A Dynamic Driver Management Scheme for

Less-than-Truckload Carriers

Alan Erera; Burak Karacik, Martin Savelsbergh
The Logistics Institute
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

*corresponding author; email: alerera@isye.gatech.edu, fax: +1 404 894 2301
Abstract

This paper describes a scheme for the dynamic management of linehaul drivers developed for a large U.S. LTL carrier. Virtually all scheduling problems faced by transportation service providers are complicated by time-constrained vehicle operators that can be renewed only after resting. LTL driver scheduling is further complicated by the fact that trucking moves, unlike passenger airline flights or train dispatches, are not pre-scheduled. The technology developed in this paper combines greedy search with enumeration of time-feasible driver duties, and is capable of generating in a matter of minutes cost-effective driver schedules covering 15,000–20,000 loads satisfying a variety of real-life driver constraints. Computational results justify the algorithmic design choices made in the development of the scheme, and a comparison with real-world dispatch data indicates that the technology produces driver schedules of very high quality.
1 Introduction

The trucking industry is vitally important to the economy of the United States, and provides an essential service by transporting goods from business to business and from business to consumer. The less-than-truckload (LTL) industry is an important segment, serving businesses that ship quantities ranging from 150 lbs to 10,000 lbs, i.e., less-than-truckload quantities. Large LTL carriers use thousands of drivers daily to move loads between terminals in the linehaul network, and each driver may be used for multiple consecutive load dispatches between rest periods.

In this paper, we describe a dynamic scheme for the management of linehaul drivers that we have developed for a large U.S. LTL carrier. The technology is capable of generating cost effective driver schedules satisfying a variety of real-life constraints in a matter of minutes. Large-scale dynamic scheduling problems with renewable resources like drivers are among the most challenging scheduling problems. A renewable resource is one that is available for use for a limited duration after some start time, but can be renewed by recharging (in this case, resting) for a minimum recharge duration. Virtually all operator scheduling problems faced by transportation service providers are complicated by renewable resources. However, unlike passenger airlines and train services, LTL driver scheduling is further complicated by the fact that LTL trucking moves are not pre-scheduled. The driving tasks to be assigned to drivers have flexible dispatch times, but some tasks are linked by precedence relationships. In addition, LTL trucking service is usually “demand-responsive”; if enough freight has accumulated at a terminal bound for a specific next terminal, a load is ready to be dispatched as soon as an appropriate driver can be assigned.

The goal of the dynamic driver management scheme developed herein is to determine the best load dispatch times and driver-to-load assignments for some fixed-duration planning period (usually 48–72 hours, a time frame that we will justify later in the paper). The methodology is to be re-run periodically to take advantage of the availability of new data to improve existing plans. Importantly, the notion of the best solution is tricky to define in this context. The goal of the firm is to minimize the driver costs required to meet customer service targets, but the true costs of a driver schedule are often difficult to quantify.

The scheme we develop combines greedy search ideas with partial enumeration to determine cost-effective solutions in minutes. The scale of the problems typically encountered for
large trucking firms—a few thousand driver resources and between ten and twenty thousand dispatch tasks—renders a mathematical programming approach impractical; a column generation methodology, even one that leads only to a heuristic solution, would likely require too much computation time. Further, a non-greedy local improvement strategy such as tabu search is also difficult to implement successfully in this context, since small changes to a small set of driver schedules can propagate throughout the solution. Since we rely on greedy search, the primary research question we would like to answer is: can a creative application of greedy search be used to determine high-quality solutions for dynamic LTL driver management?

The primary contributions of this paper can be summarized as follows:

- This paper introduces the *driver scheduling and load dispatching problem* (DSLDP), a generalization of the operator scheduling problems with fixed task dispatch times typically considered in transportation research and an important dynamic resource management problem. The DSLDP is applicable not only to LTL carriers, but also to small package express carriers.

- This paper describes a fast, effective heuristic combining greedy search with enumeration to determine cost-effective solutions to the DSLDP in minutes.

- This paper provides a computational study that shows that (1) combining enumeration with greedy search can lead to significant improvements in solution quality over simpler greedy procedures for large-scale resource management problems, and that (2) greedy approaches can be used in a dynamic framework to develop good solutions to the DSLDP.

The remainder of this paper is organized as follows. In Section 2, we begin by describing LTL trucking operations to motivate the problem that we study. In Section 3, we review related literature. Section 4 provides a high-level description of the driver management challenges faced by LTL carriers, and Section 5 provides a formal problem definition for the DSLDP. Section 6 describes the developed solution approach, and Section 7 discusses issues in implementing the approach for dynamic decision-making. Finally, Section 8 presents the results of a computational study using the approach with historical data from a large U.S. LTL carrier.
2 LTL System Description

Like small package and post carriers, LTL carriers are freight *consolidators*, combining and recombining shipments from many customers into truckloads of freight at various terminals. In this way, LTL carriers spread fixed transportation costs (*e.g.*, vehicle and operator costs) over many customers.

To provide service, LTL trucking firms operate fixed terminal networks. Carriers pick up shipments from various shippers in a relatively small geographical area, say a city, and transport them to a terminal serving the area, referred to as a satellite or *end-of-line* terminal. These satellite terminals serve as sorting centers and loading facilities for outbound freight. Since there usually is not enough freight collected at a satellite terminal to build full truckloads to satellite terminals serving other areas, another layer of consolidation is introduced in the system. Outbound freight from a satellite is sent to a terminal that consolidates freight from different satellite terminals, referred to as a *breakbulk terminal*. Breakbulk terminals handle enough freight to build and dispatch cost-efficient loads, *i.e.*, loads that completely or almost completely fill up two trailers. A typical shipment travels from an origin satellite to an origin breakbulk, then to a destination breakbulk and finally to a destination satellite. This hub–and–spoke network is referred to as the (main) *linehaul network*. When the distance between the origin and destination breakbulk of a load is too long for a single driver to cover in the driving hours allowed by regulation, a sequence of intermediate stops is introduced at so-called *relay terminals*. Usually, relay terminals are breakbulk terminals on the path from one breakbulk to another, although they may be special facilities. To ensure high service levels, the load is transferred at a relay to another driver and continues with minimal delay.

The path of a shipment through the terminal network is identified by the sequence of terminals where the shipment is handled, where handling refers to unloading and loading of the shipment into some trailer. For freight moving between each origin satellite–destination satellite pair, the *load plan* specifies the shipment path. If enough freight is available, it may be possible at an origin breakbulk to build a load for a satellite terminal and bypass the destination breakbulk specified by the load plan. Loads dispatched from an origin breakbulk to a satellite terminal are called *direct loads*, because they do not need to be handled at an destination breakbulk. LTL carriers prefer that loads from breakbulks are
dispatched directly to destination satellites, since it reduces handling costs and time (which
in turn increases the chance of delivering before the due date). Usually, a direct load follows
the same path as a regular load in order to allow driver changes at relay points, but it is
not handled until it reaches the destination satellite. Hence, direct loads may not decrease
transportation costs. An important advantage of a direct load is that it typically reduces
the time freight spends in the linehaul system (because it is handled less), which improves
service. Since the freight patterns observed by LTL carriers are fairly regular, LTL carriers
usually identify and operate direct loads on certain origin–destination pairs. Direct loads
from an origin satellite to a destination breakbulk are possible, but are used less frequently.
Furthermore, if trailers are used for pickup and delivery of freight, then it may be possible
to build loads for the origin breakbulk without unloading and loading at satellite terminals.
This practice, which reduces handling cost and may improve service, is another reason
why breakbulk–satellite direct loads are more likely to occur than satellite–breakbulk direct
loads.

LTL carriers pack freight into 28-foot trailers (or pups) or 48-foot vans. Typically, a
single driver will pull a single 48-foot van or two 28-foot trailers. One advantage of the use
of 28-foot trailers is that they allow a single driver to pull 56 feet of trailers. In addition,
the use of 28-foot trailers also improves service and reduces handling costs. It is easier
to fill a 28-foot trailer than a 48-foot trailer. By combining trailers with different final
destinations, it also is possible to build loads that can be dispatched earlier. Furthermore,
filling an entire 28-foot trailer with shipments with a common final destination (which is
more likely to occur with a 28-foot trailer than a 48-foot van) allows simple drop-and-hook
operations at the next terminal as opposed to loading and unloading. Effectively exploiting
the advantages of the use of 28-foot trailers requires proper "pup matching" at breakbulks
and relays, i.e., deciding which 28-foot trailers to combine into loads.

Due to imbalances in freight flows, it is impossible to avoid empty trailer repositioning
moves in an LTL operation. To a certain degree, long-run empty trailer balance occurs
naturally in LTL systems if drivers usually pull a van or two trailers, loaded or empty,
whenever they are dispatched between terminals. In this case, since the driver assigned to
any outbound load likely arrived inbound with trailers, trailer balance is generally preserved
simply by ensuring that drivers are positioned correctly to move loads. Sometimes it is
necessary for drivers to move single pups at a time or to "bobtail," i.e., move a tractor without

6
any trailers, and specific actions may then be required to restore trailer balance. Bobtailing is often used, for example, between a terminal and a rail head to accommodate freight movements by rail. Most national LTL carriers rely on rail to move freight across long distances.

As mentioned earlier, a load plan specifies how a carrier plans to move freight through its terminal network. However, such a plan is not enough to guarantee an effective LTL operation. In fact, a crucial component, one that has a significant impact on overall performance, is still missing: scheduling drivers to actually move the loads and empties through the linehaul network. Effective driver management is one of the most challenging aspects of an LTL operation since drivers must abide by a large number of rules that limit their use. The first source of such rules is the U.S. Department of Transportation (DOT), which limits driving hours and duty hours to ensure highway safety. The second source of such rules are labor unions. Most carriers are unionized and the labor unions have negotiated rules that restrict, sometimes severely, the use of drivers at terminals. Today, most LTL driver management is performed locally by terminal managers who dynamically determine assignments of drivers to loads and dispatch times.

Although driver management is of primary importance for efficient and effective LTL operations, it has received relatively little attention by the research community. This is partly due to a historical tendency to apply operations research methodology to planning problems rather than operational problems, and partly due to the challenges of driver management at large LTL carriers, which at first glance seem to be almost insurmountable. Given the current low cost of computing power and the growing trend in the academic community to focus on dynamic operational problems, the time seems right to see what can be done to assist national LTL carriers with driver management.

3 Related Literature

Most of the operations research literature focused on the LTL trucking industry and other consolidation transportation services has addressed tactical freight flow planning, or what is often called service network design. Crainic ([1] and [2]) provides excellent overviews of the large body of research in this area. Recent research considers time-critical transportation services such as those provided by the small package express industry (see e.g., Barnhart
and Schneur [3] and Armacost et al. [4]). Dynamic service network design has received very little research attention, but has recently begun to attract attention again (see e.g., Papadaki and Powell [5] and Crainic et al. [6]).

In the area of transportation operator scheduling, there is a significant body of research literature focusing on crew scheduling and rostering problems for transportation systems operating fixed schedules, such as passenger airlines, transit systems, and passenger rail services. Again, most research addresses tactical planning problems and applies a set covering or partitioning integer programming model to choose a subset of partial schedules, solved exactly or heuristically by enumeration or column generation. Barnhart et al. [7] provides a thorough overview of airline crew scheduling; recent advances in this field have addressed tactical crew planning under uncertainty (see e.g., Schaefer et al. [8]). Importantly, the solution times required by this general approach makes it difficult to apply in an operational setting with dynamically-changing data. Advances in solution speed using specialized solution techniques still lead to long computation times. Elhallaoui et al. [9] reports computation times greater than an hour for scheduling problems with more than 1,500 tasks.

Perhaps the most closely related research to this paper is that which focuses on dynamic resource/asset allocation problems, typically in the context of managing drivers for truckload transportation firms. The body of research by Powell and co-authors focuses primarily on methods for handling data stochasticity; see [10] and [11] for summaries of the most recent research. The adaptive dynamic programming approach applied to problems with a discretized time dimension outlined in papers by Godfrey and Powell ([12] and [13]) appears to be the most promising approach to these problems. Other recent research in Yang et al. [14] considers a dynamic truckload assignment problem, and shows using a simulation that myopic rolling horizon reoptimization policies can perform quite well when compared to an a posteriori optimization. This work, however, ignores the resource time constraints crucial to driver management. It should also be noted that truckload driver management problems are quite different from the LTL problem, in that load requests do not require movement between distinct terminals and tend to arrive with somewhat less predictability than LTL loads. One paper, Powell et al. [15], considers a deterministic LTL driver management problem quite similar to the problem that we study. After concluding that integer-programming techniques are impossible to apply due to problem scale,
the authors instead apply an approximate dynamic programming methodology similar to those developed for stochastic problems. While the approach yields promising results with computation times in the range of an hour when all constraints are included, it relies on a discrete representation of time (with computation times that depend on the fineness of the discretization) and a discrete representation of the state of a driver, including the remaining drive hours. In this work, we will not rely on such a discretization.

4 Driver Management

The discussion of LTL operations in Section 2 shows that the primary transportation tasks faced by carriers is the movement of combinations of one or more pups and vans, containing shipments with a variety of origins and destinations, between terminals that are connected in the linehaul network. Two terminals are said to be connected if the travel time between them is no greater than the maximum drive time allowed by the DOT. A movement of freight between two connected terminals requiring a single driver will be referred to as a load; as described in Section 2, a load will usually consist of a single 48-foot van or two 28-foot trailers. Each load has a starting terminal (origin), an ending terminal (destination), a ready time (the earliest time the load can be dispatched), and a cut time (the latest time the load can be dispatched). The ready and cut time of a load are determined by the shipments contained in the load. Before a load can be dispatched, all contained shipments must have arrived at the origin terminal and undergone any necessary processing such as unloading, sorting, and reloading. The latest time that a load can be dispatched depends on the shipment contained in the load with the most constraining service requirement, where a service requirement typically specifies the due date of the shipment at its final destination.

Loads are said to be built at the origin terminal. Building a load may involve physically loading shipments into a van or into one or more trailers, or may involve matching of pups, or may simply involve relaying a van or a combination of trailers. Since shipments are usually contained in multiple loads along the path from their origin satellites to their destination satellites, precedence relations exist between loads. A precedence relation exists between every pair of loads that contain a common shipment. Fortunately, due to transitivity of precedence relations, it suffices to track only precedence relations between “consecutive” loads.
Each load requires a driver to move it through the linehaul network. Each driver has a **domicile** where he is based, and to where he must return with some regularity. If we consider a single epoch in time, each linehaul driver has a current **state** including a “next” location where he will become available for assignment, a ready-to-assign time at the next location, and remaining available drive and duty hours at the next location. Drivers can only be used for a limited duration before they need to be renewed by taking a **long rest**, during which time the driver sleeps. The minimum long rest time is specified by DOT regulations. If a driver is currently resting, then the next location is his current location, his ready time is his **off-rest time**, and his remaining drive and duty hours are given by the maximum DOT limits allowed between long rests. If a driver is **en-route**, then the next location is the terminal he reaches when completing his current move, his ready time is the time at which he completes his current move, and the remaining drive and duty time are set appropriately.

Driver management technology is designed to take load and driver information for some time horizon beginning at some time epoch and attempt to make the following related decisions:

- Which driver to assign to each load?
- When to dispatch each load?

Any driver management system must consider a mixture of goals when assigning drivers to loads and deciding dispatch times, *e.g.*, 

- Maximize the number of loads that are dispatched on time
- Minimize the number of driver long rests at non-domicile terminals (*foreign beds*)
- Construct driver **duties** (sequences of dispatches between long rests) with drive times close to the allowed maximum
- Construct driver duties with little or no waiting time between consecutive load dispatches
- Minimize the use of empty repositioning
Each of these goals relates to the fundamental trade-off faced by LTL carriers: service versus cost. Load dispatch windows ensure a desired level of service for each shipment. On the other hand, all driver-related goals tie back to costs, sometimes indirectly through “quality-of-job” considerations that reduce costs related to driver turnover. For example, sometimes the frequency with which a driver must return to his domicile for rest is specified by contract. However, even when it is not, it is desirable for the carrier to return a driver to his domicile frequently to improve quality-of-life. Note also that minimizing empty repositioning is an explicit goal. Some repositioning is necessary to allow each load to meet its desired service level, but carriers know that it is extremely difficult, if not impossible, to determine the appropriate amount of empty travel. Therefore, they aim to at least control the empty repositioning introduced into the system.

In the next section, we present a more formal definition of a driver management problem, which forms the basis for the algorithmic approach we have developed.

5 Problem Description

The *driver scheduling and load dispatching problem* (DSLDP) is to determine a set of driver movement schedules for a set of loads during a given planning period. Each load is characterized by an origin terminal, a destination terminal, a ready time, a cut time, a set of preceding loads which must arrive at the origin terminal before a dispatch can take place, and a set of succeeding loads which can depart from the destination terminal only after the load has arrived there. A set of drivers is available to move the loads. Each driver has a domicile, a current location at the beginning of the planning period, and a ready time at the current location. A driver schedule consists of a sequence of moves (loaded or empty) with associated dispatch times and long rest periods with start times. Sequences of moves between long rests are known as *duties*; when a driver has been assigned to a duty, we refer to the duty as a *trip*. A feasible driver schedule must satisfy regulations governing maximum drive hours and duty hours between long rest periods. The objective of the DSLDP is to find a set of driver schedules with minimum total operating costs, such that all loads are dispatched between their ready and cut times.

Formally, we use the following notation to model the driver scheduling problem:
\( \mathcal{T} \) The set of all terminals.
\( \mathcal{C} \) The set of all connections.
\( \mathcal{L} \) The set of all loads.
\( \mathcal{D} \) The set of all drivers.

The following parameters are used to model DOT driver regulations:

- \( \Delta_{\text{DR}} \) The maximum drive hours between long rests.
- \( \Delta_{\text{DU}} \) The maximum duty hours (including waiting) between long rests.
- \( \Delta_{R} \) The minimum duration (hours) of a long rest.

In the remainder of this paper, we use current U.S. DOT driver regulations: \( \Delta_{\text{DR}} = 11 \), \( \Delta_{\text{DU}} = 14 \), and \( \Delta_{R} = 10 \).

Each load \( l \in \mathcal{L} \) has the following attributes:

- \( \text{ORIG}(l) \) The origin terminal.
- \( \text{DEST}(l) \) The destination terminal.
- \( \text{READY}(l) \) Earliest dispatch time.
- \( \text{CUT}(l) \) Latest dispatch time.
- \( \text{PREC}(l) \) The set of immediate predecessor loads.
- \( \text{SUC}(l) \) The set of immediate successor loads.

Note that an immediate predecessor load \( p \in \text{PREC}(l) \) must have \( \text{DEST}(p) = \text{ORIG}(l) \), and an immediate successor load \( s \in \text{SUC}(l) \) must have \( \text{ORIG}(s) = \text{DEST}(l) \).

Each driver \( d \in \mathcal{D} \) has the following attributes, representing his state at the beginning of the planning period:

- \( \text{DOMICILE}(d) \) The domicile.
- \( \text{LOCATION}(d) \) The current location.
- \( \text{READY}(d) \) Earliest dispatch time at the current location.
- \( \text{MAXDRIVE}(d) \) Remaining drive time.
- \( \text{MAXDUTY}(d) \) Remaining duty time.

Note that driver \( d \) may be enroute to the current location \( \text{LOCATION}(d) \) at the beginning of the planning period. In this case, and in the case where a driver is waiting (not resting) at the current location, \( \text{MAXDRIVE}(d) \) and \( \text{MAXDUTY}(d) \) may be less than the DOT maxi-
mums. In the case in which the driver is resting at the beginning of the planning period, \( \text{MAXDRIVE}(d) = \Delta_{DR} \) and \( \text{MAXDUTY}(d) = \Delta_{DU} \).

Each connection in \( C \) links two terminals \( a, b \in T \) and has the following attributes:

- \( \text{COST}(a, b) \) The cost (in dollars) of driving (loaded or empty) between terminals \( a \) and \( b \).
- \( \text{TIME}(a, b) \) The drive time (in minutes) between terminals \( a \) and \( b \).
- \( \text{RTIME}(a, b) \) The run time (in minutes) between terminals \( a \) and \( b \).

The set of connections defines the linehaul network. As mentioned earlier, if two terminals are connected, the drive time between them is no greater than \( \Delta_{DR} \), and there is frequent transportation between the two terminals. Depending on the drive time between two terminals, a driver may need short rests along the way. These short breaks are incorporated in the run time and count towards duty time. Note that \( \text{RTIME}(a, b) \leq \Delta_{DU} \) for all connected terminals.

In addition to the cost incurred while driving between two terminals, a fixed cost is incurred for a driver resting at a location other than his domicile (foreign bed cost).

6 Solution Approach

Real-world instances of the DSLDP for LTL carriers are likely to be quite large, since the problem is to create driver schedules for the entire linehaul network based on the latest available information (status of all drivers, and all actual and forecasted loads). To ensure that driver schedules are not overly influenced by end-of-horizon effects, real-world instances should also plan for multiple days. Experience with historical dispatch data from a large U.S. LTL carrier indicates that typical instances may include 15,000–20,000 loads for dispatch, and thousands of drivers. Therefore, to be able to construct driver schedules in an acceptable amount of time we have chosen to develop a solution approach that combines greedy search with enumeration.

An important feature of the solution approach is that we assign a complete duty to a driver each time we make an assign-and-dispatch decision; a simpler alternative would be to simply assign a single loaded move to a driver at a time. Assigning complete duties facilitates the evaluation of decisions since many cost-minimizing “goals” relate to complete duties. For example, a very “good” duty has at least 9 hours of driving, less than than
2 hours of waiting time between moves, and returns the driver to his domicile. A second important feature of the approach is that we attempt to enumerate potentially large sets of prospective duties each iteration. However, unlike exact approaches for transportation scheduling problems that enumerate all partial solutions, we restrict the enumeration at each iteration to only those feasible duties that include a specific selected load as the initial loaded move. Further, instead of combining a subset of the partial solutions into a complete solution using a set covering or partitioning integer program, we simply make a greedy choice each iteration using a hierarchical scoring mechanism.

A high level description of our heuristic algorithm for the driver scheduling and load dispatch problem can be found in Algorithm 1.

Algorithm 1 High level overview of the heuristic algorithm.

while There exist undispatched loads do
    Select a load to be dispatched next.
    Generate a set of duties that include the selected load as the first loaded move of the trip
    Generate a set of trips by matching drivers with duties
    Evaluate the quality of each trip with respect to objectives
    Select and dispatch a high-quality trip
    Update system information
end while

For efficiency reasons, the generation of driver trips is split into two phases. We observed that generating a set of trips separately for each driver that includes the selected load as the first loaded move sometimes resulted in tremendous duplication of effort. In many situations, a number of drivers with identical characteristics (e.g., ready time, remaining driving hours, and remaining duty hours) is available at the origin terminal of the selected load. For each of these drivers, the same set of duties was generated through enumeration. By first generating duty sequences of loads and empty moves, and then matching drivers with duties, i.e., checking whether the driver ready time, remaining driving hours, and remaining duty hours are compatible with the duty, a significant speedup was obtained.

Next, we discuss the key steps in the algorithm in more detail.

Load selection
We iterate over the loads in order of increasing cut times. That is, we always select the load with the earliest cut time (breaking ties arbitrarily) and attempt to dispatch it. This choice is motivated by the importance of meeting service. If a load is not dispatched before its cut time, at least one shipment in the load will not be delivered on time.

**Duty generation**

Once a load has been selected, we attempt to generate all time-feasible duties that include the selected load as the first loaded move. A time-feasible duty consists of one or more loads, each dispatched between its ready and cut time, such that maximum limits on drive time and duty time are not violated.

We generate time-feasible duties using a depth-first recursive procedure. For a load $\ell \in L$, we construct all duties starting with $\ell$. To control the number of trips generated, we restrict the waiting time between consecutive loaded movements to be smaller than a predefined value $W$. Sometimes, we will also truncate the recursion to limit the number of duties generated. Each recursive step starts with a base duty, initially just the load $\ell$. Then, to enforce the limit on the waiting time, we find the loads originating at the last terminal in the base duty with a ready time that is less than the arrival time at this terminal plus $W$. For all such loads $\hat{\ell}$, we check if the duty consisting of the base duty and the load $\hat{\ell}$ is time-feasible. If so, we add the duty to the list of generated time-feasible duties and invoke a new recursion where the base duty is the newly generated duty, i.e., load $\hat{\ell}$ is the last load in the extended base duty.

For a duty to be time-feasible, it must not violate the load dispatch windows, and the drive hours and duty period length must be no greater than the DOT maximums $\Delta_{DR}$ and $\Delta_{DU}$. Because of the dispatch windows, the duty may include waiting time. The waiting time of a duty depends on the dispatch times of each of the loads in the duty. Therefore, these dispatch times should be chosen in such a way that the total waiting time in the duty is minimum. Note that the set of dispatch times resulting in the minimum total waiting in a duty need not be unique. We set the dispatch time of the first load in the duty to be the earliest time which results in the minimum total waiting time of the duty, and the dispatch times of the remaining loads in the duty to the earliest feasible dispatch time. During the construction of time-feasible duties we monitor and update, if necessary, the dispatch times on the partial duty to maintain this (invariant) property.
Given a duty with minimum wait time, there still may be flexibility in the dispatch time of the first load in the duty, in the sense that departing later may not affect the total waiting time of the duty. We define the difference between the latest and the earliest dispatch times of the first load in the duty $D$ that results in a duty with minimum total wait as the flexibility of the duty, denoted by $\phi_D$. Figure 1 illustrates the concept of flexibility for a duty with three loads, A, B, and C. The bracketed ranges represent the feasible dispatch time windows between $\text{READY}(l)$ and $\text{CUT}(l)$ for each load $l$.

![Diagram of flexibility for a duty with three loads](image)

**Figure 1:** Flexibility $\phi_D$ for a duty with three loads; delaying the dispatch of load A no more than $\phi_D$ results in a minimum wait duty.

While it is trivial to determine whether a base duty $D$ can be feasibly extended with load $\hat{l}$ considering only the drive hour maximum $\Delta_{DR}$ and the latest dispatch time $\text{CUT}(\hat{l})$, it is a bit more difficult to determine whether the duty hour maximum $\Delta_{DU}$ is not violated. Let $\sigma_D$ denote the dispatch time of the first load of some base duty $D$, and let $\tau_D$ denote the duration of $D$. Further, suppose $\hat{l}$ is a candidate load for extending $D$ which is feasible with respect to maximum drive hours and latest dispatch time. Let $t_{\hat{l}}$ be the run time of $\hat{l}$, and let $\delta_{\hat{l}}$ denote the width of its dispatch window: $\text{CUT}(\hat{l}) - \text{READY}(\hat{l})$. The duration of the duty created when $\hat{l}$ is added to $D$ is computed as follows. First, we compute the beginning and end of the dispatch window of $\hat{l}$ relative to $\sigma_D$, denoted $t_1$ and $t_2$ respectively: $t_2 = \text{CUT}(\hat{l}) - \sigma_D$ and $t_1 = t_2 - \delta_{\hat{l}}$. Then, the dispatch time of $\hat{l}$ relative to $\sigma_D$, $h_{\hat{l}}$, is equal to $\max\{\tau_D, t_1\}$. If $t_1$ is strictly greater than $\tau_D$, then there may potentially be a waiting
time at the origin of \( \hat{\ell} \). Let \( w_{\hat{\ell}} = t_1 - \tau_D \) denote this potential waiting time. Some or all of this waiting time, however, may be eliminated by departing later at the origin of the first load of \( D \). The dispatch time of the first load of the base duty can be delayed by at most \( \phi_D \), the flexibility of the base duty. Thus, the minimum waiting time at the origin of load \( \hat{\ell} \) is \( \bar{w}_{\hat{\ell}} = \max\{w_{\hat{\ell}} - \phi_D, 0\} \). Note that by departing later, the total wait time of \( D \) does not change, and that the total wait time of the extended duty is equal to the wait time of \( D \) added to \( \bar{w}_{\hat{\ell}} \). We now have all the information we need to compute the duration of the extended duty. If the extended duty is feasible with respect to \( \Delta_{DU} \), we also compute its dispatch time and flexibility. Letting \( D_{\text{ext}} \) denote the extended duty, we have

\[
\begin{align*}
\tau_{D_{\text{ext}}} &= \tau_D + \bar{w}_{\hat{\ell}} + t_{\hat{\ell}} \\
\sigma_{D_{\text{ext}}} &= \sigma_D + (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}) \\
\phi_{D_{\text{ext}}} &= \min\{\phi_D - (w_{\hat{\ell}} - \bar{w}_{\hat{\ell}}), t_2 - h_{\hat{\ell}}\}.
\end{align*}
\]

**Trip generation**

Once all time-feasible duties are generated, we generate the set of all feasible trips by matching drivers available at the origin terminal of the selected load (and thus the origin terminal of all the duties) with duties, i.e., by checking whether the driver ready time, remaining driving hours, and remaining duty hours are compatible with the duty. Note that driver \( x \) has a compatible ready time with duty \( D \) if \( \text{READY}(x) \leq \sigma_D + \phi_D \).

If no trips are generated for drivers at the origin terminal of the selected load, trips are created for drivers that start at another terminal and thus “bring in” a driver with an empty move. To control empty repositioning, this process is conducted in two phases. First, all drivers at a predefined set of nearby terminals are considered. Second, if no trips can be generated for drivers at nearby terminals, trips are generated for drivers at more distant terminals with historical empty repositioning to this terminal; this process considers terminals one at a time in order of increasing distance from the origin terminal.

Since one of the objectives is to minimize the number of foreign beds, additional trips are generated that add an empty move as the final leg to return a driver to his domicile when such a move is time-feasible. We note that we do not discard the original trip. These trips compete with all other generated trips.

**Trip scoring and selection**
Each generated trip is evaluated based on how well it contributes to the various problem objectives. Once the best trip is identified, it is scheduled for dispatch, i.e., the driver and duty are dispatched to cover the initial load (and all other loads included in the duty).

Since the cost of a driver schedule is driven by many competing factors, defining an appropriate trip scoring and selection scheme is a challenge. After extensive experimentation, a two-level hierarchical scheme was adopted. The hierarchical scheme reflects a difference in importance between the various goals. The primary performance indicators for a cost-effective operation plan are low empty repositioning miles and high driver utilization. A secondary indicator is the number of driver foreign beds.

Suppose trip $T$ is formed by matching duty $D$ to driver $x$. Let $d_T$ denote the loaded drive time of trip $T$ and $r_T$ the loaded run time of trip $T$. Further, recall that $\tau_D$ is the duration of trip $T$. Furthermore, let $d_T^{\text{max}} \leq \Delta_{DR}$ be the maximum number of available driving hours for driver $x$; we note that $d_T^{\text{max}} = \Delta_{DR}$ in the typical case where the driver $x$ begins the trip after a long rest.

At the first level of the hierarchy, we assign the following initial score value to each trip:

$$s_1^T = \frac{d_T}{d_T^{\text{max}}} + \frac{r_T}{\tau_D}$$

Note that $s_1^T$ will take values between 0 and 2. Let $s_{\text{max}}$ denote the maximum initial score over all trips. Next, we select up to 300 trips with the highest scores among those with scores in the range $[0.9s_{\text{max}}, s_{\text{max}}]$, forming the candidate trip set $C$. At the second level of the hierarchy, we select a single trip for dispatch from $C$. To do so, we assign a second score $s_2^T$ between 0 and 10 to each trip $T \in C$ as follows:

- Add 3 points if trip $T$ is not the first assigned trip for driver $x$ during the planning horizon;
- Add 3 points if driver $x$ begins the trip away from his domicile terminal, DOMICILE($x$);
- Add 2 points if the trip returns to DOMICILE($x$) at the end of the duty;
- Add 2 points if driver $x$ begins away from domicile, and the trip dispatch time, $\max\{\text{READY}(x), \sigma_D\}$, is no greater than six hours after the driver ready time, READY($x$).

We then select for dispatch a trip $T$ with a maximum value of $s_2^T$, breaking ties using the initial score $s_1^T$. Note that giving priority to trips that reuse drivers leads to solutions that
dispatch fewer drivers. Giving priority to trips with drivers away from domicile and that return drivers to domicile leads to solutions with smaller numbers of foreign beds. Finally, prioritizing short waits before dispatch for drivers away from domicile reduces layover costs.

**Updating system information**

Once a trip has been selected for dispatch, the system information is updated to reflect this selection. The system information to be updated includes:

- Driver information at origin of the trip and the destination of the trip. The ready time of the driver at the destination is computed as the arrival time at the destination plus the mandatory time of a long rest, $\Delta R$.

- Load ready and cut time information of affected loads. Recall that there are precedence relations between loads. Therefore, the ready times of successor loads and the cut times of predecessor loads may have to be updated based on the dispatch times for loads to be executed in the selected trip.

Of course, all loads included in the selected trip are marked as dispatched.

**A tabu mechanism**

It is possible that no trip can be generated for a selected load. In this case, the load is declared tabu and unavailable for selection until some trip has been generated and the system information has been updated. After a system update, a driver has been relocated and it may therefore now be possible to generate a trip that includes the load that was declared tabu. Finally, to avoid infinite loops, a load that has been declared tabu more than five times is removed from the list of loads, and therefore will not be dispatched.

**Other issues**

Note that the basic heuristic described in this section does not allow any load to be dispatched after its cut time; service is treated as a hard constraint. We have also implemented versions of the heuristic that attempt to dispatch a load late by modifying its cut time (and the cut times of any successors) by fixed intervals after a load has been declared tabu more than five times; for brevity, results for these versions are not presented in this paper.

Further, it should be noted that the basic heuristic does not explicitly prevent the
dispatch of loads at times that will make undispatched predecessor loads impossible to feasibly dispatch. In general, it is rare for a successor load to be dispatched prior to a predecessor, since we select loads in order of increasing cut time. We have also experimented with a version of the heuristic that prevents this possibility by simply not allowing duties to be extended with loads that have undispatched predecessors; again, for brevity, results for this version are not presented herein.

7 Dynamic Implementation

The solution approach we have developed for the DSLDP is intended for implementation in a dynamic environment using a rolling planning horizon, to be used to provide decision support to linehaul dispatchers. There are several important challenges that need to be addressed to realize this vision.

First, in order for the technology to be useful, solution speeds must be very fast. We set a target of 10 minutes of run time to complete the full driver schedule for a large system. Given that the technology might be used to resolve a DSLDP every few hours, we believe that this run time is acceptable.

Second, it is important to determine an appropriate planning period length and resolve interval. We believe that a three-day planning period is appropriate for large network carriers in order to allow planning of multiple trips for each driver and to minimize end-of-horizon effects. Our system assumes that information is available for all loads to be dispatched within the planning period. In reality, a large number of the loads will be known, and another large number can be predicted fairly accurately. Importantly, shipment-level forecasts are not necessary, although reasonable forecasts of trailer-level ready and cut times are important. A small set of completely unknown loads should not affect solutions drastically, and can be captured by reasonably frequent resolves. In the LTL context, resolves every few hours should be sufficient.

Third and lastly, in a dynamic implementation it is important to carefully choose which feasible prior decisions to maintain, and which to discard before resolving. For the DSLDP, we believe it is appropriate to preserve at the resolve epoch only each driver’s current and next planned load dispatch (as long as they are still feasible). Another reasonable choice might be to preserve each driver’s current planned trip, or preserve at least the high quality
current planned trips for some subset of the drivers.

8 Computational Study

We now present the results of a set of computational experiments used to analyze the performance of the proposed driver management technology. The basis for our computational experiments is a month of historical data from a large national LTL carrier in the U.S. The linehaul network for this carrier contains 648 terminals and 16,405 connections. The terminal set includes breakbulk terminals, end-of-line terminals, relay terminals, rail yards, marshalling yards, and meet-and-turn locations. From a driver management perspective, the inclusion of rail yards, marshalling yards, and meet-and-turn locations does not create significant complications. The primary difference between these locations and terminals is that a driver may not rest to recharge his drive and duty hours at these locations. Since LTL truck traffic varies by month of the year, we chose data for a representative medium-traffic month, March 2005, to test the methodology.

There are a number of challenges that must be handled in the process of generating data sets. The LTL carrier provided the actual movements of loads, the shipments contained in these loads, and the drivers that moved the loads. Using information for the shipments contained in a load, we determine a load cut time, i.e., the latest time at the origin when the load can depart and still make service for all the shipments contained in the load, by considering for each shipment the difference between the due date and the minimum time required for all downstream movement and handling activities. Ready times for loads were calculated differently for two types of historical loads. For loads that were built by loading one or more trailers, the ready time is given by the latest completion time of the trailer loading process. For loads that were built only by pup-matching existing trailers in the system, the ready time is determined by adding for each trailer the minimum upstream movement time along the path from the original trailer build location to its original ready time, and taking the maximum. It is important to note that due to certain historical decisions regarding which trailers to match into a load, it is possible that this method of generating ready and cut times may result in cases where the ready time of a load is later than its cut time. Since this occurs rarely, we arbitrarily set the cut time to be 30 minutes after the ready time for such loads. We also generate a precedence relationship between
each load built by pup-matching trailers and the inbound loads moving those trailers.

8.1 Initial Analysis

An initial analysis of the developed driver management technology was performed using 3-day instances. To create an instance, we extract from the historical data all dispatched loads with a ready time that falls within a 72-hour period. We then use our driver management technology to schedule the movements of these loads (which implies scheduling of empty movements as well). A typical 3-day instance has between 15,000 and 18,000 loads and a pool of approximately 3,500 drivers is available. Table 1 gives the number of loads in each of the eight 3-day instances considered here. Since traffic varies significantly by day of the week, we extracted four 3-day instances \{MW1, MW2, MW3, MW4\} covering Monday through Wednesday loads, and four instances \{WF1, WF2, WF3, WF4\} covering Wednesday through Friday loads. Wednesday and Thursday tend to have the most traffic, while Monday traffic is light; the table reflects this fact since the end-of-week instances have approximately 20% more loads to cover.

<table>
<thead>
<tr>
<th>Instance</th>
<th># Loads</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW1</td>
<td>15,275</td>
</tr>
<tr>
<td>MW2</td>
<td>14,906</td>
</tr>
<tr>
<td>MW3</td>
<td>15,023</td>
</tr>
<tr>
<td>MW4</td>
<td>15,135</td>
</tr>
<tr>
<td>WF1</td>
<td>18,392</td>
</tr>
<tr>
<td>WF2</td>
<td>17,656</td>
</tr>
<tr>
<td>WF3</td>
<td>17,666</td>
</tr>
<tr>
<td>WF4</td>
<td>17,493</td>
</tr>
</tbody>
</table>

Table 1: Number of loads to be dispatched in test 3-day instances

We now present results that aim to validate the algorithmic design choices made for the driver management scheme, \textit{i.e.}, the use of a duty-based approach and the use of enumeration during the generation of duties. To do so, we compare the performance of three different heuristics. The first heuristic (H1) is the one described in detail in the previous section. Recall that H1 greedily selects for dispatch the outstanding load with the
earliest cut time, enumerates a large tree of time-feasible driver duties that begin with the selected load, creates trips by matching drivers with duties, scores the trips, and finally selects the best trip. The second heuristic (H2) is a simplification of the first. Heuristic H2 greedily selects the next load for dispatch using the same criterion as H1. Next, a small set of nested time-feasible driver duties are generated by recursively adding only the outbound load with maximum feasible drive time, breaking ties by choosing the one that leads to minimum wait; this method builds a single “path” of driver duties rather than a tree. This small set of duties is again matched with drivers to create trips which are scored and the best trip selected using the same approach as H1. The third heuristic (H3) is yet a further simplification which does not build a complete duty during each dispatch step. Heuristic H3 again uses the same greedy criterion for load selection, and then simply greedily selects a driver to perform the load by extending his duty (or beginning a new duty if this load is the driver’s first off rest) to include the selected load. To select a driver, the heuristic first considers all drivers with partial duties and checks to see if the load can by added without violating the drive time and duty time restrictions. If so, then among the eligible drivers, the driver with the (partial) duty with maximum loaded drive time is selected, breaking ties by choosing the duty with minimum wait time. If there are no drivers with partial duties that can accommodate the selected load, then the heuristic considers drivers coming off-rest and assigns the driver with the off-rest time closest to the ready time of the selected load is chosen. If no driver is available at the origin terminal, the nearest feasible driver at another terminal is repositioned to serve the load.

Comparative results are presented in Table 2. Each heuristic was run on a 2.4 GHz Pentium Linux PC with 2 GB of memory. There are two groups of results: the first group provides averages of key performance statistics over the four Monday-Wednesday instances MW1–MW4, and the second group gives averages over the four Wednesday-Friday instances WF1–WF4. The first column indicates the heuristic used. The second column gives the percentage of loads that are dispatched. The third column shows the percentage of loads dispatched before the historical (actual) dispatch time provided in the dispatch data set. The fourth column shows the number of empty driver miles as a percentage of the number of total driver miles. The fifth column shows the number of foreign beds as a percentage of the total number of long rests. The sixth column shows the number of drivers used. The seventh and eighth columns show the average drive time and duty time of all assigned driver
trips, in hours. Finally, the last column shows the run time of the heuristic in seconds.

<table>
<thead>
<tr>
<th></th>
<th>%Dispatched</th>
<th>%Dispatched before Historical</th>
<th>%Empty</th>
<th>%Foreign Beds</th>
<th># Drivers</th>
<th>Average Drive Time</th>
<th>Average Duty Time</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>99.52</td>
<td>64.31</td>
<td>11.80</td>
<td>67.49</td>
<td>2.824</td>
<td>8.87</td>
<td>10.29</td>
<td>303.21</td>
</tr>
<tr>
<td>H2</td>
<td>99.49</td>
<td>59.28</td>
<td>12.98</td>
<td>75.76</td>
<td>2.921</td>
<td>7.25</td>
<td>8.35</td>
<td>61.31</td>
</tr>
<tr>
<td>H3</td>
<td>99.50</td>
<td>59.23</td>
<td>12.51</td>
<td>75.60</td>
<td>2.932</td>
<td>7.24</td>
<td>8.34</td>
<td>61.75</td>
</tr>
<tr>
<td>H1</td>
<td>99.21</td>
<td>61.46</td>
<td>9.58</td>
<td>65.10</td>
<td>3.009</td>
<td>8.99</td>
<td>10.26</td>
<td>224.68</td>
</tr>
<tr>
<td>H2</td>
<td>99.22</td>
<td>55.46</td>
<td>10.51</td>
<td>76.91</td>
<td>3.174</td>
<td>7.51</td>
<td>8.62</td>
<td>85.67</td>
</tr>
<tr>
<td>H3</td>
<td>99.24</td>
<td>55.65</td>
<td>11.62</td>
<td>76.90</td>
<td>3.187</td>
<td>7.52</td>
<td>8.64</td>
<td>86.12</td>
</tr>
</tbody>
</table>

Table 2: Comparison of three greedy heuristic solution approaches for DSLDP

A number of observations can be made. First and foremost, it is clear that expending the time to enumerate and evaluate a large set of feasible duties is worth the effort. Heuristic H1 produces substantial improvement over the two other heuristics in all performance indicators. The number of drivers used decreases, the average drive and duty time increase, the empty mileage percentage decreases, and the percentage of foreign beds decreases. It is important to note that the duty enumeration, when implemented carefully, did not significantly increase the computation times, which are still well below the target of 10 minutes. Perhaps surprisingly, heuristics H2 and H3 produced remarkably similar average results. Given that H2 greedily assigns a complete driver duty built to maximize loaded drive time with each load dispatch, we initially expected it to perform somewhat better than H1. In any event, the results show that a greedy search approach for DSLDP can be significantly improved by enumerating a large set of driver duties that cover a load segment to be dispatched.

It is also interesting to observe that the results vary somewhat between the Monday-Wednesday instances and the Wednesday-Friday instances, and that the differences can likely be explained by the fact that the latter include approximately 20% more loads. Each heuristic is able to reduce the percentage of empty miles for the larger instances, although the pure greedy approach H3 produces the least relative reduction. It is also worth noting that heuristics H2 and H3 both perform better relative to H1 measured by average drive and duty hours on the larger instances.
8.2 Rolling Horizon Analysis

The results of the initial analysis are promising, in the sense that the driver management technology is more effective when solving 3-day instances than simpler greedy approaches. Next, we investigate the performance of the technology (heuristic H1) when it is embedded in a rolling horizon framework.

The following rolling horizon scheme was used in our computational experiments: create driver schedules for the next three days, implement the decisions for the first day, roll the horizon forward by one day and repeat. This scheme was used to determine driver schedules for a full week, which implies that seven 3-day instances were solved in the process. The instances were obtained by extracting from the historical data all dispatched loads with a ready time that falls within a 9-day period.

The computational experiment outlined above does not necessarily represent a realistic rolling horizon implementation, because of the assumption of complete load visibility in the 3-day period under consideration. In reality, of course, only a subset of these loads will be known at the time the schedule is constructed. To study the impact of reduced future load visibility, we also consider scenarios where a given percentage of loads from the second and third days of the planning horizon are selected at random and ignored. That is, a driver schedule will be constructed for a 3-day period under the assumption that all loads for the first day are known, but only a subset of loads for the next two days is known. By varying the percentage of ignored loads on the second and third day, we gain some insight on the value of information about future loads.

We have conducted experiments for three different weeks in March of 2005. Each instance determines dispatch decisions for Monday through Sunday, using data from Monday through the following Tuesday. The results are presented in Table 3.

A number of observations can be made. First and foremost, the weekly performance statistics deviate very little from the performance statistics we obtained for a 3-day period, especially in the best-case scenarios in which no future loads are ignored. The driver management technology is still able to dispatch more than 98% of the loads, and it does so with roughly the same number of drivers, with about the same average drive and duty times, and with slight increases in empty mileage percentage and foreign bed percentage. Note that the increase in empty miles is to be expected, since our heuristic partially uses a reactive
<table>
<thead>
<tr>
<th>Week</th>
<th>%Deleted</th>
<th>%Dispatched</th>
<th>%Dispatched</th>
<th>%Empty</th>
<th>%Foreign</th>
<th># Drivers</th>
<th>Average Drive Time</th>
<th>Average Duty Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 7–13</td>
<td>0</td>
<td>98.03</td>
<td>67.62</td>
<td>12.69</td>
<td>70.75</td>
<td>2950</td>
<td>8.84</td>
<td>10.16</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>97.89</td>
<td>67.45</td>
<td>12.68</td>
<td>71.28</td>
<td>2881</td>
<td>8.86</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>97.74</td>
<td>66.87</td>
<td>12.50</td>
<td>71.33</td>
<td>2813</td>
<td>8.89</td>
<td>10.21</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>97.28</td>
<td>66.49</td>
<td>12.49</td>
<td>71.99</td>
<td>2741</td>
<td>8.89</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>96.74</td>
<td>65.90</td>
<td>12.47</td>
<td>71.77</td>
<td>2657</td>
<td>8.91</td>
<td>10.22</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>95.12</td>
<td>64.26</td>
<td>12.42</td>
<td>72.13</td>
<td>2507</td>
<td>9.02</td>
<td>10.32</td>
</tr>
<tr>
<td>Mar 14–20</td>
<td>0</td>
<td>97.97</td>
<td>68.24</td>
<td>12.72</td>
<td>71.83</td>
<td>2897</td>
<td>8.86</td>
<td>10.21</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>97.88</td>
<td>67.19</td>
<td>12.53</td>
<td>71.62</td>
<td>2832</td>
<td>8.88</td>
<td>10.24</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>97.38</td>
<td>66.90</td>
<td>12.79</td>
<td>71.84</td>
<td>2753</td>
<td>8.91</td>
<td>10.25</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>96.93</td>
<td>66.35</td>
<td>12.87</td>
<td>71.93</td>
<td>2677</td>
<td>8.93</td>
<td>10.27</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>96.46</td>
<td>66.12</td>
<td>12.74</td>
<td>71.84</td>
<td>2618</td>
<td>8.96</td>
<td>10.29</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>94.55</td>
<td>64.75</td>
<td>12.38</td>
<td>72.71</td>
<td>2443</td>
<td>8.99</td>
<td>10.32</td>
</tr>
<tr>
<td>Mar 21–27</td>
<td>0</td>
<td>98.13</td>
<td>68.05</td>
<td>13.57</td>
<td>70.29</td>
<td>2940</td>
<td>8.87</td>
<td>10.24</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>97.90</td>
<td>68.16</td>
<td>13.49</td>
<td>70.77</td>
<td>2882</td>
<td>8.9</td>
<td>10.25</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>97.65</td>
<td>67.92</td>
<td>13.24</td>
<td>71.15</td>
<td>2788</td>
<td>8.9</td>
<td>10.23</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>97.21</td>
<td>67.00</td>
<td>13.24</td>
<td>71.19</td>
<td>2719</td>
<td>8.93</td>
<td>10.24</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>96.41</td>
<td>66.07</td>
<td>13.30</td>
<td>71.67</td>
<td>2637</td>
<td>8.96</td>
<td>10.29</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>94.55</td>
<td>65.20</td>
<td>13.07</td>
<td>72.27</td>
<td>2468</td>
<td>9.04</td>
<td>10.35</td>
</tr>
</tbody>
</table>

Table 3: Results from rolling horizon experiments

mechanism to bring drivers empty back to locations that are outbound-heavy; this effect is masked somewhat in the 3-day instances, where drivers are reset to the initial state for each run.

Furthermore, it seems that the driver management technology is fairly robust in the sense that reduced visibility of future loads does not affect the performance significantly. The percentage of loads dispatched decreases only by about 3 percentage points when 30% of the loads are ignored on the second and third day, a fairly small change given the amount of data ignored. The empty mileage percentage and average drive and duty times do not seem to be affected at all. The most striking statistic is the reduction in the number of drivers required to move loads, which of course is due in part to the reduced number of loads dispatched. However, it should be noted that the average number of loads moved by a driver during the week increases from a little over 12 to almost 14.
8.3 Comparison with Historical Dispatch Data

Finally, we evaluate the relative effectiveness of the driver management technology by comparing the results to statistics generated from the historical data. Although such a comparison is not without flaw, we believe it does provide some insights into the potential value of the driver management technology under development. There are two primary issues to keep in mind when analyzing comparative results. First, our driver management scheme has a bit more flexibility when assigning drivers to loads, since we do not explicitly account for certain union regulations. Union regulations come in the form of driver bids, which restrict the locations that drivers can visit. An example of a driver bid is an “AB-bid.” This indicates that the first destination of a driver after he comes off rest at his domicile is restricted. More specifically, if the domicile of the driver is breakbulk A, then the first destination has to be either breakbulk B or one of its aligned satellites. Second, in this section we assume complete knowledge of all loads that must be moved during each planning period.

Table 4 summarizes a comparison between the solutions generated by our driver management scheme H1 and actual statistics generated from March 2005 historical dispatch data. The columns of the table summarize key performance metrics. The first column reports the average number of unique drivers dispatched during a day, a good indication of the required driver pool size. The second column reports the average empty miles per day required by the dispatch schedule. Finally, the third and fourth columns report average drive and duty times for dispatched drivers.

<table>
<thead>
<tr>
<th></th>
<th># Drivers</th>
<th>Empty Miles (per day)</th>
<th>Average Drive Time</th>
<th>Average Duty Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Historical Dispatch</td>
<td>3209</td>
<td>181,577</td>
<td>8.41</td>
<td>9.20</td>
</tr>
<tr>
<td>H1 3-day Mon-Wed</td>
<td>2824</td>
<td>122,983</td>
<td>8.87</td>
<td>10.29</td>
</tr>
<tr>
<td>H1 3-day Wed-Fri</td>
<td>3009</td>
<td>119,610</td>
<td>8.99</td>
<td>10.26</td>
</tr>
<tr>
<td>H1 Rolling</td>
<td>2929</td>
<td>146,169</td>
<td>8.86</td>
<td>10.20</td>
</tr>
</tbody>
</table>

Table 4: Comparison with historical dispatch data for a large U.S. LTL carrier using average daily statistics for March 2005

The results in the table indicate that the developed solution technology is quite promis-
ing. In each experimental setting and for each performance metric, the heuristic solution outperforms the actual dispatch data, often by a substantial margin. Importantly, the schedules determined by the heuristic solution approach typically require approximately 6–10% fewer drivers, with each driving an additional 1/2 hour during each duty. Driver schedules with high drive times are not only good for the carrier, since they potentially reduce the fixed costs of drivers such as benefit compensation but also since they potentially decrease turnover costs by increasing driver job satisfaction since they are paid by the mile driven.

Note also that in the historical dispatch data, the average duty and drive times are separated by about 0.8 hours whereas for the heuristic they are separated by about 1.3 hours. Thus, the heuristic solutions include an additional half hour of waiting time. In contrast, real-world dispatchers tend to get drivers on the road again quickly when they complete a load but do not require long rest. The heuristic appears more likely to choose a duty for a driver that includes some “wait” to find a better load candidate.

9 Conclusions and Future Work

This paper describes a promising driver management scheme for large-scale network dispatch problems based on combining greedy search ideas with partial solution enumeration. Results from the computational study justify the algorithmic choices made, and a comparison with real-world dispatch data indicates that the quality of the solutions produced is very good. The heuristic can solve scheduling problems with 15,000–20,000 loads to be dispatched and thousands of drivers in approximately 5 minutes of PC computing time.

Although the scheme is designed to be implemented in a dynamic setting to provide real-time decision support, the technology is also currently being used for tactical planning such as determining appropriate driver pool levels at various network terminals. Other uses include providing tactical driver planning support during periods of “unusual” operation, generating new driver plans when terminals change/close due to merger/divestiture activities, and generating new driver plans when DOT driver regulations change.

Finally, we are currently developing several extensions to enhance the functionality of the driver management technology. The primary modeling extension that we are focusing on is incorporating basic union driver rules into the system. In addition, we also intend to
model pup-matching and load rerouting decisions in the future. We are also considering several potential algorithmic extensions, including speeding up solution time through reuse of the enumeration trees and improving the solution quality by implementing the search in a greedy randomized adaptive search procedure (GRASP).

Acknowledgements

The work described in this paper was supported, in part, by a grant from the Sloan Trucking Industry Program.

References


