ABSTRACT
Optimizing delivery routes for last-mile logistics service is challenging and has attracted the attention of many researchers. These problems are usually modeled and solved as variants of vehicle routing problems (VRPs) with challenging real-world constraints (e.g., time windows, precedence). However, despite many decades of solid research on solving these VRP instances, we still see significant gaps between optimized routes and the routes that are actually preferred by the practitioners. Most of these gaps are due to the difference between what’s being optimized, and what the practitioners actually care about, which is hard to be defined exactly in many instances. In this paper, we propose a novel hierarchical route optimizer with learnable parameters that combines the strength of both the optimization and machine learning approaches. Our hierarchical router first solves a zone-level Traveling Salesman Problem with learnable weights on various zone-level features; with the zone visit sequence fixed, we then solve the stop-level vehicle routing problem as a Shortest Hamiltonian Path problem. The Bayesian optimization approach is then introduced to allow us to adjust the weights to be assigned to different zone features used in solving the zone-level Traveling Salesman Problem. By using a real-world delivery dataset provided by the Amazon Last Mile Routing Research Challenge, we demonstrate the importance of having both the optimization and the machine learning components. We also demonstrate how we can use route-related features to identify instances that we might have difficulty with. This paves ways to further research on how we can tackle these difficult instances.

KEYWORDS
learning from demonstrations; autonomous planning; last-mile logistics

ACM Reference Format:

1 INTRODUCTION
Optimizing delivery routes for last-mile logistics service is challenging and has attracted the attention of many researchers. Most existing approaches from the literature (e.g., see [7]) aim to solve a wide variety of vehicle routing problems (VRPs), usually with many complicating real-world constraints (e.g., time windows, precedence). However, despite many decades of solid research on how to solve these VRP instances as efficiently and effectively as possible, we still see significant gaps between optimized routes and the routes that are actually preferred by the practitioners.

Most of these gaps are due to the difference between what’s being optimized, and what the practitioners actually care about. From the literature, typical objectives being optimized could include travel time, distance, or cost. However, experienced delivery drivers have first-hand knowledge of the area and the customers that they are serving, and thus could plan their routes based on a wide variety of additional factors that are difficult to formalize and quantify. As a result, drivers often deviate from the planned routes. While drivers could alter the actual routes to satisfy their own constraints or incorporate their personal knowledge, they could potentially sacrifice overall system metrics, such as fuel consumption, delivery time, and the order packing operations (onto the delivery vehicle) that are closely related to the routing sequence. Therefore, it will be much more desirable if we could incorporate drivers’ tacit knowledge about the delivery area into our route planning algorithm.

In this paper, we propose a novel hierarchical router with learnable parameters that combines the strength of both the optimization and machine learning approaches. Our hierarchical router first solves a zone-level Traveling Salesman Problem (TSP) with learnable weights on various zone-level features; with the zone visit sequence fixed, we then solve the stop-level vehicle routing problem as a Shortest Hamiltonian Path problem. The Bayesian optimization approach is then introduced to allow us to adjust the weights to be assigned to different zone features used in solving the zone-level TSP. By using a real-world delivery dataset provided by the Amazon Last Mile Routing Research Challenge [21], we demonstrate the importance of having both the optimization and the machine learning components. A critical difference between the problem we are solving and most past work is how we evaluate the solution quality, which is based on how close our generated route sequences are to the highly-rated route sequences from the historical dataset. Our key contributions are summarized as follows:

• We proposed a novel hierarchical route optimizer with learnable parameters that combines the strength of both the optimization and machine learning approaches.
• We demonstrate the effectiveness of our approach using a real-world delivery dataset provided by the Amazon Last Mile Routing Research Challenge.
• Finally, we also demonstrate how we can use route-related features to identify instances that we might have difficulty with.
2 RELATED WORK

2.1 Last-mile Delivery Problem

The challenging target service levels, the small dimension of parcels, and the high level of dispersal of destinations make the last-mile delivery problem become a tricky part of the delivery process [20]. Therefore, how to increase the efficiency of last-mile delivery has received growing attention in recent years. Some research focuses on how to optimize the traditional delivery mode and propose different versions of the vehicle routing problem that calculate the optimal route to deliver a set of demands to dispersed destinations [1]. Some methods focus on innovative solutions to increase the efficiency of last-mile delivery, such methods include parcel lockers [18], crowdsourcing logistics [9, 14], and drone-based solutions [23].

2.2 Traveling Salesman Problem

The problem we study in this paper is similar to the classical TSP, which aims to find the shortest tour that visits all cities in a given set exactly once and returns to the origin. The TSP has been extensively studied [13] and has been proved as an NP-hard problem [15]. Classic approaches to solving the TSP can be classified into exact methods and heuristic methods. The former has been studied using integer linear programming (ILP) [2] that are guaranteed to find an optimal solution, but ILP problems are computationally expensive to be used in practice. To address this computational challenge, many heuristic methods are proposed, which include the Savings algorithm [10], Tabu Search [12], Greedy Randomized Adaptive Search Procedure [11], Simulated Annealing [8, 19], and Genetic Algorithms [16].

More recently, deep learning approaches have been proposed as part of the heuristic for solving hard combinatorial optimization problems such as the TSP [4]. For example, Pointer Networks (PtrNet) [27] learns a sequence model coupled with an attention mechanism trained to output TSP tours using solutions generated by Concorde [2]. The PtrNet is further extended to learn without supervision using Policy Gradient, trained to output a distribution over node permutations [3].

As we aim to generate route sequences that are similar to highly rated routes in practice, our objective is different from that of the classical TSP instances and thus past approaches are not directly applicable.

2.3 Bayesian Optimization

Bayesian optimization is widely adopted for hyper-parameter tuning in optimizing objective functions. It is applied widely in areas such as combinatorial optimization [17, 28], automatic machine learning [5, 24–26] and reinforcement learning [6]. Bayesian optimization assumes the unknown function is sampled from a Gaussian process and maintains a posterior distribution for this function as observations are based [24]. Then an acquisition function is used to decide where to sample for the next step. The acquisition function measures the value that would be generated by the evaluation of the objective function at a new point, based on the current posterior distribution. The acquisition function can be expected improvement, entropy search, and knowledge gradient. In our work, we use expected improvement as the acquisition function.

3 PROBLEM DESCRIPTION

Our problem formulation is inspired by the dataset provided by the Amazon Last Mile Routing Research Challenge [21]. The objective is to learn from experienced delivery drivers so that the last-mile route optimizer is capable of generating optimized routes that drivers would also rate highly. To concretely and quantitatively define what constitutes a good route (from the driver’s perspective), Amazon provides a labeled dataset where around 6,000 actual delivery routes are rated by drivers to be of high-, medium-, and low-quality (more details on the dataset will be presented shortly). The evaluation of generated routes depends on how closely a generated route resembles a highly rated benchmark route from history.

3.1 Dataset Description

The Amazon dataset contains historical delivery routes from five metropolitan areas in the USA: Austin, Boston, Chicago, Los Angeles, and Seattle. In the dataset, there are 2718, 3292, and 102 routes that are of high-, medium-, and low-quality respectively. The dataset contains three major components: 1) the routes, 2) the stops, and 3) the packages. We elaborate on important information in all three components below.

3.1.1 The Route Information. For each route, we have: a) a unique route ID, b) the departure time, c) the volume capacity of the vehicle, d) the actual sequence of stops visited, and e) the rating label, which can be high, medium, or low.

3.1.2 The Stop Information. For each stop, we have: a) a unique stop ID, b) the geographical coordinate of the stop, and c) the zone ID (a planning unit defined by the dataset) where this stop is located. The travel times between stops are also provided.

3.1.3 The Package Information. For each package, we have: a) a unique package ID, b) the status, which can be rejected, delivered, or delivery-attempted, c) the stop ID this package belongs to, d) the dimensions (maximal width, length, and height), e) delivery time window (if any), and f) the estimated service time.

3.2 Evaluation Metric

The evaluation metric is whether a generated route closely resembles the highly-rated benchmark route. Formally speaking, the score of a route is composed of two components: the Sequence Deviation (SD) and the Edit Distance with Real Penalty (ERP). The SD component measures how different a given sequence is from the benchmark sequence, and takes values from 0 to 1, with 0 indicating that the two sequences are identical. On the other hand, the ERP component measures how many single-element operations (insertions, deletions, and substitutions) are required to transform the given sequence into the benchmark sequence. To reflect the travel distance, the operations are further weighted by the physical distance among affected stops. In other words, the ERP component measures how far apart a given sequence is from the benchmark sequence.

---

1The Amazon dataset is available at https://registry.opendata.aws/amazon-last-mile-challenges/
Let \( A \) be the benchmark sequence, and \( B \) be the sequence we would like to evaluate, the route score of \( B \) is defined as:

\[
\text{route\_score}(A, B) = \frac{SD(A, B)}{\text{ERP}_n(A, B)}. \tag{1}
\]

The SD component is defined as:

\[
SD(A, B) = \frac{2}{n(n-1)} \sum_{i=1}^{n} |a_i - a_{i-1}| - 1,
\]

where \( n \) is the total number of stops in the sequence (\( A \) and \( B \) should have equal number of stops), and \( a_i \) is the position where the \( i \)-th stop in \( B \) appears in \( A \).

The \( \text{ERP}_n(A, B) \) component captures normalized travel time between stops at the same location of both sequences and is recursively defined as:

\[
\text{ERP}_n(tail(A), tail(B)) + \text{time}_n(A[0], B[0]), \tag{3}
\]

where \( \text{time}_n(x, y) \) is the normalized travel time between stop \( x \) and stop \( y \), tail(X) is the sequence \( X \) without its first stop, \( X[0] \) is the first stop in the sequence \( X \). The implicit assumption of the above definition is that \( A \) and \( B \) should include the same set of stops.

The \( \text{ERP}_n(A, B) \) component captures the number of edit operations (insertions, substitutions, or deletions) required to transform sequence \( B \) to sequence \( A \). The ratio \( \text{ERP}_n(A, B) / \text{ERP}_e(A, B) \) can be intuitively interpreted as the average normalized travel time per edit operation.

The performance of the routing engine is simply the average score of all routes:

\[
\text{score} = \frac{1}{|I|} \sum_{i \in I} \text{route\_score}_i, \tag{4}
\]

where \( I \) contains all route instances being evaluated.

4 SOLUTION APPROACH

To explore potential design ideas that would optimize the routing of stops and at the same time incorporate drivers’ tacit knowledge of what constitutes a good route, we sample a wide collection of routes that are highly rated by drivers and compare them against routes generated via a simple TSP heuristic (see Figure 1 for a typical example). What we see from the comparison is strong visual evidence that drivers prefer to schedule their visits in blocks of zones, i.e., visiting all stops within a zone before moving on to the next zone. While the routes generated by the TSP heuristic tend to mix stops from different zones.

Given this observation, we propose to design a hierarchical router, where the zone visitation sequence is decided first, after which stops within respective zones are then routed. The most important information we would harvest from benchmark routes is how drivers would sequence zones. To achieve this, we identify a collection of features associated with zones and define the cost function at the zone level as a linear combination of zone features (with feature weights as parameters). We use the Savings algorithm to find out the zone visitation sequence, and utilize a Bayesian optimization method to iteratively update the weights associated with all zone features using historical benchmark sequences.

4.1 A Hierarchical Router with Learnable Parameter

We call our approach the Hierarchical Router with Learnable Parameters (HR-LP), since it generates a route at the zone level first, before progressing to route the stops within each zone. To formulate the zone sequencing problem, we define \( c_{ij} \) to represent the inclination a driver would have in choosing zone \( j \) when in zone \( i \) (the lower the \( c_{ij} \), the higher the inclination). We define \( c_{ij} \) as:

\[
\begin{align*}
    c_{ij} & = \sum_{k=1}^{K} \theta_k \phi_k(i, j), \tag{5}
\end{align*}
\]

where \( \phi_k(i, j) \) and \( \theta_k \) represent the value of feature \( k \) and the weight associated with feature \( k \) respectively. The feature weights will be learned via a Bayesian optimization approach, and we will defer its introduction to the later section.

We have in total 14 zone features, which can be either native (purely based on the characteristics of the zone) or derived. Among these 14 feature candidates, we select 5 features that are most significant, which are introduced below:

- \( \phi_1(i, j) \): average travel time from any stop in zone \( i \) to any stop in zone \( j \).
- \( \phi_2(i, j) \): Euclidean travel distance from the centroid of zone \( i \) to the centroid of zone \( j \). The x-coordinate of the zone centroid is calculated as: \( x_c = \frac{\sum_{i \in I} x_i}{n} \), in which \( n \) is the number of stops in the zone, and \( x_i \) is the x-coordinate of stop \( i \). The y-coordinate of the zone centroid is calculated similarly.
- \( \phi_3(i, j) \): the ratio of the average travel time from the depot to all stops in zone \( j \) over the average travel time from the depot to all stops in zone \( i \).
- \( \phi_4(i, j) \): the ratio of the average travel time from all stops in zone \( j \) to depot over the average travel time from all stops in zone \( i \) to depot.
- \( \phi_5(i, j) \): 1 if zones \( i \) and \( j \) are in the same main zones, and 0 otherwise. (The zone ID is denoted as X-N-MY, and two zones are considered to be in the same main zone if their X and N are identical.)

With \( c_{ij} \) defined, we can formulate the zone sequencing problem following the standard TSP formulation:

\[
\begin{align*}
    \min & \sum_{i \in N} \sum_{j \in N} c_{ij}z_{ij}, \tag{6}
    \\
    \text{s.t.} & \sum_{j \in N} z_{ij} = 0, \forall i \in N, \tag{7}
    \\
    & z_{ij} \in \{0, 1\}, \forall i \in N, \tag{8}
    \\
    & \sum_{i \in N} z_{ij} \leq |R| - 1, \forall R \subseteq N, \tag{9}
    \\
    & z_{ij} \in \{0, 1\}, \forall (i, j) \in A, \tag{10}
\end{align*}
\]

where the decision variable \( z_{ij} = 1 \) represents that the driver traverses from zones \( i \) to \( j \). (7) ensures that each zone will be visited exactly once, (8) eliminates self loop, and (9) prevents subtours.

We use the Savings algorithm [10] to solve the above zone-level TSP and the open-source OR-tools solver [22] is used for the implementation. The optimal zone visit sequence is denoted as \( Z^* = (0, z_1, z_2, ..., z_n) \), where 0 represents the depot. With the zone
visit sequence, we then solve for the stop visit sequence within each zone and decide which stops should be used as the starting and ending stops (to connect to the next and from the previous zones).

Since the starting and the ending stops are different when routing stops within a zone, the problem is formulated as a Shortest Hamiltonian Path (SHP) problem below:

$$\min \sum_{i \in M} \sum_{j \in M} t_{ij} x_{ij},$$  \hspace{1cm} (11)

s.t. $$\sum_{j \in M} x_{ij} = \sum_{j \in M} x_{ji} = 1, \forall i \in M \setminus \{m_0, m_d\},$$  \hspace{1cm} (12)

$$\sum_{i \in M} x_{im_d} = \sum_{j \in M} x_{m_i,j} = 1,$$  \hspace{1cm} (13)

$$\sum_{i \in M} x_{im_d} = \sum_{j \in M} x_{m_d,j} = 0,$$  \hspace{1cm} (14)

$$\sum_{i \in R, j \in R} x_{ij} \leq |R| - 1, \forall R \subseteq M,$$  \hspace{1cm} (15)

$$x_{ii} = 0, \forall i \in M, \ x_{ij} = \{0, 1\}, \forall (i, j) \in A.$$  \hspace{1cm} (16)

The stop-level routing problem for each zone is similar to the zone-level TSP formulation. For each zone, given the directed graph $G = (M, A)$, where $M$ denotes the location of $m$ stops in the zone. The decision variable $x_{ij} = 1$ represents that the driver traverses from stops $i$ to $j$, with $t_{ij}$ representing the travel time. The objective function (11) minimizes the total travel time within the zone. We denote $m_o$ and $m_d$ as the first stop when entering the zone and the last stop when leaving the zone; this is enforced by (14).

We further elaborate on the process of generating the complete stop sequence below:

- **Step 1:** For a route, obtain the optimal zone visit sequence $Z^* = (z_1, ..., z_n)$ by solving the zone-level TSP.

- **Step 2:** Initialize the optimal stop sequence $S^*$ with the depot '0' as the first element.

- **Step 3:** For each stop in zone $z_{i-1}$, calculate the average travel time to all stops $x \in z_i$. Sort stops in $z_{i-1}$ in the ascending order following the computed average travel time. Insert the top-$h$ stops into the candidate starting set $A$.

- **Step 4:** For each stop in zone $z_{i+1}$, calculate the average travel time from all stops $x \in z_i$. Sort stops in $z_{i+1}$ in the ascending order following the computed average travel time. Insert the top-$h$ stops into the candidate ending set $B$. If zone $z_i$ is the last visited zone, insert depot '0' into $B$.

- **Step 5:** For each pair of possible starting-ending stops from $A \times B$, solve the corresponding stop-level SHP. Except for the first and the last zones, we should have $h^2$ stop-level SHP instances for each zone (each instance comes with different starting and ending stops).

- **Step 6:** Choose the stop sequence with minimum total travel time as the optimal sequence $S^*_i$ for zone $i$.

- **Step 7:** For each zone $z_i$ in $Z^*$, repeat Steps 3-6 to obtain the optimal stop sequence $S^*_i$ and add it into the optimal stop sequence $S^*$.

As before, we utilize the Savings algorithm [10] to solve all stop-level SHP instances and OR-tools solver [22] is used for the implementation. $h$ is a hyper-parameter which will be discussed in the following experiments.

### 4.2 Learning Feature Weights

As mentioned before, the Bayesian optimization method is used to find the optimal weight parameters $\Theta$ of zone features by iteratively updating the weight parameters and generating route sequences. For the given high-quality routes, we first generate the proposed sequence for each route by executing our HR-LP solver as described before and then calculate the 'score' of the proposed route sequence using (1). We then apply the Bayesian Optimization procedures to search for a new set of parameter values in order to minimize the
average score of all given routes. These two steps are implemented iteratively to find the optimal weight parameters.

4.2.1 Bayesian Optimization Algorithm. Given a set of known high-quality routes and initial zone feature weights $\theta$, we generate proposed route sequences $R^* = \{r_1^*, ..., r_N^*\}$ by executing the HR-LP solver. The average score can be calculated according to (2) – (4). We then utilize the Bayesian optimization approach (see Algorithm 1) to update the feature weights iteratively to minimize the average score. In other words, we are minimizing the following objective function:

$$l(\theta) = \frac{1}{M} \sum_{m \in M} \text{route} \_ \text{score}(r_m, f(D_m, \theta)).$$

(17)

where $M$ denotes the number of routes, $r_m$ is the given benchmark sequence of route $m$, and $f(D_m, \theta)$ denotes the generated sequence of route $m$ given route data $D_m$ and weight $\theta = \{\theta_1, ..., \theta_k\}$. Bayesian optimization is thus defined as

$$\theta^* = \arg \min_{\theta \in \Theta} l(\theta),$$

(18)

where $\Theta$ is all $\theta_k$ such that $1 \leq \theta_k \leq 10$. The actual steps we adopt to solve (18) are described in Algorithm 1, with expected improvement as our acquisition function.

5 EXPERIMENTS

To evaluate our proposed approach, we use the route data from the dataset. We only incorporate 2,718 high-quality routes. There are 17 depots from the dataset, and we train a different set of weight parameters for each of these 17 depots. For each depot, we split the route data into 70% training set and 30% testing set. In the training phase, we execute the Bayesian optimization process for each depot separately to obtain the optimal weight parameters $\theta$. In the testing phase, we use the obtained $\theta^*$ on the unseen testing dataset to generate proposed route sequences by executing our HR-LP solver and obtain the average score. The implementation can be found at https://github.com/SHAOQIAN12/HR-LP.git.

To illustrate the importance of having both the optimization and learning components, We compare our proposed approach against the following two baselines:

Standard TSP: We solve the standard TSP using the Savings algorithm directly. For each route, the driver starts from the depot, visits all stops, and returns to the depot. The objective is to minimize the total travel time. The resulting routing sequence is evaluated using the same metric.

Stop-level Bayesian Optimization: In order to illustrate the effectiveness of the hierarchical routing formulation, we use the Bayesian Optimization method to update the weight parameters of stop features, however, we generate the route using only the standard TSP formulation at the stop-level. The TSP formulation is similar to (6)–(10). And the stop features we used are designed as follows:

- $\varphi_1(i, j)$: travel time from stop $i$ to stop $j$.
- $\varphi_2(i, j)$: ratio of the travel time from depot to stop $j$ to the travel time from depot to stop $i$.
- $\varphi_3(i, j)$: ratio of the travel time from stop $j$ to depot to the travel time from stop $i$ to depot.
- $\varphi_4(i, j)$: ratio of the number of packages in stop $j$ to the number of packages in stop $i$.

Figure 2: Average score from the Bayesian optimization (for Chicago).

In our experiments, the number of initial points is set to 20, and the iteration number is set to 100. We normalize each feature value into $[0, 1]$. Figure 2 shows the average score $l(\theta)$ of training routes departing from depots in Chicago (the results from other cities look very similar). From Figure 2 we can see that incorporating our observation that drivers reason at the zone level is crucial in getting good results.

Figure 3 shows the score distribution of the 808 routes testing dataset, and we can see that our HR-LP approach has the majority of scores lower than 0.06 compared to baselines. Table 1 shows the average score of the testing data using three competing methods.
Table 1: Average scores comparison. Lower score means better performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Testing data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard TSP</td>
<td>0.0898</td>
</tr>
<tr>
<td>Stop-level Bayesian Optimization</td>
<td>0.0992</td>
</tr>
<tr>
<td>HR-LP</td>
<td>0.0496</td>
</tr>
</tbody>
</table>

We can see again that our proposed HR-LP approach outperforms the two baselines by nearly 50%. Figure 4 shows the box plot of score distribution in all 5 cities. We notice that the proposed method performs well in Seattle and Los Angeles (with scores lower than 0.05). However, Austin has the highest average score, probably due to insufficient route instances for us to learn the optimal weight parameters.

We also look at the impact of hyperparameter $h$ (number of candidate starting/ending stops to enumerate) and the computational time. As shown in Table 2, increasing the value of $h$ does not seem to impact the outcomes much, which implies that enumerating more starting and ending points do not seem to help us in getting better results.

Table 2: Mean score and computational time for all $h$.

<table>
<thead>
<tr>
<th>$h$</th>
<th>Testing data</th>
<th>Training time</th>
<th>Testing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0496</td>
<td>3h12m</td>
<td>9m</td>
</tr>
<tr>
<td>3</td>
<td>0.0491</td>
<td>3h47m</td>
<td>9m</td>
</tr>
<tr>
<td>4</td>
<td>0.0492</td>
<td>4h43m</td>
<td>9m</td>
</tr>
<tr>
<td>5</td>
<td>0.0489</td>
<td>5h47m</td>
<td>10m</td>
</tr>
</tbody>
</table>

6 DISCUSSION

In this section, we discuss whether the characteristics of route instances would impact the performance of our HR-LP approach. This could help us further improve our proposed approach. We first look at the score distribution of our approach on all 2,718 high-quality routes in Figure 5. We can see that while most routes have decent route scores that are below 0.06, there is a long tail in the score distribution.

6.1 Feature Selection

We have created 42 features, all of which are normalized into $[0, 1]$. The features can be classified into the following categories:

**Route features:** There are 5 features defined at the route level. The capacity of a vehicle, the number of zones, stops and packages of a route, and the total travel time of the actual sequence.
Zone features: There are 18 features defined at the zone level. The stop number distribution (mean, standard deviation, maximum and minimum) and package number distribution among zones, the average travel time distribution between two zones (the average travel time from zone 1 to zone 2 is defined as the average travel time from any stops in zone 1 to any stops in zone 2), the average travel time distribution from depot to all zones, and the average travel time from depot to the first visiting zone and the last visiting zone.

Stop features: There are 19 features defined at the stop level. The package volume distribution and package service time distribution of stops, the travel time distribution from depot to all stops, the mean, standard deviation, and maximum package number of stops, the mean, standard deviation, and minimum travel time between two stops, and the travel time from depot to the first visiting stop.

6.2 Multiple Linear Regression
We define the logarithm of route score as the dependent variable, we have included 2,718 high-quality route instances in our regression. We choose the independent variables that are statistically significant (p-value less than 0.01) to be the final set of candidate features. These chosen features are: the number of stops (denoted as stop_number), the total travel time of the actual sequence (denoted as actual_seq_cost), the average travel time from depot to the first visiting zone (denoted as depot_first_zone), the average travel time from depot to the last visiting zone (denoted as depot_last_zone), the mean package volume of stops (denoted as mean_pac_volume), the standard deviation package volume of stops (denoted as std_pac_volume), the standard deviation travel time from depot to all stops (denoted as std_dep), and the standard deviation travel time between two stops (denoted as std_tra_stops).

We perform the multiple linear regression again using these 8 features as independent variables, and the result is summarized in Table 3. The features stop_number, depot_first_zone, std_pac_volume and std_tra_stops have positive effects on the score, while the remaining features have negative effects.

Table 3: Multiple linear regression results (standard errors are in parentheses).

<table>
<thead>
<tr>
<th>Feature</th>
<th>log_score</th>
<th>Feature</th>
<th>log_score</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop_number</td>
<td>-1.8365**</td>
<td>mean_pac_volume</td>
<td>2.3787**</td>
</tr>
<tr>
<td></td>
<td>(0.129)</td>
<td></td>
<td>(0.528)</td>
</tr>
<tr>
<td>actual_seq_cost</td>
<td>3.2024**</td>
<td>std_pac_volume</td>
<td>-1.6656**</td>
</tr>
<tr>
<td></td>
<td>(0.154)</td>
<td></td>
<td>(0.539)</td>
</tr>
<tr>
<td>depot_first_zone</td>
<td>-1.9557**</td>
<td>std_dep</td>
<td>0.5127*</td>
</tr>
<tr>
<td></td>
<td>(0.118)</td>
<td></td>
<td>(0.208)</td>
</tr>
<tr>
<td>depot_last_zone</td>
<td>1.6719**</td>
<td>std_tra_stops</td>
<td>-2.5941**</td>
</tr>
<tr>
<td></td>
<td>(0.125)</td>
<td></td>
<td>(0.207)</td>
</tr>
<tr>
<td>No. of observations</td>
<td>2718</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.3 Separating Instances using the Support Vector Machine Approach
Finally, we use the Support Vector Machine (SVM) approach to see whether these 8 features from the route instances can help us distinguish the route instances with good and bad performances. To focus on extreme cases, we define the best-performing instances for routes whose scores are less than 0.01 (240 routes), and the worst-performing instances for routes whose scores are more than 0.1 (199 routes). They are labeled as low-score and high-score classes respectively. We again divide them into 80% training set and 20% testing set to evaluate the feasibility of using SVM as a separation approach.

Table 4: SVM result.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>lowest scores</td>
<td>0.83</td>
<td>0.98</td>
<td>0.90</td>
</tr>
<tr>
<td>highest scores</td>
<td>0.97</td>
<td>0.79</td>
<td>0.87</td>
</tr>
<tr>
<td>accuracy</td>
<td>/</td>
<td>/</td>
<td>0.89</td>
</tr>
<tr>
<td>macro avg</td>
<td>0.90</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>weighted avg</td>
<td>0.90</td>
<td>0.89</td>
<td>0.89</td>
</tr>
</tbody>
</table>

Table 5: Coefficient of the linear kernel.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Coefficient</th>
<th>Feature</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop_number</td>
<td>-3.27</td>
<td>mean_pac_volume</td>
<td>1.05</td>
</tr>
<tr>
<td>actual_seq_cost</td>
<td>4.80</td>
<td>std_pac_volume</td>
<td>0.37</td>
</tr>
<tr>
<td>depot_first_zone</td>
<td>-3.97</td>
<td>std_dep</td>
<td>1.78</td>
</tr>
<tr>
<td>depot_last_zone</td>
<td>2.46</td>
<td>std_tra_stops</td>
<td>-1.69</td>
</tr>
</tbody>
</table>

To allow us to interpret the learning outcomes we adopt the linear kernel for our SVM process. As we can see from Table 4, the precision and recall are high, which means the two classes of route instances can be well separated by using these 8 features. The precision of the class ‘high-score’ is 0.97, implying that the poorly-performed route instances can be easily identified. The coefficient of the linear kernel is summarized in Table 5.

6.4 Observations
To see how the mean values of these 8 features are different in the low-score and the high-score classes, we conduct the mean difference test and see that almost all feature means are statistically different, with p-values < 0.01 (the only exception is depot_last_zone, where the p-value is 0.48, results summarized in Table 6). The distributions of feature values are also visualized using box plots in Figure 6. Based on these analyses, we conclude that low-score instances typically have fewer stops, larger actual total travel time, shorter distance from the depot to the first zone, larger package sizes on average, and greater variations in package sizes, the travel time between stops, and from depot to stops.

In summary, we investigate whether route-level features would have an impact on the performance of our HR-LP approach, and we see from our analyses that the actual total travel time of a given route sequence, the average travel time from the depot to the last visiting zone, the mean package volume of stops, and the standard
deviation of travel time from depot to stops have a positive correlation with route scores while the number of stops, the standard deviation of package volumes of stops, the average time from depot to the first visiting zone, and standard deviation of travel time between stops all have a negative correlation with scores. Therefore, given the values of these route features, we can predict whether our approach would perform well.

### 7 CONCLUSION

In this paper, we propose a novel hierarchical route optimizer with learnable parameters that combines the strength of both the optimization and machine learning approaches. Through numerical evaluations using real-world data, we demonstrate that it is crucial to have optimization and machine learning working together. Besides solving real-world instances well, we also demonstrate how we can use route-related features to identify instances we might have difficulty with. This paves the way to further research on how we can tackle these difficult instances.

### ACKNOWLEDGMENTS

This research is supported in part by the Ministry of Education, Singapore, under its Social Science Research Thematic Grant (Grant Number MOE2020-SSRTG-018).

### REFERENCES


