Learn Global and Optimize Local: A Data-Driven Methodology for Last-Mile Routing

Mayukh Ghosh\textsuperscript{1}, Alex Kuiper\textsuperscript{1}, Roshan Mahes\textsuperscript{1,2}, Donato Maragno\textsuperscript{1}

\textsuperscript{1} Amsterdam Business School, University of Amsterdam, 1018 TV Amsterdam, the Netherlands
\textsuperscript{2} Korteweg-de Vries Institute for Mathematics, University of Amsterdam, 1098 XG Amsterdam, the Netherlands

m.ghosh@uva.nl; a.v.mahes@uva.nl; d.maragno@uva.nl; a.kuiper@uva.nl

In last-mile routing, the task of finding a route is often framed as a Traveling Salesman Problem to minimize travel time and associated cost. However, solutions stemming from this approach do not match the realized paths as drivers deviate due to navigational considerations and preferences. To prescribe routes that incorporate this tacit knowledge, a data-driven model is proposed that aligns well with the hierarchical structure of delivery data wherein each stop belongs to a zone—a geographical area. First, on the global level, a zone sequence is established as a result of a minimization over a cost matrix which is a weighted combination of historical information and distances (travel times) between zones. Subsequently, within zones, sequences of stops are determined, such that, integrated with the predetermined zone sequence, a full solution is obtained.

The methodology is particularly promising as it propels itself within the top-tier of submissions to the Last-Mile Routing Research Challenge, while it maintains an elegant decomposition that ensures a feasible implementation into practice. The concurrence between prescribed and realized routes underpins the adequateness of a hierarchical breakdown of the problem, and the fact that drivers make a series of locally optimal decisions when navigating. Furthermore, experimenting with the balance between historical information and distance exposes that historic information is pivotal in deciding a starting zone of a route. The experiments also reveal that at the end of a route, historical information can best be discarded, making the time it takes to return to the station the primary concern.

Key words: last-mile logistics, route prediction, navigation, traveling salesman problem, data-driven optimization
1. Introduction

Last-mile routing is often the shortest, yet most complex and expensive leg in distribution systems. It accounts for about 28% of the total costs of transportation from the distribution center to the final customer (Goodman 2005, Gevaers, Van de Voorde, and Vanslander 2011). Last-mile operations also have a large impact on the livability, because of traffic congestion and pollution it creates (Taniguchi and Thompson 2002, Chopra 2003). This pressure is set to increase, because the trend of urbanization continues; around 68% of the global population will reside in urban areas by 2050, according to a report by the United Nations (2018). Currently e-commerce sales are estimated at about $709.78 billion, which is projected to increase as the coronavirus has accelerated a channel shift to e-commerce, see eMarketer (2020). The same trends are echoed by a recent report of the World Economic Forum (Deloiton et al. 2020), which further points out that also the number of same-day deliveries will grow due to customers ever-increasing expectations, amplifying the importance of last-mile operations and its impact on the ecosystem.

Considering parcel delivery in last-mile operations, the main goal is to ensure that products arrive in good shape. In addition, it should be on time (Gunasekaran, Patel, and McGaughey 2004), i.e., the time windows that have been communicated will be met. As routes require a multitude of stops to be visited, it is critical that a proposed route needs to be followed up as deviations can incur redeliveries or dissatisfied customers (by not meeting time slots), unforeseen time losses along the route and additional operational expenses (Boyer, Prud’homme, and Chung 2009, Gevaers, Van de Voorde, and Vanslander 2014).

Despite the benefits of adhering to a proposed ‘optimal’ route are clear, in practice—reported by managers in last-mile delivery (e.g., Amazon)—drivers often deviate from proposed routes. This discrepancy between theoretical route planning and real-life route execution is an important gap in the literature on last-mile routing, and has remained an open research question, because of the typical inaccessibility of real-life data. Recently, Amazon in combination with MIT’s Center for Transportation & Logistics hosted a challenge (Amazon and MIT Center for Transportation &
A Data-Driven Methodology for Last-Mile Routing

In the US, Amazon steadily holds the number one position as the largest e-commerce retailer having a market share of 38.0% (eMarketer 2020), and inherently is responsible for the majority of traffic associated to last-mile delivery. Therefore, by applying the methodology on the data associated to Amazon will set an example for many other organizations facing the bewildering setting of last-mile delivery. Although we consider it as a predictive problem, which helps uncovering on a conceptual level the driver’s decision process when determining a route, we emphasize that the approach can also be used to prescribe routes, which are more feasible from a driver’s perspective.

To consider the problem at a more detailed level, two delivery drivers’ routes are given in Figure 1 as an increasing sequence of numbers; each number represents a stop. Note that not all stops are shown as for readability purposes. Along the route the driver traverses through different zones, given by their zone ids, which are (preset) geographical areas. Comparing the routes in the left...
Leveraging historical routes to learn recurrent patterns is feasible on a zone level rather than at a stop level. This is due to the fact that a zone appears in multiple routes while a stop does not (or rarely). The intrinsic importance of a zone sequence depends on how the zone is established. Ideally, a zone should be constructed such that every stop in it is visited before moving to the next zone (as for example in Figure 1). However, with this presumption, classical heuristics and optimization models fall short to exploit this information. Two classical examples are illustrated in Figure 2. On the left, we find the route obtained by applying the nearest neighbor heuristic, which is sometimes argued to mimic the driver’s behavior, while on the right the route that minimizes distance, as the solution of the corresponding Traveling Salesman Problem (TSP).

On top of the condition that drivers visit all stops in a zone before moving to another, the challenge is to determine the sequence of zones a driver will follow. In Figure 2, there are six possible permutations of zone sequences, but in real-life settings (Figure 1) there are easily 30 zones leading to 30! unique zone sequences. The sequence might be based on information at ground level, navigational considerations, or merely a preference, but most importantly an approach should be able to learn this tacit information from realized routes.

The contributed methodology divides the problem into two parts: on the zone level, it aims to infer the sequence of zones after which, within each zone, the sequence of stops is determined. It
A Data-Driven Methodology for Last-Mile Routing

stems from the idea that first a routing problem is abstractified into zones, for which a sequence is established by solving a TSP formulation that balances historical zone transition information, resembling the tacit knowledge, and distances. So the methodology learns on a global level, which aligns with the intuition that humans congregate route sequences on a higher level than remembering exact stop sequences. Secondly, the problem in each zone is of relatively low complexity and thus a driver is capable to determine an optimal sequence out of the set of stops that minimizes distance or travel time. In the end, the performance of the methodology confirms the competency of the approach.

In the next section, we provide the literature review, which outlines important elements to our methodology. Section 3 describes the methodology, and is split into learning and prediction. In Section 4, we introduce the dataset on which the methodology will be applied and provide the route prediction metrics aligned with the challenge. Using the case study data we carry out various experiments and demonstrate the methodology. Furthermore, we compare the performance to the aforementioned benchmarks as well as to the submissions of the competition. In Section 5, we present our concluding remarks and implications to practice, followed by a discussion and some suggestions for future research.

2. Literature Review

Solving a last-mile routing problem can be translated to a TSP, i.e., a set of stops should be visited—starting from and ending at the station. However, in this research the goal is to predict a driver’s route using historic data. Therefore, in this pursuit, elements of different disciplines are combined. More specifically, we consider classical TSP optimization frameworks, hierarchical structures as an approximation (such as cluster first, routing second), human approaches to solving TSPs, and lastly, learning from route data. In relation to our work, we elaborate further on these themes below.

2.1. Traveling Salesman Problem

The TSP with Euclidean distance is notorious for being hard and is proven to be NP-complete (Papadimitriou 1977), but for finding the optimal or near-optimal tour there is a rich literature (Lawler 1985, Applegate et al. 2011). Also, several variations of the standard TSP have been
introduced in literature, to name a few: an open TSP (OTSP) where the driver does not necessarily return to the depot; TSP with time windows to meet (Gendreau et al. 1998); additionally allowing pick-ups such that the capacity of the truck is not exceeded (Gendreau, Laporte, and Vigo 1999); or the covering Salesman problem (CSP) (Current and Schilling 1989), where a minimal tour is found such that all stops are in the vicinity of a stop on the tour (e.g., dropping goods at neighbors).

Acquainted to the TSP is its generalization known as Vehicle Routing Problem (VRP); see Dantzig and Ramser (1959) for its formulation and Bräysy and Gendreau (2005a), Montoya-Torres et al. (2015) for recent surveys on multiple depots VRPs with variants that include time windows, split delivery, heterogeneous fleet, periodic deliveries, and pick-up. Also, various studies have considered the ecological footprint as presented in the literature review given by Lin et al. (2014). Finally, also for the more general VRP, numerous heuristics and metaheuristics have been proposed over the last decades (Bräysy and Gendreau 2005b, Speranza and Archetti 2014, Purkayastha et al. 2020).

Both TSP and VRP have seen many different integer programming formulations (Orman and Williams 2007). In our model, we adopt the Miller-Tucker-Zemlin (MTZ) formulation to solve the TSP, introduced for the first time by Miller, Tucker, and Zemlin (1960). The MTZ formulation has the advantage of being intuitive and straightforward to implement and works well when the number of stops (variables) to consider is relatively small, fitting our instances. As a downside, it has been proven to lead to a weak relaxation, leading to higher computation times, see e.g., Campuzano, Obreque, and Aguayo (2020) for a discussion. In our case, as we break down the full problem into smaller ones, this is not an issue, but as desired any other TSP formulation can be adopted.

2.2. Hierarchical Structure

To reduce the computational complexity, the problem can be broken down into smaller instances, which are computationally much less involved, and connect these together. For example, in the seminal work by Karp (1977), the proposed partitioning algorithms are asymptotically optimal.
heuristics in the case of uniformly distributed stop locations, i.e., the error tends to zero with probability one as the length of a route (i.e., the number of stops) increases.

Both Liao and Liu (2018) and Jiang et al. (2014) decompose the problem with small-scale nodes by relying on clustering algorithms. Each subproblem is subsequently optimized and the center nodes of the subproblems constitute a TSP in itself. Connecting all local tours in the order of the upper layer problem generates approximative solutions with significantly reduced computation times. Such an approach is also found from an empirical point of view, see Vickers et al. (2003).

Moreover, evidence to support the use of a hierarchical structure in practice is found in Graham, Joshi, and Pizlo (2000). They show that solution methods originating from artificial intelligence or operations research algorithms are insufficiently capable to mimic the human approach of solving TSPs. They introduce a hierarchical model by means of a pyramid algorithm on the visual representation. This algorithm is capable to render human-alike solutions to the various TSP instances where classical algorithms fail in this task, see also Pizlo et al. (2006) for a more refined algorithm. More importantly, the aforementioned works hint at the use of a hierarchical breakdown, because of lower computational complexity and being concurrent with practice.

2.3. Human Navigation

The TSP lends itself to a broad range of experimental studies as its goal of minimization of the route is easy to understand and visualize. Many experimental studies have been devoted to visual versions of the TSP (MacGregor and Ormerod 1996, Van Rooij, Stege, and Schachtman 2003, Vickers et al. 2003). MacGregor and Ormerod (1996) show that humans outperform well-known TSP heuristics and are in small problem sizes capable to be in 1% from optimality.

Humans rely on various tactics to generate near-optimal solutions for the TSP. The potential of these tactics is further demonstrated by the fact that the time needed increases only in a linear fashion compared to the problem size (Pizlo et al. 2006, Dry et al. 2006). One of these tactics is to consider the problem first globally, considering the tour that visits all ‘exterior’ points, to which interior points are inserted; this resembles the so-called convex hull approach (MacGregor
and Ormerod 1996, Vickers et al. 2003). Argued as the underlying motivation for the convex hull approach, is the avoidance of crossings in a tour (Van Rooij, Stege, and Schactman 2003). The motivation behind this tactic is the intuition that a cross in a tour is suboptimal, which is even a fact for metric TSPs. In an OTSP, where the solution is not a tour, the starting point can have a profound impact on the performance, see Sengupta, Mariescu-Istodor, and Fränti (2018). However, they also report that humans are quite capable to select a ‘right’ starting point.

Wiener, Ehbauer, and Mallot (2009) find in a series of experiments that, when navigating, a coarse route is stipulated first. This route visits a set of ‘regions’ after which it is ‘optimized’ on a detailed level along the way. Furthermore, if the problem size increases, the problem is divided into more regions to make the problem manageable and approachable. Such a global-to-local approach echoes the hierarchical decomposition of first determining the zone sequence on a global level, and subsequently a series of local problems to determine the sequence in which stops are to be visited in each zone. Additionally, another experiment from Wiener and Mallot (2003) reveals that a segmentation into zones affects the route planning and navigational behavior as it primes a driver to approach the problem from such a structure. A similar effect can be expected when drivers are presented a route, for example a solution from routing software as in the Amazon challenge, from which we take the data to demonstrate our methodology (Amazon and MIT Center for Transportation & Logistics 2021a).

2.4. Route Prediction

Typically, the problem of finding a sequence of stops is positioned in the field of combinatorial optimization, but in this research the goal of find an optimal sequence to minimize an objective function does not take center stage. Here, the goal is how route data can be used to predict routes that drivers (will) follow; for an example where learning takes place in order to find the optimal route, see Kool, van Hoof, and Welling (2019).

The capability to learn and predict (part of) the routes is demonstrated in Krumm (2008). The model is trained from drivers’ long-term trip history using GPS data. It uses a Markovian approach,
i.e., on the basis of the last road segment, which can also be adjusted to incorporate more history, it predicts the next segment the driver will take. In the same vein, Ye et al. (2015) propose a route prediction method based on a hidden Markov model that can accurately predict an entire route early in the trip. Wang et al. (2015) also employ a Markov model; their algorithm relies on a probability transition matrix that is developed to represent the knowledge of the driver’s preferred links and routes. For the VRP, Canoy and Guns (2019) show the potential of using a Markov model in an optimization framework. By constructing a transition probability matrix, based on historical data, and by exploiting the VRP structure, they render solutions that resemble actual route plans much better than relying on a distance metric.

The previous works demonstrate that a Markov model is a powerful tool to learn from historical data and to use these in predictions. We adopt such a model by using historical information, which is weighted against a distance metric. The full methodology is detailed in the next section.

3. Methodology

The methodology that we propose relies on the fourfold of elements reviewed in Section 2. We impose a hierarchical structure on two levels: finding the zone sequence on a global level and finding the stop sequence within each zone locally. Such an approach is in line with evidence from human navigation, and more importantly matches the structure of typical routing data—each stop belongs to a zone. We learn on the global level the preference to traverse from one zone to another by adopting a Markov model, which we combine with the distances between zones to form the cost matrix for the global problem. Relying on standard methods, we solve the TSP on the global level to determine the zone sequence, and a series of TSPs on the local level, which mimics the driver capabilities to find near-optimal solutions in small instances. Finally, by patching the local solutions, we deliver a full prediction of the route.

To learn the drivers’ preference over zones we use the fact that each route starts at a station \( s \)—also commonly known as depots, where \( s \) belongs to the set of stations \( \mathcal{S} \). Then for each stop in a route, the data consists of the \( GPS \) coordinates, which can be used to compute distances between
stops, but the methodology can also be used when the costs of traversing from one stop to another are given, e.g., (expected) travel times. Furthermore, each stop within the route should be given an identifier, which relates the stop to belong to a geographical area, i.e., a zone. The zones should not overlap and therefore they form a segmentation. As a side note, the zone ids should be consistent within the set of routes in the dataset. Specifically in relation to zones there are two phases, which form the basis of our approach:

- **Learning.** Extracting zone sequences from historical data enables the counting and collection of zone transitions in a count matrix.
- **Prediction.** For new routes, a combination of a distance and count matrix is used in a TSP formulation which solution provides the zone sequence; after which within zones the stop sequences are iteratively determined by means of a series of OTSPs.

### 3.1. Learning Zone Preferences

For learning, we exploit the information contained in route data to extract information about drivers’ preferences at the level of zones. Each route consists of an ordered series of \(n+1\) stops, namely \(n\) delivery nodes (this number differs per route) and the station. Therefore, a route is represented as a sequence of tuples, \(\text{route} = ((x_0, y_0, z_0), \ldots, (x_n, y_n, z_n))\), with \((x_i, y_i) \in \mathbb{R}^2\) indicating the stop’s geographical location, and \(z_i\) the corresponding zone id, with \(z_i \in \{Z_1, \ldots, Z_M\}\) for \(i \in \{0, \ldots, n\}\). As the data provides the stop sequence per route, we have to convert it to the higher-level sequence of zones.

Given the observations in the introduction and a navigational perspective that all stops within a zone are completed before a driver moves to another zone (Section 2.3), we determine for each route the sequence of unique zones. To illustrate the procedure let us consider a route with \(n\) delivery nodes. First, we consider the sequence of zones for each stop \((z_1, \ldots, z_n)\). Then, the route’s zone sequence is reduced to the sequence of pairs \(((\zeta_1, \tau_1), \ldots, (\zeta_\ell, \tau_\ell))\), where each pair \((\zeta_i, \tau_i)\) represents how many nodes \(\tau_i\) belonging to the same zone \(\zeta_i\) are visited in a row. When the \(\zeta_i\) are pairwise distinct, we consider \((\zeta_1, \ldots, \zeta_\ell)\) to be the desired sequence of unique zones.
Example 1. The following route consisting of six stops over three zones is reduced into a sequence of three unique zones:

\[(z_1, z_2, z_3, z_4, z_5, z_6) = (Z_1, Z_1, Z_1, Z_2, Z_3, Z_3) \rightarrow ((\zeta_1, 3), (\zeta_2, 1), (\zeta_3, 2)) \rightarrow (\zeta_1, \zeta_2, \zeta_3) = (Z_1, Z_2, Z_3).\]

In case of a zone’s reoccurrence (\(\zeta_i = \zeta_j\) for an \(i < j\)), we only keep the occurrence with the highest number of stops by comparing \(\tau_i\) to \(\tau_j\), or keep the earliest appearance in case of a tie, here that is \(\zeta_i\). This procedure is repeated until we are left with a sequence of singly occurring zones. Two extreme, yet illustrative, examples of this procedure are given below.

Example 2. The reduction of two routes consisting of six stops over three zones under the procedure outlined above is illustrated below. First, zone \(Z_3\) is put last as \(\tau_1 = 1\) and \(\tau_4 = 2\):

\[(Z_3, Z_1, Z_1, Z_2, Z_3, Z_3) \rightarrow ((\zeta_1, 1), (\zeta_2, 2), (\zeta_3, 1), (\zeta_4, 2)) \rightarrow (\zeta_2, \zeta_3, \zeta_4) = (Z_1, Z_2, Z_3).\]

In the next case, zone \(Z_1\) occurs thrice, but breaking the tie twice puts zone \(Z_1\) at its first occurrence in the sequence:

\[(Z_1, Z_3, Z_1, Z_2, Z_2, Z_1) \rightarrow ((\zeta_1, 1), (\zeta_2, 1), (\zeta_3, 1), (\zeta_4, 2), (\zeta_5, 1)) \rightarrow (\zeta_1, \zeta_2, \zeta_4) = (Z_1, Z_3, Z_2).\]

Evidently, not all zones are visited within each route, therefore we keep track of all \(M\) zones that have been visited at least once in any route. Having reduced each route to its corresponding zone sequence, we compute an asymmetric count matrix \(N\) where each entry \(N_{ij}\) represents the number of times a driver went from the \(i\)-th to the \(j\)-th zone, so that \(i\) and \(j\) range from 1 to \(M + S\), where \(S\) is the number of stations in the dataset. The pseudocode related to the procedures outlined is provided in the Appendix.

### 3.2. Predicting the Zone Sequence

In the next phase, we predict the route that a driver will take. For that purpose, we translate the count matrix, created in the previous phase, to a transition matrix \(P\) via \(P_{ij} = \frac{N_{ij}}{\sum_{j=1}^{m} N_{ij}}\), so that \(P_{ij}\) reflects the Markovian probability of transitioning from zone \(i\) to zone \(j\).
Besides the zone sequences, the locations of the station and zones are necessary ingredients in the suspected trade-off a driver makes. For this purpose, we define the center of a zone, if not given, by computing the mean latitude and longitude of all stops that were visited in that zone. Having the zone centers’ location, we can compute the distances $\delta_{ij}$ between zone $i$ and zone $j$. We normalize the distances by dividing each element through the largest, i.e., $\max_{i,j} \delta_{ij}$, and store these distances in a matrix $D \in \mathbb{R}^{(M+S) \times (M+S)}$.

As an alternative to the Euclidean distances in $D$, a sensible choice is to use the travel times (or another measure for costs between stops) as a link to measure ‘distance’ between zones. However, in the case when we are only given route data, we can proceed with a two-step procedure to have an approximation of the travel times between zones. Since stops are typically assigned to zones, we first identify for each zone the closest stop within the route to the zone center by using the Euclidean distance metric. Second, once each zone has a stop assigned, we extract the travel times between these stops to compose a matrix $T$ which consists of elements $T_{ij}$ corresponding to normalized travel times from station/zone $i$ to $j$, i.e., $T_{ij} = \frac{t_{ij}}{\max_{i,j} t_{ij}}$ where $t_{ij}$ are the provided travel times. We will compare the performances of both options.

Since we are to minimize a cost matrix, we reverse the transition matrix to $1 - P_{ij}$ in the cost matrix to arrive at the following linear combination with $\omega \in [0,1]$ being a weight parameter:

$$C_{ij} = \omega D_{ij} + (1 - \omega)(1 - P_{ij}),$$

(1)

where $D_{ij}$ can be interchanged with $T_{ij}$, and additionally we set the diagonal entries $C_{ii} = 0$. In fact this problem can be considered as a TSP with unknown costs as we do not a priori know how distance is evaluated against likelihood of zone-to-zone transitions, which contain the implicit tacit knowledge and drivers’ preferences. So a good value of $\omega$ is still to be determined. Next, we are able to use this cost matrix in a TSP formulation, which optimal solution constitutes a tour through all zones in which one or more stops are located. For details about the implementation we refer the reader to the Appendix.
3.3. Predicting the Stop Sequence

At this point, we have a sequence of zones that yet has to be transformed into a sequence of stops, e.g., we are in Figure 3a where the zone sequence is \((Z_A, Z_B, Z_C)\). As the problem size in each zone is considerably smaller, drivers are able to make (near-)optimal navigational decisions that minimize travel time or distance, see Section 2.3. To mimic that behavior, we employ again the framework for solving TSPs, but with the additional flexibility to be an open TSP (OTSP) as we do not need to close the loop within a zone, but rather traverse from one zone to another. Although there are several options available, the procedure detailed in Figure 3 is in accordance with practice. The approach builds on two elements, which are applied iteratively in each zone:

- The last stop of a previous zone (or station, Figure 3b) is used as the starting point of the OTSP for the current zone.

- An additional stop of the next zone is added to each OTSP to add sense of direction to find a good stop to finish its zone.

As seen in Figure 3a, the extra stop (not filled) is selected as the one closest (in terms of Euclidean distance) to the computed zone centers \((\times)\). Naturally, after the solution for an OTSP is found with the additional stop added, the last stop is removed from the solution such that for the subsequent zone the starting point is again a stop in its previous zone. The procedure continues until the last zone is reached for which the station is used as the additional ‘stop’.

Because of clarity of the exposition, we considered Euclidean distances in the figure, but the implementation can also be fed with travel times. The resulting solution thus forms a logical sequence from a navigational point of view as it avoids path crossings and mimics human behavior when solving (O)TSPs, see Section 2.3. A full implementation of the approach is given in the Appendix and is tailored such that it fits the dataset corresponding to the Amazon Last Mile Research Challenge. Also, we use this dataset to show the adequacy of the approach, as the achieved performance puts the approach among the top-tier of submissions for this challenge.
4. Performance Evaluation

To assess the performance of the proposed methodology we employ it on a routing dataset, which was part of the Amazon Last Mile Research Challenge (Amazon and MIT Center for Transportation & Logistics 2021a). Also the metrics that we use are adopted from this challenge and are provided in Section 4.1.

The dataset counts 6,112 routes from 17 different stations. We split the data such that 5,112 routes are used for learning and 1,000 routes are reserved as an out-of-sample test set. Furthermore, we impute missing zone ids by taking over the id of the closest stop in terms of travel time—a sensible choice from the driver’s perspective. The experiments related to our methodology are run using an Intel Core i7-10850H 2.70 GHz CPU and 32 GB RAM. The learning takes approximately
1 minute and with a test set of 1,000 routes, it takes on average 30 minutes to render all routes. However, for choosing a right value of $\omega$ for Eq. (1) and reconsidering it in 4.3, we rely on 5-fold cross validation over the training set of 5,112 routes.

4.1. Route Prediction Metrics

The route prediction performance can be evaluated in two dimensions, that is, how often and by how much a prediction $B$ deviates from the realized stop sequence $A := (0, 1, \ldots, n)$, and thus $B$ is in essence a permutation of $A$. Below, we describe these different components and how they together form an overall route performance metric.

- The sequence deviation $SD_{stop}(A, B)$ measures the difference in stop sequences $A$ and $B$. Given that in both cases all $n + 1$ stops have been visited, we take the stop sequence of $B$, and for each position we trace back when it has been visited in $A$ to create a vector $(a_0, a_1, \ldots, a_n) = \pi(A)$ so that

$$SD_{stop}(A, B) := \frac{2}{n(n - 1)} \sum_{i=1}^{n} (|a_i - a_{i-1}| - 1).$$

Note that indeed if $A \equiv \pi(A) = B$, the deviation becomes 0. Lastly, due to the fact that we deduce the zone sequence from a route by running the procedure outlined in Section 3.1, we can also compute the zone sequence deviation $SD_{zone}(A, B)$ that on the higher level compares sequence deviations between two zone sequences $A$ and $B$.

- Besides considering the position in the sequence, one can also count the number of edit operations needed to come from the predicted route to the realized route, familiarly known as the Levenshtein distance as introduced by Levenshtein et al. (1966). This distance metric, renamed to $ERP_{edit}(A, B)$, counts the number of deletions and insertions to get the same sequence.

- Apart from considering the concurrence and concordance of stop sequences, the size of deviation between two sequences $A$ and $B$ should be evaluated as well. To this end, normalized travel time is introduced to evaluate the difference between two routes on a stop basis; let $A[i]$ and $B[j]$ be the $i$-th stop of $A$ and the $j$-th stop of $B$, with $i, j = 0, 1, \ldots, n$, then it is defined as

$$time_{norm}(A[i], B[j]) := Y_{A[i], B[j]} - \min_{i, j} Y_{A[i], B[j]},$$

$$Y_{A[i], B[j]} := \frac{t_{A[i], B[j]} - \bar{t}}{std(t)},$$
where $t_{A[i], B[j]}$ is the travel time from stop $A[i]$ to stop $B[j]$, and $\bar{t}$ and $\text{std}(t)$ are the mean and standard deviation over all travel times between the stops in the route. Considering per stop the difference between two sequences, we compute the so-called Edit Distance with Real Penalty component ($\text{ERP}_{\text{norm}}$). Formally, it is computed as

$$\text{ERP}_{\text{norm}}(A, B) := \sum_{i=0}^{n} \text{time}_{\text{norm}}(A[i], B[i]).$$

As an aside, dividing $\text{ERP}_{\text{norm}}(A, B)$ by $\text{ERP}_{\text{edit}}(A, B)$ translates to the average additional travel time incurred by deviating, and is called $\text{ERP}_{\text{ratio}}(A, B)$.

Besides these separate metrics, they are combined to create a comprehensive score

$$\text{route_score} = \frac{\text{SD}_{\text{stop}}(A, B) \cdot \text{ERP}_{\text{norm}}(A, B)}{\text{ERP}_{\text{edit}}(A, B)},$$

so that a performance score is obtained by averaging over all $I$ routes to be predicted:

$$\text{Performance} = \frac{1}{I} \sum_{i=1}^{I} \text{route_score}_i. \quad (2)$$

Evidently, a lower route score means that the predicted sequence coincides more with the realized sequence; 0 means they are exactly the same.

### 4.2. Choosing the Weight

As a first experiment, we consider varying the weight parameter $\omega$ in Eq. (1). Furthermore, we can replace distance by travel times $T$ as these are provided in Amazon’s dataset. The performances when varying $\omega$ in both versions are displayed in Figure 4.

As expected, we find that only relying on historical information ($\omega = 0$), or only using distance or travel times ($\omega = 1$) does not perform as well as having a trade-off between the two. Overall, we observe that an $\omega$ value around 0.9 achieves the best trade-off, but actually any value between 0.1 and 0.9 in Eq. (1) (dashed and solid lines) will result in a performance of around 0.035. This observation demonstrates the robustness of the model as being insensitive to misspecification of $\omega$. In addition, we conclude that the difference between travel times or Euclidean distance is small and thus distance can be used as a good proxy for travel time.
4.3. Adjusting the Station’s Weights

Reconsidering Eq. (1), wherein we define an Ω matrix to weigh historical information against travel times, we further study the role of the station as the starting and ending point of a route. For the sake of readability, we consider a matrix $\Omega^{(r)}$ which contains the weights in Ω corresponding to the station and zones that are visited in a specific route $r$ where the station is related to the first row and column. So as to refine the role of the station, we define $\Omega^{(r)}$ matrix as

$$
\Omega^{(r)} = \begin{pmatrix}
0 & \omega_F & \ldots & \omega_F \\
\omega_L & \omega_Z & \ldots & \omega_Z \\
\vdots & \vdots & \ddots & \vdots \\
\omega_L & \omega_Z & \ldots & \omega_Z
\end{pmatrix},
$$

where $\omega_F \in [0, 1]$ is the weight in the cost function from the station to the first zone. Analogously, $\omega_Z \in [0, 1]$ sets the balance for zone-to-zone transitions, and lastly $\omega_L \in [0, 1]$ is the weight corre-
responding to the return to the station. The elements $\Omega_{ij}$ from this matrix are put in the following refinement of Eq. (1) with travel times:

$$C_{ij} = \Omega_{ij} T_{ij} + (1 - \Omega_{ij})(1 - P_{ij}),$$

(3)

where, similarly to Eq. (1), the diagonal values are set to zero. With this generalized cost matrix we first set $\omega_L = 1$ as we anticipate that at the end of a route heading back to the station should be the only concern, and thus the number of times that a specific zone acted as the finish point of a route can be disregarded. This resonates with the thought that a driver is only focused at the end on the travel time it takes to return to station. Having this parameter restrained, we study combinations of $\omega_F$ and $\omega_Z$ to obtain the contour plot in Figure 5. In the plot, we observe that the optimal point should lie somewhere in the region where $\omega_F \in [0.1, 0.2]$ and $\omega_Z \in [0.7, 0.8]$. So when starting a route in the first transition from the station, historical information should be weighted more important than regular zone-to-zone transitions. This implicates that when determining the route globally preference and navigational considerations have a greater impact than the standard trade-off with travel times.

The choice of $\omega_L = 1$ is justified by the degradation in performance for values less than 1. This is shown in Figure 6 by taking four combinations of $(\omega_F, \omega_Z)$ that lie around the inner contour—close to the optimum. In all, when comparing the points in detail, we conclude that the best performance in our cross-validation is obtained when $(\omega_F, \omega_Z, \omega_L) = (0.2, 0.8, 1)$, which results in the reported performance of the final model in Figure 7.

### 4.4. Comparison against Benchmarks

In Table 1, we provide the performance of our approach abbreviated to LG-OL and compare it to two benchmarks to shed some light onto the range of performance scores that can be attained. Besides the metrics outlined in Section 4 we also added the mean travel time in seconds in the **Time** column. First, we apply the Nearest Neighbor algorithm (Figure 2a) on the out-of-sample test set of 1,000 routes. This algorithm sequentially adds a connection from the last stop to the
nearest unvisited stop; a strategy that is sometimes considered to model human route planning behavior (Graham, Joshi, and Pizlo 2000, Wiener, Ehbauser, and Mallot 2009). As another idea, consider the route which minimizes the travel times as the route a driver follows (Figure 2b). Such route is generally hard to compute, but within the range of state-of-the-art solvers (e.g., Gurobi). Surprising is that drivers average travel time (9496 seconds) is about 18% more than when travel time minimization was the only objective (Full TSP); indeed showing that drivers’ navigational decisions cannot be approached as a standard optimization problem.

Our methodology obtains a prediction performance (Performance) of 0.0299—a more than 64% reduction compared to the aforementioned benchmarks. Interestingly, the Nearest Neighbor and full TSP models have similar ERP\textsubscript{ratio} scores, but the full TSP renders better sequences (lower SD\textsubscript{zone} and SD\textsubscript{stop} scores). When considering our individual route scores of the test set in Figure 7
A Data-Driven Methodology for Last-Mile Routing

Figure 6  A sensitivity study on $\omega_L$, (zone-to-station), for selected combinations of $\omega_F$ and $\omega_Z$ that lie around the inner contour of Figure 5. The performances, as defined in Eq. (2), are obtained by averaging over the 5-fold cross-validation scores.

against the benchmarks, we find that our model succeeds to have 40% of the test routes to be near-perfectly predicted, as they are below 0.01; and even gets 80% of them below the 0.05 threshold. In retrospect, reconsidering Figure 4 we find that even for any $\omega$ the unadjusted approach outperforms these benchmarks, supporting the fundamental idea of our learn global, optimize local approach.

Evaluating our performance to the submissions for the challenge (Amazon and MIT Center for Transportation & Logistics 2021b), we find that the approach propels itself in the top-tier of submissions, which performances range from 0.025 to 0.050. Finally, to verify the correctness and potential of the approach, we did an additional hypothetical experiment on the test set of which the results are provided in Table 1 as LG-OL (hypothetical). We extracted the zone sequences using Section 3.1 on the test set so as to consider the situation that we are fully capable to predict the zone sequence right. Thus remaining are the inconsistencies between the design choices made in Section 3, and drivers’ behavior that is not reflected in the methodology. In the experiment, we find that the performance becomes 0.0094, which is well below all submissions for this challenge.


Table 1  Comparison of the methodology against benchmarks on the out-of-sample test set.

| Model                  | SD zone | SD stop | ERP ratio | Time (s) | Performance |
|------------------------|---------|---------|-----------|----------|-------------|
| Nearest Neighbor       | 0.2441  | 0.0873  | 1.3382    | 9865     | 0.1119      |
| Full TSP               | 0.1912  | 0.0650  | 1.3330    | 8036     | 0.0826      |
| LG-OL                  | 0.0940  | 0.0335  | 0.7389    | 8994     | 0.0299      |
| LG-OL (hypothetical)   | 0       | 0.0238  | 0.3831    | 9535     | 0.0094      |
| Driver                 | 0       | 0       | 0         | 9496     | 0           |

Figure 7  Boxplot representing the route score distribution on the out-of-sample test set of 1,000 routes for our model, and the two benchmarks of Table 1.

5. Conclusion and Discussion

The costs associated with last-mile delivery surmount any other shipping costs. Hence, good predictions about the route taken are crucial from an operations management point of view. The proposed approach leads to several broadly applicable insights about delivery driving.

The approach first determines a zone sequence by solving a traveling salesman problem (TSP) over a cost matrix which is a weighted combination of historical information, which has to be learnt from realized routes, and distance (or travel time) between zones. Next, on the local level, adhering to the established zone sequence on the global level, the order of stops to be visited in a zone is found by solving an open TSP (OTSP), which generates locally optimal solutions. Repeating this for all subsequent zones and patching them together generates a feasible route. Therefore this approach can be considered to learn global (over zones) and optimize local (within zones).
Besides that the approach can be argued to follow the logic of human navigation, the approach is convenient, because it only requires learning on the zone level as by tracking the zone transitions. The breakdown makes the approach relatively fast, as the zone-sequence TSP and within-zone OTSPs are not so computationally involved compared to solving a single TSP for the entire route. This also renders the approach amenable to real-time training using newly realized routes of the same day.

The key element in the approach is the computation of a cost matrix that combines distance and historical information when establishing the zone sequence as a TSP. The experiments show that a weighted combination with travel times provides a slight performance improvement over using Euclidean distance. This fact underpins that when determining the zone sequence an implicit trade-off is made between the driver’s zone-to-zone preferences and a measure of distance. Studying the weight value by changing it from 0 to 1, where the setting of 1 corresponds to only considering travel time, uncovers that the exact value is not critical. Overall, the best performance is obtained when the value is close to 0.9. However, the weighting of station-to-zone and zone-to-station transitions can be further adapted in the cost matrix to better mimic driver’s behavior resulting in another performance improvement. In fact, these adaptations are motivated by the logic that at the beginning of a route the driver is more concerned with where to start, whereas at the end, when all stops are visited, the only consideration is the return to the station. Indeed, the experiments confirm these hypotheses and we find that the travel time matrix should be weighted less important in the station-to-zone transitions than in the zone-to-zone transitions, and from zone-to-station it is the only factor. These insights come with the caveat that drivers were suggested a route beforehand (see Amazon and MIT Center for Transportation & Logistics 2021a), which might have tainted the driver’s decisions.

The approach as-is only utilizes historical stop data (with zone ids) in making predictions. Using additional data which are part of the case study, such as package size, start time, quality of the route, did not provide any clear starting points for improvement. Since the zone ids were given, it is
worthwhile to further study the underlying segmentation, because it determines the breakdown of the problem into a global and a series of local problems and as such it is pivotal to the performance of the approach. We have seen that occurrence of zone revisitations are rare. However, in 2.6% of the routes there are more than five revisitations, which hints that in these routes it might be valuable to reconsider the zone segmentation.

The test set also provides us an opportunity to quantify the remaining learning gap. By first extracting the zone sequences in the test set after which we apply the approach, we arrive at a score just below 0.01. This shows that the learn global and optimize local approach can yield a score as low as 0.009 (as a reference we get it to 0.0299, as shown in Section 4). The learning took place on only 5,112 routes, which results in a sparse transition matrix as the zone transition probabilities, when available, are learnt based on the relatively few routes available for each region. Increasing the number of routes for learning, for example by daily tracking of realized routes and updating the count matrix accordingly, the approach can likely improve its accuracy on zone sequence prediction, which have a profound impact on the overall performance.

Another endeavor is to further study the cost matrix structure. The methodology readily incorporates other functional forms, and also different weightings between historic information and distance-based measures. These weightings might depend on the size of the dataset available and even on the length of the route itself. Nevertheless, we find that a linear structure between distance or travel time and historical information is robust to misspecification of this weight parameter and is practically sound.

Finally, the methodology can be employed in other (last-mile) delivery concepts, see for example Boysen, Fedtke, and Schwerdfeger (2021), or generalized to other operational optimization problems, which are susceptible to human interference in the decision making process—a logical starting point is the allied Vehicle Routing Problem, e.g., Bräysy and Gendreau (2005a). In all, this research demonstrates that integrating learning in an optimization pipeline leverages its connection and value to practice.
References

Amazon and MIT Center for Transportation & Logistics, 2021a Amazon Last-Mile Routing Research Challenge. URL https://routingchallenge.mit.edu/about-the-challenge/, (accessed on 30 April 2022).

Amazon and MIT Center for Transportation & Logistics, 2021b Amazon Last-Mile Routing Research Challenge. URL https://routingchallenge.mit.edu/last-mile-routing-challenge-team-performance-and-leaderboard/, (accessed on 30 April 2022).

Applegate DL, Bixby RE, Chvátal V, Cook WJ, 2011 The Traveling Salesman Problem (Princeton University Press).

Boyer KK, Prud’homme AM, Chung W, 2009 The last mile challenge: Evaluating the effects of customer density and delivery window patterns. Journal of Business Logistics 30(1):185–201.

Boysen N, Fettke S, Schwerdfeger S, 2021 Last-mile delivery concepts: a survey from an operational research perspective. OR Spectrum 43(1):1–58.

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

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

Campuzano G, Obreque C, Aguayo MM, 2020 Accelerating the Miller-Tucker-Zemlin model for the asymmetric traveling salesman problem. Expert Systems with Applications 148:113229.

Canoy R, Guns T, 2019 Vehicle routing by learning from historical solutions. International Conference on Principles and Practice of Constraint Programming, 54–70 (Springer).

Chopra S, 2003 Designing the distribution network in a supply chain. Transportation Research Part E: Logistics and Transportation Review 39(2):123–140.

Current JR, Schilling DA, 1989 The covering salesman problem. Transportation Science 23(3):208–213.

Dantzig GB, Ramser JH, 1959 The truck dispatching problem. Management Science 6(1):80–91.
Deloison T, Hannon E, Huber A, Heid B, Klink C, Sahay R, Wolff C, 2020 *The future of the last-mile ecosystem: Transition roadmaps for public-and private-sector players* (World Economic Forum).

Dry M, Lee MD, Vickers D, Hughes P, 2006 *Human performance on visually presented traveling salesperson problems with varying numbers of nodes*. The Journal of Problem Solving 1(1):4.

eMarketer, 2020 *US ecommerce 2020, coronavirus boosts ecommerce forecast and will accelerate channel-shift*. URL https://www.emarketer.com/content/us-ecommerce-2020, (accessed on 30 April 2022).

Gendreau M, Hertz A, Laporte G, Stan M, 1998 *A generalized insertion heuristic for the traveling salesman problem with time windows*. Operations Research 46(3):330–335.

Gendreau M, Laporte G, Vigo D, 1999 *Heuristics for the traveling salesman problem with pickup and delivery*. Computers & Operations Research 26(7):699–714.

Gevaers R, Van de Voorde E, Vanelslander T, 2011 *Characteristics and typology of last-mile logistics from an innovation perspective in an urban context*. City Distribution and Urban Freight Transport (Edward Elgar Publishing).

Gevaers R, Van de Voorde E, Vanelslander T, 2014 *Cost modelling and simulation of last-mile characteristics in an innovative b2c supply chain environment with implications on urban areas and cities*. Procedia - Social and Behavioral Sciences 125:398–411, eighth International Conference on City Logistics 17-19 June 2013, Bali, Indonesia.

Goodman RW, 2005 *Whatever you call it, just don’t think of last-mile logistics, last*. Global Logistics & Supply Chain Strategies 9(12).

Graham SM, Joshi A, Pizlo Z, 2000 *The traveling salesman problem: A hierarchical model*. Memory & Cognition 28(7):1191–1204.

Gunasekaran A, Patel C, McGaughey RE, 2004 *A framework for supply chain performance measurement*. International Journal of Production Economics 87(3):333–347.

Jiang J, Gao J, Li G, Wu C, Pei Z, 2014 *Hierarchical solving method for large scale tsp problems*. International symposium on neural networks, 252–261 (Springer).

Karp RM, 1977 *Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane*. Mathematics of Operations Research 2(3):209–224.
Kool W, van Hoof H, Welling M, 2019 Attention, learn to solve routing problems! International Conference on Learning Representations, URL https://openreview.net/forum?id=ByxBFsRqYm.

Krumm J, 2008 A Markov model for driver turn prediction. Society of Automotive Engineers (SAE) World Congress, April 2008.

Lawler EL, 1985 The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization (Wiley-Interscience Series in Discrete Mathematics).

Levenshtein VI, et al., 1966 Binary codes capable of correcting deletions, insertions, and reversals 10:707–710.

Liao E, Liu C, 2018 A hierarchical algorithm based on density peaks clustering and ant colony optimization for traveling salesman problem. IEEE Access 6:38921–38933.

Lin C, Choy K, Ho G, Chung SH, Lam H, 2014 Survey of green vehicle routing problem: Past and future trends. Expert Systems with Applications 41:1118–1138.

MacGregor JN, Ormerod T, 1996 Human performance on the traveling salesman problem. Perception & Psychophysics 58(4):527–539.

Miller CE, Tucker AW, Zemlin RA, 1960 Integer programming formulation of traveling salesman problems. Journal of the ACM 7(4):326–329.

Montoya-Torres JR, López Franco J, Nieto Isaza S, Felizzola Jiménez H, Herazo-Padilla N, 2015 A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering 79:115–129.

OpenStreetMap, 2021 Planet dump retrieved from https://planet.osm.org. URL https://www.openstreetmap.org, (accessed on 2021-20-11).

Orman A, Williams HP, 2007 A survey of different integer programming formulations of the travelling salesman problem. Advances in Computational Management Science, volume 9, 91–104 (Springer Berlin Heidelberg).

Papadimitriou CH, 1977 The Euclidean traveling salesman problem is NP-complete. Theoretical Computer Science 4(3):237–244.

Pizlo Z, Stefanov E, Saalweachter J, Li Z, Haxhimusa Y, Kropatsch WG, 2006 Traveling salesman problem: A foveating pyramid model. The Journal of Problem Solving 1(1):8.
Purkayastha R, Chakraborty T, Saha A, Mukhopadhyay D, 2020 Study and analysis of various heuristic algorithms for solving travelling salesman problem—a survey. Mandal JK, Mukhopadhyay S, eds., Proceedings of the Global AI Congress 2019, 61–70 (Springer Singapore).

Sengupta L, Mariescu-Istodor R, Franti P, 2018 Planning your route: Where to start? Computational Brain & Behavior 1(3-4):252–265.

Speranza M, Archetti C, 2014 A survey on matheuristics for routing problems. EURO Journal on Computational Optimization 2:223–246.

Taniguchi E, Thompson RG, 2002 Modeling city logistics. Transportation Research Record 1790(1):45–51.

United Nations, 2018 68% of the world population projected to live in urban areas by 2050, says UN. URL https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html, (accessed on 2021-10-11).

Van Rooij I, Stege U, Schactman A, 2003 Convex hull and tour crossings in the Euclidean traveling salesperson problem: Implications for human performance studies. Memory & Cognition 31(2):215–220.

Vickers D, Lee MD, Dry M, Hughes P, 2003 The roles of the convex hull and the number of potential intersections in performance on visually presented traveling salesperson problems. Memory & Cognition 31(7):1094–1104.

Wang X, Ma Y, Di J, Murphey YL, Qiu S, Kristinsson J, Meyer J, Tseng F, Feldkamp T, 2015 Building efficient probability transition matrix using machine learning from big data for personalized route prediction. Procedia Computer Science 53:284–291.

Wiener J, Ehbauer N, Mallot H, 2009 Planning paths to multiple targets: Memory involvement and planning heuristics in spatial problem solving. Psychological Research PRPF 73(5):644–658.

Wiener JM, Mallot HA, 2003 ‘Fine-to-coarse’ route planning and navigation in regionalized environments. Spatial cognition and computation 3(4):331–358.

Ye N, Wang Zq, Malekian R, Lin Q, Wang Rc, 2015 A method for driving route predictions based on hidden Markov model. Mathematical Problems in Engineering 2015.
Appendix. Pseudocode

In this Appendix, we provide the pseudocode to the methodology presented in Section 3. Although the pseudocode has been designed on the basis of the Amazon Last-Mile Routing Research Challenge dataset, it can be easily adapted to different data structures. As a side note, the zone ids are not consistent for different stations, therefore, we run our code for each \( s \in S \), with \( S \) the set of stations. The pseudocode used for the learning phase is reported in Algorithm 1, where the function ToZoneSeq converts the realized sequence of stops into a sequence of zones (see Section 3.1), while function ComputeCountMatrix computes an asymmetric matrix where the \((i,j)\)-th entry represents the number of times a driver went from the \(i\)-th to the \(j\)-th zone. Algorithm 2 illustrates the pseudocode used in the prediction phase. Here, in order to compute the cost matrix, as thoroughly described in Section 3.2, we compute the distance matrix where each entry refers to the travel time between two different zone centers (computed with EstimateZoneCenter).

The code used to run the experiments presented in Section 4 is open source and will be made available.

---

**Algorithm 1 Learning Zone Preferences and Preprocessing**

**Input:** \( \mathcal{D}_s \): route data set corresponding to station \( s \)

**Input:** StopSeqs: actual sequence of stops for each route in \( \mathcal{D}_s \)

**Output:** CountMatrix \( \in \mathbb{R}^{(M+1) \times (M+1)} \): counts of station/zone transitions

1: ZoneSequences \( \leftarrow \emptyset \)

2: AllZones \( \leftarrow \{s\} \)

3: for route in \( \mathcal{D}_s \) do

4: ZoneSequences[route] \( \leftarrow \) ToZoneSeq(StopSeqs[route])

5: AllZones \( \leftarrow \) AllZones \( \cup \) \{ZoneSequences[route]\}

6: end for

7: CountMatrix \( \leftarrow \) ComputeCountMatrix(AllZones, ZoneSequences)
Algorithm 2 Predicting Routes

**Input:** CountMatrix: output of Algorithm 1

**Input:** Zones, Stops: zones and stops characterizing the route

**Input:** station, TravelTimes

**Output:** PredictedStops: predicted sequence of stops

1: ZoneCenters ← EstimateZoneCenter(Zones, Stops)
2: DistanceMatrix ← ComputeDistMatrix(Zones, ZoneCenters)
3: CostMatrix ← ComputeCostMatrix(DistanceMatrix, CountMatrix)
4: PredictedZoneSeq ← TSP(station, Zones, CostMatrix)
5: PredictedStopSeq ← [station]
6: for zone in PredictedZoneSeq do
7: PrevStop ← PredictedStopSeq[-1]  \(\triangleright\) Last element of PredictedStopSeq
8: AddStop ← ClosestToNextZoneCenter(zone, Stops)
9: PredictedStopSeq ← PredictedStopSeq + [OTSP(PrevStop, Stops[zone], AddStop, TravelTimes)]
10: end for