for the over-arching SAA algorithm, although we will study the effect of increasing $M$ on finding better solutions in Section 6.4.

For the separation algorithms described in Section 5.2.2 (in both the exact and heuristic cases), we set the following parameter values:
Table 3: Aggregated demand data for instance represented in Figure 5

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

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

- **REnum**: We set $REMaxIter = 30$ and $\gamma^{RE} = 20$ for the root node, while we have $REMaxIter = 15$ and $\gamma^{RE} = 10$ elsewhere. We also set $M = 4$ (the size of cutsets we enumerate up front).

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

In all three approaches, we are more aggressive in finding and adding cuts at the root node. We set $REMaxIter$ and $RGMaxIter$ to be lower than $RCMaxIter$ at the root node because their corresponding separation algorithms are much slower to execute than 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 and using the RandCut algorithm 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>No. of EOLs</th>
<th>No. of BBs</th>
<th>No. of arcs</th>
<th>% OD density</th>
<th>No. of OD pairs</th>
<th>Q</th>
<th>Expected demand range</th>
<th>Demand std. dev. range</th>
</tr>
</thead>
<tbody>
<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 presented in Section 5 for solving the sample problems in each iteration of the SAA algorithm. We specifically compare the approaches Exact-RandCut (RC), Exact-REnum (RE), Exact-RGContraction (RG), and Exact-Select (ES). 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 seek good designs, and the second-stage variables are of lesser importance, 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, i.e. we allow each method a total of 10 hours of computational time for a method to find feasible designs for an instance-model combination. This time does not include the evaluation phases which, in the worst case, may take up an additional 2-3 hours in total. Since we are solving a design problem that is only solved once for every operating season, time is not that much of a concern, but we impose these limits in this work for practical reasons. 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 $100 \times \frac{\text{Total Cost} - \text{Best Cost}}{\text{Best Cost}}$.

- **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 the first-stage cost as a proxy to check for design changes during the solve of a sample problem. This measure is useful since in Section 6.4 we experiment with reducing the time allotted to each sample problem while increasing the number of sample problems solved, and hence favor methods that are able to find high-quality designs in a short amount of time so as not to be affected to a great degree by the reduction of computational time.

- **TTB**: the average time to best solution, defined as the time until best solution is found during the solve of a sample problem (this includes the second-stage cost in contrast to TFD).

- **PI**: the average primal integral for the sample problems. The primal integral measure was introduced and defined in [54] as a single value that quantifies the quality of the solutions that a primal solution method finds during the solution process as well as the time at which these solutions are found. Consider the primal gap of a MIP (measured with respect to a best known solution for that MIP) as a function of time, then the primal integral is defined to be the integral of that function. For this reason, a smaller primal integral is preferred since a smaller area under the curve would suggest that good solutions are found quicker in the solution process.

- **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 (networks in practice can have have 300+ terminals), these sample problems are difficult to solve to optimality. Indeed, 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, especially with the help of the cut inequalities (Exact-RandCut and Exact-RGContraction, in particular). The RGContraction 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. Moreover, we observe that the three methods NoCuts, RandCut, and Select have lower primal
<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>NoCuts (NC)</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>GTB (%)</td>
<td>0.00</td>
<td>0.03</td>
<td>0.19</td>
<td>0.03</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>TTB (s)</td>
<td>178.54</td>
<td>186.96</td>
<td>715.33</td>
<td>772.14</td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>0.82</td>
<td>1.42</td>
<td>5.47</td>
<td>7.55</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>IPGap (%)</td>
<td>3.37</td>
<td>0.26</td>
<td>4.67</td>
<td>1.20</td>
</tr>
<tr>
<td></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>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>TTB (s)</td>
<td>1,158.48</td>
<td>717.39</td>
<td>1,663.63</td>
<td>1,907.84</td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>6.55</td>
<td>3.44</td>
<td>14.95</td>
<td>22.92</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>IPGap (%)</td>
<td>3.11</td>
<td>0.30</td>
<td>4.34</td>
<td>1.60</td>
</tr>
<tr>
<td></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>TFD (s)</td>
<td>303.67</td>
<td>140.84</td>
<td>535.58</td>
<td>873.63</td>
</tr>
<tr>
<td></td>
<td>TTB (s)</td>
<td>561.91</td>
<td>181.94</td>
<td>541.29</td>
<td>893.35</td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>1.45</td>
<td>3.91</td>
<td>6.00</td>
<td>9.81</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>Renum (RE)</td>
<td>IPGap (%)</td>
<td>4.69</td>
<td>0.51</td>
<td>5.30</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>GTB (%)</td>
<td>0.02</td>
<td>0.07</td>
<td>0.19</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>1,460.80</td>
<td>938.97</td>
<td>1,652.34</td>
<td>2,094.40</td>
</tr>
<tr>
<td></td>
<td>TTB (s)</td>
<td>1,460.80</td>
<td>1,164.76</td>
<td>1,734.49</td>
<td>2,945.52</td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>26.42</td>
<td>94.55</td>
<td>36.44</td>
<td>222.52</td>
</tr>
<tr>
<td></td>
<td>No. B&amp;B nodes</td>
<td>658.40</td>
<td>673.50</td>
<td>464.40</td>
<td>486.80</td>
</tr>
<tr>
<td>Select (ES)</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>GTB (%)</td>
<td>0.00</td>
<td>0.06</td>
<td>0.00</td>
<td>0.07</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>TTB (s)</td>
<td>412.07</td>
<td>166.31</td>
<td>883.97</td>
<td>485.85</td>
</tr>
<tr>
<td></td>
<td>PI</td>
<td>2.35</td>
<td>2.87</td>
<td>7.64</td>
<td>9.98</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>
integral values compared to the other three. Since these methods purposefully spend less
time looking for cut inequalities (if any at all) during the solution process, they are able to
use that extra time to explore more of the search tree and find better solutions much faster
than the other two approaches, which results in a lower primal integral value. This is also
corroborated by observing that the number of branch-and-bound nodes explored (No. B&B
nodes) by NoCuts, RandCut, and Select are much greater than that of RGContraction and
REnum.

To visually understand the trade-off between solution time and solution quality, in Fig-
ure 6, 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. In
Figure 7, we plot for each instance and each method the average total cost estimate against
the average time to final design (TFD) taken over the $M = 10$ sample problems. Each row
of plots in Figures 6 and 7 corresponds to a single instance, with the two plots in that row
representing the different alt structures considered (1-alt and 2-alt).

From Figure 6, we see that there is very little variability in terms of these exact approaches
in both solution time and solution quality. Indeed, only in the plots for instance 14-7-35-1-0.2,
does one approach (REnum) seem to be considerably worse than the others in either solution
time (1-alt) or solution quality (2-alt). Due to both 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. The increase in runtime exhibited by REnum in Figure 6e is due to
an increase in the time spent in the evaluation phase for that instance. The deterioration
in solution quality exhibited by REnum in Figure 6f is due to REnum finding designs that,
on average, were 7% cheaper in first-stage cost compared to the average first-stage cost of
the other four methods on that instance. This translates to less scheduled trailer capacity
installed in the first-stage, which in turn, resulted in a considerable increase in the second-
stage cost in the evaluation phase compared to the other approaches, thereby causing the
overall expected total cost to be worse than the other exact approaches.

More important for our analysis in Section 6.4, however, is the trade-off between a design’s
quality and when that design found in the solution process, which is represented in Figure 7.
The figure shows more variability in TFD among the five exact approaches. Overall, REnum
is usually the worst approach in terms of TFD and sometimes in terms of design quality
(Figure 7f). On the 1-alt instances, the Select approach is consistently the best at finding
designs with the lowest expected total cost estimate. This can also be seen by looking at
the GTB statistic in Table 5. On the other hand, on the 2-alt instances, RGContraction is
the best, on average, in terms of design quality. Usually, however, this comes at the expense
of some additional time to obtain the final design. Table 5 shows that on average Select
achieves a better TFD, on average, on both 1-alt and 2-alt instances.
Although there does not seem to an exact method that clearly and consistently outperform the others on both metrics, it is clear that Exact-Select is a non-dominated approach in the 2-alt cases, i.e. no other method is strictly better than Exact-Select in both criteria. It also finds the best solutions on average in all three 1-alt problems, and is also non-dominated in two out of the three 1-alt instances (the exception being Figure 7a) where all methods except RE have the same expected total cost. However, even there it does not lose out too much in terms of TFD compared to RC and NC. In Exact-Select, we only include a small number of important of cut inequalities, and therefore, the method was designed to enable the solver to spend more time in the branch-and-bound search tree exploring new nodes rather than spending that time separating for cut inequalities, potentially finding better designs in the process – which seems to be the case. For these reasons, we will use the Exact-Select method for the upcoming experiment. in Section 6.4.

Overall, these results demonstrate the difficulty of solving these sample problems, and even though good solutions were found relatively early in the search process on the smaller instances, the larger instance suggests that exploring heuristics might be worthwhile.

### 6.3 Comparison of heuristic approaches

The objective of this experiment is to compare the heuristic approaches presented in Section 5 for solving the sample problems in each iteration of the SAA algorithm. We specifically compare the approaches Heuristic-RandCut (RC), Heuristic-REnum (RE), Heuristic-RGContraction (RG), and Heuristic-SS (SS). In addition, we include Heuristic-NoCuts (NC), where we solve the relaxed sample problems 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 the approaches Heuristic-NoCuts, Heuristic-RandCut, Heuristic-REnum, and Heuristic-RGContraction, we set an hour for the computational time limit for each of their sample problems. For Heuristic-SS, to solve each sample problem, we now have to possibly perform several iterations of slope scaling. In each of these iterations, we solve a MIP given by (10), hereby referred to as SSMIP, with a certain set of multipliers before adjusting the multipliers again using (11) until a stopping criterion is met. For this reason, we implement a timing scheme for Heuristic-SS that takes this difference into consideration. Specifically, this scheme consists of two components:

1. A total time limit for all the SSMIPs of a single sample problem: the total computational time that a single sample problem can take (with all of its SSMIPs) is set to an hour.

   However, with just this restriction, we observed that Heuristic-SS reaches this time limit after solving just two or three SSMIPs. Although the first one or two SSMIPs were typically easy to solve to optimality in little time, the SSMIPs encountered in subsequent slope scaling iterations were generally a lot more difficult to solve to opti-
Figure 6: Total Expected Cost vs. Average Time per SAA Iteration
Figure 7: Total Expected Cost vs. TFD
mality, thereby causing the heuristic to exhaust the time limit having performed only one or two slope scaling updates. However, we also observed that in most cases the incumbent solution of each SSMIP is found relatively early in the solution process, and that most of the time spent on these harder SSMIPs was likely spent to prove optimality. Because of this observation, we also implemented an individual time limit for each SSMIP.

2. An *individual* time limit for each SSMIP in a single sample problem solve: To encourage the heuristic to perform more slope scaling iterations, we set a time limit on each individual SSMIP in addition to the overall total time limit. Our observations from experimenting with Heuristic-SS on smaller instances suggested that the average number of slope scaling iterations taken by a sample problem was around five. We then used this to allocate the one hour time limit among the individual SSMIPs in a cumulative fashion: 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, that remaining time is carried over to the next SSMIP solved 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: $\rho_{ij\omega}^m = 0.7\rho_{ij\omega}^{m-1} + 0.3\hat{c}_{ij}^Q \forall i, j, \omega$, where $\rho_{ij\omega}^{m-1}$ represents the final multipliers at the final SSMIP of sample problem $m - 1$ for arc $(i, j)$ 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. We again refer the reader to Figure 6 which includes plots of the average expected total cost taken over the $M = 10$ sample problems against the average time per SAA iteration for these heuristic approaches. Figure 7 includes plots of the average expected total cost taken over the $M = 10$ sample problems against the time to final design (TFD) for these approaches. Furthermore, in Table 6 we show additional statistics about 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.

The first thing to note from Figure 6 is that there is much more variability in terms of solution quality and runtime among the different heuristic approaches compared to the exact approaches. Moreover, the average runtime of the heuristic approaches, particularly in Figures 6a–6d, indicate 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 performance 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,
<table>
<thead>
<tr>
<th>Method</th>
<th>Statistic</th>
<th>Instances</th>
<th>Averages</th>
<th></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></td>
<td></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>REnum (RE)</td>
<td>GTB (%)</td>
<td>2.93</td>
<td>0.97</td>
<td>8.14</td>
<td>0.55</td>
<td>12.11</td>
</tr>
<tr>
<td></td>
<td>TFD (s)</td>
<td>1368.17</td>
<td>598.29</td>
<td>1544.90</td>
<td>1198.09</td>
<td>2328.90</td>
</tr>
<tr>
<td></td>
<td>RCP (%)</td>
<td>14.20</td>
<td>6.20</td>
<td>17.80</td>
<td>5.90</td>
<td>27.30</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>
Heuristic-NoCuts 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-RGContraction, Heuristic-REnum, and Heuristic-SS is significant. Furthermore, the algorithm RC doesn’t seem to offer much improvement on the quality of the solution obtained by the NC approach. On the other hand, the NC and RC algorithms are generally the among the fastest to run when compared with the other three. This is easily seen in Figures 6a–6d. However, on the larger 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 these two methods to terminate before exhausting the time limit. In the larger 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-NoCuts and Heuristic-RandCut exhaust this limit (as is the case with the other three) causing all five methods to have more comparable solution times. Figure 6f is a curious case in which RC takes the longest time among the five heuristic methods. Upon inspection, we noticed that RC took an unusually long time in the evaluation phase – roughly 2.5 times the time taken by the evaluation phase of NC. It is not clear why RC takes this unusual amount of time for evaluation. On the other hand, SS is the method that consistently finds the best solutions in terms of expected total cost. This can also be seen in the GTB values of Table 6. This quality does not always necessarily come at the expense of more computational time; SS is competitive in terms of average time in Figures 6a–6d, but is actually the fastest method on the final two figures corresponding to the largest instance.

In Figure 7, we plot the expected total cost vs. TFD for the heuristic approaches. Again, we notice reasonable variability in TFD between these methods. Here, we notice that RE is dominated in five out of the six cases in the sense that there is at least one other approach that achieves a better total expected cost and/or a better TFD (the exception being the 2-alt version of Instance 12-5-35-1-0.2 where RE achieves a slightly less expected total cost compared to RG – as verified by Table 6). The interesting comparison is between RG and SS. Despite SS consistently finding the best quality solution, it takes longer to reach this final design. On the other hand, RG achieves a much smaller TFD value, 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 7e).

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. While four of the five methods attempt to mitigate the effects of the relaxation, it is clear that they vary in the degree to which they are successful in doing so. This observation related to RCP is also consistent with what we observed in solutions obtained via the exact methods (which range from 3% to 7%).

It is not clear a priori how Heuristic-RG and Heuristic-SS will fare if the computational
time allotted to each sample problem is reduced, which is part of the setup of our next experiment. Although the figures indicate that Heuristic-SS has a larger TFD value but achieves better solution quality, it is not clear what the quality of the design is, say, halfway through the solution process. In some cases (Figure 7e is a good example), the gap in solution quality between Heuristic-RG and Heuristic-SS is such that even reducing the computational time limit for the sample problems for Heuristic-SS might still yield designs with better quality than Heuristic-RG. In fact, in some early experiments we performed with Heuristic-SS, we allowed three times as much computational time for the Heuristic-SS sample problems compared to the other methods. The results showed that Heuristic-SS found designs with much better quality compared to these approaches, but even when reducing the computational time for the sample problems to an hour, Heuristic-SS still maintained its superiority in design quality. For that reason, we favored Heuristic-SS over Heuristic-RG and select is as our heuristic method of choice for the experiment presented in the next subsection.

6.4 Experimenting with the number of sample problems in SAA

The objective of this experiment is to study the effect of increasing the number of sample problems in the SAA scheme, $M$, on finding better designs to the 1-alt and 2-alt models. We use this experiment to select the algorithm and the SAA setup that we will use for the final experiments in which 2-alt designs are analyzed. Specifically, we will use the chosen exact method, Exact-Select (ES), and the chosen heuristic method, Heuristic-SS (SS), to compare two SAA setups, namely:

1. Setting $M = 10$ and allowing an hour of computational time for each sample problem (which was our setup in the experiments above).

2. Setting $M = 20$ and allowing half an hour of computational time for each sample problem.

For this experiment, and to facilitate making direct comparisons between designs, we generate 1,000 scenarios up front to represent all the possible realizations of demand that can occur, and use these 1,000 scenarios exclusively for all the runs of this 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. The first 10 sample problems in each approach have the same 10 sample sampled scenarios for their sample problems. Moreover, instance 14-7-35-1-0.2 was used for this experiment.

In Figure 8, we plot the chosen design’s total expected cost against the total runtime of each algorithm in seconds. We use the total runtime as the measure of time for this experiment because increasing the number of sample problems in SAA will lead to an increase in the number of evaluation phases, thereby increasing the overall runtime of the $M = 20$ SAA case (provided that all sample problems take the same amount time). Therefore, we wish to understand if this increase in runtime does come with noticeable advantages in
solution quality. We can clearly see the difference in runtime between the two cases in Figure 8. This is especially visible in the 2-alt case where, on average, the $M = 20$ runs take about 7,500 seconds (approx. 2 hours) more to run.

Furthermore, this increased runtime does not seem to produce designs that are of better quality than their $M = 10$ counterparts. In the 1-alt case, although going from $M = 10$ to $M = 20$ seemed to help Heuristic-SS achieve a slight improvement in solution quality, the Exact-Select method’s solution gets worse in quality when the number of sample problems is increased. In the 2-alt case, the solution quality of Exact-Select is about the same under both values of $M$, but the solution quality of Heuristic-SS deteriorates this time. This deterioration in solution quality is due to the decreased computational time for each sample problem. Interestingly enough, Heuristic-SS doing better with $M = 20$ on the 1-alt model is due to this same reason. In particular, when $M = 20$, Heuristic-SS finds the chosen design in the second sample problem, and none of the additional sample problems 11-20 find designs that are better than the chosen $M = 10$ design. By terminating the solution process of the second sample problem after half an hour as opposed to an hour, we obtain a resulting design that may not necessarily be a good one for the sampled scenarios (recall that these are the same for the first 10 sample problems for both the $M = 10$ and $M = 20$ cases), but turns out to be very a good design for the 1,000 scenarios that represent the set of all possible realizations. Overall, there doesn’t seem to be a clear benefit from increasing the number of sample problems from $M = 10$ to $M = 20$ (while halving the sample problem time limit), and so we elect to keep the $M = 10$ setup we initially started with.

What remains is to select a final method to use in this chosen SAA setup by comparing Exact-Select and Heuristic-SS. Looking at the values for $M = 10$ in Figure 8a, it is evident that Exact-Select outperforms Heuristic-SS in solution quality, and the increase in time is only about 2000s. In Figure 8b, for the 2-alt case, the two approaches exhibit very similar performance, but Heuristic-SS performs marginally better overall. Therefore, we choose Exact-Select to be the method we use to solve the $M = 10$ sample problems (each with an hour time limit) for analyzing 2-alt designs in the next subsection.

6.5 Analysis of 2-alt designs

The primary objective of these 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 these experiments is to demonstrate the value of obtaining a design by using stochastic models instead of deterministic ones.

For this experiment, and for each instance, we again generate 1,000 scenarios up front (different from the 1,000 scenarios used in Section 6.4), and we use that set to represent all possible demand realizations for that instance. We select $M = 10$ and we run all sample problems with a time limit of one hour. The results of this experiments come from running all six instances whose details we presented in Table 4.
6.5.1 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 from adopting 2-alt designs. For each model and instance, we compute the following metrics associated with the chosen design that is selected from the $M = 10$ iterations of the SAA algorithm:

1. Cost: This is the primary metric that we use to compare these designs. The costs reported here for each 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 measures: LTL carriers are consolidation carriers, and hence, improving the consolidation of commodities/shipments naturally results in more savings for these carriers. We use consolidation measures, therefore, to shed some light on where cost savings come from. As it is difficult to define a single consolidation measure that accurately captures the level of consolidation of a design, papers in the literature have used proxies to measure consolidation levels (see [42] and [47], for example). These proxies generally fall under 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 the more commodities that use multiple paths to get from their origins.
to their destinations, and the more that commodities share these paths, the higher the likelihood that freight is indeed consolidated on the services of the network. We use the following four metrics as consolidation measures:

(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. This is an example of a path sharing measure.

(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. Then, we average these numbers over the set of all commodities. This is an example of a multipath usage measure.

(c) **# Hops**: For each design, we report the average number of intermediate terminals that a commodity can transit through on its way to its final destination. This is done by computing, for each commodity, the average number of intermediate terminals that each commodity can transit through, then averaging those numbers over the set of all commodities. The greater the number of hops, the more handling a commodity requires on its way to its final destination. However, this also likely indicates that the commodity is more likely to undergo a number of rounds of consolidation on its journey. That is, as a commodity transits through more terminals, its chances of being combined with other commodities are increased.

(d) **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:

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

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 measure specifically to compare 1-alt and 2-alt load plans. This measure 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 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” column shows the percentage of the gap
### Table 7: Cost results

<table>
<thead>
<tr>
<th>Instance</th>
<th>Model</th>
<th>Savings (%)</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>Infinite-alt</td>
</tr>
<tr>
<td>12-5-20-1-0.2</td>
<td>849.47</td>
<td>798.97</td>
<td>797.37</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>879.43</td>
<td>823.48</td>
<td>822.52</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>1243.57</td>
<td>1174.60</td>
<td>1172.78</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>1290.88</td>
<td>1215.10</td>
<td>1208.78</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>1745.59</td>
<td>1605.16</td>
<td>1593.53</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>1814.37</td>
<td>1658.49</td>
<td>1651.27</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 8: Comparison of design costs

<table>
<thead>
<tr>
<th>Instance</th>
<th>Design Costs</th>
<th>DCP</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1-alt</td>
<td>2-alt</td>
<td>Infinite-alt</td>
<td>1-alt</td>
</tr>
<tr>
<td>12-5-20-1-0.2</td>
<td>800.31</td>
<td>764.54</td>
<td>762.89</td>
<td></td>
<td>94.21</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>800.31</td>
<td>761.92</td>
<td>761.92</td>
<td></td>
<td>91.00</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>1219.36</td>
<td>1137.07</td>
<td>1140.89</td>
<td></td>
<td>98.05</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>1200.12</td>
<td>1120.38</td>
<td>1108.63</td>
<td></td>
<td>92.97</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>1668.25</td>
<td>1505.16</td>
<td>1513.49</td>
<td></td>
<td>95.57</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>1688.53</td>
<td>1483.78</td>
<td>1489.46</td>
<td></td>
<td>93.06</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1229.48</td>
<td>1128.81</td>
<td>1129.55</td>
<td>94.15</td>
</tr>
</tbody>
</table>
between the 1-alt and Infinite-alt costs that the 2-alt model is able to close. Specifically, we calculate that gap as follows:

\[ \text{Gap} = 100 \left( \frac{C_1 - C_2}{C_1 - C_{\text{Inf}}} \right) \]

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 (or first-stage) costs for all pairs of instances and models. We also report in the table, the Design Cost Percentage (DCP), defined as the percentage of the total expected cost that is made up of the first-stage cost.

The first observation we make is that, on these instances, adopting a 2-alt design results in cost savings of 6.73%. In fact, on the largest two instances, the savings were in the order of 8-9%. 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 [38]. 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.

Therefore, a natural question to ask 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 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 much greater than the cost savings gained by further allowing more alts (including allowing all alts in the network). While this result might be an artifact of the sizes and structure of the instances we use (e.g. disallowing freight movement between EOL terminals), we still think the benefits would exist if larger networks are studied (although the percentage of the cost gap closed may be different from the one we report). In fact, our findings are also consistent with what was observed in [37] 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 used utilized 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 case where allowing a system some limited flexibility gives rise to benefits that are very close to allowing full flexibility in the system. This comes as good
news for carriers since operating an Infinite-alt model in practice will require big investments in support systems that can make real-time decisions about which alt (and therefore, which trailer) to choose for a particular shipment arriving at a terminal. On the other hand, carriers primarily prefer traditional load plans for their operational simplicity. What these instances show is that with only two alts allowed for each node and ultimate destination, the carrier can reap most of the benefits of the Infinite-alt model with only a fraction of the complexity associated with operating the network with a such a fully-flexible design. In fact, some carriers have already adapted their operations to the presence of these second alts at some terminals (as briefly described above).

When comparing the design costs of the different models and the DCP values in Table 8, it seems that, on average, the 2-alt and Infinite-alt designs are very comparable on both metrics, while the 1-alt designs are higher than the both of them. Therefore, we observe that 1-alt designs tend to invest more heavily in first-stage capacity by adding more scheduled trailers, whereas 2-alt and Infinite-alt designs make a little bit more use of the outsourced trailers depending on the demand realization. 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.

Table 9: Consolidation Statistics

<table>
<thead>
<tr>
<th>Instance</th>
<th># Comms. Consolidated</th>
<th># OD Paths</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>1.60</td>
<td>1.90</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>1.60</td>
<td>1.91</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>2.71</td>
<td>3.07</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>2.73</td>
<td>3.02</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>2.44</td>
<td>2.83</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>2.50</td>
<td>2.84</td>
</tr>
<tr>
<td>Average</td>
<td>2.26</td>
<td>2.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th># Hops</th>
<th>Utilization (%)</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>2.07</td>
<td>2.68</td>
</tr>
<tr>
<td>12-5-20-1-0.6</td>
<td>2.07</td>
<td>2.90</td>
</tr>
<tr>
<td>12-5-35-1-0.2</td>
<td>2.00</td>
<td>2.41</td>
</tr>
<tr>
<td>12-5-35-1-0.6</td>
<td>2.02</td>
<td>2.53</td>
</tr>
<tr>
<td>14-7-35-1-0.2</td>
<td>2.13</td>
<td>2.63</td>
</tr>
<tr>
<td>14-7-35-1-0.6</td>
<td>2.20</td>
<td>3.18</td>
</tr>
<tr>
<td>Average</td>
<td>2.08</td>
<td>2.72</td>
</tr>
</tbody>
</table>

To shed some light on where these cost savings come from, we report in Table 9, the consolidation measures described above for all three models and all six instances. On average,
the 2-alt designs manage to consolidate a similar number of commodities on their arcs as the Infinite-alt designs, and achieve a higher number compared to their 1-alt counterparts.

Another interesting observation from Table 9 is that the number of OD paths that a commodity can take from origin to destination in the 2-alt case is less by more than a factor of ten when compared with the Infinite-alt case. While we naturally expect the number of paths in the 2-alt case to be more than the one path that is implied by the 1-alt load plan structure, this increase allows for only 2-3 more paths for each 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 is a natural way to plan for demand volatility as more routing options are available (as discussed in [42], for example). To see a concrete example of this difference between 1-alt and 2-alt designs, consider the example shown in Figure 9 where we show the load plans for both the 1-alt and 2-alt models for destination terminal EOL05 in the same 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 purposes of this discussion, we will refer to commodities by their origin terminals. In Figure 9, 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 2-alt counterpart. Whereas the 1-alt load plan only allows a single path for each OD pair, thereby restricting its ability to provide multiple paths for the OD pairs, the 2-alt load plan is capable of taking advantage of the additional routing options to provide this form of 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 to use 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, would bypass the congested link.

In the # Hops columns, we also observe that the number of terminals a commodity transits through is increased in the 2-alt case when compared to its 1-alt counterpart. Again, the increase is not as drastic as the increase from the 1-alt case to its Infinite-alt counterpart. This suggests two things: (1) by increasing the number of hops, commodities have more opportunities for consolidation (since commodities are only consolidated at BB terminals), and (2) the commodities need to be handled more as they pass through more breakbulk terminals. Handling freight at terminals incurs cost, and increases the chances of freight damage, and therefore, carriers would greatly prefer to minimize freight handling. Luckily, the average number of hops in the 2-alt case is only about 30% higher than in the 1-alt case, which suggest that the increase in the number of times a commodity is handled is not an alarming one.

In the Utilization columns of the table, we note that the average utilization value of the 2-alt designs is not very far from that of the Infinite-alt designs (and is much higher than the 1-alt utilization value). Greater utilization of trailers for these carriers translates naturally into cost savings. So, this observation also corroborates the one made above where we noted
Figure 9: 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 that the total expected cost of the 2-alt designs are very close to that of their Infinite-alt counterparts.

Finally, in Table 10, we compare the percentages of arcs in the 1-alt chosen design (averaged over all destinations) with those of their 2-alt counterparts, % Alt Subset. On average, a very 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. 95%). For example, the load plan for EOL05 shown in Figure 9 shows a case where all the 1-alt load plan arcs also appear in the 2-alt load plan for that terminal.

We note there may very well be multiple optimal solutions that have different choices for the alt variables, especially as alt variables don’t incur any cost in the model, and therefore, these percentages are not definitive and can 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 a very 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 performs so well, and how these designs harness these benefits. These observations are useful to take into consideration when designing heuristics for this model.

### 6.5.2 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 very 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 this experiment, and for each instance, 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 then evaluated on the same 1,000 scenarios that were used to evaluate the stochastic model, and a total expected cost is calculated.

In Figure 10, 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 11, 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
Therefore, these instances show that carriers can generate considerable savings by moving from solving a deterministic 1-alt model to a stochastic one, 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

In this paper, we introduced the $p$-alt model which is 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 investigated a number of exact and heuristic approaches to solve the two-stage stochastic $p$-alt problem within the context of a Sample Average Approximation framework. The aim of this investigation was to select an approach that provided good quality solutions within the computational time limit we set for our experiment. The Exact-Select method was then chosen to carry out the final analysis of alts where we 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 our instances, we showed 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 for these carriers, but just adding that 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 very close to that of the Infinite-alt model, but still maintain a good degree of the operational simplicity that is a 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; an increase in the number of hops 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 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 and enrich this work. While we investigated a number of heuristic approaches to solve the stochastic p-alt model, we retained the integrability 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 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


