Near Real-Time Loadplan Adjustments for Less-than-Truckload Carriers

Yassine Ridouane, Ian Herszterg, Natashia Boland, Alan Erera, and Martin Savelsbergh

Georgia Institute of Technology, USA

Abstract

We design and implement decision support technology to assist dispatchers in the daily management of loadplans in less-than-truckload service networks. The freight volume that enters a service network on the day of operations deviates from the forecast freight volume used to create the loadplan. These deviations cause inefficiencies when the capacity on planned freight paths is no longer sufficient and delays result in missed service promises. Near real-time loadplan adjustments, i.e., rerouting freight on alternate paths, can improve on-time performance without incurring additional cost (e.g., without purchasing additional capacity). The problem of identifying effective alternate freight paths is modeled on a time-expanded network and fast heuristics are developed for its solution in order to ensure that there is sufficient time to put the adjusted loadplan in place. The loadplan adjustment technology has been extensively tested using data from a large US less-than-truckload carrier. The results show that on-time performance can be improved without increasing cost, i.e., by rerouting freight and using existing capacity in the service network.

keywords: Large scale optimization; Transportation; Load planning; Heuristics.

1 Introduction

Ground transportation forms the backbone of many economies as it can cost-effectively connect dispersed supply and demand. We focus on the Less-than-truckload (LTL) segment, which, in the United States, represents a $40 billion industry with about 25 major players. It handles shipments with a weight ranging between 120 and 10,000, more than what parcel carriers handle, but too small for truckload transportation. To be profitable, LTL carriers have to consolidate shipments from different shippers so as to increase trailer utilization and minimize the “air” transported.
In the last decade, the share of Less-than-Truckload within the trucking industry has grown as a result of economic trends, e.g., e-commerce with its aggressive service guarantees. The rise in B-to-B and B-to-C freight handled by LTL carriers has put additional pressure on daily operations. Unfortunately, the use of weekly driver schedules, prepared in advance (typically a few days in advance for many major US carriers), provides little, if any, flexibility to adjust them on the day of operations. Given increasing day-to-day freight volume fluctuations, this is a major challenge, which is exacerbated by more and more aggressive service promises and a highly competition environment. Thus, there is need for decision support technology that can, in near real-time, suggest changes to a loadplan on the day of operations based on observed freight volumes.

Fortunately, access to more and more real-time information regarding operations has become available, e.g., shipment volume and vehicle location information. Many, if not all, major players have invested heavily in equipping their facilities and vehicles with technology and intelligence that meets their needs for real-time information. Moreover, advances in computing power and infrastructure have made it more realistic to expect that near real-time optimization-based decision support is feasible.

In this paper, we discuss the design and implementation of several efficient heuristics for re-routing shipments to their destination without altering driver schedules (i.e., planned vehicle movements) so as to improve on-time performance.

Currently, shipments arriving at a terminal are typically handled in a first in, first out (FIFO) fashion. Terminal operators assign arriving shipments to the trailer on the planned path to the shipment’s destination with the earliest (feasible) departure time in order to avoid accumulation of (too many) shipments in front of dock doors and to get the shipment closer to its destination as soon as possible. Such a strategy helps achieve performance targets set for the terminal, but ignores how these decisions affect the performance of the entire system. Furthermore, such a strategy facilitates accommodating last minute customer reservations and unexpected shipment arrivals (e.g., sent by other terminals that had to deal with an unexpectedly high freight volume). Our heuristics seek to find the right balance between local and system-wide performance.

The contributions of the research discussed in this paper can be summarized as follows:

- We introduce a variety of metrics related to service quality that help quantify the benefits of real-time adjustments to loadplans;
- We present a number of fast heuristics to suggest loadplan adjustments that substantially improve on-time performance;
- We demonstrate the practical viability and value of these heuristics in a computational study using real-life data from a large US LTL carrier.

The remainder of the paper is organized as follows. In Section 2, we formally define the problem, present a formulation of the problem, and discuss the scope and goals of the research. In Section 3, we introduce different heuristics to solve realistic, large-size instance of the problem. In Section 4, we summarize and analyze the results of a set of computational
experiments. In Section 5, we highlight the practical value of the proposed heuristics and discuss ongoing research to further enhance the technology.

2 Problem Statement

LTL carriers seek to achieve high trailer utilization, especially when moving trailers over longer distances, by employing a hub-and-spoke network, represented schematically in Figure 1. After being collected from customers, shipments are brought to an End-of-Line terminal, where they are sorted, consolidated, and dispatched to a Breakbulk terminal. Breakbulk terminals serve as central hubs for a region and consolidate shipments from different End-of-Line terminals in order to ensure high load factors (i.e., high utilization) for trailers departing to other Breakbulk terminals. Shipments in trailers arriving at a Breakbulk terminal are sorted, consolidated, and either dispatched to another Breakbulk terminal or to an End-of-Line terminal in the region for final delivery. For a more detailed description of LTL operations see Erera et al. (2013b).

Figure 1: A hub-and-spoke LTL network. The dotted red path represents a load from End-of-Line E1 to End-of-Line E2 involving four schedule legs $E_1 \rightarrow B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow E_2$.

Consolidation carriers have traditionally focused their decision support efforts on planning rather than execution. In load planning, given a forecast of freight volumes between origins and destinations, the goal is to identify origin-destination paths that meet service guarantees, that are likely to lead to effective consolidations, and that are not too costly, i.e., that provide the right balance between route circuity and trailer utilization.

3
An origin-destination path is defined by a sequence of (intermediate) terminals (possibly empty) where shipments are unloaded from one trailer and loaded into another trailer. The set of all origin-destination paths is commonly referred to as the **planned flow**. The planned flow dictates where a shipment is sent next, given that it becomes available at a specific terminal at a specific time of day. If actual freight flows match forecast freight flows, then using the planned flow paths should minimize cost while ensuring service guarantees are met. However, in practice, actual freight flows rarely match forecast freight flows, and it may not be possible to meet service guarantees relying solely on the planned flow paths. Therefore, in practice, a set of alternate origin-destination paths is constructed and used when capacity on a planned flow path is insufficient. The trailer movements are not affected and are executed as planned, but the path from origin to destination for (some) shipments is changed. The set of all alternate origin-destination paths is commonly referred to as the **alternate flow**.

There is abundant literature on network design for the LTL trucking industry. However, there is scant literature on near real-time dynamic load planning for service networks. A taxonomy of service network planning has been introduced in Crainic and Laporte (1997) and is widely used to delineate strategic, tactical, and operational planning. The bulk of the literature focuses on strategic and tactical planning, and only a few papers consider operational planning, which is the topic of our research. To the best of our knowledge, no prior work has investigated the use of alternative paths to handle daily demand fluctuations in an operational setting. The value of introducing (planned) alternative paths in a service network to hedge against demand uncertainty has been highlighted in Baubaid et al. (2019). The authors demonstrate that it is sufficient to have a single alternative option at the terminals visited along an origin-destination path to absorb most of the demand uncertainty. Due to the fact that integer programming formulations (based on a network or a time-expanded network representation) of tactical service network design problems are difficult to solve, especially for realistic instance sizes, most solution approaches are heuristic. The survey papers by Crainic (2000) and Wieberneit (2008) summarize much of the research in this area. Examples of papers proposing meta-heuristics include Farvolden and Powell (1994), Barcos et al. (2010), Crainic et al. (2000), Ghamlouche et al. (2004) and Erera et al. (2013a), proposing Lagrangean heuristics include Holmberg and Yuan (1998) and Katayama and Yurimoto (2016), and proposing slope scaling heuristics include Jarrah et al. (2009).

The importance of using near real-time information to optimize freight transportation operations on the day of operations is highlighted in Gendreau et al. (1996), Regan et al. (1996), Yang et al. (1999), Cheung and Muralidharan (1999), Cheung and Muralidharan (2000), Roy (2001), Roy (2005), Powell (2003), and Hejazi and Haghighi (2007). Powell et al. (2002) discuss the benefits and difficulties associated with implementing near real-time optimization models in the motor carrier industry, stressing the importance of acknowledging the human factor in decision making. The survey paper on intelligent freight network systems by Crainic et al. (2009) includes an overview of the opportunities, but also the challenges, that access to real-time information offers. The authors argue that operations research techniques need to be leveraged to bring additional value to motor carriers and increase
their agility in the modern fast-paced environment. In the aforementioned papers, different types of formulations are used to model network design problems. Arc-based formulations (in which an arc represents a direct trailer movement between two terminals) are suggested by Ghamlouche et al. (2004), Holmberg and Yuan (1998), and Katayama and Yurimoto (2016). Path-based formulations (where a path represents a sequence of direct movements that take a shipment from its origin to its destination) are suggested in Rothenbächer et al. (2016) and Crainic et al. (2000). Tree-based formulations (where a tree represents direct movements that take shipments from their origin to a particular destination) are suggested in Barnhart et al. (2002) and Jarrah et al. (2009).

In a setting more similar to the one we consider, Simao and Powell (2018) propose an integer programming based look-ahead formulation to solve a dynamic load planning problem. The formulation is based on the current state of the network, i.e., the shipments in the system, the shipments forecast to enter the system, the set of drivers and their assigned trailer movements, and is solved using a type of “relax-and-fix” approach, in which the formulation is decomposed into multiple subproblems based on a discretization of the time horizon, and subproblems are solved in order of time. The solution approach is tested on data from a reasonably large LTL carrier in the U.S. (with a network of about 300 terminals with about 40,000 daily shipments) and shows positive results in terms of trailer utilization and on-time delivery performance.

We focus on the design and implementation of decision technology to route shipments on their planned or alternate paths given the latest information on the freight already in or anticipated to enter the service network so as to ensure the highest possible on-time performance. The technology will support central dispatchers, in near real-time, when they are faced with deviations from expected freight volumes. When the decision technology is invoked it considers three sets of shipments: shipments that have been picked up and are available for dispatch at a terminal at the start the planning period, shipments that are en-route at the start of the planning period and are expected to reach their next intermediate terminal within the planning period, and shipments that are expected to be available for dispatch at a terminal at some later time during the planning period (referred to as forecast shipments). Given the trailer movements that are to be executed during the planning period, we may find that all shipments can be delivered at their destination on-time using the planned flow, or some shipments will be delivered late if only the planned flow is used.

We assume a planning period of 48 hours starting at a specific time during the day, typically 6pm. The choice of the length of the planning period is motivated by the fact that the fastest growing offerings of freight transportation companies are next-day and two-day delivery. Consequently, the majority of freight in the system will reach its final destination within 48 hours. In the remainder, for convenience and ease of presentation, we assume that there are no late shipments at the start of the planning period, and that for each shipment there is a (unique) planned path along which the shipment can reach its final destination on time (if it would be the only shipment in the system and it would not have to compete for capacity). We also assume that planned flow and alternate flow paths are given, and that scheduled capacity is fixed (i.e., there is a given number of planned trailers with known...
dispatch and arrival times between pairs of terminals). The scheduled capacity is such that if the anticipated daily demand realizes the capacity is sufficient to move all the shipments along their planned paths.

To model the problem, we use a time-expanded directed graph \( G = (V, A) \). Let \( S \) be the set of shipments and let \( L \) be the set of trailer movements during the planning period. Each shipment \( s \in S \) has an associated quantity \( qty_s \), origin terminal \( org_s \) (where it enters the network), origin time \( otm_s \) (when it enters the network), destination terminal \( dst_s \) (where it needs to end up), and due time \( due_s \) (when it needs to end up there). Each trailer movement \( l \in L \) has an associated capacity \( cap_l \), origin terminal \( org_l \), departure time at the origin \( dtm_l \), destination terminal \( dst_l \), and arrival time at the destination \( atm_l \). All times are relative to the start of the planning period. A node \( (u, t) \in V \) represents a location \( u \) at a point in time \( t \). The node set \( V \) is partitioned into three subsets: \( V_D, V_A, \) and \( V_E \). The node set \( V_D \) and \( V_A \), respectively, represent the departures and the arrivals of trailers, and the node set \( V_E \), with nodes \( (u, +\infty) \) – one for each terminal \( u \), represents the end of the planning period. The arc set \( A \) is also partitioned into three subsets: \( A_L, A_H, \) and \( A_E \). The arc set \( A_L \) represents the trailer movements in \( L \). An arc \( ((u, t), (u, t + 1)) \in A_H \) models the possibility for shipments to remain at terminal \( u \) from a trailer arrival or departure time \( t \) to the next trailer departure time \( t + 1 \), where the last “holding” arc is \( ((u, t), (u, \infty)) \). An arc \( ((u', \infty), (u, \infty)) \in A_E \) models a shipment being at an intermediate terminal \( u' \) at the end of the planning horizon and its transfer from \( u' \) to its destination terminal \( u \). An arc \( ((u', t), (u, \infty)) \in A_E \) for any terminal \( u' \) and \( t \) after the end of the planning horizon, models the situation where a shipment is en-route at the end of the planning horizon, reaches terminal \( u' \) at time \( t \), and is transferred from \( u' \) to its destination terminal \( u \). Figure 2 shows a part of a time-expanded network for shipments destined to terminal \( C \).

Each shipment enters the system at a source node \( (org_s, otm_s) \in V_D \) and leaves the system at a sink node \( (dst_s, +\infty) \in V_E \). Let \( a = (u, v) \) with \( u, v \in V \) denote an arc in \( A \). Let \( A(s) \) denote the set of feasible arcs for shipment \( s \), i.e., the relevant arcs in \( A_L \) along the planned and the alternate paths for \( s \) and the relevant arcs in \( A_H \) and \( A_E \) at terminals along the planned and the alternate paths for \( s \).

Let \( x_a^s \) denote a binary decision variable representing assigning of shipment \( s \in S \) to arc \( a \in A(s) \) \( (x_a^s = 1) \) or not \( (x_a^s = 0) \). Then the dynamic load planning problem can be
Figure 2: Part of a time-expanded network for shipments destined to Terminal C.

formulated as an arc-based multi-commodity flow problem on $G$ as follows:

\[
(ABF) \min \sum_{s \in S} \left[ \left( \sum_{a \in A(s)} c^s_a x^s_a \right) - due_s \right] + \\
\text{s.t.} \quad \sum_{s \in S} \sum_{a \in A(s)} qty_s x^s_a \leq cap_a \quad \forall a \in A_L \\
\sum_{w \in V} x^s_{(u,w)} = \sum_{w \in V} x^s_{(w,u)} \quad \forall s \in S, u \neq (org_s, otm), u \neq (dst_s, +\infty) \\
\sum_{w \in V} x^s_{((org_s, otm), w)} = 1 \quad \forall s \in S \\
\sum_{w \in V} x^s_{(w, (dst_s, +\infty))} = 1 \quad \forall s \in S \\
x^s_a \in \{0, 1\} \quad \forall s \in S, a \in A(s),
\]

where $cap_a$ represents the capacity of arc $a \in A$, i.e., the capacity of the trailer movement, and $c^s_a$ represents an appropriately chosen time-related cost for sending shipment $s$ along arc $a \in A(s)$. More specifically, let $Est\text{Arrival}(s, o, t)$ be the estimated arrival time of shipment $s$ at its destination given that it will be available at terminal $o$ at time $t$, will follow its planned path, and will use the earliest possible trailer movements along the planned path.
Then the time-related cost $c_a^s$ is given by:

$$c_a^s = \begin{cases} 
0, & \forall a = ((u, t_u), (v, t_v)) \in (A_L \cup A_H) \cap A(s), v \neq dst_s \\
t_v, & \forall a = ((u, t_u), (v, t_v)) \in A_L \cap A(s), v = dst_s \\
EstArrival(s, u, 48), & \forall a = ((u, +\infty), (v, +\infty)) \in A_E \\
EstArrival(s, u, t), & \forall a = ((u, t), (dst_s, +\infty)) \in A_E \text{ with } t > 48
\end{cases}$$

The objective in $ABF$ seeks to minimize the total lateness of shipments. No incentive is given for delivering shipments early, i.e., it suffices to deliver a shipment at or before its due time. If a shipment that does not reach its destination during the planning period, its path ends with an arc $((u, \infty), (dst_s, \infty)) \in A_E$. Consequently, the model seeks to get such shipments as close as possible to their destination (as it captures the projected arrival time at the destination after the end of the planning period). Constraints (1) ensure that trailer movements do not exceed their capacity and Constraints (2), (3), and (4) ensure flow conservation for shipments (i.e., that there is a unique origin-destination path for each shipment).

The formulation above is presented only to provide a formal statement of the problem. Solving this formulation for large-scale instances in a short amount of time is impossible (and the same is true for the path-based formulation). A typical instance of interest has a network with 300 terminals, 10,000 trailer movements, and 75 thousand daily shipments; the formulation will yield a model with 1.5 billion binary variables and 45 million constraints. Given that the goal is to solve the problem several times during the day of operations in less than 15 minutes, it is simply impossible to use a (commercial) IP solver. Consequently, we focus on developing computationally efficient heuristics.

Another layer of complexity stems from the use of sorts to simplify and manage terminal operations. A sort is a time period at a terminal during which shipments are handled and loaded in trailers that will transport them to their next destination. The concept of a sort originated in the small package business where parcels go through a conveyor sortation station in order to be consolidated with other parcels and directed to their respective dock doors. It has been adopted by some carriers in their LTL freight business as well. The idea behind a sort is that all shipments that arrive at the terminal before the cut-off time associated with the sort will be processed during that sort and will be ready to be dispatched by the end of that sort. Shipments that arrive after the cut-off time associated with a sort are held and will be processed in the next sort. Sorts will be considered in the methodologies presented in the next section and provide a mechanism for capturing and handling different organization of operations at terminals. In our motivating setting, there are four sorts within a day of operations: Day, Twilight, Night, and Sunrise. Sort-based strategies seek to minimize the number of shipments held (or rolled over) to subsequent sorts. An example of the organization and use of sorts is given in Figure 3. Cut-off and processing times at terminals can be handled by adjustments to the arcs representing trailer movements, i.e., an arrival after a cut-off time is mapped to the start time of the next sort and processing times are incorporated in the trailer movement times. Thus, for the remainder, we assume that
all trailers arriving in a sort arrive before the cut-off time and shipments are available for dispatch immediately upon arrival.

(a) Type of the activities in a sort. Blue arrows represent shipments that arrive before the cutoff time and can be handled in the sort. Red arrows represent shipments that arrive after the cutoff time and will be rolled over to the next sort.

(b) Example of a shipment delivered through a sequence of terminal sorts from its origin A to its final destination C. $t_o$ represents the time the shipment was made available at terminal A, $t_d$ represents the arrival time at terminal C.

Figure 3: Example of the organization and use of sorts.

3 Methodology

To determine whether adjustments to the loadplan are advantageous given up to date information on the shipments in and entering the service network, we seek to find time-feasible paths for all shipments such that the total (expected) lateness is minimized. That is, given the planned trailer movements, we find a path for each shipment during the planning period, using only planned or alternate flow options, that minimizes the total expected lateness, where, for shipments reaching their destination during the planning period, we will know the actual lateness, and for shipments that do not reach their destination during the planning period, we use an optimistic estimate of their lateness. Given the size of problem, i.e., the large number of planned trailer movements in the planning period and the large number of shipments in or entering the service network during the planning period, and the limited time available, i.e., at most 15 minutes of computing time, we develop greedy, but intelligent, trailer loading heuristics that balance the need for efficiency with the desire for quality.

We start by presenting a baseline heuristic that is shipment focused and assumes that shipments in a sort at a terminal are processed first-in, first-out (FIFO). Although naive, it reflects the viewpoint, which has been popular in practice, that it is beneficial to keep freight moving in the direction of its final destination. Next, we present a heuristic that is trailer movement focused and processes trailer movements in non-decreasing order of departure.
times. After observing that a simple upfront analysis of the trailer movements can identify blocks of trailer movements that can be considered together, we present a heuristic that is block focused and processes blocks in non-decreasing order of departure times of the first trailer movement in a block.

3.1 FIFO Loading

For a given shipment $s \in S$ let $P_s$ denote the \{terminal, sort\} pair where $s$ has become available for dispatch. This can be at the terminal where the shipment enters the linehaul system or at an intermediate terminal on the shipment’s journey from origin to destination. We first check to see if there is a trailer movement departing in $P_s$ after the arrival of $s$ on the planned path for $s$ that has capacity remaining to accommodate $s$. If such a trailer movement exists, we load $s$ in the first such trailer movement, i.e., the one with the earliest departure time, and update $P_s$ (i.e., we set $P_s$ to the \{terminal, sort\} pair defined by the destination of the trailer movement and the sort in which it arrives). If no such trailer movement exists, we check to see if there is a trailer movement departing in $P_s$ after the arrival of $s$ on an alternative path for $s$ that has enough remaining capacity to accommodate $s$. If such a trailer movement exists, we load $s$ in the first such trailer movement and update $P_s$. Finally, if no such trailer movement exists either, we hold $s$ until the next sort (we update $P_s$ accordingly). Heuristic FIFO-Push processes shipments in $S$ non-increasing order of arrival time (at their current location). When a shipment arrives at its destination, it is not reinserted in $S$, which ensures that the heuristic terminates after a finite number of steps. Algorithm 1 shows the pseudo-code for FIFO-Push.
Algorithm 1 FIFO-Push

1: $S \leftarrow$ list of all shipments in the network, sorted by arrival time
2: $L \leftarrow$ list of all trailers in the network departing during the time horizon
3: for each shipment $s$ in $S$ do
4:   $P_s \leftarrow$ pair $\{\text{terminal, sort}\}$ where $s$ is available for pickup
5:   $L^P_s \leftarrow$ subset of trailers in $L$ departing during $P_s$ going though the planned path for $s$
6:   $l_P \leftarrow$ earliest feasible trailer in $L^P_s$, with enough capacity left to load $s$
7:   if $l_P \neq \emptyset$ then
8:     load $s$ in $l_P$ and update $S$
9:   else
10:      $L^A_s \leftarrow$ subset of trailers in $L$ departing during $P_s$ going though one of the alternative paths for $s$
11:      $l_A \leftarrow$ earliest feasible trailer in $L^A_s$, with enough capacity left to load $s$
12:      if $l_A \neq \emptyset$ then
13:         load $s$ in $l_A$ and update $S$
14:      else
15:         hold $s$ and postpone the loading decision to the next sort at the terminal

3.2 Urgency Loading

The second heuristic focuses on trailer movements rather than shipments and processes trailer movements in order of nondecreasing departure times. For a given trailer movement, we have to decide which of the available shipments to load in that trailer movement. As our goal is to minimize total lateness, we use the urgency of a shipment as the basis for making loading decisions.

Consider, again, the function $EstArrival(s, o, t)$, which estimates the arrival time at destination terminal $dst_s$ of a shipment $s$ that is currently at terminal $o$, that departs from that terminal after time $t$, that follows its planned path, and that uses the earliest possible trailer movements along the planned path. We define $Urgency(s, o, t)$ as the urgency of shipment $s$ given that it is currently at terminal $o$ at time $t$:

$$Urgency(s, o, t) = EstArrival(s, o, t) - due_s. \quad (5)$$

A positive value of $Urgency(s, o, t)$ means that the shipment will be late even in the best case scenario, i.e., that it can follow its planned path and can always depart on the earliest trailer movements along the path. A non-positive value means that the shipment is expected to arrive on time at its destination.

For convenience, we will, in the remainder, refer to a trailer movement simply as a trailer. Let $L$ be the set of trailers departing within the planning period in order of non-decreasing dispatch times. For a given trailer $l \in L$, let $S_l$ be the set of shipments available for loading at $org_l$, i.e., every shipment in $S_l$ has arrived at $org_l$ before $dtm_l$. Let $S^P_l$ be the subset
of planned path shipments in \( S_l \), i.e., the subset of shipments for which \( l \) is on the planned path. Let \( S^A_l \) be the subset of alternate path shipments in \( S_l \), i.e., the subset of shipments for which \( l \) is on the alternate path. We start by loading \( l \) with the shipments in \( S^P_l \) in order of non-increasing urgency, and, in case of ties, in order of non-increasing size until there is no capacity left in \( l \) and/or all shipments in \( S^P_l \) have been loaded. If \( l \) has any remaining capacity after processing shipments in \( S^P_l \), we repeat the loading process with the shipments in \( S^A_l \), again loading in order of non-increasing urgency, and, in case of ties, in non-increasing order of size, before moving on to the next trailer. Shipments not loaded during a sort will naturally be processed in subsequent sorts. Algorithm 2 gives the pseudo-code for Urg-Pull. Urg-Pull focuses on moving shipments towards their destination as early as possible and on using as much of the available capacity as possible. Both are “rules of thumb” often used in practice.

**Algorithm 2 Urg-Pull**

1: \( L \leftarrow \) list of all trailers in the network departing during the time horizon, sorted by dispatch time
2: for each trailer \( l \) in \( L \) do
3: \( S^P_l \leftarrow \) list of planned path shipments available for pickup at the origin of \( l \), sorted by urgency then by quantity, both in descending order
4: while there is capacity left in \( l \) and \( S^P_l \neq \emptyset \) do
5: load shipments from \( S^P_l \) in \( l \)
6: if \( l \) has capacity left then
7: \( S^A_l \leftarrow \) list of alternative path shipments available for pickup at the origin of \( l \), sorted by urgency then by quantity in descending order
8: while there is capacity left in \( l \) and \( S^A_l \neq \emptyset \) do
9: load shipments from \( S^A_l \) in \( l \)

Urg-Pull can be implemented efficiently. The trailers are processed one by one in order of nonincreasing departure times. The shipments are maintained in unordered lists, one for each terminal. Initially, a shipment is placed in the list associated with the terminal where it enters the system. Whenever a shipment is loaded and dispatched in a trailer, we update the lists associated with the origin and destination of the trailer by removing the shipment from the list of shipments at the trailer’s origin and inserting it in the list of shipments at the trailer’s destination. Processing a trailer involves going through the list of shipments at its origin, which takes linear time. If a shipment is loaded onto a trailer, deleting it from the list at the origin and inserting it in the list at the destination takes constant time.

Even though moving shipments towards their destination as early as possible is desirable, using alternate flow to do so may not necessarily be best. Therefore, we also consider the variant Urg-Pull-PF which only loads shipments on their planned flow path. Similar to Urg-Pull, we sort trailers in order of nondecreasing departure times and process them one after the other. Each trailer \( l \) is loaded with planned path shipments only (i.e., with shipments in \( S^P_l \)). Even when \( l \) has a remaining capacity, no additional shipments are loaded.
3.3 Block Loading

The myopic nature of Urg-Pull, which attempts to move shipments closer to their destination whenever possible, may result in the use of unnecessarily many alternate paths, which, in turn, may result in too many shipments arriving at a terminal at a time when that terminal does not have enough outbound trailer capacity available to send these shipments on towards their final destination. This reflects the fact that Urg-Pull considers only one trailer at a time and does not look ahead. Next, we introduce block loading, which considers a number of consecutive trailers departing from a terminal simultaneously, which, therefore, addresses one of the limitations of Urg-Pull. Furthermore, we consider several variants of block loading in which we incorporate different look-ahead strategies, which, therefore, addresses another limitation of Urg-Pull.

Let \( PTS \) be the list of \{terminal, sort\} pairs in the planning period. Given a pair \( \{t, s\} \) in \( PTS \), let \( K \) denote the set of shipments available at terminal \( t \) at the start of the sort \( s \) plus any forecast shipments that will become available for loading during sort \( s \) (i.e., before the cut-off time of the sort). Let \( L^D \) denote the set of outbound trailers departing from \( t \) during sort \( s \) and \( L^A \) be the set of inbound trailers arriving at \( t \) during sort \( s \). If \( L^A = \emptyset \), then every shipment available for loading during the sort is known and one could solve a single optimization problem that assigns the shipments in \( K \) to the trailers in \( L^D \). However, when \( L^A \neq \emptyset \), trailers will arrive during the sort and some of the shipments available for loading during the sort are not yet known, namely those that arrive on the trailers in \( L^A \). Consequently, solving a single optimization problem that assigns the shipments in \( K \) to the trailers in \( L^D \) is no longer advisable, as only partial information about the shipments available for loading during the sort is available. The two insights that underpin block loading are (1) that the information regarding shipments available for loading during a sort depends on the order in which we process the \{terminal, sort\} pairs in \( PTS \), and (2) that by partitioning sorts into blocks (of trailers), and by processing these blocks in a specific order, we can ensure that every shipment available for loading during a block is known.

More specifically, we define a block as the set of consecutive outbound trailer departures between two consecutive inbound trailer arrivals within a sort (where the start time of the sort and the end time of the sort are also considered inbound trailer arrivals). In other words, a block is the largest possible set of departing trailers within a sort such that all shipments that can be loaded in these trailers are known at the time of the first trailer departure. Therefore, we can solve a single optimization problem to assign these shipments to the trailers in the block. In the following subsections, we describe in more detail how blocks are created and how shipments are assigned to trailers in a block.

3.3.1 Block generation

We start by observing that for a set of consecutive trailer departures at a terminal occurring between two consecutive trailer arrivals at that terminal, all the information about the
shipments that can be loaded onto these trailers is known at the time of the departure of the first trailer (in fact at the time of the arrival of the trailer that precedes it). We call such a set of trailers a block. Importantly, we can assign the shipments available at the start of the block to the trailers in the block simultaneously, rather than one by one, which may be beneficial. We consider the start and end of a sort as “trailer arrivals” to ensure that every block occurs within a sort. It is easy to generate these blocks in nondecreasing order of the departure time of the first trailer in the block. In Figure 4, we illustrate the concept of blocks on a small example (assuming a single sort per terminal).

![Figure 4: An example of blocks. The labels of the blocks represents the order in which the blocks have been created.](image)

Next, we observe that it may be possible to create larger blocks. This is the case when the shipments arriving in a trailer that would define the end of a block are already known by the time we start creating the block. This happens when the block that contains the (departure of such) trailer has already been processed. For example, in Figure 4, we see that the trailer that defines the end of Block 3 has already been processed during the creation of Block 1 and thus its shipments are known. Therefore, when creating blocks, we will use the most up-to-date information to determine the end of each block. Let \( \mathcal{L} = \{l_1, \ldots, l_m\} \) be the set of trailers dispatched during the planning period in nondecreasing order of departure times. After we initiate a new block and the terminal associated with the block is known, we continue to add trailers from \( \mathcal{L} \) that depart at that terminal to the block until we encounter a trailer that arrives at that terminal but has no shipments assigned to it yet. The first block \( B_1 \) is initiated with \( l_1 \). Note that this block generation scheme implicitly assumes that the assignment of shipments to the trailers in a block is performed block by block in the order that the blocks are generated. Figure 5 shows the blocks created using this scheme for the
example as previously.

Figure 5: An example of the adjusted generation of blocks. The numbers at the top left corners of the blocks represents the order in which the blocks are created.

### 3.3.2 Assigning shipments to trailers

Recall that the planned flow and the planned trailer movements are designed to meet the service guarantees of the shipments in the system on a day with average demand. Therefore, the premise of all our approaches is that assigning shipments to trailers on their planned flow paths is desirable.

Therefore, all block loading approaches start by loading trailers with only planned path shipments and only after that, if there is remaining capacity, load alternate path shipments. The loading of the trailers in a block with planned path shipments proceeds as follows. For a given block \( B \), let \( L_B \) be the set of trailers and let \( S_B \) be the set of shipments available for loading. Furthermore, let \( L_N \) be the set of trailers departing after the end of the block but before the end of the sort. We load the trailers in \( L_B \cup L_N \) in order of nondecreasing departure time with planned path shipments in \( S_B \) in order of nonincreasing urgency, and, in case of ties, in order of nonincreasing size (see Algorithm 3 for details of procedure BLOCK-PF). The loading decisions for trailers in \( L_B \) are final whereas the loading decisions for trailers in \( L_N \) are tentative (as new shipments may become available for loading in subsequent blocks, e.g., shipments with higher urgency). The reason for loading shipments in \( S_B \) in trailers in \( L_N \) is that we do not want to load too many shipments in \( S_B \) on alternate paths, especially when these shipments can be loaded in trailers along their planned flow path later in the sort (but not in the block). Once shipments in \( S_B \) have been loaded in trailers along their
Algorithm 3 Block-PF

1: for each trailer \( l \in \mathcal{L}_B \) do
2: \( S^P_l \leftarrow \) list of planned path shipments in \( S_B \) available for pickup, sorted by urgency then by quantity, in descending order
3: while there’s still capacity left in \( l \) or \( S^P_l = \emptyset \) do
4: load shipments from \( S^P_l \) in \( l \)
5: for each trailer \( l \in \mathcal{L}_N \) do
6: \( S^P_l \leftarrow \) list of planned path shipments in \( S_B \) available for pickup, sorted by urgency then by quantity, both in descending order
7: while there’s still capacity left in \( l \) or \( S^P_l = \emptyset \) do
8: load shipments from \( S^P_l \) in \( l \) without carrying the decisions over to subsequent blocks

planned path, we have to decide whether any remaining capacity in the trailers in the block should be used to load any remaining shipments in \( S_B \) in these trailers if they happen to be on their alternate path, or whether to postpone their loading to subsequent blocks. In the following, we will present different approaches for making these decisions.

A simple look ahead heuristic

Let \( S^n_B \subseteq S_B \) be the set of as-yet unloaded shipments in nonincreasing order of urgency, and, in case of ties, in order of nonincreasing size. For each shipment \( s \in S^n_B \), let \( L^A_s \subseteq \mathcal{L}_B \) be the set of trailers along the alternate path of \( s \) that have sufficient remaining capacity to load \( s \). Furthermore, let

\[
t^A_s = \min_{l \in L^A_s} \text{EstArrival}(s, dst_l, atm_l),
\]

i.e., \( t^A_s \) is the earliest estimated arrival time of \( s \) at its destination if it is loaded in one of trailers in \( L^A_s \), and let

\[
l^A_s = \arg \min_{l \in L^A_s} \text{EstArrival}(s, dst_l, atm_l).
\]

Finally, let \( t^P_s \) be the estimated arrival time of \( s \) at its final destination if, instead, it is loaded in the first trailer departing in the next sort that is on its planned flow path. If \( t^A_s < t^P_s \), we load \( s \) into \( l^A_s \), otherwise, we do not load \( s \) in this block. That is, we only load a shipment on a trailer on its alternate path if the arrival at its destination is expected to be earlier than when loading is postponed until the next sort. Algorithm 4 gives the pseudo code of Blk-LookAhead.

Blk-LookAhead can also be implemented efficiently. The estimated arrival times at the destination for shipments \( s \in S_B \) and trailers \( l \in L^A_s \) are pre-computed and stored in a look-up table using \( \text{EstArrival}(s, dst_l, atm_l) \). The trailer in \( L^A_s \) resulting in the earliest estimated arrival time, \( l^A_s \), can be determined at the same time at no extra cost.

Optimization: Basic formulation

16
Algorithm 4 Blk-LookAhead

1: $S^u_B \leftarrow$ list of shipments in $S_B$, sorted by urgency, then by quantity in descending order
2: for each shipment $s$ in $S^u_B$ do
3: $L^A_s \leftarrow$ list of alternative path trailers available within the block with enough capacity left to load $s$
4: $l^P_s \leftarrow$ the first planned path trailer departing after the end of the sort containing the block
5: $t^A_s \leftarrow \min_{l \in L^A_s} EstArrival(s, dst_l, atm_l)$
6: $t^P_s \leftarrow$ estimated arrival of $s$ at its final destination if loaded in $l^P_s$
7: if $t^A_s < t^P_s$ then
8: load $s$ in $l^A_s$
9: else
10: postpone the loading decision for $s$ to the subsequent block

Rather than deciding whether to assign a shipment to a trailer on its alternate path one shipment at a time, we next present an optimization model that decides whether to assign shipments to trailers on their alternate paths simultaneously. To allow postponing the loading of shipments to the next sort, which, at the same time, accommodates situations in which there is insufficient capacity to load all shipments on trailers on their alternate paths, we introduce a dummy trailer $l^*$ with infinite capacity. For a given shipment $s \in S^u_B$, the feasible assignments, other than to $l^*$, are to trailers departing after the time that $s$ becomes available and that are on the alternate path for $s$. We define the cost of assigning a shipment $s$ to a trailer $l \in L^A_s$ by the function $C(s, l)$ given by

$$C(s, l) = \max\{EstArrival(dst_s, dst_l, atm_l) - due_s, 0\}.$$  

The cost of assigning $s$ to the dummy trailer $l^*$ is set to $C(s, \bar{l})$, where $\bar{l}$ is the first trailer on the planned path of $s$ in the next sort or infinity if no such trailer exists. Recall that it is desirable to dispatch shipments in the sort in which they arrive, but that rolling over shipments is possible if it reduces the total lateness. We model the assignment of shipments to trailers within a block as the following integer program

$$\min \sum_{s \in S^u_B} \sum_{l \in L(s) \cup \{l^*\}} qty_s C(s, l)x_{sl}$$

s.t. $\sum_{l \in L^A_s \cup l^*} x_{sl} = 1$ \hspace{1cm} $\forall s \in S^u_B$ \hspace{1cm} (6)

$$\sum_{s \in S^u_B} \sum_{l \in L^A_s} qty_s x_{sl} \leq cap_l$$ \hspace{1cm} $\forall l \in L_B$ \hspace{1cm} (7)

$$x_{sl} \in \{0, 1\}$$ \hspace{1cm} $\forall s \in S^u_B \forall l \in L^A_s$
where $cap_l$ is the remaining capacity of trailer $l$ and $x_{sl}$ is the decision variable that models whether to assign $s$ to trailer $l$ ($x_{sl} = 1$) or not ($x_{sl} = 0$). The objective function seeks to minimize the lateness of the shipments in the block using estimates of a shipment’s arrival time at its destination when it is loaded on a particular trailer in the block. The lateness of a shipment is weighted by its size, thus a higher priority is given to large shipments. Constraints (6) ensure that every shipment in the block is assigned to a trailer (possibly the dummy trailer $l^*$) and Constraints (7) ensure that the capacity of each trailer departing within the block is not exceeded.

**Optimization: Extended formulation**

The basic formulation seeks to minimize the lateness of the shipments in a block at a terminal, but ignores the impact that the loading of these shipments may have on the terminals where they end up and where they may not have been expected and where there may be insufficient capacity to handle them. To address this limitation, we propose an extended formulation that also considers shipments and capacity at the destinations of the trailers departing in the block, and, thus, estimates the available capacity at these destinations to handle the shipments in the block. As a consequence, shipments are less likely to be send to destinations on their alternate paths if the available capacity at that destination is limited.

Let $L_i^+$ be the set of trailers departing from the destination of $l$ after its arrival there and before the end of the sort in which it arrives, and let $L_B^+ = \bigcup_{l \in L_B} L_i^+$. Let $L_s^+$ for $s \in S_B$ be the set of trailer pairs $\{l_1, l_2\}$ with $l_1 \in L_B$ a trailer on the alternate path for shipment $s$ and $l_2 \in L_i^+$ a trailer on the planned path for shipment $s$. Let $S_B^+$ be the set of shipments expected to be available at the destinations of trailers in $L_B$, which includes shipments that arrive there on their planned path and shipments that arrive there on one of their alternate paths from blocks that were processed previously. For each shipment $s \in S_B^+$, let $L_s^P$ be the set of trailers departing after the time $s$ becomes available, but before the end of the sort in which $s$ becomes available, and that are on the planned path of $s$. We model the assignment
problem of shipments to pairs of trailers as the following integer program:

\[
\begin{align*}
\min & \quad \sum_{s \in S_B^u} \sum_{\{l_1, l_2\} \in L^+_u \cup \{l^*, l^*\}} qty_s C_1(s, \{l_1, l_2\}) x_{s(l_1, l_2)} + \sum_{s \in S_B^u} \sum_{l \in L^+_u \cup l^*} qty_s C_2(s, l) y_{sl} \\
\text{s.t.} & \quad \sum_{\{l_1, l_2\} \in L^+_u \cup \{l^*, l^*\}} x_{s(l_1, l_2)} = 1 \quad \forall s \in S_B^u \tag{8} \\
& \quad \sum_{l \in L^+_u \cup l^*} y_{sl} = 1 \quad \forall s \in S_B^+ \tag{9} \\
& \quad \sum_{s \in S_B^u: \{l, l'\} \in L^+_u} qty_s x_{s(l, l')} \leq \text{cap}_l \quad \forall l \in L_B \tag{10} \\
& \quad \sum_{s \in S_B^u: \{l', l\} \in L^+_u} qty_s x_{s(l', l)} + \sum_{s \in S_B^u: l \in L^+_u} qty_s y_{sl} \leq \text{cap}_l \quad \forall l \in L_B^+ \tag{11} \\
& \quad x_{s(l_1, l_2)} \in \{0, 1\} \quad \forall s \in S_B^u \forall \{l_1, l_2\} \in L^+_u \\
& \quad y_{sl} \in \{0, 1\} \quad \forall s \in S_B^+ \forall l \in L_B^+ 
\end{align*}
\]

In this extended formulation, we not only consider the capacity of trailers in the block, i.e., Constraints (10), but also the capacity of the outbound trailers at the destinations of the trailers in the block, i.e., Constraints (11). This ensures that we load shipments in trailers on their alternate paths only if their is likely sufficient capacity at the destinations of these trailers. As in the basic formulation we allow postponing shipments through the introduction of dummy trailer pairs \(\{l^*, l^*\}\) and \(\{l, l^*\}\) for shipments in \(S_B\) and dummy trailer \(l^*\) for shipments in \(S_B^+\). The cost of assigning a shipment \(s\) in \(S_B\) to a trailer pair \(\{l_1, l_2\} \in L^+_u\) is given by:

\[
C_1(s, \{l_1, l_2\}) = \begin{cases} 
C(s, \bar{l}), & \text{if } \{l_1, l_2\} = \{l^*, l^*\} \\
C(s, \hat{l}), & \text{if } l_1 \neq l^* \text{ and } l_2 = l^* \\
C(s, l_2), & \text{o.w}
\end{cases}
\]

and the cost of assigning a shipment \(s\) in \(S_B^+\) to a trailer \(l_2 \in L_B^+\) is given by:

\[
C_2(s, l_2) = \begin{cases} 
C(s, \hat{l}), & \text{if } l_2 = l^* \\
C(s, l_2), & \text{o.w}
\end{cases}
\]

where \(\bar{l}\) is the first planned path trailer leaving after the end of the current sort containing the block and \(\hat{l}\) is the first planned path trailer leaving after the end of the sort that contains the trailer \(l_2\). The set of forecast shipments in \(S_B^+\) is obtained as follows: before creating and solving blocks, we (tentatively) load the trailers in the network with planned path shipments using URG-PULL-PF. The resulting loading decisions are only used for estimating
the remaining capacity of trailers. After solving a block using the extended formulation, the new location and arrival time for a shipment assigned to pair \( \{l_1, l_2\} \) are set to \( dst_{l_1} \) and \( atm_{l_1} \), respectively.

### 3.3.3 Block Loading

Once the blocks have been created, they are sorted in order of nonincreasing departure time of the first trailer in the block, and the blocks are processed one after the other. The construction of the blocks and the processing order guarantee that all shipments that can be loaded into the trailers of a block are known at the start time of the block. For blocks that contain a large number of shipments and trailers, the optimization models may become too large to be solved in an acceptable amount of time. In order to prevent spending too much time on a single block, we set a time limit for solving the integer program. If the optimal solution is not obtained within the time limit, we check if the gap between the upper and lower bound is less than a given threshold (discussed in Section 4). If so, then we accept the incumbent solution and move on to the next block. Otherwise, we solve the block using the look ahead loading strategy for blocks. Algorithm 5 shows the overall procedure for a block loading heuristic.

**Algorithm 5** Block Loading

1: \( Blocks \leftarrow \text{GenerateBlocks}() \);
2: for each block \( b \) in \( Blocks \), sorted by the dispatch time of the first trailer departing within the block do
3: \( \text{Block-PF}(b) \); \text{Solves the assignment problem for planned path trailers}
4: \( \text{SolveBlock}(b) \); \text{Solves the assignment problem for the remaining shipments in the block using either lookahead heuristic or the basic/extended formulation}
5: if optimization models are used, timeLimit is exceeded and optimality gap > \( p \) then
6: \( \text{Blk-LookAhead}(b) \); \text{Solves the assignment problem for the remaining shipments using the look ahead loading heuristic for blocks}

### 4 Computational Experiments

#### 4.1 Performance metrics

In order to assess the quality of the different loading strategies, we use the following set of performance metrics, where we only take into account shipments that enter the system in the first 24 hours of the planning period. These shipments matter most as they have been and still are in the system or are just entering the system and they are directly impacted by the decisions we make (at time zero). For completeness sake, we present results for all shipments, i.e., including forecast shipments in Table ?? in the appendix.

- **Total Lateness (TL)**. For a given shipment \( s \), we define the lateness \( lat_e \) as the difference between the arrival time of the shipment at its destination and its due time.
There are two cases to consider: (i) the shipment arrives at its destination during the planning period, and (ii) the shipment does not arrive at its destination during the planning period. In the second case, the arrival time of the shipment is estimated based on the use of the earliest departing trailers along the shipment’s planned path after the planning period. If a shipment arrives on time (before its due time), \( late_s \) is set to zero as we are only interested in the shipments that arrive late. The total lateness for a loading strategy is then defined as the sum of each shipment’s lateness weighted by its size:

\[
TL = \frac{1}{\sum_{s \in S_{24}} qty_s} \sum_{s \in S_{24}} qty_s late_s,
\]

where \( S_{24} \) is the set of shipments that enter the system in the first 24 hours of the planning period.

- \( %D \), \( %D_{OT} \), and \( %D_L \): the fraction of shipments delivered, the fraction of shipments delivered or expected to be delivered on time, and the fraction of shipments delivered late or expected to be delivered late.

- \( %PF \) and \( %AF \): the fraction of shipments that only used trailers on their planned path during the planning period and the fraction of shipments that used at least one trailer on their alternate path during the planning period.

- \( %S \): the fraction of shipments that have a due time within the planning period and that have “stalled”, i.e., have not been loaded into any trailer during the planning period.

- **RO-AVG**: the average number of sorts that shipments were rolled over during planning period weighted by size; let \( I^s \) be the set of terminals visited by shipment \( s \) during the planning period and let \( nst_i^s \) be the number of sorts used by shipment \( s \) at terminal \( i \in I^s \), then

\[
RO-AVG = \frac{1}{\sum_{s \in S_{24}} qty_s} \sum_{s \in S_{24}} qty_s \sum_{i \in I^s} nst_i^s / |I^s|,
\]

### 4.2 Instances

The set of ten instances used in the computational experiments are derived from snapshots of historical data from a major U.S. LTL carrier. Each snapshot corresponds to a set of consecutive days at different times of the year. The number of shipments varies, but the planned and alternate paths as well as the trailer movements are similar. Each instance contains shipment and trailer information from more than 72 continuous hours of operations. We set the start of the planning period to be 52 hours before the last departing trailer so as to ensure that we have information on all trailers departing during the 48 hour planning period. Given that each snapshot represents historical real-world data with a considerable number of imperfections, we have decided to make the following modifications:
• Shipments meeting at least one of the following criteria are removed from the instance: (a) shipments with the same origin and destination, (b) shipments with an incomplete planned path, (c) shipments with no trailer departing along their planned path at the terminal where they enter the network; (d) shipments with a size larger than the capacity of a trailer; (e) shipments that are already late at time zero; (f) shipments that have been in the system for more than 96 hours before the start of the planning period;

• The due time of a shipment \( s \) that will arrive late at its destination if loaded in the earliest possible departing trailers along the planned path is changed as follows. We estimate the arrival time \( atm_s \) of the shipment at its destination given its origin location and the time it entered the network by always considering the earliest departing trailers in the planned path and set the due time \( due_s \) as:

\[
due_s = atm_s + 4
\]

These changes ensure that each shipment can reach its destination before its due time if it would be the only shipment in the system. Table 1 shows the number of shipments \(|S|\) and the number of trailers \(|L|\) for each instance. The instances are based on a network with about 350 terminals. A terminal can operate up to four sorts (Day, Twilight, Night, and Sunrise). About 85% of the terminals, mostly End-of-Lines, operate two sorts, about 10%, operate three sorts, and about 5%, mostly Breakbulks, operate four sorts.

| Instance | \( |L| \) | \( |S| \) |
|----------|--------|--------|
| I1       | 12,134 | 92,329 |
| I2       | 12,072 | 73,608 |
| I3       | 12,105 | 77,290 |
| I4       | 10,953 | 71,592 |
| I5       | 11,966 | 91,747 |
| I6       | 11,677 | 92,001 |
| I7       | 11,723 | 89,617 |
| I8       | 11,911 | 91,391 |
| I9       | 11,806 | 88,937 |
| I10      | 11,806 | 89,619 |

Table 1: Information on the instances used in the computational experiments.

4.3 Analysis

We compare the performance of the following loading strategies: FIFO-Push, Urg-Pull, Urg-Pull-PF, i.e., the variant of Urg-Pull in which shipments are only loaded on trailers
along their planning path, Blk-LookAhead, block loading with look ahead, Blk-IP-Basic, block loading using the basic assignment formulation, and Blk-IP-Extended, block loading using the extended assignment formulation. When solving an integer program in one of the formulation-based block loading variants, a time limit of 300 seconds is imposed and solutions with an integrality gap of less than 10% are considered acceptable.

All loading strategies are coded in C# and integer programs are solved using IBM CPLEX Optimizer 12.6. All experiments were conducted in a single thread of a dedicated Intel Core i5-73000U 2.60GHz CPU with 16GB RAM running Microsoft Windows 10.

The results can be found in Table 2.

We see that using alternate paths too aggressively, as is done in Urg-Pull, results in poor performance. However, the results also show that using alternate paths too sparingly, as is done in Urg-Pull-PF, which does not use alternate paths at all, does not result in strong performance either (although clearly better). Furthermore, the results for Urg-Pull-PF show that not using alternate paths leads to a considerable increase in the number of stalled shipments, and, more general, a higher number of roll-overs.

All block loading strategies, which seek to balance the use of planned flow and alternate flow paths, perform noticeably better. For Blk-LookAhead, we see an average improvement of 48.41% in total lateness over the total lateness of FIFO-Push, 29.54% over the total lateness of Urg-Pull, and 13.79% over the total lateness of Urg-Pull-PF (with largest improvements of 57.36% and 43.09% for Instance 7, and 26.06% for Instance 4, respectively). We also see higher fractions of delivered shipments.

In all block loading strategies, the trailers are loaded first with shipments for which the trailers are on their planned flow path, and the same algorithm, Block-PF, is used for all block loading strategies. Therefore, the difference is a result of the choice of alternative paths. Block loading strategies are conservative in their use of alternative paths: they are only used when a shipment is expected to arrive earlier at its destination compared to waiting for the next planned path trailer. This accounts for the improvements not only in velocity and total lateness, but also in number of rolled over shipments.

Not surprisingly, using optimization models to refine the assignment of shipments to trailers improves performance. Using Blk-IP-Basic, we see an average improvement of 2.82% in total lateness over Blk-LookAhead (with largest improvement of 4.34% for Instance 5). Blk-IP-Extended performs even better, which shows the importance of evaluating the impact of sending shipments to an alternative destination. By taking into account the available capacity in the planned path trailers at an alternative destination, better loading decisions are made. More precisely, resorting to alternative paths is only allowed in case there is sufficient capacity in the planned path trailers at the destination of an alternative path trailer. As a result of this careful examination of alternative paths, more shipments are rolled over and pushed into their planned path trailers in subsequent sorts. This explains the improvement in planned flow percentage of Blk-IP-Extended over Blk-IP-Basic. However, note that the average number of sorts used per shipment in Blk-IP-Extended is still comparable to Blk-IP-Basic. This is explained by the fact that we are making loading decisions that give priority to rolling over shipments to subsequent
Table 2: Results for the set of instances used in the computational experiments considering different metrics. The best results for each instance in terms of $TL$, $\%D$, $\%D_{OT}$, and $RO-AVG$ are highlighted in bold. $TL$ is given in hours, and total runtime $TT$ is in seconds.
sorts where they can be potentially loaded in a planned path trailer, rather than sending them to an alternative destination with no immediate available capacity. When compared to Urg-Pull-PF, Blk-IP-Extended shows an average improvement of 17.10% in total lateness (with a largest improvement of 28.18% for Instance 4). It does better in most other metrics as well. However, the improvement in performance of Blk-IP-Extended comes at a price. The computing time increases by a factor of 10, on average, compared to Blk-IP-Basic.

Next, we investigate the formulation-based block loading strategies in some more detail. Table 3 presents statistics on the blocks generated for both the basic and the extended formulations. We report the number generated (#B), the average duration (DAvg), the average number of trailers (LAvg), the average number of pairs of trailers in the extended formulation (PAvg), the average number of shipments using the basic and the extended formulations (SBAvg and SEAvg, respectively), the average solution time for the basic and extended formulations (TBMax and TEBMax, respectively), and the maximum IP solve time for the basic and extended formulations (TBMax and TEBMax, respectively). We see that the average number of trailers per block is more than ten for all the instances. This explains, to some extent, why the block-based loading strategies, and, thus, the formulation-based loading strategies, perform better than trailer-based loading strategies. We also see that the solution time for the integer programs solved when using Blk-IP-Extended is, on average, about five times more than when using Blk-IP-Basic.

<table>
<thead>
<tr>
<th>Instance</th>
<th>#B</th>
<th>DAvg</th>
<th>LAvg</th>
<th>SBAvg</th>
<th>TBAvg</th>
<th>TBMax</th>
<th>PAvg</th>
<th>SEAvg</th>
<th>TEBAvg</th>
<th>TEBMax</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>949</td>
<td>2.06</td>
<td>13.79</td>
<td>123.47</td>
<td>3.08</td>
<td>74.22</td>
<td>211.11</td>
<td>123.39</td>
<td>14.48</td>
<td>176.61</td>
</tr>
<tr>
<td>I2</td>
<td>953</td>
<td>2.09</td>
<td>13.67</td>
<td>71.64</td>
<td>2.05</td>
<td>41.05</td>
<td>210.91</td>
<td>71.75</td>
<td>9.02</td>
<td>150.97</td>
</tr>
<tr>
<td>I3</td>
<td>953</td>
<td>2.07</td>
<td>13.70</td>
<td>77.10</td>
<td>2.09</td>
<td>39.88</td>
<td>209.29</td>
<td>77.18</td>
<td>9.77</td>
<td>119.39</td>
</tr>
<tr>
<td>I4</td>
<td>859</td>
<td>2.13</td>
<td>13.75</td>
<td>69.17</td>
<td>2.03</td>
<td>44.25</td>
<td>228.11</td>
<td>69.26</td>
<td>11.00</td>
<td>195.46</td>
</tr>
<tr>
<td>I5</td>
<td>926</td>
<td>2.11</td>
<td>13.92</td>
<td>77.58</td>
<td>2.35</td>
<td>49.71</td>
<td>224.19</td>
<td>77.91</td>
<td>12.08</td>
<td>157.79</td>
</tr>
<tr>
<td>I6</td>
<td>914</td>
<td>2.11</td>
<td>13.78</td>
<td>67.04</td>
<td>1.80</td>
<td>23.82</td>
<td>224.49</td>
<td>67.24</td>
<td>8.69</td>
<td>89.08</td>
</tr>
<tr>
<td>I7</td>
<td>921</td>
<td>2.07</td>
<td>13.73</td>
<td>77.39</td>
<td>2.40</td>
<td>47.37</td>
<td>218.85</td>
<td>77.98</td>
<td>11.29</td>
<td>134.95</td>
</tr>
<tr>
<td>I8</td>
<td>925</td>
<td>2.06</td>
<td>13.88</td>
<td>80.47</td>
<td>2.35</td>
<td>40.40</td>
<td>223.06</td>
<td>81.09</td>
<td>11.87</td>
<td>172.02</td>
</tr>
<tr>
<td>I9</td>
<td>923</td>
<td>2.05</td>
<td>13.79</td>
<td>79.72</td>
<td>2.42</td>
<td>38.14</td>
<td>220.93</td>
<td>80.10</td>
<td>11.70</td>
<td>146.47</td>
</tr>
<tr>
<td>I10</td>
<td>919</td>
<td>2.05</td>
<td>13.87</td>
<td>88.87</td>
<td>2.55</td>
<td>33.78</td>
<td>221.22</td>
<td>89.28</td>
<td>12.11</td>
<td>136.16</td>
</tr>
</tbody>
</table>

Table 3: Statistics on the dynamic generation of blocks for each instance.

average number of trailers per block is more than ten for all the instances. This explains, to some extent, why the block-based loading strategies, and, thus, the formulation-based loading strategies, perform better than trailer-based loading strategies. We also see that the solution time for the integer programs solved when using Blk-IP-Extended is, on average, about five times more than when using Blk-IP-Basic.

5 Final remarks

We have designed and implemented heuristics that can be used for near real-time loadplan adjustments. Consolidation carriers have long recognized the opportunity and value of near real-time loadplan adjustments, but have also acknowledged the challenges of doing so in
practice. These challenges relate to the data needs and the computational requirements. Detailed information on the system status, e.g., what pallets have already been loaded into a truck at a loading dock at a breakbulk terminal, is not always readily available, and to be able to react quickly to observed changes into anticipated freight volumes decision support tools have to be efficient, e.g., propose loadplan adjustments in 15 minutes or less. With the advances in data collection technology, the advances in computing power, and the advances in algorithms, we have reached the point where near real-time loadplan adjustments are possible. The heuristics described in this paper are in daily use at a large national US LTL carrier and are generated significant benefits.

The next phase of this research is to extend the technology to offer additional functionality: adding or canceling schedules. For example, under-utilized trailers and schedules can be detected and the heuristics can be used to evaluate whether the shipments in these trailers and can be rerouted on alternate paths, and, if so, the schedules can be canceled. Or, when too many shipments stall at a terminal during a sort, the heuristics can be used to evaluate whether adding a schedule (one or more trailers) results in significantly fewer shipments stalling. Such functionality would further enhance a carrier’s ability to better manage daily operational costs while maintaining the service guarantee promised to its customers.

Acknowledgment

We are grateful for the support of the Transportation Analytics and Operations Research team at UPS. We want to thank them for the many informative discussions and help with questions regarding near real-time loadplan adjustments in freight transportation networks.

References


Appendix 1

Tables 4 and 5 present statistics for all shipments, i.e., shipments known at time zero and shipments forecast to enter at time 24.

<table>
<thead>
<tr>
<th>Ins.</th>
<th>Algorithm</th>
<th>TL</th>
<th>%D</th>
<th>%DOT</th>
<th>%DL</th>
<th>%PF</th>
<th>%AF</th>
<th>%S</th>
<th>RO-AVG</th>
<th>TT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FIFO-Push</td>
<td>3.00</td>
<td>92.30</td>
<td>65.72</td>
<td>15.58</td>
<td>68.54</td>
<td>1.58</td>
<td>1.00</td>
<td>535.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Urg-Pull</td>
<td>2.06</td>
<td>94.87</td>
<td>61.96</td>
<td>12.92</td>
<td>71.87</td>
<td>20.45</td>
<td>3.31</td>
<td>370.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Urg-Pull-PF</td>
<td>1.81</td>
<td>95.25</td>
<td>63.08</td>
<td>12.17</td>
<td>82.85</td>
<td>11.99</td>
<td>3.33</td>
<td>290.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-LookAhead</td>
<td>1.39</td>
<td>96.09</td>
<td>55.40</td>
<td>10.70</td>
<td>80.39</td>
<td>11.99</td>
<td>3.33</td>
<td>366.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Basic</td>
<td>1.33</td>
<td>96.14</td>
<td>55.72</td>
<td>10.42</td>
<td>85.50</td>
<td>3.28</td>
<td>0.56</td>
<td>550.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Extended</td>
<td>3.05</td>
<td>93.48</td>
<td>75.49</td>
<td>17.99</td>
<td>91.85</td>
<td>6.98</td>
<td>0.53</td>
<td>3.75</td>
<td>5,468.78</td>
</tr>
<tr>
<td>11</td>
<td>FIFO-Push</td>
<td>3.02</td>
<td>92.30</td>
<td>65.72</td>
<td>15.58</td>
<td>68.54</td>
<td>1.58</td>
<td>1.00</td>
<td>535.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Urg-Pull</td>
<td>2.06</td>
<td>94.87</td>
<td>61.96</td>
<td>12.92</td>
<td>71.87</td>
<td>20.45</td>
<td>3.31</td>
<td>370.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Urg-Pull-PF</td>
<td>1.81</td>
<td>95.25</td>
<td>63.08</td>
<td>12.17</td>
<td>82.85</td>
<td>11.99</td>
<td>3.33</td>
<td>290.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-LookAhead</td>
<td>1.39</td>
<td>96.09</td>
<td>55.40</td>
<td>10.70</td>
<td>80.39</td>
<td>11.99</td>
<td>3.33</td>
<td>366.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Basic</td>
<td>1.33</td>
<td>96.14</td>
<td>55.72</td>
<td>10.42</td>
<td>85.50</td>
<td>3.28</td>
<td>0.56</td>
<td>550.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Extended</td>
<td>3.05</td>
<td>93.48</td>
<td>75.49</td>
<td>17.99</td>
<td>91.85</td>
<td>6.98</td>
<td>0.53</td>
<td>3.75</td>
<td>5,468.78</td>
</tr>
<tr>
<td>12</td>
<td>FIFO-Push</td>
<td>3.50</td>
<td>90.18</td>
<td>74.86</td>
<td>15.32</td>
<td>69.90</td>
<td>23.33</td>
<td>3.83</td>
<td>0.76</td>
<td>363.97</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull</td>
<td>2.31</td>
<td>93.77</td>
<td>80.89</td>
<td>12.88</td>
<td>74.39</td>
<td>19.26</td>
<td>2.74</td>
<td>0.75</td>
<td>323.12</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull-PF</td>
<td>2.04</td>
<td>94.16</td>
<td>81.94</td>
<td>12.22</td>
<td>90.67</td>
<td>0.00</td>
<td>3.51</td>
<td>0.89</td>
<td>140.40</td>
</tr>
<tr>
<td></td>
<td>Blk-LookAhead</td>
<td>1.65</td>
<td>94.90</td>
<td>84.06</td>
<td>10.84</td>
<td>82.22</td>
<td>11.57</td>
<td>2.63</td>
<td>0.45</td>
<td>369.80</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Basic</td>
<td>1.61</td>
<td>95.10</td>
<td>84.37</td>
<td>10.73</td>
<td>87.40</td>
<td>5.32</td>
<td>2.51</td>
<td>0.47</td>
<td>570.22</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Extended</td>
<td>3.16</td>
<td>95.13</td>
<td>84.49</td>
<td>10.64</td>
<td>87.51</td>
<td>5.10</td>
<td>2.61</td>
<td>0.4</td>
<td>5,842.59</td>
</tr>
<tr>
<td>13</td>
<td>FIFO-Push</td>
<td>5.06</td>
<td>88.91</td>
<td>70.52</td>
<td>18.38</td>
<td>74.45</td>
<td>21.53</td>
<td>3.25</td>
<td>0.70</td>
<td>324.61</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull</td>
<td>3.51</td>
<td>92.56</td>
<td>76.44</td>
<td>16.12</td>
<td>75.82</td>
<td>20.08</td>
<td>3.02</td>
<td>0.73</td>
<td>256.43</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull-PF</td>
<td>3.20</td>
<td>92.84</td>
<td>77.67</td>
<td>15.17</td>
<td>93.32</td>
<td>0.00</td>
<td>4.19</td>
<td>0.88</td>
<td>117.97</td>
</tr>
<tr>
<td></td>
<td>Blk-LookAhead</td>
<td>2.39</td>
<td>94.31</td>
<td>80.28</td>
<td>14.04</td>
<td>86.11</td>
<td>11.39</td>
<td>2.84</td>
<td>0.59</td>
<td>319.58</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Basic</td>
<td>2.33</td>
<td>94.47</td>
<td>80.68</td>
<td>13.79</td>
<td>89.30</td>
<td>6.38</td>
<td>2.83</td>
<td>0.59</td>
<td>467.32</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Extended</td>
<td>2.31</td>
<td>94.49</td>
<td>80.70</td>
<td>13.79</td>
<td>89.62</td>
<td>6.05</td>
<td>2.93</td>
<td>0.54</td>
<td>5,531.15</td>
</tr>
<tr>
<td>14</td>
<td>FIFO-Push</td>
<td>2.97</td>
<td>95.43</td>
<td>80.20</td>
<td>15.23</td>
<td>75.98</td>
<td>20.04</td>
<td>1.72</td>
<td>0.91</td>
<td>404.95</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull</td>
<td>2.30</td>
<td>96.41</td>
<td>84.62</td>
<td>11.79</td>
<td>73.69</td>
<td>22.00</td>
<td>1.53</td>
<td>0.91</td>
<td>284.41</td>
</tr>
<tr>
<td></td>
<td>Urg-Pull-PF</td>
<td>1.60</td>
<td>97.53</td>
<td>87.75</td>
<td>9.78</td>
<td>94.04</td>
<td>0.00</td>
<td>1.63</td>
<td>1.06</td>
<td>127.23</td>
</tr>
<tr>
<td></td>
<td>Blk-LookAhead</td>
<td>1.39</td>
<td>97.81</td>
<td>88.87</td>
<td>8.95</td>
<td>84.91</td>
<td>10.91</td>
<td>1.55</td>
<td>0.75</td>
<td>368.22</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Basic</td>
<td>1.33</td>
<td>98.06</td>
<td>89.28</td>
<td>8.78</td>
<td>91.22</td>
<td>3.94</td>
<td>1.35</td>
<td>0.78</td>
<td>519.31</td>
</tr>
<tr>
<td></td>
<td>Blk-IP-Extended</td>
<td>1.31</td>
<td>98.09</td>
<td>89.25</td>
<td>8.84</td>
<td>91.50</td>
<td>3.58</td>
<td>1.42</td>
<td>0.75</td>
<td>5,372.42</td>
</tr>
</tbody>
</table>

Table 4: Results for the set of instances I1-I5 used in the computational experiments considering different metrics for all shipments in the system. The best results for each instance in terms of $TL$, $%D$, $%DOT$, and RO-AVG are highlighted in bold. $TL$ is in hours, and total runtime TT is in seconds.
Table 5: Results for the set of instances I6-I10 used in the computational experiments considering different metrics for all shipments in the system. The best results for each instance in terms of $TL$, $D$, $D_{OT}$, and $RO-AVG$ are highlighted in bold. $TL$ is in hours, and total runtime $TT$ is in seconds.