Optimization-based Learning for Dynamic Load Planning in Trucking Service Networks

Ritesh Ojha[a,*], Wenbo Chen[a,*], Hanyu Zhang[a], Reem Khir[b], Alan Erera[a], Pascal Van Hentenryck[a]

[a]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Tech, Atlanta, GA

[b]School of Industrial Engineering, Purdue University, West Lafayette, IN

{rojha8, wenbo.chen, hzhang747}@gatech.edu, {rkhir}@purdue.edu, {alan.erera, pvh}@isye.gatech.edu

Abstract

The load planning problem is a critical challenge in service network design for parcel carriers: it decides how many trailers (or loads), perhaps of different types, to assign for dispatch over time between pairs of terminals. Another key challenge is to determine a flow plan, which specifies how parcel volumes are assigned to planned loads. This paper considers the Dynamic Load Planning Problem (DLPP) that considers both flow and load planning challenges jointly in order to adjust loads and flows as the demand forecast changes over time before the day of operations. The paper aims at developing a decision-support tool to inform planners making these decisions at terminals across the network. The paper formulates the DLPP as a MIP and shows that it admits a large number of symmetries in a network where each commodity can be routed through primary and alternate paths. As a result, an optimization solver may return fundamentally different solutions to closely related problems (i.e., DLPPs with slightly different inputs), confusing planners and reducing trust in optimization. To remedy this limitation, the paper proposes a Goal-Directed Optimization (GDO) that eliminates those symmetries by generating optimal solutions staying close to a reference plan. The paper also proposes an optimization proxy to address the computational challenges of the optimization models. The proxy combines a machine learning model and a feasibility restoration model and finds solutions that satisfy real-time constraints imposed by planners-in-the-loop. An extensive computational study on industrial instances shows that the optimization proxy is around 10 *times faster than the commercial solver in obtaining the same quality solutions and orders of magnitude faster for generating solutions that are consistent with each other. The proposed approach also demonstrates the benefits of the DLPP for load consolidation, and the significant savings obtained from combining machine learning and optimization.*

1 Introduction

The e-commerce market continues to show robust growth and leading analysts project that today's $3.3 trillion market could grow further to $5.4 trillion annually by 2026 (Stanley, 2022). Much of e-commerce

[*]Co-first authors

relies on home delivery of small packages or parcels and other boxed freight. Key freight carriers like UPS and FedEx continually seek to redesign and operate profitable logistic networks that meet e-commerce customer service expectations. Beyond physical network design including the location and sizing of various freight processing terminals, these companies face challenging service network design problems. A critical service network design challenge for package carriers are the so-called *load planning* problems (for background, see Bakir et al., 2021). Here, load planning refers to decisions related to the number of trailers or container loads, perhaps of different types, to plan for dispatch over time between pairs of terminals. Such planned loads are the transportation capacity of the network. *Flow planning* decisions represent another key challenge, where the flow plan specifies how to allocate parcel volumes to planned loads to feasibly and cost-effectively serve network demand. As each package moves from its origin to destination, it is transported by a sequence of planned loads where it is unloaded and sorted at a transfer (hub) terminal between each loaded dispatch. Together, the flow and load plan decisions define a service network that moves package volume from origins to destinations in order to meet customer service expectations. The research described in this paper is conducted directly with a leading global parcel carrier that operates a massive network moving large volumes of packages each day. Figure 1 illustrates the load planning operations at an example terminal. It highlights the planner-in-the-loop environment in which load planning takes place; an important consideration underlying this research.

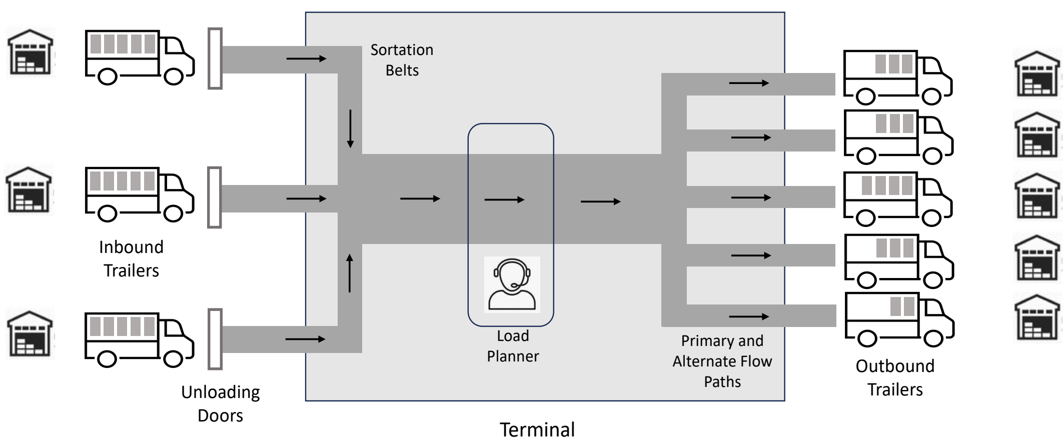


Figure 1: *An illustration of daily operations at a terminal. A load planner determines the number of trailers or loads to operate on each outbound path from the terminal and allocates packages to their primary or alternate flow paths.*

Packages at a terminal with the same destination and service class are referred to as a *commodity*. A *flow plan* defines flow rules for each commodity in the service network; these flow rules specify how a commodity is routed through the network over time. Since parcel carriers operate massive terminal networks with large numbers of transfer locations, a flow plan may include *alternate flow rules* that specify loading paths for commodities in addition to the default path specified by the *primary flow rules*. Both the primary (default) and alternate paths specify how a commodity moves through the network, and these planned paths are *service feasible*, i.e., they ensure that commodities arrive on time given their service guarantees.

This paper considers the *Dynamic Load Planning Problem* (DLPP) faced by the load planner at a terminal as depicted in Figure 1 during a short time period (one or two weeks) leading up to the day of operations. The goal of the planner, and thus of the DLPP, is to decide (1) how many loads should be planned for outbound dispatch to other terminals at various times during the day of operations and (2) how to allocate commodity volumes across planned loads respecting the capacity constraints and the primary and alternate flow rules. These two decisions define what is called a *load plan* in this paper. The objective of the DLPP is to obtain a load plan that minimizes the number of loads, consolidating the commodities as best as possible. In practice, the DLPP is solved by planners, who adjust existing load plans manually to reflect changes in commodity volumes arriving at the terminal. This process is typically *myopic* and creates inefficiencies across the network.

The goal of this research is to develop a decision support tool to assist planners in solving the DLPP, suggesting load plans that remove existing inefficiencies. Moreover, for terminals that do not have a planner, the tool can fully automate the DLPP, bridging the gap between network design and operations. To develop such a tool, this paper first investigates optimization models for the DLPP. In its general form, the DLPP is strongly NP-hard and its MIP formulation is challenging for state-of-the-art solvers given the size of the instances encountered in practice. Moreover, the natural MIP model exhibits significant symmetries which is highly undesirable for the planner-in-the-loop environment of the industrial partner. Indeed, planners will be extremely confused if small changes in commodities result in completely different load plans. To address this challenge, this paper presents a Goal-Directed Optimization (GDO) that solves a first model to find the optimal solution to the DLPP and uses a second model to find a plan that is as close as possible to a reference plan. GDO is shown to produce consistent plans, i.e., plans that are close for inputs that only differ slightly. Unfortunately, the GDO approach is too time-consuming to be used in planner-in-the-loop environments. To address this final difficulty, this research proposes the use of optimization proxies that combine a Machine-Learning (ML) model and a feasibility restoration procedure to obtain near-optimal solutions in a few seconds, even for the largest terminals. The ML model uses supervised learning to mimic the GDO approach and predicts the optimal set of planned loads. The feasibility restoration procedure then solves a small MIP model to determine the final allocation of commodity volumes to planned loads, adding extra capacity as needed to ensure feasibility. The proposed approach is practical since it produces high-quality plans that are consistent with each other, where small changes in inputs leads to very similar load plans by virtue of the ML training that mimics the GDO optimization.

The main contributions of the paper can be summarized as follows:

1. The paper formalizes the DLPP and develops a natural MIP formulation to solve it.

2. The paper proposes a Goal-Directed Optimization approach to remedy the limitations of the MIP formulation; it uses a 2-stage approach to eliminate symmetries and provide optimal load plans that are close to a reference plan.

3. The paper proposes an optimization proxy to address the computational difficulties of the GDO approach; the optimization proxy uses a machine learning model to predict the loads and a feasibility restoration procedure to adjust the predictions to satisfy the problem constraints and determine the commodity flows. Once trained, the optimization proxy provides high-quality solutions in a few seconds.

4. The paper presents extensive computational results on industrial instances, including some of the largest terminals in the network; the results demonstrate the significant benefits of optimization and the ability of the optimization proxy to find high-quality and consistent solutions in real time. More precisely, the paper shows that the optimization proxy outperforms a greedy heuristic and the MIP model solved by a commercial solver both in terms of the objective function value and *consistency* metrics. The optimization proxy is around 10 times faster than the commercial solver in obtaining solutions with the same objective function value and orders of magnitude faster in terms of generating solutions that are consistent with a reference plan. Empirical experiments show the value of breaking symmetries by GDO, which helps the proxy to produce high-quality and consistent load plans.

5. From a business and sustainability perspective, the experiments demonstrate the value of having alternate flow paths for the commodities, in addition to the primary flow paths. The proposed load plans allocate approximately 17% commodity volume to the alternate flow paths and reduce the required load capacity by $12\% - 15\%$.

The rest of this paper is organized as follows. Section 2 summarizes related work. Sections 3 and 3.3 introduces the DLPP and its modeling. Sections 4 and 5 present the GDO approach and the optimization proxy. Section 6 describes a heuristic that mimic human planners and serve as a baseline. Section 7 describes the computational results. Section 8 discusses the benefits of the DLPP formulation, optimization, and machine learning, quantifying the cost and sustainability benefits and the important factors driving them.

2 Related Work

Service Network Design. There is abundant research on network design for the *Less-than-truckload* (LTL) trucking industry (see Crainic, 2000; Crainic and Laporte, 1997). Interested readers can consult (Erera et al., 2013a) for a detailed description of LTL operations. Bakir et al., 2021 present a detailed description of the mathematical models and heuristics for the problems arising in trucking service network design. The authors describe the tactical *flow and load planning problem* which is solved weeks in advance for "typical" commodity volume (e.g., average daily origin-destination commodity volume) for a network of terminals. The goal of the flow and load planning problem is to determine effective primary flow paths for the commodity volume and the total trailer capacity required on each flow path in a network of terminals. Most of these network design problems are formulated over time-space networks using integer programming models. The flow and load planning problem with both primary and alternate flow paths for industry-scale instances can be modeled as large-scale integer programming models which, unfortunately, cannot be solved directly by commercial solvers. Therefore, previous work in this area focused mainly on finding a single cost-effective primary flow path for the commodities. Exact approaches to solve these problems have been proposed by Boland et al., 2017, Marshall et al., 2021, and Hewitt, 2019. However, these approaches can only solve instances with a few thousand packages. For industry-scale instances, researchers have resorted to various heuristics including variants of local search heuristic algorithms (Erera et al., 2013b; Lindsey et al., 2016, Powell, 1986) and greedy algorithms (Ulch, 2022).

Flow and Load Planning with Alternate Paths. Tactical flow and load planning is typically based on average daily estimates of origin-destination commodity volume. However, commodity volume differ substantially from day to day and from week to week (Lindsey et al., 2016). Hence, planners at a terminal locally modify the load plans on a daily basis, using the latest estimates of commodity volume until the day of operations. More specifically, the planners take advantage of both primary and alternate flow paths to improve trailer consolidation at their respective terminals. It is worth highlighting that the primary flow paths come from flow and load planning. Once primary options are available alternate flow paths, that are *time feasible*, are identified. To the best of our knowledge no paper carefully studies the problem of allocating volume across alternate flow paths in operations. Alternate flow paths are useful to reduce trailers when commodity volume can be split across paths. This is especially useful because of demand uncertainty. Baubaid et al., 2021 present a study on the value of having these alternate flow paths to hedge against demand uncertainty. They show that it is sufficient to have just one alternate to contain the impact of most of the fluctuations in demand; the authors refer to such a load plan as a $2-$alt load plan. Subsequently, the authors in Baubaid et al., 2023 study the operational decisions that *LTL* carriers need to make to effectively operate a $2-$alt load plan when demand changes dynamically on a day-to-day basis. However, the proposed approach cannot be solved for practical sized instances. This paper proposes a ML-based solution approach for the allocation of volume across both primary and multiple alternate flow paths; the proposed approach is shown to be effective for large scale instances experienced in practice.

Dynamic Load Planning. Network-wide simultaneous optimization of load planning adjustments is a daunting challenge due to the scale of the network, number of commodities and the number of transfer hubs for the commodities. Existing research in the literature may be applicable to the problem of selecting a single primary flow path (non-splittable) for each commodity at each terminal for each sorting period in order to minimize the cost of the resulting load plan. Splitting commodity volume across alternate flow paths is likely to improve trailer utilization as it introduces more flexibility in the load planning process. This research considers the DLPP problem at a terminal in which the commodity volume can be split (among primary and alternate flow paths) to promote better trailer utilization, lower transportation cost, and increased sustainability. The flexibility to adjust plans enables terminal planners to better manage daily operations while maintaining service guarantees. This problem is mentioned as an interesting and useful future research direction by Lindsey et al., 2016.

One paper in the literature, Herszterg et al., 2022, does introduce the problem of re-routing freight volume on alternate flow paths to improve on-time performance of load plans on the day-of-operations; this becomes necessary when the actual volume deviates from the forecasted volume on the day-of-operations. In this work, commodity volume is assigned to exactly one flow path (*it is not splittable*) such that the total (fixed) trailer capacity is respected and the objective is to minimize the total lateness of shipments. The authors develop MIP models for this problem and propose heuristic algorithms to solve them. Note that a key difference between this approach and the approach proposed in the current paper is that we allow volume to be split across multiple flow paths on the day-of-operations. Furthermore, we also adjust the load plan to identify opportunities to reduce outbound capacity (and improve utilization) as demand forecasts are updated.

The DLPP is also similar to the variable-sized bin packing problem described by Friesen and Langston, 1986 where the objective is to minimize the total space used to pack a set of items into bins (available in different sizes), such that each item is packed into exactly one bin. In the DLPP, the packages are the items and trailers are bins but the key difference is that the DLPP allows for the splitting of the package volume into compatible trailers in order to further reduce the transportation cost by promoting better consolidation or packing.

Machine Learning for Optimization. In recent years, there has been a notable surge of interest among researchers in the development of ML surrogates for solving MIPs. This emerging field has attracted attention due to the potential of ML techniques to provide efficient approximations for computationally intensive calculations involved in solving MIPs. We refer the reader to (Bengio et al., 2021,Kotary, Fioretto, Van Hentenryck, and Wilder, 2021) for a comprehensive overview on the topic. The techniques can fall into one of the two categories. The first category includes methods based on reinforcement learning (Fu et al., 2021; Khalil et al., 2017; Kool et al., 2018; Song et al., 2022; Yuan et al., 2022), where the ML model is trained by interacting with simulation environments. The second category comprises supervised learning (Chen, Khir, et al., 2023; Fioretto et al., 2020; Kotary, Fioretto, and Van Hentenryck, 2021, 2022; Park et al., 2022), where the ML model imitates the optimization model and replaces expensive calculations with a quick approximation. This research focuses on the latter category since the proposed optimization model could be used as the expert for supervised learning. Optimization proxies, which combine learning with feasibility restoration, has emerged from supervised learning. Recent work in this area includes (Chen, Khir, et al., 2023; Chen, Tanneau, et al., 2023; Kotary et al., 2022; Park et al., 2022).

3 Problem Description and Modeling

Parcel carriers operate massive terminal networks with hundreds of facilities to move large volumes of parcels each day. Each day at a terminal is divided into time windows (typically three to four hours in length), called *sort periods* or *sorts*, during which parcels are sorted. A typical operational day includes "day", "twilight", "night" and "sunrise" sorts that are non-overlapping in time. All parcels sorted at a terminal during a given sort with the same service class (e.g., one-day service or two-day service) and the same destination are referred to as a *commodity*. Suppose then that each commodity has a primary flow path and one or more alternate flow paths that each specify a sequence terminals and sorts that parcels will traverse en route from origin to destination. For a specific commodity at a specific terminal at a specific sort, each flow path will determine the next terminal and sort to which packages will be loaded. Typically, shipments are loaded on trailers moving along the primary flow path for the commodity; however, when there are better consolidation opportunities, commodity volume can be split over primary and alternate flow paths, or completely allocated to alternate flow paths. The rest of this section describes the main concepts underlying the DLPP. Section 3.1 describes some key terminology and presents examples to illustrate the operations at terminals. Section 3.2 describes the DLPP that includes splitting of commodity volume across primary and alternate flow paths.

3.1 Definitions

Let $\mathcal{G} = (\mathcal{N}, \mathcal{S})$ denote a time-space network. Each node $n \in \mathcal{N}$ represents a terminal location at a particular time period and is defined by a tuple, i.e. $n =$(terminal, sort, day). Each arc $s \in \mathcal{S}$ represents a directed dispatch of loads from one timed node to another. Henceforth in the paper, we refer to each such an arc as a *sort pair*. Figure 2 illustrates an example time-space network for terminal A during a single *twilight* sort period. In this example, three sort pairs are outbound from terminal A on day 1, namely, $(A, Twilight, 1) \rightarrow (X, Twilight, 2)$, $(A, Twilight, 1) \rightarrow (Y, Twilight, 2)$, and $(A, Twilight, 1) \rightarrow (Z, Twilight, 3)$. Figure 3 illustrates another example of terminal B that operates multiple sort periods, i.e., the *day, twilight, night* sorts on a given day, and seven sort pairs $(b_1, b_2, b_3, b_4, b_5, b_6, b_7)$ outbound from terminal B.

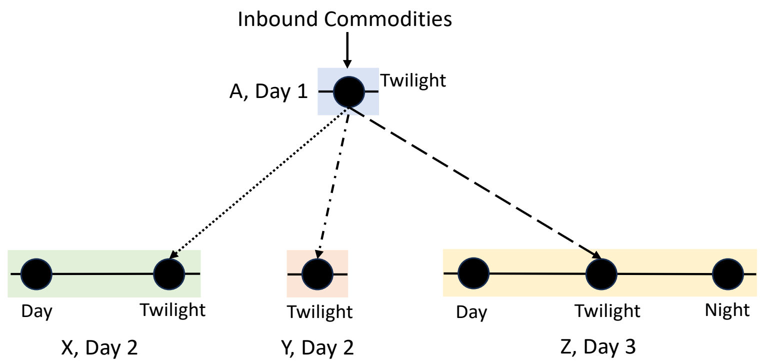


Figure 2: *An illustration of terminal A with one sorting period (twilight sort) on day* 1

A key objective in load planning is to determine the number of trailers (possibly of different types) to operate on each sort pair to containerize the total commodity volume allocated to the sort pair. During a sort, each loading door at a terminal *builds/loads* trailers for a specific sort pair destination. In a single sort facility, as shown in Figure 2, if there is commodity volume allocated on each of the three sort pairs, then at least three trailers (one on each sort pair) should be opened at the loading doors corresponding to the sort pair destinations.

In practice, commodities outbound from an origin terminal that arrive over consecutive sorts and that are heading to the same time-space destination can be consolidated together. For that, the concept of *load pairs* is introduced, where a load pair represents a set of consecutive sort pairs that share the same destination node. Combining sort pairs into load pairs allows better consolidation and trailer utilization, since trailers can be held partially loaded from one sort to the next prior to dispatch to the destination. Figure 4 illustrates an example of a load pair that is composed of three different sort pairs.

We now relate primary and alternate flow paths to sort pairs. If we consider volume for commodity $k \in \mathcal{K}$ at some time-space location n, its primary flow path specifies the next (terminal, sort, day) to which it should be loaded. Thus, the primary path identifies a unique outbound sort pair for k at n. Similarly, each alternate flow path identifies a (possibly different) outbound sort pair for k. Recall that primary and alternate flow paths for each k at n are specified in advance, and we assume that loading outbound on any of these options will lead to volume arriving on-time to its destination.

We will define *compatible* sort pairs for k at n to be the primary path sort pair (the primary sort pair) and any alternate path sort pair (an alternate sort pair). Furthermore, any sort pairs that are in load pairs with compatible sort pairs with an *earlier* origin sort are also compatible. When volume is assigned to such earlier sort pairs, the decision is to assign volume to trailers that are opened first for loading in those earlier sorts and held for dispatch. Figure 5 illustrates four compatible sort pairs (outbound from terminal B) for a commodity k sorted in the twilight sort at terminal B.

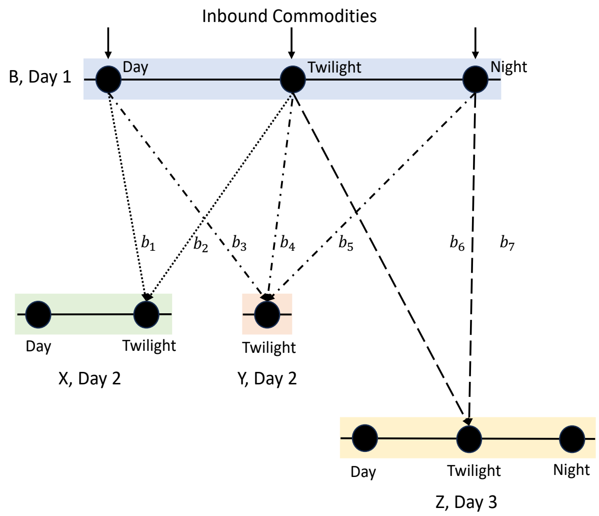
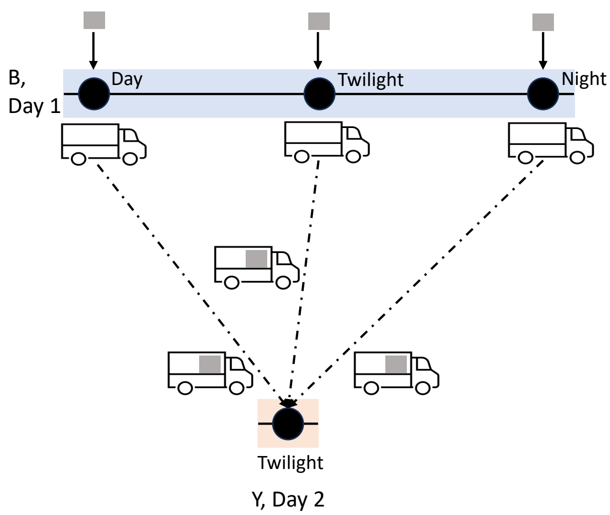
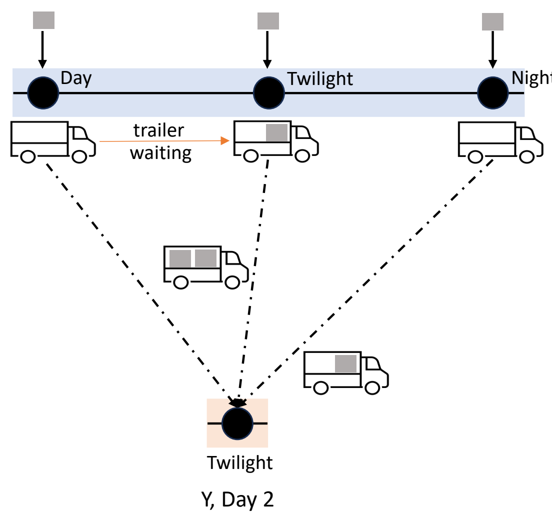


Figure 3: *An illustration of a terminal B with multiple sorting periods on day* 1*. The figure has three load pairs one for each of the three terminals, X, Y, Z, and seven sort pairs.*



(a) *Operating a trailer on each sort pair, independent of other sort pairs in the same load pair requires three trailers*



(b) *Better consolidation can be achieved by keeping a trailer waiting from Day to Twilight sort*

Figure 4: *This example illustrates the concept of load pairs. If two sort pairs have the same destination node, and hence belong to a load pair, then the volume sorted at an earlier sort can be held on a partially-loaded trailer until more volume becomes available during the next sort. This practice promotes better trailer utilization and entails lower transportation cost.*

3.2 Dynamic Load Planning Problem (DLPP)

Parcel carriers typically build a load plan in two phases: (1) the *tactical flow and load planning* phase specifies an initial plan and provides an input to the scheduling team; and (2) the *load plan adjustment* allows adjustments to the initial plans up to the day-of-operation. The scheduling and load dispatching teams then execute the adjusted load plan. Weekly plans that determine the number of loads or trailers to operate

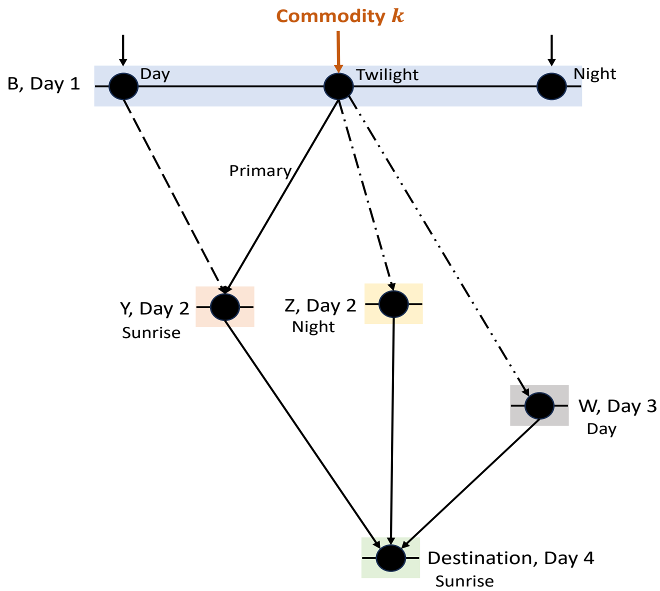
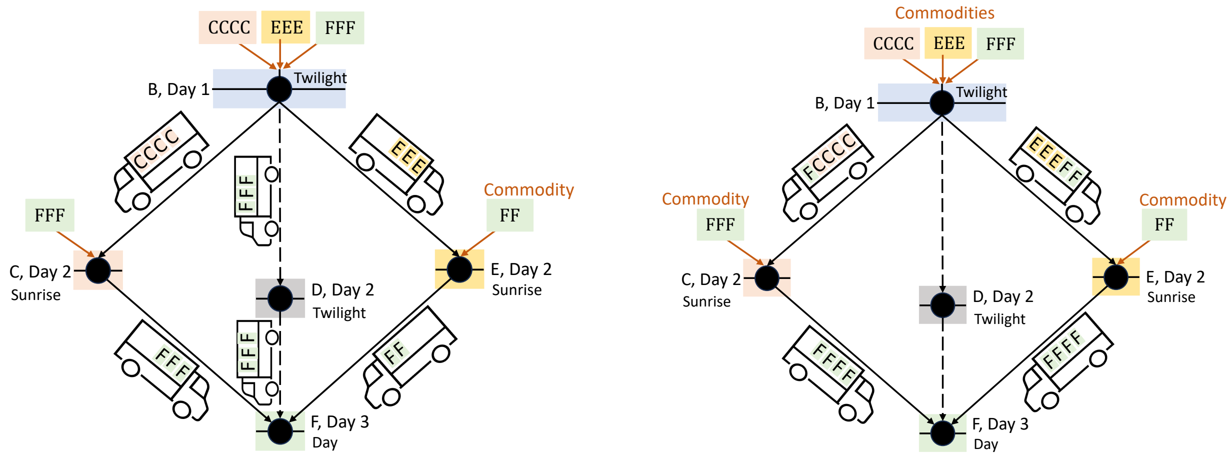


Figure 5: *This example illustrates four compatible sort pairs to terminals Y, Z, W) for commodity k sorted in the twilight sort at terminal B. The four different sort pairs define four different paths to the destination of the commodity.*

on each sort pair are fixed approximately two weeks in advance of the operating week. However, due to demand uncertainty, the volume forecast for commodities may change, and adjustments to the load plan may be necessary to accommodate actual volumes. These adjustments may lead to cost decreases when unnecessary load capacity is removed from the plan.

Consider the following optimization problem during the two weeks leading into the day-of-operation. Each terminal in the network has a set of forecasted inbound commodities during some time period (for example, a single operating day and multiple sorting periods). Each such commodity arrives during a specific sorting period and has a destination terminal and service class (specifying a due date at the destination). Given this information, the fixed flow plan specifies a primary flow path (next terminal and arriving sorting period) for each commodity, and possibly also one or more alternate flow paths. Recall that, if the commodity is assigned to any of these flow paths, then it will reach its final destination on time according to plan. The adjustment optimization problem is to assign each commodity to its primary and/or one of its alternate flow paths while simultaneously determining how many loads of different types are required for each proposed flow paths. Note that existing flow and load planning literature typically assumes that *all* commodities, arriving at a terminal during a specific sorting period should be assigned to the primary flow path. Here, the challenge is different, and is instead to determine specifically how to split each commodity volume among its possible compatible flow paths or sort pairs to drive high load utilization levels and low costs while still meeting service promises.

Consider the example shown in Figure 6 with three commodities (4 units destined to terminal C, 3



(a) *Assigning commodities to their primary sort pair*

(b) *Splitting volume of commodity F across alternate sort pairs*

Figure 6: *The figure shows how splitting of commodity volume can yield better solution. Splitting commodity volume of F between the two alternate sort pairs to C and D yields better consolidation as the solution requires one fewer trailer on each of the arcs from B to F via D.*

units destined to terminal E, 3 units destined to F) sorted in the twilight sort of day 1 at terminal B. In this example, we denote each commodity by its destination terminal name. The commodity destined for terminal F has three compatible sort pairs: $(B, Twilight, 1) \to (C, Sunrise, 2)$ is the primary sort pair, and $(B, Twilight, 1) \to (E, Sunrise, 2)$ and $(B, Twilight, 1) \to (D, Twilight, 2)$ are the alternate sort pairs. Splitting commodity volume destined to terminal F between the two alternate sort pairs to C and D yields better consolidation (and lower transportation cost) as the solution requires one less trailer on the two arcs: $(B, Twilight, 1) \to (D, Twilight, 2)$ and $(D, Twilight, 2) \to (F, Day, 3)$.

For a given terminal, define S to be the set of outbound sort pairs and let K be the set of commodities sorted at the terminal. Each commodity $k \in K$ has a cubic volume of q^k, and a set of compatible sort pairs S^k. For every outbound sort pair s, there is a set V_s of trailer types, that can be used to containerize the total commodity volume allocated to the sort pair. Each sort pair can have different set of allowed trailer types, i.e., V_{s_1} can be different from V_{s_2} for two different sort pairs $s_1, s_2 \in S$. Each trailer type $v \in V_s$ has a cubic capacity Q_v and has a per-unit transportation cost c_v. A solution of the DLPP determines the number of trailers of each type assigned to each sort pair, as well as the volume of each commodity allocated to each trailer. A solution must ensure that all the volume is assigned to trailers and that the capacities of the trailers are not violated. The goal of the DLPP is to find a solution that minimizes the costs of the trailers. Appendix 10.1 provides the complexity results. The DLPP is strongly NP-hard. It becomes weakly NP-hard when each commodity is compatible with exactly one or with all sort pairs and there are multiple trailer types. It becomes polynomial when each commodity is compatible with exactly one or with all sort pairs and there is only one type of trailer.

3.3 A Mixed-Integer Programming Formulation

An optimization model for the DLPP can be defined as follows in Model 1:

$$\underset{x,y}{\text{Minimize}} \quad \sum_{s \in S} \sum_{v \in V_s} c_v y_{s,v} \tag{1a}$$

$$\text{subject to} \quad \sum_{s \in S^k} \sum_{v \in V_s} x^k_{s,v} = q^k, \qquad \forall k \in K, \tag{1b}$$

$$\sum_{k \in K: s \in S^k} x^k_{s,v} \leq Q_v \, y_{s,v}, \qquad \forall s \in S, v \in V_s, \tag{1c}$$

$$x^k_{s,v} \geq 0 \qquad \forall k \in K, s \in S^k, v \in V_s, \tag{1d}$$

$$y_{s,v} \in \mathbb{Z}_{\geq 0} \qquad \forall s \in S, v \in V. \tag{1e}$$

It uses a non-negative continuous decision variable $x^k_{s,v}$ to represent the volume of commodity k allocated to trailer type v operating on a sort pair s, and an integer decision variable $y_{s,v}$ to determine the number of trailers of type v installed on sort pair s. The objective (1a) minimizes the total cost of creating loads. In the experiments, $c_v = Q_v \; \forall \; v \in V$, i.e., the model minimizes the total trailer capacity required to containerize the total commodity volume in the problem instances. Constraints (1b) ensure that the total volume of each commodity is assigned to its compatible sort pairs. Constraints (1c) ensure that the total volume on a sort pair respects the installed trailer capacity on it. Constraints (1d)-(1e) define the domain and range of variables.

4 Goal-Directed Optimization

The optimization model of the DLPP has a large number of symmetries. Figure 7 depicts a simple instance with multiple optimal solutions that are operationally different from one another, yet they are equivalent from Model 1 perspective as they require the same number of trailers of the same type. This is because in Model in 1, commodities are indifferent to the sort pairs they are assigned to, as the volume allocation decisions (x-variables) do not incur any cost.

Such symmetries are undesirable for many reasons. Paramount among them are the realities in the field: the model is intended to be used and validated by planners. If small variations of inputs produce fundamentally different solutions, planners are unlikely to trust the model. Indeed, since the model is used multiple times a day, it is important to ensure that the successive optimal solutions are as consistent as possible with each other. Fortunately, in practice, a *reference plan* is always available and the DLPP should ideally produce optimal solutions that are as close as possible to the reference plan.

This section explores how to refine the model presented earlier to satisfy this requirement, and presents

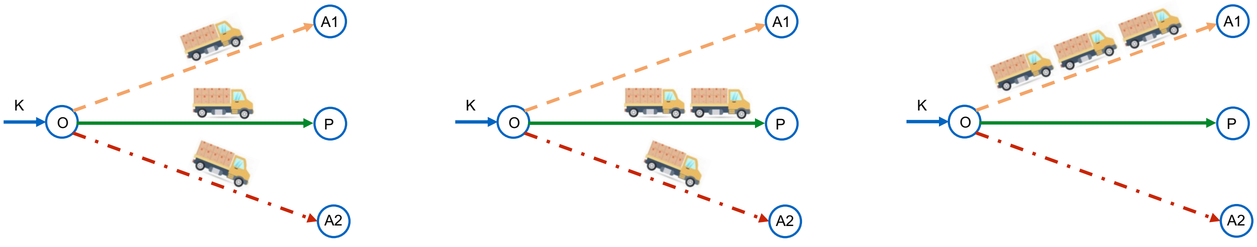


Figure 7: *An example to highlight the symmetries in Model 1. K denotes the set of commodities processed at terminal O with primary flow path P and alternate flow paths A1 and A2*

a Goal-Directed Optimization (GDO) approach to the DLPP. It uses a reference plan to eliminate symmetries and ensure that the solution is compatible with the planner-in-the-loop reality in the field. The use of a reference plan eliminates many symmetries but not all. To break more symmetries, the GDO approach also adds a *flow diversion cost* that captures the cost of using alternate paths instead of the primary path. For instance, in the example depicted in Figure 7, only the solution shown in Figure 7b is optimal following our assumptions. The flow diversion cost is chosen to be proportional to the distance between the next alternate terminal and the destination of the commodity, as there is incentive to move commodities as close as possible to their destination. For example, suppose a commodity k is in *Atlanta* and is destined for *Chicago*. Let the primary next terminal be *Louisville* (with flow diversion cost 0), alternate 1 be *Nashville*, and alternate 2 be *Memphis*. As Nashville is closer to *Chicago* than *Memphis*, the flow diversion cost of allocating volume to alternate 1 is lower than that of alternate 2. As a result, the GDO approach has at its disposal a reference plan γ, where $\gamma_{s,v}$ denotes the number of trailers of type v planned to operate on sort pair s. It also leverages the *flow diversion cost* d_s^k that denotes the cost of allocating a per-unit volume of commodity $k \in K$ to a compatible sort pair $s \in S^k$.

The GDO approach first solves Model 1 to obtain the optimal objective value Z^*. It then solves a second MIP Model to bias the trailer decisions so that they are as close as possible to the reference plan and minimize diversion costs. The second-stage model is defined as follows:

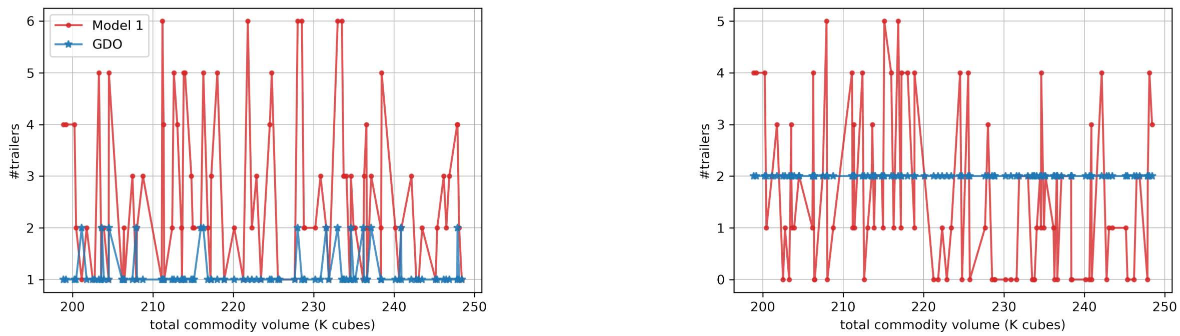
$$\underset{x,y}{\text{Minimize}} \quad \sum_{s\in S}\sum_{v\in V_s} |y_{s,v} - \gamma_{s,v}| + \epsilon \sum_{k\in K}\sum_{s\in S_k}\sum_{v\in V_s} d_s^k x_{s,v}^k \tag{2a}$$

$$\text{subject to} \quad \sum_{s\in S^k}\sum_{v\in V_s} x_{s,v}^k = q^k, \qquad \forall k \in K, \tag{2b}$$

$$\sum_{k\in K: s\in S^k} x_{s,v}^k \le Q_v(y_{s,v}), \qquad \forall s \in S, v \in V_s, \tag{2c}$$

$$\sum_{s\in S}\sum_{v\in V_s} c_v y_{s,v} \le Z^*, \tag{2d}$$

$$x_{s,v}^k \ge 0 \qquad \forall k \in K, s \in S^k, v \in V_s, \tag{2e}$$



(a) *Sort pair 1* **(b)** *Sort pair 2*

Figure 8: *Sensitivity analysis of Model 1 and GDO on real-world medium-sized instance from the industrial partner. The optimal trailer decisions from Model 1 oscillate dramatically for a small change in total commodity volume, which might confuse the planner. On the contrary, the proposed GDO generates more consistent plans.*

$$y_{s,v} \in \mathbb{Z}_{\geq 0} \qquad \forall s \in S, v \in V. \tag{2f}$$

The objective function (2a) minimizes the weighted sum of the Hamming distance of the trailer decisions from the reference plan γ and the flow diversion costs. The weight ϵ for the flow diversion cost is sufficiently small such that the cost does not dominate over the Hamming distance term in the objective function. The purpose of the flow diversion cost in (2a) is to break the symmetry between solutions with the same Hamming distance; it biases the solution to have more volume allocated to primary sort pairs than alternate sort pairs. Constraints (2b), (2c), (2e) and (2f) are the same as in Model 1. Constraint (2d) ensures that the optimal solution does not use more trailer capacity than Z^*. Note that the objective function is non-linear due to the Hamming distance term. It can be linearized by replacing $|y_{s,v} - \gamma_{s,v}|$ with new variables $w_{s,v} \geq 0 (s \in S, v \in V_s)$ and imposing the following constraints

$$y_{s,v} - \gamma_{s,v} \leq w_{s,v} \quad \forall s \in S, v \in V_s, \tag{3a}$$

$$\gamma_{s,v} - y_{s,v} \leq w_{s,v} \quad \forall s \in S, v \in V_s, \tag{3b}$$

Figure 8 illustrates the sensitivity of the trailer decisions (y-variables) subject to increases in the total commodity volume ($\sum_{k \in K} q^k$) (x-axis) for the two models: Model 1 (red plot) and the GDO approach (blue plot). As the total commodity volume increases, Model 1 exhibits solutions where the trailer decisions fluctuate dramatically between 1 and 6 trailers for sort pair 1, and between 1 and 5 trailers for sort pair 2. However, when using GDO, the trailer decisions in GDO are more consistent and vary between 1 and 2 trailers on sort pair 1, and is constant at 2 trailers on sort pair 2.

5 Learning-based Optimization Proxies

The GDO approach produces consistent solutions to the DLPP, but it is too slow to be used with planners in the loop. This section proposes a Machine Learning (ML) approach to the DLPP. Its goal is to move some of the optimization burden offline and produce high-quality solutions in real time. More precisely, the approach uses the concept of *optimization proxies* to produce high-quality solutions to an optimization problem by learning its input/output mapping (see, for instance, (Chatzos et al., 2022; Chen, Khir, et al., 2023; Chen et al., 2022; Chen, Tanneau, et al., 2023; Fioretto et al., 2020; Kotary, Fioretto, and Van Hentenryck, 2021, 2022; Park et al., 2022; Park and Van Hentenryck, 2023) for an overview of this concept and its applications).

The overall methodology underlying optimization proxies is depicted in Figure 9. It consists of two stages,

1. an *offline* stage where an ML model learns the input/output mapping of the optimization problem;
2. an *online* stage which is used in real time: it receives an instance, applies the ML model to predict a (possibly infeasible) solution and uses a *repair procedure* to deliver a feasible solution.

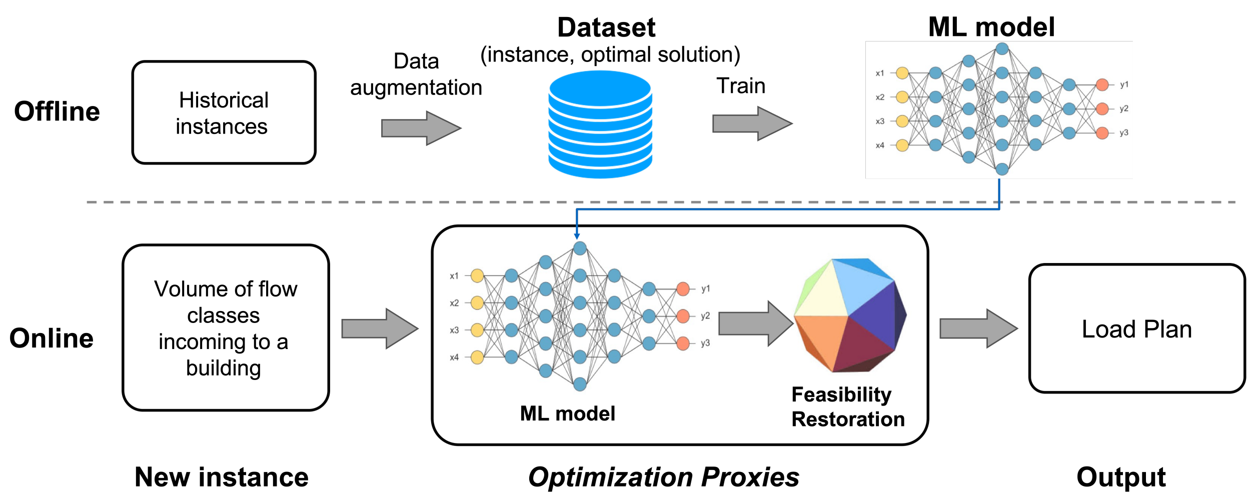


Figure 9: *The Overall Pipeline of Learning-Based Optimization Proxies: In the offline setting, the dataset is generated and augmented by solving historical instances with the proposed GDO model. The ML model is trained in a supervised learning fashion. In the online setting, the optimization proxies take as input a new instance and output near-optimal feasible load plan in seconds.*

For the DLPP, the ML model learns the mapping between the (input) commodity volumes and the (output) trailer decisions; in other words, given the commodity volumes, the ML model predicts trailer decisions for every sort pair. The trained ML model may sometimes *underestimate* the number of trailers on some sort pairs when executed in real time. To circumvent this issue, the *feasibility restoration* step projects the predicted trailer decisions back into the feasible region; in addition, the feasibility restoration also computes the volume allocation on the sort pairs. A key element in the ML training is *data augmentation* that complements historical data by generating realistic instances through input perturbations. The ML model formulation is introduced and discussed in more details in what follows.

5.1 The ML Model Formulation

This section defines a machine learning model f, parameterized by θ, that maps the input parameters, i.e., the commodity volume, to the optimal trailer decisions: (4a)-(4b).

$$f_\theta : \mathbb{R}_{\geq 0}^{|K|} \longrightarrow \mathbb{Z}_{\geq 0}^{|S|\times|V|} \tag{4a}$$

$$\mathbf{p} \longmapsto \mathbf{y} \tag{4b}$$

The ML inputs are assumed to be taken from a distribution $\mathcal{P}$ that captures the actual instances.

Given a dataset of input parameters $\{\mathbf{p}_i\}_{i\in N} \sim \mathcal{P}$, where N is the set of instances, parametrization θ^* can be obtained by minimizing the empirical risk shown in (5a), where (5b) denotes the optimization problem solved by Model 2, and l denotes the loss function that measures the L1-distance of the predicted ($f_\theta(\mathbf{p})$) and optimal (y^*) trailer decisions.

$$\underset{\theta}{\text{Minimize}} \quad \frac{1}{N}\sum_{i\in N} l(f_\theta(\mathbf{p}_i), \mathbf{y}_i^*) \tag{5a}$$

$$\text{subject to} \quad (\mathbf{x}_i^*, \mathbf{y}_i^*) = \underset{\mathbf{x},\mathbf{y}\in\mathcal{C}(\mathbf{p}_i)}{\text{argmin}}\ c(\mathbf{x}, \mathbf{y}), \tag{5b}$$

It is important to highlight that an ML model could be used to predict commodity volume allocation on the sort pairs (x-variables) instead of the trailer decisions (y-variables). This may seem to be a good approach since, after predicting volume allocation, one can easily recover the trailer decisions and hence a feasible solution, by setting $y_{s,v} = \left\lceil \frac{\sum_{k\in K: s\in S^k} x_{s,v}^k}{Q_v} \right\rceil \ \forall s \in S, v \in V_s$. However, this approach has some shortcomings. First, the output dimension is significantly larger than the input dimension which makes it very difficult to develop an effective ML model even for the smallest instances. Second, recovering trailer decisions is very sensitive to the predicted volume allocation decisions. Consider an example where 100 cubic volume is allocated to a sort pair which requires two trailers, each with capacity 50 cubic volume, in the optimal solution. If the ML model predicts the volume on the sort pair to be 100.5, then the total number of trailers required is $\left\lceil \frac{100.5}{50} \right\rceil = 3$ which generates a poor solution in terms of the objective function value of Model 1. Experimental results confirmed that it is beneficial to learn the mapping from input parameters to the trailer decisions rather than the volume decisions. The trailer decisions $\mathbf{y} \in \mathbb{Z}_{\geq 0}^{|S|\times|V|}$ are more aggregated than the volume allocation decisions $\mathbb{R}_{\geq 0}^{|K|\times|S|\times|V|}$. The benefits comes from the significant reductions in output dimensionality and variability. In addition, as presented in section 5.2, once the trailer decisions are known, restoring the feasibility of the solution is relatively easy as the feasibility restoration MIP has a small number of binary decision variables and therefore, it is easy to solve.

The ML model used in this paper is a deep neural network as illustrated in Figure 10. It uses a Multi-Layer Perceptron (MLP), where each dense layer is followed with a batch normalization (Ioffe and Szegedy, 2015), a dropout (Srivastava et al., 2014), and a ReLU (Rectified Linear Unit) function. It maps the input parameter $\mathbf{p}$ to the flattened trailer decision $\tilde{\tilde{\mathbf{y}}}$. The last ReLU guarantees that the output of the neural network is non-negative. The compatible trailer decisions $\tilde{\mathbf{y}}$ are then generated by reshaping the flattened decision $\tilde{\tilde{\mathbf{y}}}$ and masking it with the compatible trailer mask $\mathbf{m}$, where $m_{s,v} = 1$ indicates that equipment type $v \in V$ is compatible with sort pair $s \in S$. In the training phase, the loss function is computed by measuring the distance of predicted compatible trailer decision $\tilde{\mathbf{y}}$ with the optimal trailer decisions. Specifically, this work used smooth l_1 loss. The loss is used to update the parameters of the MLP using stochastic gradient descent (Kingma and Ba, 2014) with back propagation (Rumelhart et al., 1986). At inference time (i.e., in real time), the compatible trailer decisions are rounded to an integer value.

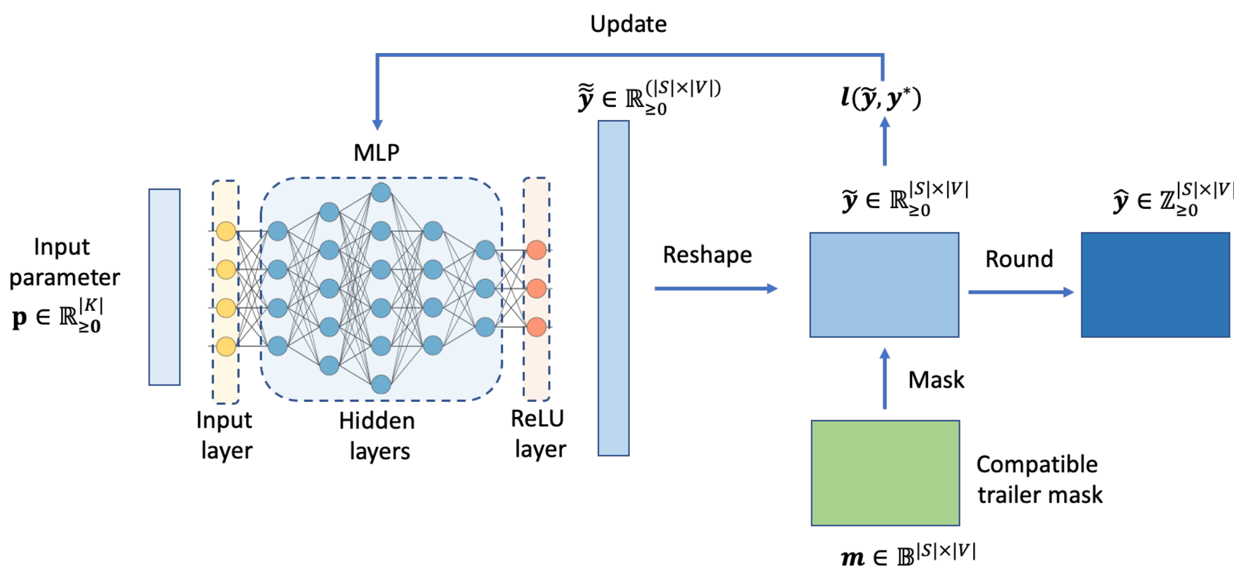


Figure 10: *The Machine Learning Model for DLPP.*

5.2 MIP-based Feasibility Restoration

The proposed ML model predicts the number of trailers $\hat{y}_{s,v}$ for each sort pair $s \in S$ and equipment type $v \in V_s$. Let the total trailer capacity installed on each sort pair $s \in S$ be $\Lambda_s = \sum_{v \in V_s} Q_v(\hat{y}_{s,v})$. The system of equations

$$\sum_{s \in S^k} \sum_{v \in V_s} x^k_{s,v} = q^k, \qquad \forall k \in K, \tag{6a}$$

$$\sum_{v \in V_s} \sum_{k \in K: s \in S^k} x^k_{s,v} \leq \Lambda_s, \qquad \forall s \in S, \tag{6b}$$

$$x^k_{s,v} \geq 0 \qquad \forall k \in K, s \in S^k, v \in V_s, \tag{6c}$$

is then used to determine the volume of every commodity $k \in K$ allocated to its compatible sort pairs. However, it is possible that some of the sort pairs do not have sufficient trailer capacity because the ML model may *underestimate* the capacity. In that case, (6) is infeasible. The following linear program

$$\underset{z}{\text{Minimize}} \quad \sum_{s \in S} z_s \tag{7a}$$

$$\text{subject to} \quad \sum_{s \in S^k} \sum_{v \in V_s} x^k_{s,v} = q^k, \qquad \forall k \in K, \tag{7b}$$

$$\sum_{v \in V_s} \sum_{k \in K: s \in S^k} x^k_{s,v} - z_s \leq \Lambda_s, \qquad \forall s \in S, \tag{7c}$$

$$x^k_{s,v}, z_s \geq 0 \qquad \forall k \in K, s \in S^k, v \in V_s, \tag{7d}$$

can be used determine the sort pairs with trailer capacity violations. Its objective function (7a) minimizes the capacity violations on the sort pairs. Constraints (7b) ensure that total volume of every commodity is assigned to compatible sort pairs. Constraints (7c) determine the sort pair capacity violations. Constraints (7d) define the domain and range of variables. When Model 7 has an optimal objective value equal to 0, it has recovered a feasible solution to Model 1. Otherwise, additional trailer capacity is required on sort pairs with capacity violations.

This paper proposes a two-stage MIP-based feasibility restoration process. In the first stage, Model 7 is solved to obtain an optimal solution z^*. Let the set of sort pairs with trailer capacity violation be $\widehat{S} = \{s \in S : z^*_s > 0\}$. The feasibility restoration then identifies the cheapest equipment v to serve the excess volume on sort pair $s \in \widehat{S}$. The extra trailer capacity is given by $\xi_s = \left(\left\lceil \frac{z_s}{Q_v} \right\rceil * Q_v \right)$ and the option to add the extra capacity to sort pair $s \in \widehat{S}$ is added using a binary decision variable. The second stage solves the following MIP model:

$$\underset{u}{\text{Minimize}} \quad \sum_{s \in \widehat{S}} u_s \xi_s \tag{8a}$$

$$\text{subject to} \quad \sum_{s \in S^k} \sum_{v \in V_s} x^k_{s,v} = q^k, \qquad \forall k \in K, \tag{8b}$$

$$\sum_{v \in V_s} \sum_{k \in K: s \in S^k} x_{s,v}^k \leq \Lambda_s + u_s \xi_s, \qquad \forall s \in \widehat{S}, \tag{8c}$$

$$\sum_{v \in V_s} \sum_{k \in K: s \in S^k} x_{s,v}^k \leq \Lambda_s, \qquad \forall s \in S \backslash \widehat{S}, \tag{8d}$$

$$x_{s,v}^k \geq 0 \qquad \forall k \in K, s \in S^k, v \in V_s, \tag{8e}$$

$$u_s \in \{0, 1\} \qquad \forall s \in \widehat{S}, \tag{8f}$$

The objective function in (8a) minimizes the total trailer capacity added on each sort pair. Constraints (8b) ensure that the commodity volume is assigned to the compatible sort pairs. Constraints (8c) and (8d) ensure that commodity volume allocated to each sort pair respects the trailer capacity. Constraints (8e) and (8f) define domain and range of variables. The number of binary variables in this model is at most the number of sort pairs in the instance, i.e. $|S|$. The main difference between Model 8 and Model 7 is that Model 8 uses binary variables u_s instead of continuous variable z_s, to denote the option of adding extra trailer capacity ξ_s on sort pairs $s \in \widehat{S}$. When $u_s = 1$, extra trailer capacity is added to sort pair s.

After solving Model (7), adding ξ_s capacity on every sort pair $s \in \widehat{S}$ yields a feasible solution to Model (1). However, the goal is to use Model (8) to obtain a better feasible solution. Consider an example with a set of commodities all of which can be allocated to any of the two sort pairs s_1 and s_2 and trailer with capacity 2 units. Suppose the optimal solution of Model 7 is $z_{s_1} = z_{s_2} = 1$. In this case, a feasible solution to Model 1 can be recovered by adding two trailers, one on each sort pair. However, Model (8) (which has two binary variables) yields a solution with only one trailer on any one of the two sort pairs. Algorithm 1 provides a summary of the feasibility restoration procedure.

Algorithm 1 MIP-based Feasibility Restoration

Solve Model 7
$z^* \leftarrow$ optimal solution value $\forall\, s \in S$ in Model 7
For $(s \in S : z_s^* > 0)$
    $v \leftarrow$ equipment to containerize excess volume on sort pair s at minimum cost
    $\xi_s \leftarrow \left(\left\lceil \frac{z_s^*}{Q_v} \right\rceil * Q_v \right)$
End For
Solve Model 8

5.3 Value of Symmetry-Breaking Data Generation for Learning

The optimization proxies are trained using the solutions provided by the GDO which uses the same reference plan for all instances of a given terminal. *As a result, the proxies are consistent by design and do not rely on a reference plan*. The GDO approach is not only critical for environments with planners in the loop, but it also has an additional benefit: it makes the learning problem easier. This section provides theoretical insights about why the data generation using GDO results in better function approximation than data generation from Model (1) alone.

Observe that the solution trajectory associated with different instances can often be effectively approximated by piecewise linear functions, as depicted in Figure 8. This approximation becomes exact in the case of linear programs and mixed integer programs when the input reflects incremental changes in the objective coefficients or right-hand sides of the constraints. This paper utilizes ReLU-based neural networks to approximate the solutions of optimization problems. These neural networks are capable of capturing piecewise linear functions, which makes them well-suited for this purpose. However, the ability of representing a target piecewise linear function accurately depends on the model capacity. As the complexity of the function grows with more pieces, a larger model is required to obtain a high-quality approximations.

Theorem 1. *(Model Capacity) (Arora et al., 2016) Let $f : \mathbb{R}^d \to \mathbb{R}$ be a piecewise linear function with p pieces. If f is represented by a ReLU network with depth $k+1$, then it must have size at least $\frac{1}{2}kp^{\frac{1}{k}} - 1$. Conversely, any piecewise linear function f that is represented by a ReLU network of depth $k+1$ and size at most s, can have at most $(\frac{2s}{k})^k$ pieces.*

Due to the symmetry in optimal solutions of Model (1), as shown in Figure 8, the solution trajectory varies dramatically. Theorem 1 states that the approximation of a more volatile solution trajectory (i.e., a piecewise linear function with more pieces) requires a deep neural network with greater capacity, which makes the learning task more challenging. In other words, given a fixed-size ReLU network, higher variability of the solution trajectory typically results in higher approximation errors. These errors are bounded by the following theorem.

Theorem 2. *(Approximation Error) (Kotary, Fioretto, and Van Hentenryck, 2021) Suppose a piecewise linear function $f_{p'}$, with p' pieces each of width h_k for $k \in [p']$, is used to approximate a piecewise linear f_p with p pieces, where $p' \le p$. Then the approximation error*

$$\|f_p - f_{p'}\|_q \le \frac{1}{2}h_{max}^2 \sum_{1 \le k \le p} |L_{k+1} - L_k|,$$

holds where L_k is the slope of f_p on piece k and h_{max} is the maximum width of all pieces.

Theorem 2 relates the approximation error of a piecewise linear function with the total variation of its slopes. It implies that the data generated using GDO (which exhibits lower sensitivity than the data from Model (1)) should facilitate learning and result in lower approximation errors.

6 Greedy Heuristic (GH)

This section proposes a greedy heuristic to construct feasible solutions for Model (1) and benchmark the quality of the solution obtained from optimization proxies. This heuristic iteratively solves linear programs

(LP) until all the y-variables are integers, i.e., they satisfy the integrality tolerance (10^{-5}). In each iteration, the algorithm identifies fractional variables with minimum $(\lceil y_{s,v} \rceil - y_{s,v})$ value, updates the lower bound of variable $y_{s,v}$ to $\lceil y_{s,v} \rceil$, and re-solves the LP as shown in Algorithm 2. The main idea is that for a given sort pair $s \in S$ and trailer type $v \in V_s$, if $y_{s,v}$ has a fractional value very close to an integer $\lceil y_{s,v} \rceil$, then, this indicates that there is enough commodity volume to have *at least* $\lceil y_{s,v} \rceil$ trailers on the sort pair. GH *greedily* adjusts the lower bound of a y-variable in each iteration till all y-variables can be labelled as integers, in which case a feasible solution to Model (1) has been found.

Algorithm 2 Greedy Heuristic

```
While(number of fractional y-variables > 0)do
  Solve LP relaxation of Model 1
  If(all y-variables in the LP solution are integers, i.e., ⌈y_{s,v}⌉ − y_{s,v} ≤ 10^{-5} ∀s ∈ S, v ∈ V_s)
    Integer Feasible Solution Found
    Stop
  Else
    list ← {(s, v) : ⌈y_{s,v}⌉ − y_{s,v} > 10^{-5}}
    (s*, v*) ← argmin_{(s,v)∈list}(⌈y_{s,v}⌉ − y_{s,v})
    Update the lower-bound of y_{s*,v*} to ⌈y_{s*,v*}⌉
  End If
End While
```

7 Computational Study

This section reports a series of experiments conducted on real-life instances provided by our industry partner. Section 7.1 presents statistics for the problem instances. Section 7.2 discusses the experimental setup for the optimization models and proxies. Section 7.3 evaluates the computational performance of the optimization proxies against the greedy heuristic (GH) and the optimization models (Model (1) and GDO). Section 7.4 evaluates the benefits of GDO for learning.

7.1 Instances

The experiments are based on industrial instances for three different terminals in the service network of our industry partner: *medium* (M), *large* (L), and *extra-large*(XL). Each category has a reference plan for a terminal on a particular day as provided by our industry partner. Table 1 reports the statistics of the instances: #*Arcs* denoting the total number of unique outgoing sort pairs or arcs from the terminal, #*Commodities* denoting the number of commodities that are sorted at the terminal and loaded into outbound trailers (rounded to nearest multiple of 1,000), and #*Loads* denoting the number of planned loads in the reference load plan for the corresponding terminals (rounded to the nearest multiple of 50). Note that, in addition to the planned loads, small package companies typically operate *empty* trailers on the outbound sort pairs for trailer repositioning. This study only considers trailers that are filled with commodity volume and do not include empty trailer capacity.

Table 1: *Data Statistics for Test Instances*

Category	#Arcs	#Commodities	#Loads
M	92	9,000	150
L	399	15,000	550
XL	1,602	20,000	2,000

It is worth highlighting that the XL instance operates more volume and capacity than the M and L instances combined. Table 2 reports some statistics for Model (1) for the three instances: #Integer-Vars and #Continuous-Vars denoting the number of integer and continuous decision variables, respectively, and #Constraints denoting the total number of constraints.

Table 2: *Optimization Model Statistics for Model 1*

Instance	#Integer-Vars	#Continuous-Vars	#Constraints
M	187	50,713	8,962
L	846	60,023	15,583
XL	4,475	107,854	24,931

7.2 Experimental Setup

Parameters for GDO The cost of assigning commodity k to a sort pair $s \in S^k$ (denoted by d_s^k) is defined as

$$d_s^k = \begin{cases} 0, & \text{if } s \text{ is primary flow path for commodity } k \\ \left(\alpha_s^k + 10 * \beta^k\right) & \text{otherwise} \end{cases} \tag{9}$$

$$\beta_k = \begin{cases} 1, & \text{if commodity } k \text{ belongs to one-day service class} \\ 2, & \text{if commodity } k \text{ belongs to two-day service class} \\ 3, & \text{if commodity } k \text{ belongs to three-day service class} \\ 4, & \text{otherwise} \end{cases} \tag{10}$$

$$\epsilon = \frac{1}{\left(\max_{k \in K, s \in S^k} (\alpha_s^k + 10 * \beta^k)\right) \sum_{k \in K} q^k} \tag{11}$$

where α_s^k denotes the distance between the alternate next terminal and the destination of commodity $k \in K$ for sort s, and parameter β_k depends on the commodity service level. Recall that a commodity $k \in K$ is defined as all packages with the same destination and service class. The term α_s^k ensures that two commodities with different destinations have different flow diversion cost. However, two commodities with different service class can have the same destination. β^k ensures that such commodities have different flow diversion cost for the same destination. The weight for the flow diversion cost is defined in (11).

Data Generation for ML Model The dataset is generated by perturbing the input parameters of real-life instances provided by the industry partner with up to 20,000 commodities. Denote by $\mathbf{p}^{ref}$ the volume of different commodities in a given reference plan. The DLPP instances are generated by perturbing this reference commodity volume. Namely, for instance i, $\mathbf{p}^{(i)} = \gamma^{(i)} \times \eta^{(i)} \times \mathbf{p}^{ref}$, where $\gamma^{(i)} \in \mathbb{R}$ denotes a global scaling factor and $\eta \in \mathbb{R}^{|K|}$ is the commodity level multiplicative white noise. γ is sampled from a uniform distribution $U[80\%, 120\%]$, and for every commodity η is sampled from a normal distribution with mean equals to 1 and standard deviation of 0.05. For every category, 10,000 instances are generated, and a commercial solver is used to solve the GDO model for each instance. The dataset of 10,000 instances for each category is then split as follows: 80% for the training set, 10% for the validation set, and 10% for the test set.

Performance Metrics The performance metrics in this study are designed to compare the *total trailer cost* and the *consistency* of the solutions generated by the optimization proxies against the total trailer cost from Model (1) and then the consistency of solution from Model (2) of the GDO approach. Given an instance $\mathbf{p}$ with optimal trailer decision $\mathbf{y}^*$ and a feasible trailer decision $\hat{\mathbf{y}}$, the optimality gap is defined as

$$\text{Gap} = (\hat{Z} - Z^*)/|Z^*|,$$

where Z^* is the optimal trailer cost of Model (1), and $\hat{Z}$ is the trailer cost computed from $\hat{\mathbf{y}}$. Recall that the total trailer cost does not increase in Model (2) of the GDO approach due to constraint (2d). If Model (1) cannot be solved to optimality in 30 minutes, then the best lower bound obtained from the solver run is used to compute the optimality gap instead of Z^*.

This paper proposes two metrics to quantify the consistency. The first one is a normalized distance (Δ) between the optimized load plan and the load plan $\tilde{\mathbf{y}}$, using shifted geometric means as given by

$$\Delta_{s,v} = \begin{cases} |y_{s,v} - \tilde{y}_{s,v}| & \text{if } \tilde{y}_{s,v} = 0 \\ |y_{s,v} - \tilde{y}_{s,v}|/\tilde{y}_{s,v}, & \text{otherwise} \end{cases} \qquad \forall s \in S, v \in V_s \tag{12}$$

$$\Delta = \exp\left(\frac{1}{|S||V|}\sum_{s\in S}\sum_{v\in V_s}\log(\Delta_{s,v} + 0.01)\right) - 1\%. \tag{13}$$

From a planner perspective, this metric captures the deviation of the optimized load plan with respect to the reference load plan. As mentioned in Section 5.3, load plans that are as close as possible to the reference plan are highly desirable.

The second metric is the total variation of the set of trailer decisions across a set of N instances (for each terminal). For simplicity, instances are ordered such that $\sum_{k\in K} q_{i+1}^k \geq \sum_{k\in K} q_i^k \; \forall \; i \in \{1, 2, \cdots, N-1\}$. The goal is to analyze the variation in trailer decisions on sort pairs when the total commodity volume is incrementally increased from $\sum_{k\in K} q_1^k$ to $\sum_{k\in K} q_N^k$. Let $\{\mathbf{y}_i\}_{i=1}^N$ denote the set of trailer decisions of N instances. The total variation is defined as:

$$\mathrm{TV}(\{\mathbf{y}_i\}_{i=1}^N) = \sum_{i=1}^{N-1} \|\mathbf{y}_{i+1} - \mathbf{y}_i\|_p,$$

where $p = 2$. This metric captures the sensitivity of the models, i.e., the impact of changes in total commodity volume on the trailer decisions of different sort pairs. Lower total variation implies that the trailer decisions are less sensitive to changes in total commodity volume. Planners are more amenable to such solutions because fewer (but effective) load plan modifications reduce the solution evaluation effort and is also easier to execute in practice.

The computational efficiency of different models is measured by the training time of optimization proxies including the data-generation time and the inference time. Unless specified otherwise, the average metrics on the test dataset are reported in shifted geometric means:

$$\mu_s(x_1, \ldots, x_n) = \exp\left(\frac{1}{n}\sum_i \log(x_i + s)\right) - s,$$

where the shift is set as 0.01 for the optimality gap and normalized distance, 1 second for the inference/solving time, and 1 cube for the distance between the optimized load plan to the reference load plan.

Implementation Details All optimization problems are formulated using the Gurobi Python interface, and solved with Gurobi 9.5 (Gurobi Optimization, LLC, 2023) with 8 CPU threads and default parameter settings, except for *MIPFocus* which is set to a value of 3. All deep learning models are implemented using PyTorch (Paszke et al., 2019) and trained using the Adam optimizer (Kingma and Ba, 2014). The ML models are multiple layer perceptron and are hyperparameter-tuned using a grid search with learning rate in $\{10^{-1}, 10^{-2}\}$, number of layers in $\{3, 4, 5\}$, and hidden dimension in $\{128, 256\}$. For each system, the best model is selected on the validation set and the performances on the test set are reported. Experiments are conducted on dual Intel Xeon 6226@2.7GHz machines running Linux, on the PACE Phoenix cluster (PACE, 2017). The training of ML models is performed on Tesla V100-PCIE GPUs with 16GBs HBM2 RAM.

7.3 Computational Performance of the Optimization Proxies

This section presents numerical experiments used to assess the performance of the proposed optimization proxies (Proxies) against the optimization models (GDO) and the greedy heuristic (GH).

Optimality Gap Table 3 presents the optimality gaps of various approaches, including the results of Model (1) under various time constraints. In the table, the columns under "Gap of Model (1)" denote the optimality gaps of the model under various time limits. Similarly, columns *Gap* for GH and Proxies denote optimality gaps for GH and the optimization proxies. In addition, columns *Time(s)* denote the solving times for GH and Proxies.

Table 3: *Optimality Gap (%) with respect to the Total Trailer Cost*

Instance	Model (1)						GH		Proxies	
	1s	5s	10s	30s	60s	1800s	Gap	Time (s)	Gap	Time (s)
M	2.59	0.55	0.48	0.48	0.48	0.48	3.84	3.12	1.14	0.33
L	51.15	5.22	2.18	1.71	1.41	1.39	12.85	13.28	3.80	1.10
XL	77.35	14.02	10.41	2.93	2.07	0.93	17.01	121.55	5.21	2.49

Recall that Model (1) produces solutions that exhibit considerable variability when the total commodity volume is perturbed as detailed in Table 4 and 5. As such, it is unlikely to be practical in scenarios with planners in the loop. Hence, the table compares the optimization proxies and the heuristics GH with an "idealized" benchmark. With this caveat in place, observe the performance of the optimization proxies under tight time constraints. Proxies generate solutions with low optimality gaps and may be up to 10 to 50 times faster than GH, and around 10 times faster than Model (1) solved with Gurobi. Second, although Model (1) efficiently produces solutions with low optimality gaps, closing the optimizality gap proves to be a significant challenge due to the poor LP relaxation. The performance of GH is also impeded by the inefficiencies of the LP relaxation, as it solves the LP relaxations over many iterations; it takes the GH around 30 iterations for terminal M, 200 iterations for terminal L, and more than 1000 iterations for terminal XL to generate a feasible solution.

Consistency Tables 4 and 5 report the consistency of solutions obtained from different models in terms of the normalized distance to the reference load plan and the total variation of the generated solutions. As GDO requires running Model (1) and Model (2) sequentially, these experiments set the same time limits for the two stages. For example, if a time limit of 30 seconds is set, GDO runs Model (1) for 30 seconds and subsequently runs Model (2) using the best upper bound obtained from Model (1) for another 30 seconds.

The high-level result is that proxies are ideally suited to produce consistent plans. Table 4 shows that the proxies accurately predict, in a few seconds, the results produced by GDO after an hour. Furthermore, Table 5

Table 4: *Normalized Distance (%) to Reference Load Plan*

Instance	Model	Optimization Models					GDO		GH		Proxies	
		1s	5s	10s	60s	1800s	Δ	t (s)	Δ	t (s)	Δ	t (s)
M	M1	13.53	13.43	13.42	13.19	13.20	5.18	3600	14.39	3.12	5.19	0.33
	M2	12.97	11.40	9.49	6.03	5.18						
L	M1	18.19	13.31	12.69	12.76	12.69	4.01	3600	17.07	13.28	3.75	1.10
	M2	0.45	11.40	10.11	7.03	4.01						
XL	M1	21.86	7.99	7.96	7.47	7.23	3.52	3600	10.08	121.55	2.97	2.49
	M2	16.93	7.41	7.70	6.21	3.52						

Table 5: *Total Variation of Load or Trailer Decisions*

Instance	Model	Optimization Models					GDO		GH		Proxies	
		1s	5s	10s	60s	1800s	TV	t (s)	TV	t (s)	TV	t (s)
M	M1	2097	2063	2126	2040	2054	392	3600	2444	3.12	115	0.33
	M2	2071	1737	1528	579	392						
L	M1	2655	3275	2961	3134	3174	1201	3600	4797	13.28	347	1.10
	M2	684	3156	2829	2368	1201						
XL	M1	6428	9174	8228	6914	7047	5228	3600	11843	121.55	565	2.49
	M2	6519	9095	8131	6715	5228						

shows that proxies produce solutions that have at least an order of magnitude smaller total variations in trailer decisions than both GDO and GH. Proxies produce load plans that exhibit great stability with changing total commodity volume.

The fact that proxies improve the consistency of the GDO plans is especially interesting: it means that the optimization proxies, by virtue of the learning step, avoid oscillations present in the GDO approach. Of course, it does so at a small loss in objective value (if, for instance, the GDO model is allowed a minute to run instead of the 2.5 seconds of the optimizations). But the consistency benefits are substantial as shown in Table 5. The proxies also provide dramatic improvements over the GH heuristic. Note also that GDO itself brings significant improvements over Model (1).

In Table 4, observe that the normalized distance for solution from GDO for the large (L) instance first increases from 0.45% to 11.40%, and then follows the *expected* decreasing trend with increase in computational time limit. Recall that GDO first minimizes the total trailer capacity required in Model (1) and then solves Model (2) to minimize the Hamming distance of the solution (and the flow diversion cost) from the reference load plan. As shows in Table 3 the feasible solution obtained from Model 1 is of poor quality and is closer to the reference load plan in terms of the number of trailers. Hence, the normalized distance value is small. As the computational time limit increases, the feasible solution obtained from Model (1) exhibits a reduced total trailer capacity compared to the reference load plan. Hence, the

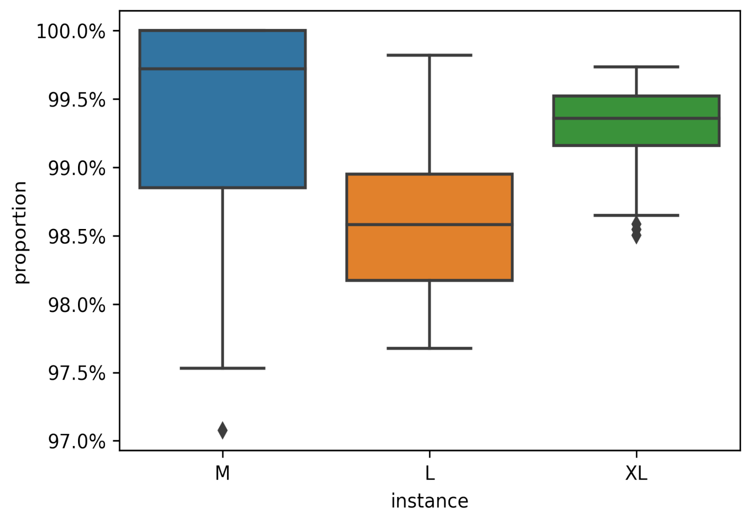


Figure 11: *Total Predicted Trailer Capacity as a Percentage of the Total Trailer Capacity in the Final Solution Obtained by Proxies.*

normalized distance increases as the model tries to find more cost effective solutions. With further increases in computational time, the normalized distances decrease as the solver finds a better solution with a smaller Hamming distance using Model 2.

It is also interesting to observe that the total trailer capacity predicted by the ML model, i.e., the capacity provided by all the trailers predicted to be needed by the ML model, is very close to a feasible solution. Only a few trailers must be added to recover a feasible solution. Figure 11 shows the distribution of the predicted trailer capacity as a percentage of the total trailer capacity in the feasible solution generated by the proxies for each type of terminal. The results show that more than 98% of the trailer capacity is predicted correctly and less than 2% comes from feasibility restoration step generated by algorithm 1. More accurate predictions might even result in a feasibility restoration model that has fewer decision variables, hence, requiring less computational time to produce a feasible solution. Appendix 10.2 shows that one of the key benefits of the optimization proxies is that it replaces a model with large number of integer decision variables with a prediction model, and requires to solve a relatively simpler feasibility restoration model with small number of binary variables.

7.4 Value of Symmetry-breaking Data Generation

As discussed in section 4, the optimal (or near-optimal) trailer decisions of Model (1) are very sensitive to changes in total commodity volume due to the presence of symmetries in the model and the randomized nature of MIP solvers. The solutions to Model (1) are reported in *red* in the plots of Figure 12, which illustrates this behavior. This is not desirable in environments with planners-in-the-loop, where similar solutions are expected for similar instances. The GDO approach is much more consistent and its solutions are shown in *orange* in the plots of Figure 12. The ML component of the optimization proxy uses GDO as the *expert* to imitate and learn solution patterns from. As shown in *blue* in the plots of Figure 12, the ML model is effective in producing solutions that are close to the solutions generated by GDO. It should be highlighted that the GDO approach has two benefits. First, it generates consistent solutions that are amenable to planner-in-the-loop environments. Second, it makes the learning problem much more tractable. Designing an ML model for (1) is really challenging due to the high sensitivities in small changes: typically an ML model for learning Model (1) would return an average value.

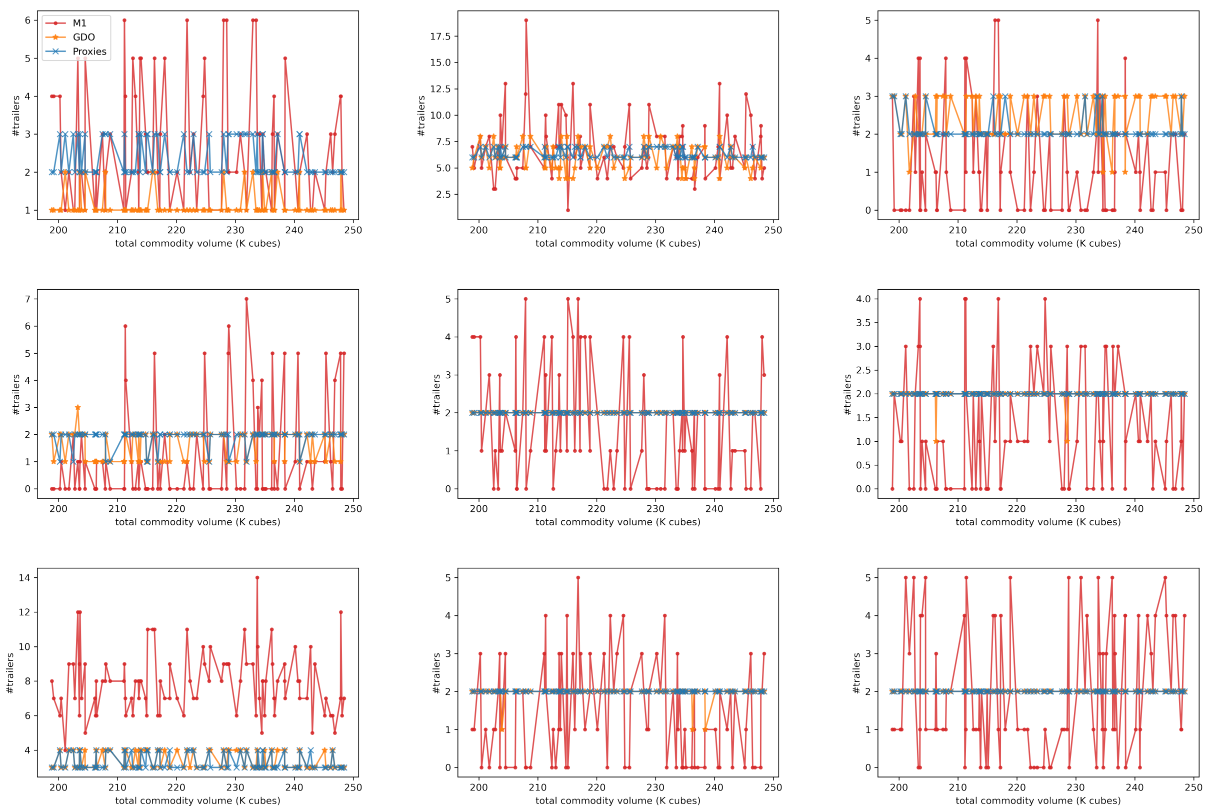


Figure 12: *Illustration of the sensitivity of trailer decisions to changes in total commodity volume on different sort pairs for medium (M) instances. Trailer decisions of Model 1 (M1) are very sensitive to changes in total commodity volume, whereas GDO generates consistent load plans. The Proxies imitate the GDO as an "expert" to learn solution patterns and produce consistent solutions.*

8 Benefits of Dynamic Load Planning, Optimization, and Learning

This section discusses the benefits of the dynamic load planning approach, optimization, and learning. The load planning methodology studied in this paper is based on the concepts of primary and alternate flow paths. With the availability of optimization models, it is possible to evaluate the benefits of this approach for load consolidation, at least from a local perspective.

The results in the paper also make it possible to evaluate the benefits of optimization compared to human planners. During operations, planners typically assign commodities to the primary flow paths. If there is no capacity available on the primary flow path, then planners allocate the remaining volume on the first alternate flow path and, if there is no capacity on the first alternate, they turn to the second alternate flow path, and so on. Observe that this is a greedy strategy of loading commodity volume on trailers, and hence, it is myopic in nature. A comparison between such a greedy approach and the optimization models help assess the value of optimization. Of course, the optimization models are too slow to be used with planners in the loop. The optimization proxies proposed in the paper are the technology enabler for translating the benefits of optimization into practice.

The first results in this section aim at quantifying the value of a network with alternate flow paths relative to a network with primary flow options only. Figure 13 presents some characteristic of the networks studied in this paper: it shows the distribution of the number of commodities with a specific number of alternate flow paths for each instance. It highlights that the network has some significant flexibility for many of its commodities.

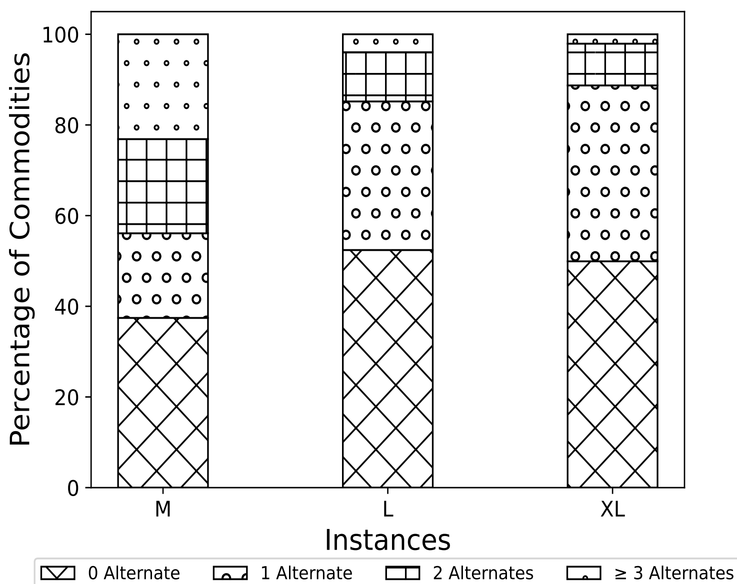
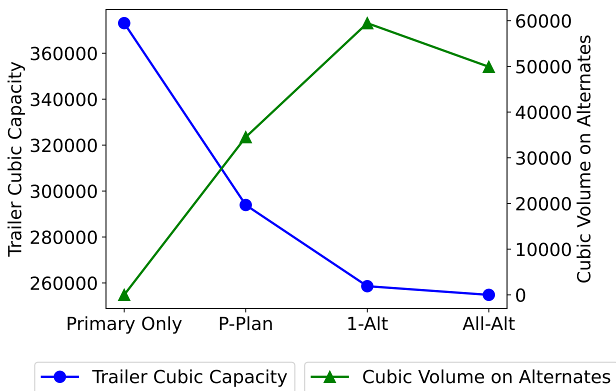
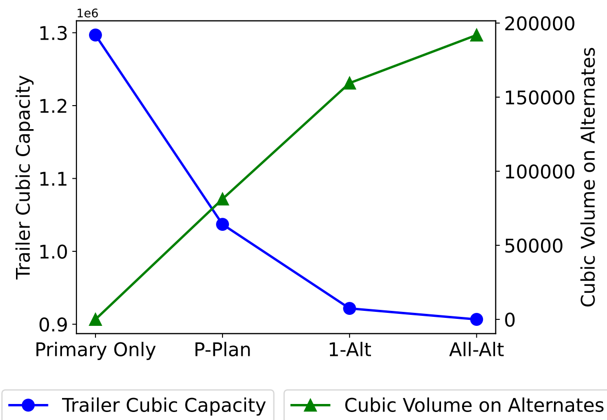


Figure 13: *Instance Statistics About the Range of Alternate Flow Paths For Commodities.*

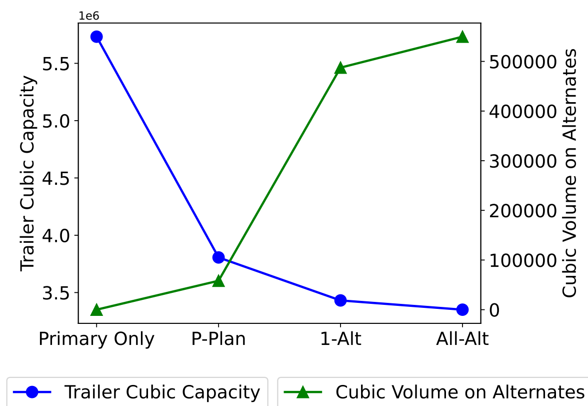
Figure 14 presents the benefits of the load planning methodology. It compares the variation in trailer cubic capacity required to containerize the *total commodity volume* (blue curve) and the *total volume allocated to alternate flow paths* (green curve) across four different load plans: *Primary Only*, *Reference Plan*, *1-Alt* and *All-Alt* for the three instance categories. In the *Primary Only* plan, each commodity can be assigned only to its primary flow path. The *Reference Plan*, referred to as the *P-Plan*, is the reference load plan from our industry partner. Note that in the reference plan each commodity can use any number of compatible alternate flow paths. In the *1-Alt* plan, each commodity can be assigned to either its primary path or the cheapest alternate path. In the *All-Alt* plan, each commodity can be assigned to all the available paths, i.e. splitting is allowed. Observe that the curves are on different scales: the left scale for the blue curve and the right scale for the green curve. The *P-Plan* is produced by the planners using the greedy approach proposed earlier.



(a) *M*



(b) *L*



(c) *XL*

Figure 14: *Variation in trailer cubic capacity and total volume allocated to alternate flow paths in four load plans.*

Figure 14 demonstrates a consistent trend in cubic capacity required in the four different load plans: the capacity monotonically decreases and the decreases are significant. Allowing spittability of commodity volume across primary and alternate flow paths improves trailer consolidation. These benefits are already apparent in the *P-Plan* of the planners, despite the fact that this is a greedy approach. The optimization model with a single alternate flow path, i.e., the *1-Alt* plan, brings another step change in consolidation, highlighting the benefits of optimization. This benefit stems for the fact that a large number of commodities have at least one alternate flow path in all instances (see Figure 13). Note also that the *1-Alt* load plan requires significantly smaller total trailer capacity than the *P-Plan*, although the *P-Plan* has the flexibility of using any number of alternate flow paths. The *All-Alt* plan brings further benefits but they are rather incremental. Part of the reasons comes from the fact that a relatively small fraction of the commodities have more than one alternate flow path. It would be interesting to study a network with more flexibility as this may bring further load consolidation benefits.

There is an interesting phenomenon that appears in the medium-sized instance *M*: the volume assigned on the alternate flow paths decreased when moving from *1-Alt* to an *All-Alt* plan. This comes from the fact that this instance has many commodities, with a smaller volume, that have new alternate flow paths options available in the *All-Alt* setting. As a result, commodities with larger volume are allocated to their primary flow path (as the flow diversion cost is proportional to the total commodity volume assigned to alternate flow paths) and the commodities with smaller volume can be allocated to the alternate flow path that is the primary for the commodities with larger volume (and not the cheapest alternate path of the *1-Alt* setting). Hence, the total volume assigned to the alternate flow paths reduces, although the total number of commodities that use alternate flow paths increases.

Figure 15 compares the percentage of the total commodity volume that is assigned to the alternate flow paths in the *P-Plan* and the *All-Alt* plan. It is undesirable to allocate a major proportion of the volume to the alternate flow paths because the downstream buildings may not be better equipped to handle or process the large inbound volume. Observe that, on average across all the instances, the *All-Alt* plan (resp. *P-Plan*) allocates around 17% (resp. 9%) commodity volume on the alternate flow paths. The *All-Alt* plan reduces the total trailer capacity by roughly $12\% - 15\%$ relative to the *P-Plan*. For the XL instance, there is a significant gap between the *P-Plan* and *All-Alt* plan statistics because most of the commodities in the *P-Plan* are allocated to the primary flow paths. This is why the total commodity volume allocated to the alternate flow paths in the *P-Plan* and the *Primary Only* have a small difference; see Figure 14c for XL category.

These results show that optimization proxies can bring substantial benefits in practice. They provide, in real time, significant improvements over the existing planning process. Moreover, by virtue of their training mimicking the GDO optimization, that makes sure that plans evolve smoothly during the planning process: small changes in inputs will not result in large changes in the proposed solutions.

These results are eminently practical. One of the challenges in the operational setting is the need for additional trailers when the total commodity volume increases. Planners can acquire these trailers either through empty trailer repositioning or by engaging in short-term trailer leasing with local companies. Conversely, if the commodity volume decreases, planners are left with a plan that has low trailer utilization. The optimization proxies address this issue directly. Planners can also use the proposed optimization proxies to obtain recommendations for load plan adjustment in the event of a disruption (due to uncertainty in commodity volume), even for the largest terminal, within a matter of seconds. Furthermore, the recommendations from the optimization proxies are consistent with existing load plans, which makes it

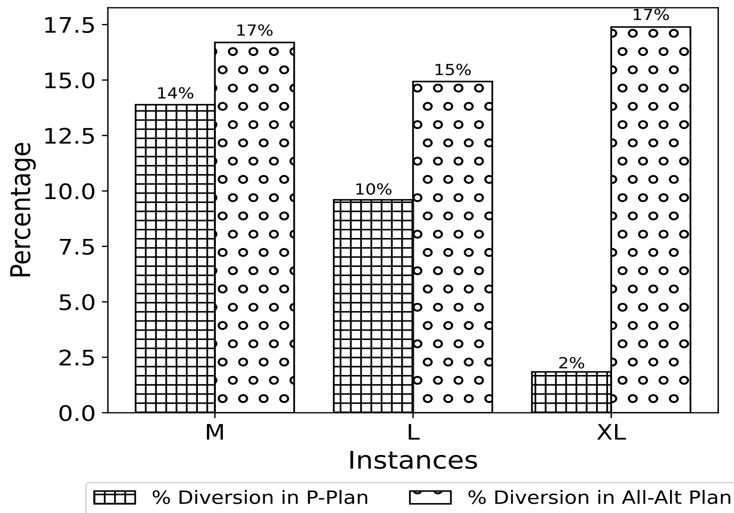


Figure 15: *Proportion of total commodity volume allocated to alternate flow paths in the P-Plan and All-Alt Plan*

easy for the planners to evaluate and implement the suggestions. Finally, new terminals in the service network often do not have dedicated planners to develop load plans and extra capacity is built in the system to handle the commodity volume in the worst-case scenario. Optimization proxies can be used as a decision support tool at such terminals.

9 Conclusions and Future Work

This paper studies the Dynamic Load Planning Problem (DLPP) that considers both load and flow planning challenges jointly in order to adjust loads and flows as the demand forecast keeps changing over time before the day of operations. The paper is motivated by the need of a decision-support tool to advice planners making these decisions at terminals across the network. The paper formulates the problem as a MIP and shows that it admits many symmetries. As a result, the optimization solver may return fundamentally different solutions to closely related problems (i.e., DLPPs with slightly different inputs), confusing planners and reducing trust in optimization. To remedy this limitation, the paper proposes a Goal-Directed Optimization (GDO) that eliminates those symmetries by generating optimal solutions staying close to a reference plan. The paper also proposes an optimization proxy, combining learning and optimization, to provide high-quality and consistent load plans. An extensive computational study on industrial instances shows that the optimization proxy is around 10 times faster than the commercial solver in obtaining the same quality solutions and orders of magnitude faster for generating solutions that are consistent with each other. The proposed approach also highlights the benefits of the DLPP for load consolidation, and the significant savings from the combination of machine learning and optimization.

This research is the first stage of a multi-stage project with our industry partner (a large parcel carrier) for solving load planning problems. Future research will extend the proposed approach to clusters of terminals, taking into account their capacities for processing commodities. The resulting problem thus requires to determine both inbound and outbound planning decisions at each terminal, which significantly complicates the optimization and learning models.

Acknowledgement

This research was partly supported by the NSF AI Institute for Advances in Optimization (Award 2112533).

References

Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*.

Bakir, I., Erera, A., & Savelsbergh, M. (2021). Motor carrier service network design. In *Network design with applications to transportation and logistics* (pp. 427–467). Springer.

Baubaid, A., Boland, N., & Savelsbergh, M. (2021). The value of limited flexibility in service network designs. *Transportation Science*, *55*(1), 52–74.

Baubaid, A., Boland, N., & Savelsbergh, M. (2023). The dynamic freight routing problem for less-than-truckload carriers. *Transportation Science*, *57*(3), 717–740.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405–421.

Boland, N., Hewitt, M., Marshall, L., & Savelsbergh, M. (2017). The continuous-time service network design problem. *Operations research*, *65*(5), 1303–1321.

Chatzos, M., Mak, T. W. K., & Van Hentenryck, P. (2022). Spatial network decomposition for fast and scalable ac-opf learning. *IEEE Transactions on Power Systems*, *37*(4), 2601–2612. https://doi.org/10.1109/TPWRS.2021.3124726

Chen, W., Khir, R., & Van Hentenryck, P. (2023). Two-stage learning for the flexible job shop scheduling problem. *arXiv preprint arXiv:2301.09703*.

Chen, W., Park, S., Tanneau, M., & Van Hentenryck, P. (2022). Learning optimization proxies for large-scale security-constrained economic dispatch. *Electric Power Systems Research*, *213*, 108566. https://doi.org/https://doi.org/10.1016/j.epsr.2022.108566

Chen, W., Tanneau, M., & Hentenryck, P. V. (2023). End-to-end feasible optimization proxies for large-scale economic dispatch. *CoRR*, *abs/2304.11726*. https://doi.org/10.48550/arXiv.2304.11726

Crainic, T. G. (2000). Service network design in freight transportation. *European journal of operational research*, *122*(2), 272–288.

Crainic, T. G., & Laporte, G. (1997). Planning models for freight transportation. *European journal of operational research*, *97*(3), 409–438.

Csirik, J. (1991). Heuristics for the 0-1 min-knapsack problem. *Acta Cybernetica*, *10*(1-2), 15–20.

Erera, A., Hewitt, M., Savelsbergh, M., & Zhang, Y. (2013a). Creating schedules and computing operating costs for ltl load plans. *Computers & Operations Research*, *40*(3), 691–702.

Erera, A., Hewitt, M., Savelsbergh, M., & Zhang, Y. (2013b). Improved load plan design through integer programming based local search. *Transportation Science*, *47*(3), 412–427.

Fioretto, F., Mak, T. W. K., & Hentenryck, P. V. (2020). Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 630–637. https://ojs.aaai.org/index.php/AAAI/article/view/5403

Friesen, D. K., & Langston, M. A. (1986). Variable sized bin packing. *SIAM journal on computing*, *15*(1), 222–230.

Fu, Z.-H., Qiu, K.-B., & Zha, H. (2021). Generalize a small pre-trained model to arbitrarily large tsp instances. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(8), 7474–7482.

Gurobi Optimization, LLC. (2023). Gurobi Optimizer Reference Manual. https://www.gurobi.com

Herszterg, I., Ridouane, Y., Boland, N., Erera, A., & Savelsbergh, M. (2022). Near real-time loadplan adjustments for less-than-truckload carriers. *European Journal of Operational Research*, *301*(3), 1021–1034.

Hewitt, M. (2019). Enhanced dynamic discretization discovery for the continuous time load plan design problem. *Transportation science*, *53*(6), 1731–1750.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456.

Karp, R. M. (2010). *Reducibility among combinatorial problems*. Springer.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, *30*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kool, W., Van Hoof, H., & Welling, M. (2018). Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Kotary, J., Fioretto, F., & Van Hentenryck, P. (2021). Learning hard optimization problems: A data generation perspective. *Advances in Neural Information Processing Systems*, *34*, 24981–24992.

Kotary, J., Fioretto, F., & Van Hentenryck, P. (2022). Fast approximations for job shop scheduling: A lagrangian dual deep learning method. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(7), 7239–7246.

Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Lindsey, K., Erera, A., & Savelsbergh, M. (2016). Improved integer programming-based neighborhood search for less-than-truckload load plan design. *Transportation science*, *50*(4), 1360–1379.

Marshall, L., Boland, N., Savelsbergh, M., & Hewitt, M. (2021). Interval-based dynamic discretization discovery for solving the continuous-time service network design problem. *Transportation science*, *55*(1), 29–51.

PACE. (2017). *Partnership for an Advanced Computing Environment (PACE)*. http://www.pace.gatech.edu

Park, S., Chen, W., Han, D., Tanneau, M., & Van Hentenryck, P. (2022). Confidence-aware graph neural networks for learning reliability assessment commitments. *arXiv preprint arXiv:2211.15755*.

Park, S., & Van Hentenryck, P. (2023). Self-supervised primal-dual learning for constrained optimization. *Proceedings of the 37th AAAI Conference on Artificial Intelligence, Washington, DC*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Powell, W. B. (1986). A local improvement heuristic for the design of less-than-truckload motor carrier networks. *Transportation Science*, *20*(4), 246–257.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Song, W., Chen, X., Li, Q., & Cao, Z. (2022). Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, *19*(2), 1600–1610.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stanley, M. (2022). https://www.morganstanley.com/ideas/global-ecommerce-growth-forecast-2022

Ulch, D. (2022). *Greedy approaches for large-scale flow and load planning* (Doctoral dissertation). Georgia Institute of Technology.

Yuan, E., Chen, W., & Van Hentenryck, P. (2022). Reinforcement learning from optimization proxy for ride-hailing vehicle relocation. *Journal of Artificial Intelligence Research*, *75*, 985–1002.

10 Appendix

10.1 Complexity Results

Model 1 is difficult to solve because in addition to determining the right combination of trailer types to contain volume on each arc, we need to determine the right splits of commodity volume on the given set of compatible arcs. We will analyze the complexity of Model 1 using the special cases described below.

Case 1: There is only one trailer type available at the terminal, i.e., $|V_s| = 1 \; \forall s \in S$. Each commodity $k \in K$ is compatible with exactly one sort pair s_k, i.e., $S^k = \{s_k\} \; \forall k \in K$

Case 2: There is only one trailer type available at the terminal, i.e., $|V_s| = 1 \; \forall s \in S$. Each commodity $k \in K$ is compatible with all sort pairs, i.e., $S^k = S \; \forall k \in K$

Proposition 3. *Cases* 1 *and* 2 *are polynomial time solvable*

Proof. In *Case* 1, the volume of each commodity k is assigned to its only compatible sort pair, s_k, i.e. $x^k_{s_k} = q^k$. Then, the optimal solution has $y_s = \left\lceil \frac{\sum_{k \in K: s \in S^k} x^k_s}{Q} \right\rceil = \left\lceil \frac{\sum_{k \in K: s \in S^k} q^k}{Q} \right\rceil \; \forall \; s \in S$.

In *Case* 2, the optimal solution is to assign the volume of all commodities on any sort pair $s \in S$ and set $x^k_s = q^k \; \forall k \in K$, $y_s = \left\lceil \frac{\sum_{k \in K} q^k}{Q} \right\rceil$, $y_{s'} = 0 \; \forall \; s' \in S, s' \neq s$. □

Case 3: Same as *Case 1*, but with more than one trailer type available at the terminal

Case 4: Same as *Case 2*, but with more than one trailer type available at the terminal

Proposition 4. *Cases* 3 *and* 4 *are weakly NP-Hard*

Proof. In the optimal solution in *Case* 3 the volume of each commodity k is assigned to its only compatible sort pair s_k. Thus, it remains to decide the optimal combination of trailer types required to containerize the volume on every sort pair. This is the minimum knapsack problem (see Csirik, 1991 for the problem definition) for each sort pair (that has more than one trailer type) as shown in 14 which is known to be weakly NP-Hard.

$$\text{For every } s \in S\text{: } \underset{y}{\text{Minimize}} \quad \sum_{v \in V_s} c_v y_{s,v} \tag{14a}$$

$$\text{subject to} \quad \sum_{k \in K: s \in S^k} q^k \leq \sum_{v \in V_s} Q_v(y_{s,v}), \tag{14b}$$

$$y_{s,v} \in \mathbb{Z}_{\geq 0} \qquad \forall v \in V_s \tag{14c}$$

Similarly, for *Case* 4 there exists an optimal solution in which the volume of all commodities is assigned to one sort pair $s^* \in S$, i.e. $x^k_{s^*} = q^k \; \forall k \in K$ and it remains to solve a minimum knapsack problem for the sort pair s^* due to which *Case* 4 is weakly NP-Hard. □

Case 5: Each commodity $k \in K$ is compatible with a subset of sort pairs, i.e., $S^k \subset S$, and has unit volume, $q^k = 1$. There is only one trailer type with per-unit cost $c_s = 1 \; \forall s \in S$ and capacity $Q = \max_{s \in S}\{\sum_{k \in K} \mathbb{1}_{s \in S^k}\}$; hence, $y_s \in \{0,1\} \forall s \in S$, as installing one unit of trailer is enough to containerize the total volume that can be assigned to the sort pair. Note that we ignore the index v for trailer because each sort pair has exactly one and the same trailer type.

Claim 5. *In the optimal solution of Case 5, each commodity is assigned to exactly one compatible sort pair (i.e. there is no splitting of volume)*

Proof. We will present a proof by contradiction. WLOG, suppose there exists an optimal solution in which the volume of a commodity $\hat{k}$ is split between two sort pairs and the volume of all other commodities $k \in K \backslash \{\hat{k}\}$ is assigned to exactly one sort pair s_k. Thus, we have $x^k_{s_k} = q^k \; \forall k \in K \backslash \{\hat{k}\}$ and $x^{\hat{k}}_{s_1} + x^{\hat{k}}_{s_2} = q^k$. Consider a solution $\underline{x}^k_s = x^k_s \; \forall k \in K \backslash \{\hat{k}\}$ and $\underline{x}^{\hat{k}}_{s_1} = x^{\hat{k}}_{s_1} + \epsilon, \underline{x}^{\hat{k}}_{s_2} = x^{\hat{k}}_{s_2} - \epsilon$, where $\epsilon > 0$ is a small real number. Note that $\underline{x}^{\hat{k}}_{s_1} + \underline{x}^{\hat{k}}_{s_2} = q$. Consider another solution $\bar{x}^k_s = x^k_s \; \forall k \in K \backslash \{\hat{k}\}$ and $\bar{x}^{\hat{k}}_{s_1} = x^{\hat{k}}_{s_1} - \epsilon, \bar{x}^{\hat{k}}_{s_2} = x^{\hat{k}}_{s_2} + \epsilon$. Note that both solutions $\underline{x}$ and $\bar{x}$ satisfy constraints (1b) and are feasible to constraints (1c) because we choose $Q = \max_{s \in S}\{\sum_{k \in K} \mathbb{1}_{s \in S^k}\}$. The solution x can be written as a convex combination of the solution $\underline{x}$ and $\bar{x}$ ($x^k_s = \frac{1}{2}\bar{x}^k_s + \frac{1}{2}\underline{x}^k_s \; \forall k \in K, s \in S^k$) which contradicts the optimality of the solution. □

Proposition 6. *Case 5 is strongly NP-Hard*

Proof. We will show that this special case can be solved as a set cover problem which is known to be strongly NP-Hard (Karp, 2010). An instance of a set cover is given by a ground set $U = \{x_1, x_2, \cdots, x_n\}$ and a collection of m subsets $E_i \subseteq U \; \forall i \in \{1, 2, \cdots, m\}$ of the ground set U. The optimization problem is to find the smallest number of subsets $i \in \{1, 2, \cdots, m\}$ such that $\bigcup_{i \in \{1,2,\cdots,m\}} E_i = U$.

From claim 5 we know that each commodity is assigned to exactly one compatible sort pair in the optimal solution. Let commodity $k \in K$ denote element $x_k \in U$, $|K| = n$ and set of sort pairs $S = \{1, 2, \cdots, m\}$. Define $K_i = \{k \in K : x_k \in E_i\}$ as the set of commodities or elements that can be covered by selecting sort pairs $i \in \{1, 2, \cdots, m\}$. Now note that finding the smallest number of sort pairs $s \in S$ such that all commodities in K are covered is equivalent to finding the smallest number of subsets $i \in \{1, 2, \cdots, m\}$ to cover all elements in U. □

10.2 Additional Experimental Results

Table 6 compares the the number of integer decision variables in Model 1 and the average number of binary decision variables in Model 8 across multiple test instances. The number of integer decision variables remain the same for each instance category because it depends in the number of arcs or sort pairs and trailer types; only the commodity volume changes across different test instances. However, the size of the feasibility restoration model 8 depends on the predictions of the ML model. Recall that the ML model predicts the value of the integer decision variables of Model 1. Hence, if the predictions are accurate, then fewer sort pairs would have capacity violations. Consequently, there would be fewer binary decision variables in Model 8; the number of binary decision variables in Model 8 is equal to the number of sort pairs with capacity violation. As the ML predictions can vary for different test instance with the same set of sort pairs due to different commodity volume, the number of binary variables in Model 8 can be different for different test instances. This is why the Table 6 reports fractional values for the average number of binary variables. It is worth highlighting that one of the key benefits of the optimization proxies is that it replaces a model with large number of integer decision variables with a prediction model, and requires to solve a relatively simpler model with small number of binary variables.

Instance	#Integer-Vars	Average #Binary-Vars
M	187	1.3
L	846	19.5
XL	4,475	35.5

Table 6: *Comparing the number of integer decision variables in Model 1 with the the average number of binary decision variables in the feasibility restoration Model 8*