in previous research. Then, as extensions, we show the additional benefits of this system: (1) to help transportation companies to make strategy-level decisions by simulating alternatives using DMDS; (2) to utilize the learned dispatching rules to improve the computational quality and speed of a sampling-based optimization algorithm for LPDPs (Pi et al. 2008); (3) to quickly generate better-than-actuals solutions by adding optimized solutions to the training data set and re-train the dispatch rules.

The rest of the paper is organized as follows. Section 2 provides the problem description and formulation of LPDPs. In Sect. 3, we introduce the DMDS framework and describe its main modules. In Sect. 4, we apply the DMDS to the intermodal industry and show the test results. Then, contributions of DMDS are further extended in Sect. 5 by applying the system to help decision support and to expedite an optimization algorithm for LPDPs. Finally, conclusions are drawn in Sect. 6.

2 Problem description

The basic operations involved in LPDPs can be described as follows (Wang and Regan 2002): On a daily basis, a fixed number of drivers (who are associated with vehicles) initially locate at certain positions. Each driver has a service time window which is imposed by work rules. A driver can only handle one load at a time. Within the time window, a driver goes to a location (origin) to pick up a load and delivers it to another location (destination). After the delivery of a load, the driver goes to serve another load or becomes idle. During each load service, there are loaded movements which generate revenues and incur costs, and empty movements which incur costs but generate no revenue.

In the following, we present a LPDP formulation with some common problem constraints and key performance indicators (Wang and Regan 2002; Pi et al. 2008). However, the system framework we provide can be potentially applied to LPDPs with more complicated constraints. We use the following notation:

Sets:

- $K = \{1, \ldots, |K|\}$: the set of drivers.
- $L = \{1, \ldots, |L|\}$: the set of nodes representing loads which need to be served.
- $S = \{|L| + 1, \ldots, |L| + |K|\}$: the set of start nodes representing the starting locations of the drivers. $\forall k \in K$, $|L| + k$ is the start node of driver k.

Parameters:

- a_l, $l \in L \cup S$: the start time of each node. For $l \in S$, a_l represents the work start time of driver $l - |L|$.
- b_l, $l \in L \cup S$: the end time of each node. $l \in S$, b_l represents the work end time of driver $l - |L|$. For $\forall l \in L$, $[a_l, b_l]$ denotes service time window of load l.
- d_l, $l \in L \cup S$: the service duration of each node, which can be assumed to be a constant value. For $l \in L$, d_l includes the pickup duration of load l, the loaded travel time from pickup location to drop-off location of load l, and the drop-off duration of load l. For $l \in S$, d_l is assumed to be 0.
- v: the vehicle speed, which is assumed to be a constant.
- s_l, $l \in L \cup S$: the loaded travel time of each node. For $l \in L$, s_l is the loaded travel time from pickup location to drop-off location of load l. Loaded travel time is proportional to the loaded travel distance. For $l \in S$, s_l is assumed to be 0.
- T_{ij}, $i \in L \cup S$, $j \in L$: the empty travel time from drop-off location of load i to pickup location of load j. Empty travel time is proportional to the empty travel distance.

Table 2 Load attributes

Name	Description
LSTime	Start time of a load
LETime	End time of a load
LCurLoc	Current location of a load
LDist	Loaded travel mileage of a load
LEstTime	Estimated service duration of a load
LTrailerType	The type of trailer a load requires

Table 3 Driver attributes

Name	Description
DBoard	Driver board number
DSTime	Start time of a driver
DETime	End time of a driver
DHomeLoc	Home location of a driver
DCurLoc	Current location of a driver
DDueTime	Estimated remaining time for a driver to finish current task
DRemTime	A driver's estimated remaining work hours
DRemMiles	A driver's estimated remaining mileage during the work hours
DTrailerType	The type of trailer a driver currently has in tow

Table 4 Load-driver pair attributes

Name	Description
DLTimeDiff	Difference between a driver's remaining work hours and a load's service duration
DLDist	Distance between a driver and a load

decisions are made, such as when a driver is available or a load enters the system, L_t and K_t are the set of available loads and drivers at time point t, respectively.

According to our observation, potentially useful attributes for LPDPs are provided in Tables 2, 3, and 4.

Given each load-driver pair $\mathcal{I}_{lk}$ at time t, a decision value $\mathcal{D}_{lk}$ is associated. The decision value is numerical $\{0, 1\}$ (or nominal $\{yes, no\}$ depending on which kind of machine learning technique we choose). At time t when load l was actually assigned to driver k, a decision value "1" is assigned to $\mathcal{D}_{lk}$; for other loads l' ($l' \in L_t \backslash \{l\}$), $\mathcal{D}_{l'k}$ is assigned "0". The load-driver pair attribute vector and the related decision value form one piece of a training sample:

$$\mathcal{S}_{lk} = [\mathcal{I}_{lk}, \mathcal{D}_{lk}], \quad \forall l \in L_t,\ \forall k \in K_t,\ \forall t \in T.$$

The general process of preparing training samples is shown in Fig. 2.

In addition to exploring the right data as the training samples, we can make the inputs more amenable for learning methods using data engineering techniques (Witten and Frank 2005; Li and Ólafsson 2005). Key data engineering techniques that we found useful to our problem are as follows.

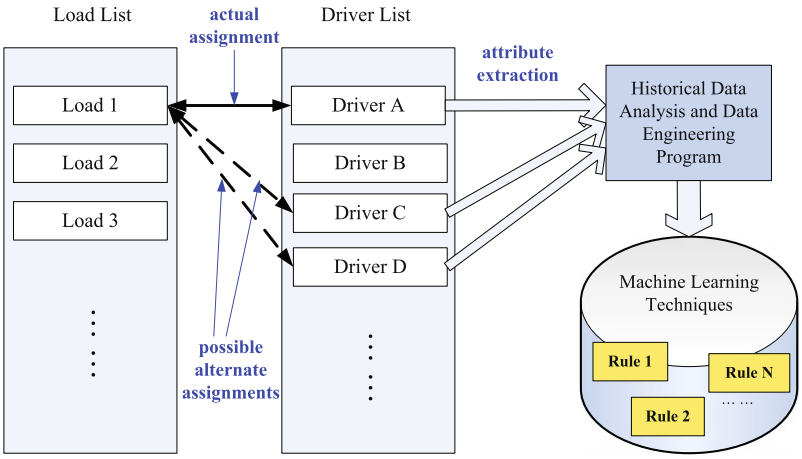


Fig. 2 General process of preparing training samples

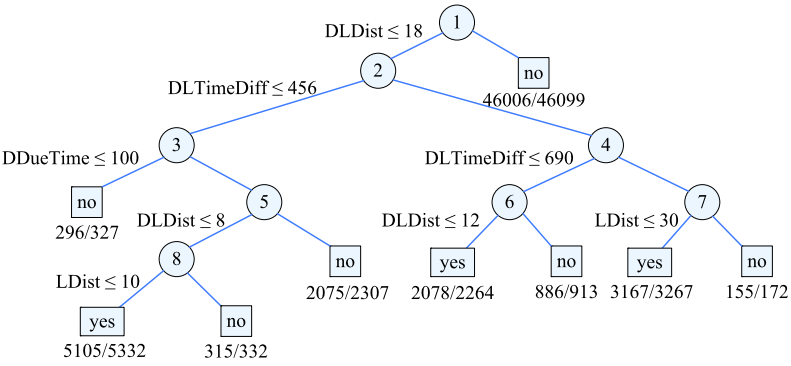
– Data cleansing: Records from the data set may contain incorrect records that caused either by the actual errors made by dispatchers in practical operations, or by mistakes that occurred when historical operations were stored into the database. Detecting and correcting/removing these inaccurate records is necessary for the success of data mining.
– Data transformation: Simple arithmetic transformations of existing attributes may yield new attributes which can benefit the learning results. An example is the load-driver pair attributes. For instance, attribute "DLDist" is the distance between a driver and a load, generated using the geographical locations of the driver and the load in the database.

3.2 Dispatching rule learning

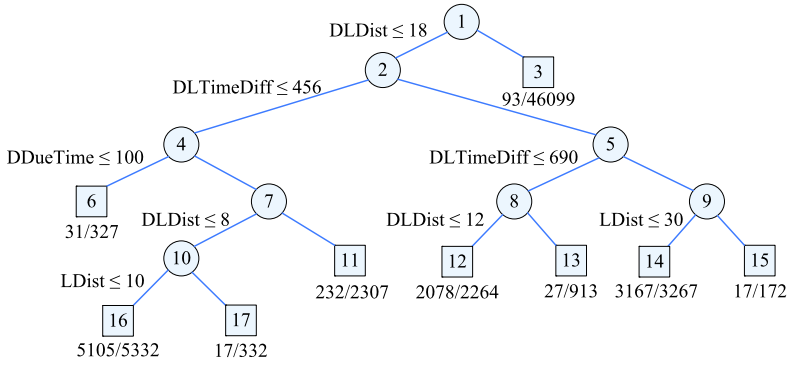
After a set of training samples are extracted from raw data, machine learning techniques are applied to learn the embedded patterns and preferences of human dispatchers. Each machine learning algorithm generates a policy π, representing the learned dispatching rules. After examining various popular learning algorithms, we decided that tree structure is the most suitable for this type of problems. Several key characteristics that we want to emphasize are as follows.

– Decision trees use a white box model, compared to the black box model used by some other algorithms, such as neural networks. The structure of a tree is transparent and easy to understand and interpret. This property can enable us to learn important attributes from a well-trained tree.
– The process of producing predictions from decision trees and implementing them is fast. Once the training process finishes and a decision tree is built, that tree can be used to predict the results for new inputs rapidly. This is important for solving large-scale LPDPs with many decisions to be made.
– The training samples have an attribute-value form, which are the preferred form of instances for decision trees.

A number of methods have been developed to induct decision trees over the past twenty years in the machine learning community. Two types of trees exist, including classification trees and regression trees (Breiman et al. 1984). We use the ID3 (Quinlan 1986) to induct



(a) A classification tree



(b) A regression tree

Fig. 3 Examples of decision trees

a classification tree, and use Logistic Regression Tree with Unbiased Selection (LOTUS) method (Chan and Loh 2004) to induct a regression tree. Although using different techniques for tree induction (Long et al. 1993), both a classification tree and a regression tree are suitable to represent dispatching rules. Simple examples of both trees are shown in Fig. 3. Note that in real applications, a generated tree consists of more nodes and is much more complicated. We use 10-fold cross validation, and it takes about 15–20 seconds to fit a tree once the data are loaded. The actual fitted trees vary from case to case. Typically, a fitted tree has 40–60 terminal nodes and 80–120 total nodes. A tree may have 8–12 levels. Each terminal node has minimum 50 training instances. The classification accuracy of the trees is more than 99 %. Although the examples in Fig. 3 are not actual trees, the most important variables are represented out of all attributes in Tables 2, 3, and 4. For most cases, "DLDist" is the top-level node, and also presents in many sub-branches. Other important attributes include "DLTimeDiff", "DDueTime" and "DRemTime". These observations are actually consistent with the thinking process of those dispatch experts. To assign a load to a driver, he/she usually checks how far away the driver's delivery location is from the load's pick-up location, how long this driver takes to serve current load, how many hours the driver can work, whether there is enough time for this driver to serve this load, etc.

In Fig. 3(a), the classification tree shows that "DLDist" and "DLTimeDiff" are among the more important attributes. For example, if one load-driver pair has a "DLDist" greater

than 18, the observation goes to the right branch, meaning that the load will not be assigned to the driver since the corresponding leaf node is "no". The number besides this leaf node is the sample proportion at this node, called correct ratio. That is, 46006 instances out of 46099 samples have a decision value "no". If multiple pairs lead to a "yes" node, the proportion can be used to break the ties. The regression tree in Fig. 3(b) has the similar interpretation, except that the leaf node represents the ratio of cases with $\mathcal{D}_{lk}$ equals 1 (or "yes") to the node sample size. The driver-load pair with the highest value of ratio will be chosen. Ties can be broken arbitrarily.

Among those available attributes in the database, some of them are irrelevant or redundant. We use domain knowledge to pre-select a set of possible attributes, and then use standard feature selection methods (Witten and Frank 2005) to eliminate unimportant attributes. Selection of useful attributes can frequently improve the performance of learning methods (Liu and Motoda 1998). A separate set of tuning data is used to perform attribute selection.

3.3 Solution generation

Using machine learning techniques, a well-trained decision tree (or other type of learning result) is generated to represent dispatching rules. Decisions are made for the new observations at each decision epoch. If a regression tree is used, a numerical value between 0 and 1, called assignment probability, is the output for each pair of new load-driver observation. If a classification tree is used, the output is expected to be a boolean value ($\{0, 1\}$ or $\{yes, no\}$) indicating the classification. But due to the large number of available pairs, more than one pair of observation can be classified as "yes", and ties have to be broken using the correct ratio, which can be easily transformed to the form of an assignment probability ("no" decisions are interpreted as 0 probability). The assignment probability is denoted by $p_{lk} = \pi(\mathcal{I}_{lk})$ $(l \in L_t,\ k \in K_t,\ t \in T)$. Here, p_{lk} represents the probability of assigning load l to driver k at time t. Generally, the bigger p_{lk} is, the more likely load l will be assigned to driver k.

Another significant problem is that dispatching rules can only make static decisions whereas the dispatch process is dynamic. The attribute vector of any single driver or load changes over time, and that can possibly change the next dispatch decision. In order to find promising assignments dynamically, the solution generation process is designed to run in a rolling horizon manner. Given a certain learned rule, several solution generation policies can be used, which provides great flexibility to generate different sets of solutions:

Driver-oriented policy: The drivers are tracked over the course of the day. For example, once driver k finishes servicing one load at time t, each available load l $(\forall l \in L_t)$ form a load-driver pair with driver k. For each attribute vector $\mathcal{I}_{lk}$, an assignment probability p_{lk} is calculated under the learned rule π. For all these load-driver pairs, load $\hat{l} = \arg\max_{l \in L_t} p_{lk}$ is assigned as driver k's next task.

Load-oriented policy: The load-oriented policy is different from the driver-oriented policy by tracking the loads during their task life cycles instead. For example, at some time t, a list of drivers k $(\forall k \in K_t)$ are available to serve load l. For all possible load-driver pairs, driver $\hat{k} = \arg\max_{k \in K_t} p_{lk}$ is assigned to serve load l. In the real world, some fraction of the loads to be moved in a given day become known only a short time before the time when they have to be served. Loads may also need to be reassigned due to traffic, weather, or facility delays (Wang and Regan 2002). For these situations, the load-oriented approach can be useful for capturing the dynamic load information.

Batch-assignment policy: If a list of loads l ($l \in L_t$) and a group of drivers k ($k \in K_t$) are previously known at certain time t, we can design a more complex policy to find out the best assignment schedule for these loads and drivers. The following deterministic assignment problem is solved at each decision epoch t.

$$\begin{aligned} \max \quad & \sum_{l \in L_t, k \in K_t} p_{lk} \cdot f_{lk} \\ \text{s.t.} \quad & \sum_{l \in L_t} f_{lk} \leq 1 \quad \forall k \in K_t \\ & \sum_{k \in K_t} f_{lk} \leq 1 \quad \forall l \in L_t \\ & f_{lk} \in \{0, 1\} \quad \forall l \in L_t,\ \forall k \in K_t. \end{aligned}$$

In the above formulation, f_{lk} is a binary variable which indicates whether load l is assigned to driver k at time point t. The binary variables can be further relaxed as continuous variables with lower bound as 0 and upper bound as 1.

In our testing cases, we use the driver-oriented policy. The reason is that an event to trigger the decision process is usually caused by the change of status of drivers, i.e., a driver started to work or just finished a load. Thus, the driver-oriented policy is more suitable for our testing scenario. A summary of the assignment process is shown in Algorithm 1.

Algorithm 1 Driver-oriented assignment

```
while end time has not been reached do
    Move to the next event time t, i.e., a driver k ∈ K_t is available
    for each available load l ∈ L_t do
        Generate the attribute list I_lk
        Calculate the assignment probability p_lk via learned rule π
    end for
    Find load l̂ = arg max_{l∈L_t} p_lk
    Mark driver k as unavailable, and load l̂ as assigned
    Add the finish of load l̂ to the event list, if driver k is still available at that time
end while
```

4 Real-world example

In this section, we present a real-world example where the DMDS is adopted for a LPDP from the intermodal freight industry.

4.1 Background

Truck intermodal operations combine the cost-effectiveness of rail for long haul (linehaul) movements and the flexibility of trucks for initial and final short haul (drayage) movements. Intermodal freight transportation has become one of the most financially attractive and environmentally responsible modes of transportation. A typical intermodal hub is usually defined within a certain area and includes the following resources (as shown in Fig. 4): intermodal

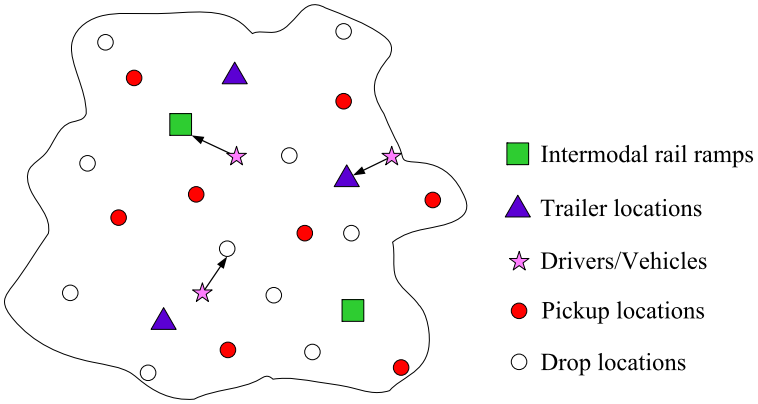


Fig. 4 Intermodal hub

rail ramps, drop locations, pickup locations, drivers (who are associated with tractors), and trailers.

Generally, the operations at each hub can be modeled as a LPDP. We have made the following assumptions in our example:

- We only consider those loads which are delivered to or from the locations within a single hub. We ignore a small fraction of loads which are delivered to locations outside the hub.
- We only consider the local drivers who usually do local deliveries. In real situations, regional drivers (who usually do long distance one way deliveries) and third party drivers can sometimes be used to do local deliveries.

4.2 DMDS setting

We received the historical data from Schneider National Inc., one of the largest trucking companies in North America. We chose the Dallas hub for demonstration purposes. It is one of Schneider National's busiest hubs.

The key data that we use include: (1) events data, which record all the pickup, drop-off, and other types of events that happened during the days; and (2) attributes data, which are the snapshot of load and driver attributes at the beginning of each day (e.g., 6AM).

After raw data processing, we have the following attributes for use in our data mining exercise.

- Load attributes: LSTime, LETime, LCurLoc, LDist, LEstTime.
- Driver attributes: DSTime, DETime, DCurLoc, DDueTime, DRemTime, DRemMiles.
- Load-driver pair attributes: DLTimeDiff, DLDist.

We used 25 days of historical data for training. The solution generation process is based on the driver-oriented policy. The key performance indicators *EM*, *LM*, *TM*, *ER*, and *LR* (refer to Table 1) are used to evaluate solution quality. Both classification tree (CT) inducted by ID3 and regression tree (RT) inducted by LOTUS are tested.

4.3 Testing results

We test the performance of a set of testing data consisting of ten instances (ten days). We also compare the performance of the solutions generated by the DMDS to the historical

Table 5 Performance of actual solutions

Instance	*EM*	*LM*	*TM*	*ER* (%)	*LR* (%)
1	2753.8	8673.6	11427.4	24.10	75.90
2	4104.5	8108.4	12212.9	33.61	66.39
3	3389.4	8322.1	11711.5	28.94	71.06
4	3074.0	8270.1	11344.0	27.10	72.90
5	2959.3	7460.2	10419.5	28.40	71.60
6	2377.2	6836.0	9213.2	25.80	74.20
7	2466.9	6872.5	9339.4	26.41	73.59
8	2810.8	7569.9	10380.7	27.08	72.92
9	2600.5	7582.6	10183.1	25.54	74.46
10	2578.0	6987.8	9565.8	26.95	73.05
Average	2911.4	7668.3	10579.8	27.52	72.48

Table 6 Performance of solutions generated by CT

Instance	*EM*	*LM*	*TM*	*ER* (%)	*LR* (%)
1	2576.6	8749.5	11326.1	22.75	77.25
2	3032.8	7754.1	10786.9	28.12	71.88
3	3199.1	8586.4	11785.5	27.14	72.86
4	2297.7	7212.8	9510.5	24.16	75.84
5	2866.0	7322.4	10188.4	28.13	71.87
6	2255.4	7017.3	9272.7	24.32	75.68
7	2371.1	6973.8	9344.8	25.37	74.63
8	2593.0	6857.1	9450.1	27.44	72.56
9	2878.1	7035.6	9913.7	29.03	70.97
10	2816.3	6973.8	9790.1	28.77	71.23
Average	2688.6	7448.3	10136.9	26.52	73.48

decisions made by dispatchers. The performance of actual solutions from dispatchers is shown in Table 5, and the performance of our computational results is shown in Tables 6 and 7. Here, one instance consists one day's data, and the performance indicators are *EM*, *LM*, *TM*, *ER* and *LR* as introduced in Table 1. (The numerical results presented in this paper are not real values due to confidentiality considerations. We manipulated the results in the following manner: all the *EM* values are multiplied by an identical coefficient ω_e, and all the *LM* values are multiplied by another coefficient ω_l.)

The solutions generated by the DMDS using "CT" and "RT" are slightly better than actual solutions from experienced dispatchers, with respect to key performance indicators. For example, the average loaded ratio (*LR*) of the actual solutions is 72.48 % and the average *LR* of the generated solutions is 73.48 % using "CT", and 73.91 % using "RT". It indicates that our DMDS is capable to learn useful dispatch knowledge from historical data. The fact that generated solutions are slightly better than actual solutions may be caused by the randomness of some parameters in real situation, which we assume to be constant in DMDS, such as vehicle speed (v) and travel time (T_{ij}). The simulation results also show that "RT" performs slightly better than "CT", but the difference is not obvious. In the remaining tests, we therefore choose "RT" as our default option. It is worth mentioning here that the computational

Table 7 Performance of solutions generated by RT

Instance	*EM*	*LM*	*TM*	*ER* (%)	*LR* (%)
1	2611.3	8749.5	11360.8	22.98	77.02
2	2936.2	7754.1	10690.3	27.47	72.53
3	3262.1	8586.4	11848.5	27.53	72.47
4	2090.7	7212.8	9303.5	22.47	77.53
5	3107.1	7322.4	10429.5	29.79	70.21
6	2077.5	7017.3	9094.9	22.84	77.16
7	2279.3	6973.8	9253.1	24.63	75.37
8	2347.0	6857.1	9204.0	25.50	74.50
9	2658.0	7035.6	9693.6	27.42	72.58
10	2916.8	6973.8	9890.5	29.49	70.51
Average	2628.6	7448.3	10076.9	26.09	73.91

time on generating the solutions for each instance is typically around 1 minute, compared to the much longer time using optimization (such as 15 minutes as we set in our optimization algorithm in next section). It is one of the benefits of using data mining-based approach.

5 Application extensions

In the previous section, DMDS targets on learning actual load assignment rules and generating solutions comparable to those generated by dispatch exporters in a rolling horizon manner. In this section, we demonstrate that DMDS is also valuable in helping strategic decision making in industry, and further improving the solution quality by combining with an optimization algorithm for LPDPs.

5.1 Strategic decision making

In trucking companies, some strategy-level decisions have to be made, in order to make the company more competitive in the industry, or to increase the total revenue by reducing certain operational costs. One important issue in trucking companies is the selection of the drivers' park locations, also called domicile locations, from several alternatives. The trucking company typically has several park locations where the drivers start and end their daily tasks. Usually, a park location is chosen to be close to a rail ramp where loads are frequently moved in and out, so that the drivers may be assigned to pickup a load from the ramp without adding too much empty mileage. Sometimes, due to the change of the company's supply chain, such as adding or removing a ramp, a new park location might be needed, or it could be necessary to relocate a park location. The decisions were usually made by the manager who chooses a location according to his/her experience. It seems to be a good way so far because no real change could be made before the decision is made.

Now, with the help of DMDS, more reliable decisions can be made by running the system to simulate the real situations. Since the DMDS generates schedules for the operational level planning quickly, it can be a great tool to facilitate such strategy-level decision making. For the park location selection problem, the manager usually has several options to choose from, based on the building cost and land availability of such locations. Then, we can mimic the real operations by adding or removing a park location to the dispatching system, and

Table 8 Simulation results of test scenarios

Scenario	*IEM*	*EM*	*LM*	*TM*	*ER* (%)	*LR* (%)
Actual	693.7	2803.3	8392.2	11195.4	25.04	74.96
0	707.3	2879.1	8422.0	11301.1	25.48	74.52
1	685.2	2818.0	8422.0	11239.9	25.07	74.93
2	642.9	2938.6	8422.0	11360.6	25.87	74.13
3	438.6	3303.2	8422.0	11725.2	28.17	71.83

observe the average performance of each scenario. It is a reliable and valuable tool to help the manager rule out some bad options, and assist him/her to make a choice among the most competitive candidates.

Due to confidentiality considerations, we present an engineered example, with results multiplied by the same coefficients as mentioned in Sect. 4. Similar data from Schneider National at Dallas hub is used, but with different days. The testing scenario is described as follows. The default domicile location at Dallas hub for Schneider National is its Operations Center (OC). Observing the historical data, it shows that a large amount of loads went through a location called XYZ (name is made up here for confidentiality reason). The proposed decision is to build up a domicile location at XYZ, so that the operational cost can be lowered. We further design four test scenarios. Scenario 0: maintain the current flow without considering XYZ; Scenario 1: 50 % drivers parked at OC move to XYZ; Scenario 2: 75 % drivers parked at OC move to XYZ; Scenario 3: 100 % drivers parked at OC move to XYZ. We simulated 10 days' data and the average results are presented in Table 8. In Scenarios 2 and 3, the drivers who move to XYZ are randomly selected. Table 8 shows that there is no obvious benefit to open a domicile location at XYZ. Although assigning more drivers to park at XYZ can reduce the initial empty mileage, which may improve drivers' satisfaction, the total loaded mileage and the loaded ratio does not increase as expected (even decrease significantly by about 3 % for Scenario 3). Therefore, the decision makers may need to further investigate the total set-up cost of XYZ, such as land cost, building cost and maintenance cost, in order to validate its financial benefit. Furthermore, we validate again that the solutions generated by DMDS is close to the solutions generated by dispatchers, by comparing Scenario 0 to the real situation (row with Scenario "Actual").

The above example demonstrates how DMDS is beneficial to assist the decision makings when alternatives are expensive to compare in the real world. The decision makers will be able to compare possible outcomes by simulating real operations without making any real change to the current system. However, current dispatching system is not optimal most of the time, and it is possible that the decision may change under an improved system. In Sect. 5.2, we will show how DMDS can help achieve better solutions in a faster manner by combining with a heuristic algorithm for LPDPs.

5.2 Improving performance of optimization algorithm

For solving large-scale LPDPs in industry, many methods have been proposed as reviewed in Sect. 1. Some methods (Dumas et al. 1991; Desrosiers et al. 1986; Lim et al. 2006) rely on the power of mathematical programming. However, these methods can only solve small-size instances. Therefore, heuristic/metaheuristic methods are developed to solve large-scale problems (Lim et al. 2006; Wang and Regan 2002; Nanry and Barnes 2000; Mourkousis et al. 2003; Funke et al. 2005; Campbell and Savelsbergh 2004). These heuristics, including metaheuristics, aim to find good solutions quickly or target on certain types of

problems difficult to formulate or/and solve by standard mathematical programming algorithms. There is the prospect of combining the strength of mathematical programming with the flexibility of metaheuristics. In Pi et al. (2008), we proposed a hybrid nested partitions and mathematical programming (HNP-MP) approach for combinatorial optimization. The HNP-MP approach adopts nested partitions (NP) (Shi and Ólafsson 2000) as the framework to partition the solution space and focus search effort on the most promising region which is further partitioned in the next iteration. The samples are drawn from each subregion to decide which one is most promising, and the standard mathematical programming is used to evaluate the quality of each sample.

We will briefly present HNP-MP for solving LPDPs, and then introduce how DMDS can help improve the performance of HNP-MP. Here, the mathematical formulation for LPDPs is represented by Eqs. (1)–(6), although there are modifications to the formulation when applying to the intermodal example in Sect. 4.

HNP-MP algorithm for LPDPs

S0: (*Initialization*) Set the most promising region as the whole solution space, that is, fix $x_{klj} = 1, \forall (k, l, j) \in Q$ where $Q = \emptyset$. Set the surrounding region, complementary region of the most promising region, as $\emptyset$. Go to S1.

S1: (*Partial sampling*) For the most promising region, solve the linear programming (LP) relaxation of formulation (1)–(6) fixing $x_{klj} = 1, \forall (k, l, j) \in Q$, and denote the LP solution by $(\hat{x}_{ijk}, \hat{t}_l)$. Randomly draw partial samples based on the LP solution. Follow similar steps for the surrounding region.

A partial sample is a feasible solution in which every load is assigned to a driver, but the service sequence and start time of each load is not determined. Mathematically, if load l is assigned to driver k, fix $x_{k'lj} = 0, \forall k' \in K, k' \neq k, j \in L$.

The *LP solution-based sampling* is performed in the following fashion: $\forall k \in K, l \in L \cup S$, define $Z_{kl} = \sum_{j \in L} \hat{x}_{klj}$. The probability that load l is assigned to k is linear correlated to Z_{kl} positively.

S2: (*Evaluation*) Solve the MIP (1)–(6) using mathematical programming solver, for each partial sample (with $x_{k'lj} = 0$ fixed in S1). Compare the objective values of all samples. If the best sample (x^*_{ijk}, t^*_l) is from the most promising region, go to S3; otherwise, go to S4.

S3: (*Further partitioning*) For all $x^*_{klj} = 1$, add N pairs of (k, l, j) to set Q, where the N selected pairs have the highest values of Z_{kl}. This is called *LP solution-based partitioning*, where N is a chosen parameter. Go to S5.

S4: (*Backtracking*) Set the superregion that contains current most promising region as the next most promising region. Go to S5.

S5: (*Stopping criteria*) If the stopping conditions satisfy, stop and output the current best solution; otherwise, go to S1.

The HNP-MP approach has been applied to solve large-scale LPDPs and has shown its advantages in solving randomly generated instances (Pi et al. 2008). However, in solving real world settings, we have found that the quality of solutions depends on the instances. For many cases, *LP solution-based sampling* and *LP solution-based partitioning* do not lead to high quality samples and partitions, which causes frequent backtracking and mediocre solution quality. On the other hand, past computational experience on nested partitions method show that combining with domain knowledge usually results in better performance. Therefore, we believe the dispatching rules learned by DMDS can play a valuable role in HNP-MP algorithm for solving LPDPs.

In Sect. 3, we have seen a decision tree derived by learning the historical data. Using a decision tree similar to that in Fig. 3, we are able to obtain a probability of load l being assigned to driver k, denoted by p_{kl}. Hereby, we can design *dispatching rule-based sampling* (Algorithm 2) and *dispatching rule-based partitioning* (Algorithm 3).

Algorithm 2 Dispatching rule-based sampling

for each load $l \in L \cup S$ **do**
  for each driver $k \in K$ **do**
    Evaluate the value of p_{kl} using the decision tree learned in DMDS
  end for
  Calculate sampling weights: $w_{kl} = Prob(\text{assign load } l \text{ to driver } k) = p_{kl} / \sum_{k' \in K} p_{k'l}$
  Draw a driver $\tilde{k}$ using weighted sampling, and assign load l to driver $\tilde{k}$
end for
Output the partial sample

Algorithm 3 Dispatching rule-based partitioning

$i \leftarrow 0$
while $i < N$ **do**
  Select the pair $(\tilde{k}, \tilde{l})$ with highest value of p_{kl} (ties can be broken arbitrarily)
  if $(\tilde{k}, \tilde{l}, j) \notin Q,\ \forall j \in L$ **and** $\exists \tilde{j} \in L$ such that $x^*_{\tilde{k}\tilde{l}\tilde{j}} = 1$ **then**
    $Q \leftarrow Q \cup \{(\tilde{k}, \tilde{l}, \tilde{j})\}$
    $i \leftarrow i + 1$
  end if
  Delete $p_{\tilde{k}\tilde{l}}$ from the list of p_{kl}
end while
Fix $x_{klj} = 1, \forall (k, l, j) \in Q$ for the next most promising region

These two algorithms fully utilize the domain knowledge derived by DMDS, and have been proven to be very useful in expediting the search for a good sample and a good partition in real world instances. By replacing LP solution-based sampling/partitioning with dispatching rule-based sampling/partitioning, we do not need to solve the LP relaxation which also saves computational time. We call the new algorithm DMDS-driven HNP-MP (DMDS-HNP-MP) approach. To test the effectiveness of proposed approaches, we use the same data source as that in Sect. 5.1. The computational time is set to 15 minutes, and each MIP is solved by CPLEX 10 with 4 parallel threads. The tests are run in a computer with Intel Xeon 3.16 GHz CPU and 16 G RAM. Notice that the DMDS-HNP-MP algorithm uses the tree learned in the data mining process. The time spent on data mining is difficult to count due to human interactions in data preparation and processing step. Furthermore, a well-trained tree can be used repeatedly in DMDS-HNP-MP algorithm without further training. Hence, we do not account for the time spent on learning the tree in this comparison.

The average performance is compared in Table 9. The table shows that both optimization algorithms obtain better results than the real dispatching results. A lower empty mileage and a higher loaded mileage have been achieved by optimization. Also, the results show that DMDS-HNP-MP outperforms HNP-MP for all performance indicators. However, the improvements are not as great as we expected. After analyzing the data, we think the reason

Table 9 Performance comparison between HNP-MP and DMDS-HNP-MP

Scenario	*EM*	*LM*	*TM*	*ER* (%)	*LR* (%)
Actual	2803.3	8392.2	11195.4	25.04	74.96
HNP-MP	2509.7	8673.9	11183.6	22.44	77.56
DMDS-HNP-MP	2390.7	8815.2	11205.9	21.33	78.67

Table 10 Performance of feeding DMDS with optimized solutions

EM	*LM*	*TM*	*ER* (%)	*LR* (%)
2645.0	8519.5	11164.5	23.69	76.31

is that we try to optimize the real scenarios which have been tailored by the dispatching rules in reality. That is, when the business accepts loads to serve, they have already considered the capacity of current dispatch system. Therefore, some loads have been declined and are not in the historical data. If we take these potential loads into consideration, we expect to achieve higher loaded mileage by running optimization algorithms.

In the above, we show how the learned dispatch rules are used to improve the performance of optimization algorithm and achieve better dispatching results. On the other hand, the optimized solutions can be fed into the DMDS system, so that an improved dispatch rule can be learned and used to generate better-than-actuals solutions. To demonstrate this conclusion, we run the optimization algorithm for each day of the training data, and use the optimized solutions as the new training set. The re-trained regression tree is used to generate new solutions for the same days in Table 9. The performance indicators are shown in Table 10. Comparing the results to those in Table 9, it is clear that we are able to achieve better performance by feeding DMDS with optimized solutions. The empty travel mileage has been reduced and the empty ratio has been reduced about 1.5 %. Therefore, DMDS provides a quick approach to the LPDPs by feeding the system with good or optimized solutions. From Tables 9 and 10, we also notice that even though the new training data are the optimized solutions, the solutions generated by DMDS are worse than those using optimization algorithms (23.69 % *ER* by DMDS vs. about 22 % *ER* by optimization algorithms). We believe this is because the solutions provided by optimization algorithms cannot be fully captured by a regression tree, while the tree structure is more natural to represent the decision processes of human dispatchers.

6 Conclusions

In this paper, a new data mining-based dispatching system (DMDS) for solving the local pickup and delivery problem is presented. The DMDS learns dispatching rules from expert dispatchers using historical data, and generates solutions to LPDPs in a rolling horizon manner. A real application in the intermodal freight industry has shown that the system can mimic the dispatchers' activities quite well. That means the dispatching rules that we derived by data mining can well represent the experienced dispatchers' policy and knowledge. The solution generation procedure is highly efficient, which can be beneficial for large-scale dynamic problems. It is valuable to assist the decision makings in trucking industry without making real changes to the system.

Based on the knowledge of field experts, the DMDS can provide a set of daily dispatch solutions as good as historical solutions given by field experts. But these schedules are not

optimal solutions most of the time. We perform some preliminary studies to combine the DMDS with an optimization algorithm for LPDPs. The learned rules from DMDS are useful inputs to an optimization algorithm to improve the computational speed and solution quality for LPDPs. More studies can be done by investigating other optimization techniques as reviewed in Sect. 1. In turn, the solutions of optimization algorithms can be used as inputs to the DMDS. These solutions contain more efficient dispatching rules than historical data do, and therefore the resulting DMDS is able to generate better-than-actuals solutions.

References

Bräsy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science*, *39*(1), 104–118.

Bräsy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science*, *39*(1), 119–139.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Campbell, A. M., & Savelsbergh, M. (2004). Efficient insertion heuristics for vehicle routing and scheduling problems. *Transportation Science*, *38*(3), 369–378.

Chan, K. Y., & Loh, W. Y. (2004). Lotus: an algorithm for building accurate and comprehensible logistic regression trees. *Journal of Computational and Graphical Statistics*, *13*(4), 826–852.

Desrosiers, J., Soumis, F., Desrochers, M., & SauvéGerad, M. (1986). Methods for routing with time windows. *European Journal of Operational Research*, *23*(2), 236–245.

Dumas, Y., Desrosiers, J., & Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, *54*(1), 7–22.

Ekins, S., Shimada, J., & Chang, C. (2006). Application of data mining approaches to drug delivery. *Advanced Drug Delivery Reviews*, *58*, 1409–1430.

Fagerholt, K., & Christiansen, M. (2000). A travelling salesman problem with allocation, time window and precedence constraints—an application to ship scheduling. *International Transactions in Operational Research*, *7*(3), 231–244.

Funke, B., Grunert, T., & Irnich, S. (2005). Local search for vehicle routing and scheduling problems: review and conceptual integration. *Journal of Heuristics*, *11*(4), 267–306.

Holsapple, C. W., Lee, A., & Otto, J. (1997). A machine learning method for multi-expert decision support. *Annals of Operations Research*, *75*, 171–188.

Li, X., & Ólafsson, S. (2005). Discovering dispatching rules using data mining. *Journal of Scheduling*, *8*, 515–527.

Lim, A., & Wang, F. (2005). Multi-depot vehicle routing problem: a one-stage approach. *IEEE Transactions on Automation Science and Engineering*, *2*(4), 397–402.

Lim, A., Wang, F., & Xu, Z. (2006). A transportation problem with minimum quantity commitment. *Transportation Science*, *40*(1), 117–129.

Liu, H., & Motoda, H. (1998). *Feature extraction, construction and selection: a data mining perspective*. Norwell: Kluwer Academic.

Long, W. J., Griffith, J. L., & Selker, H. P. (1993). A comparison of logistic regression to decision-tree induction in a medical domain. *Computers and Biomedical Research*, *26*, 74–97.

Mourkousis, G., Protonotarios, M., & Varvarigou, T. (2003). Application of genetic algorithms to a large-scale multiple-constraint vehicle routing problem. *International Journal of Computational Intelligence and Applications*, *3*(1), 1–21.

Nanry, W. P., & Barnes, J. W. (2000). Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research. Part B*, *34*(2), 107–121.

Pi, L., Pan, Y., & Shi, L. (2008). Hybrid nested partitions and mathematical programming approach and its applications. *IEEE Transactions on Automation Science and Engineering*, *5*(4), 573–586.

Piramuthu, S., Raman, N., & Shaw, M. J. (1998). Decision support system for scheduling a flexible flow system: incorporation of feature construction. *Annals of Operations Research*, *78*, 219–234.

Powell, W. B., & Carvalho, T. (1998). Dynamic control of logistics queueing networks for large scale fleet management. *Transportation Science*, *32*(2), 90–109.

Powell, W. B., Shapiro, J., & Simao, H. P. (2002). An adaptive, dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science*, *36*(2), 231–249.

Quinlan, J. R. (1986). Induction of decision tree. *Machine Learning*, *1*(1), 81–106.

Shaw, M. J., Park, S. C., & Raman, N. (1992). Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transactions*, *24*(2), 156–168.

Shi, L., & Ólafsson, S. (2000). Nested partitions method for global optimization. *Operations Research*, *48*(3), 390–407.

Wang, X., & Regan, A. C. (2002). Local truckload pickup and delivery with hard time window constraints. *Transportation Research. Part B*, *36*, 78–94.

Witten, I. H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques* (second ed.). San Mateo: Morgan Kaufmann.

Xu, H., Chen, Z. L., Rajagopal, S., & Arunapuram, S. (2001). Solving a practical pickup and delivery problem. *Transportation Science*, *37*(3), 347–364.