Incorporating Intuition in Last-Mile Routing

Mayukh Ghosh*¹, Roshan Mahes*¹,², Donato Maragno*¹, Alex Kuiper¹

1 Amsterdam Business School, University of Amsterdam, 1018 TV Amsterdam, the Netherlands
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 optimization 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 found as a result of a minimization over a cost matrix which balances historical information and distances between zones. Second, within zones, a sequence of stops is determined, such that integrated with the predetermined zone sequence a full solution is obtained.

The methodology allows for several enrichments, such as replacing distances by travel times, different cost structures, and intuition while traversing from one zone to another ensuring a seamless connection. These directions are particularly promising as they propel the approach in the top-tier of submissions to the Last-Mile Routing Research Challenge while maintaining the elegant decomposition that ensures a feasible implementation into practice. From a cognitive perspective, the concurrence between prescribed and realized routes reveals that delivery drivers adopt a hierarchical breakdown of the full problem combining intuition and locally optimal decisions when navigating. Furthermore, experimenting with the trade-off between historical information and travel times exposes that drivers rely more on their intuition at the start, whereas at the end, when returning to the station, only travel time is the concern.

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

* These authors contributed equally.
1. Introduction

This research has been instigated as part of the Last-Mile Routing Research Challenge (LMRRC) hosted by Amazon and MIT Center for Transportation & Logistics (2021). The aim is to improve Amazon’s last leg of delivery services, i.e., the determination of a route to deliver a multitude of packages, which incorporates operational conditions, such as traffic, parking availability, et cetera. Last-mile routing is often the shortest, yet most complex and expensive leg in distribution systems. For example it can account for about 28% of the total costs of transportation from the distribution center to the final customer (Goodman 2005, Gevaers, Van de Voorde, and Vanelslander 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, since 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).

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 e-commerce sales are currently estimated to be $709.78 billion, which will further increase as the coronavirus accelerates a channel shift to e-commerce. Besides, not only in mature markets but also in emerging markets, this growth creates a multitude of challenges in last-mile distribution (Janjevic and Winkenbach 2020).

Considering last-mile delivery in detail, besides letting the products arrive in good shape, the performance constitutes to be on-time (Gunasekaran, Patel, and McGaughey 2004). To ensure that a prescribed route will be followed up, a proposed route should reflect the driver’s intuition. Otherwise, the driver is likely to deviate from the prescribed route incurring unforeseen time losses, additional operational expenses and failing to meet communicated time slots (Boyer, Prud’homme, and Chung 2009, Gevaers, Van de Voorde, and Vanelslander 2014). Drivers, who mostly drive the same route, may have tacit information about the conditions at a ground level, e.g., traffic, road structure, parking space. Therefore, the specified objective of the challenge is to come up with an approach that leverages available (historical) delivery data in a way to predict the route sequence
Figure 1 Examples of two realized routes by using OpenStreetMap (2021). The numbers represent the order in which the stops are visited and the colors represent the different zones within the routes.

that is followed by a driver, so that as a consequence more viable routes can be prescribed. Although the challenge is framed as a predictive problem, we emphasize that the output of the model is also prescriptive in nature which makes the two terms prediction and prescription interchangeable.

Two delivery drivers’ routes are given in Figure 1 as an increasing sequence of numbers. 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. Observing the routes in the left (more rectangular grid) and the right map, we find that the size of a zone ranges from a part of a street to a set of multiple streets. Also, as in line with what is observed in the data, drivers typically do not revisit a zone.

To prescribe routes that incorporate the tacit information in realized route data, we contribute an approach that breaks the problem down 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. The approach stems from the intuition that a driver first simplifies the problem such that on the higher level it ensures visiting all zones in the most convenient way possible. Secondly, the problem in each zone is of much lower complexity and a driver is capable to determine the best sequence out of the set of
stops. In addition, learning at the zone level is an effective and natural way of leveraging historical information as opposed to overly detailed stop data. This breakdown reduces the computational complexity of the problem considerably and makes a real-time implementation feasible.

Ultimately, the approach obtains the zone sequence by balancing historical zone transition information and distances between zones in a cost matrix used in the formulation of a Traveling Salesman Problem (TSP) (Flood 1956); its performance confirms the competency of the approach in predicting last-mile routing. On top of that, several amendments are suggested: adjusting the distance metric, forcing the location of the first and last stop within a zone to ensure a smooth between-zone transition, and last but not least, changing the weighting between historical information and distance. These adjustments incorporate important navigational considerations that close the literature gap between stylized models and route prediction in practice.

In the next section, we provide a literature review. The subsequent section outlines the methodology of our approach in full detail and shows the baseline model performance. In Section 4 various amendments to the approach are discussed and incorporated in the final model. Finally, in Section 5, we present our concluding remarks followed by a discussion on future research directions.

2. Literature Review

In theory, solving the last-mile routing problem comes down to tackling a TSP, but drivers learn from the complex operational environment they are facing every day and act accordingly. Therefore in the pursuit of finding an approach that incorporates this tacit information elements of different disciplines are combined, which deem it a challenge: we break down the classical TSP optimization framework in smaller instances, consider the human approach to solving TSPs, and integrate it with learning from historically realized routes. 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 in 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 sub-problem is subsequently optimized and the center nodes of the sub-problems 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, see also the next section.

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 Schactman 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 sub-optimal, 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 structure.

2.4. Route Prediction

Although the challenge can be positioned in the field of combinatorial optimization, the optimization setting does not take center stage, rather the question is how data from practice can be used to predict routes. For this purpose, we propose a data-driven model that incorporates learning in the optimization framework in line with Simchi-Levi (2014).

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 the 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 a data set containing 6,112 routes is provided. Each of these routes starts at one of the 17 stations $s \in S$, also commonly known as depots, across the United States of America. For each stop, the data consists of the GPS coordinates, the (expected) travel time to any other stop within the route, the dimensions of the packages to deliver, and an identifier of the zone, called the zone id. As a side note, these ids are consistent within the set of routes that share the same station; if the station is different, a same zone id might refer to two different geographical regions. After visiting all stops, the driver ends the route at the original station.
Besides that the approach has a hierarchical structure, there is also a distinction between model building (learning from realized routes) and applying it (predicting new routes):

- On the global level, *model build phase*: zone centers are determined, which allow the computation of a distance matrix. Furthermore, extracting zone sequences from historical data enables the counting and collection of zone transitions in a count matrix.

- On the global and local level, *model apply phase*: a combination of distance and count matrices is used to determine, for new routes, the zone sequence by formulating a TSP over the zones including the station. When the zone sequence has been established, the stop sequence within each zone is determined by means of a series of OTSPs.

An overview of the framework with the hierarchy between zones and stops is provided in Figure 2.

**Figure 2**  Overview of the methodology on a station level. Historical data aggregated at a station level is used to calculate the distance and count matrices in the model build phase, which are used in the model apply phase to get predicted new routes as output.

### 3.1. Model Build Phase

In the model build phase, we exploit the available data to extract information about drivers’ preferences. In accordance with the fact that the serving areas of stations do not overlap, and to ease notation, the pseudocode in Algorithm 1 and corresponding discussion refer to a specific station \( s \in S \).

For a given station \( s \), a corresponding set of routes \( \mathcal{D}_s \) is available. Each of these routes 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 $z_i \in \{\dagger, Z_1, \ldots, Z_M\}$ for $i \in \{0, \ldots, n\}$ where a $\dagger$ indicates the case a zone id is unavailable. As the data provides the stop sequence per route, we have to convert it to the higher-level sequence of zones.

First of all, we impute missing zone ids by taking over the id of the closest stop in terms of Euclidean distance. 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 performed by function ToZoneSeq, let us consider a route with $n$ delivery nodes where the actual sequence of stops is represented by $\text{StopSeqs}[\text{route}]$. First, $\text{StopSeqs}[\text{route}]$ is used to define the sequence of zones $(z_1, \ldots, z_n)$. 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. The first example sets zone $Z_3$ last, since $\tau_1 = 1$ 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 example 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 the set $\textbf{AllZones}$ which finally consists of the station and all $M$ zones that have been visited at least once in a route that started from the corresponding station. 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 0 to $M$ as the station is considered to be the zeroth zone.

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 call the function $\text{EstimateZoneCenter}$, which defines the center of a zone by computing the mean latitude and longitude of all stops 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 $D \in \mathbb{R}^{(M+1) \times (M+1)}$.

### 3.2. Model Apply Phase

In the model apply stage, we use the model build outputs to predict the sequence of stops for new routes. The steps are summarized in Algorithm 2. Given in this stage are the two matrices from the previous phase; for now a Euclidean based distance matrix and a count matrix. As a baseline, we propose the following structure to create a cost matrix $C \in \mathbb{R}^{(M+1) \times (M+1)}$ over which we formulate a global problem:

$$C_{ij} = \frac{D_{ij}}{1 + N_{ij}}. \quad (1)$$

This is done in the function $\text{ComputeCostMatrix}$. Note that this form reduces the cost in terms of distance between two zones whenever it has been frequently traversed, but the relative impact of it decreases as $N_{ij}$ increases. So the matrix $C$ can be read as the cost of traversing from one zone to another.
Algorithm 1 Model Build Phase

Input: $D_s$: route data set corresponding to station $s$

Input: StopSeqs: actual sequence of stops for each route in $D_s$

Output: ZoneCenters $\in \mathbb{R}^{(M+1)\times 2}$: (estimated) zone centers for each zone in $D_s$

Output: DistanceMatrix, CountMatrix $\in \mathbb{R}^{(M+1)\times (M+1)}$

1: ZoneSequences $\leftarrow \emptyset$
2: AllZones $\leftarrow \{s\}$
3: for route in $D_s$ do
4: ZoneSequences[route] $\leftarrow$ ToZoneSeq(StopSeqs[route])
5: AllZones $\leftarrow$ AllZones $\cup$ {ZoneSequences[route]}
6: end for
7: ZoneCenters $\leftarrow \emptyset$
8: for zone in AllZones do
9: ZoneCenters[zone] $\leftarrow$ EstimateZoneCenter(zone, $D_s$)
10: end for
11: DistanceMatrix $\leftarrow$ ComputeDistMatrix(AllZones, ZoneCenters)
12: CountMatrix $\leftarrow$ ComputeCountMatrix(AllZones, ZoneSequences)

It might happen that the route to be predicted contains stops that miss a zone id or lie in a zone that has not been visited before. To deal with the former case, we take the zone id of the closest stop as similar to Section 3.1. When dealing with new zones, we have to enrich the cost matrix to incorporate traveling from and to this zone. To this end we compute, based on the stops in this zone, the mean latitude and longitude, and update the cost matrix by considering the distance of the new zone to all other zones. Obviously, there are no counts to and from this zone to other zones, so we only consider distances to define the new row and column in the cost matrix. This preprocessing is handled in the UpdateCostMatrix function.
Next, we are able to use the cost matrix in a TSP formulation which results the optimal tour between the zones and station and thus the predicted zone sequence (\texttt{PredictedZones}). The TSP is solved using the open source modeler PuLP (Mitchell, Consulting, and Dunning 2011) with the Coin-or Branch and Cut (CBC) solver (Lougee 2003).

At this point, we have a sequence of zones that has to be transformed into a sequence of stops. In order to do so, we solve an OTSP for each zone. The TSPs are open, because we do not need to close the loop within a zone but rather move from one zone to another. Although there are several options to ensure a smooth transition (see also Section 4), a first and natural way to do so is to start each OTSP at the stop that lies closest to the final stop of the previous zone in terms of (provided) travel times. This is done via the function \texttt{FindClosestStop}; for the first zone it takes the station as the final stop. Again, relying on the same solver, we obtain a sequence of stops that predicts the route of a driver.

\begin{algorithm}
\caption{Model Apply Phase}
\begin{algorithmic}[1]
\Input ZoneCenters, DistanceMatrix, CountMatrix: outputs of Algorithm 1
\Input Zones, Stops: zones and stops characterizing the route
\Input station, TravelTimes
\Output PredictedStops: predicted sequence of stops

1: CostMatrix $\leftarrow$ \texttt{ComputeCostMatrix}(DistanceMatrix, CountMatrix)
2: UpdatedCostMatrix $\leftarrow$ \texttt{UpdateCostMatrix}(ZoneCenters, CostMatrix, Zones)
3: PredictedZones $\leftarrow$ TSP(station, Zones, UpdatedCostMatrix)
4: PredictedStops $\leftarrow$ [station]
5: \For {zone in PredictedZones} \Do
6: \quad PrevStop $\leftarrow$ PredictedStops[-1] $\triangleright$ Last element of PredictedStops
7: \quad FirstStop $\leftarrow$ \texttt{FindClosestStop}(zone, PrevStop)
8: \quad PredictedStops $\leftarrow$ [PredictedStops, OTSP(FirstStop, Stops[zone], TravelTimes)]
9: \EndFor
\end{algorithmic}
\end{algorithm}
3.3. Performance Evaluation

Since the goal is to predict unseen routes, we split the data such that part of it can be used for training and another for measuring prediction. In order to get an accurate performance estimation, we reserve 1,000 routes for the test score of the final model and with the remaining 5,112 routes we use a 5-fold cross-validation to measure the validation performance. The model build phase is applied on the data from four folds and the remaining one is used for prediction. Averaging over the results of the five iterations, we get the validation score.

The performance is 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 the different components of the performance metric as prescribed by the organizers of the challenge (Amazon and MIT Center for Transportation & Logistics 2021).

- The sequence deviation $\text{SD}_{\text{stop}}(A, B)$ 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

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

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 $\text{SD}_{\text{zone}}(A, B)$ 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 $\text{ERP}_{\text{edit}}(A, B)$, 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

\[
\text{time}_{\text{norm}}(A_i, B_j) := Y_{A_i, B_j} := \frac{t_{A_i, B_j} - \bar{t}}{\text{std}(t)},
\]

where $t_{A_i, B_j}$ is the travel time from stop $A_i$ to stop $B_j$ and, $\bar{t}$ and 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

\[
\text{route}_{\text{score}} = \frac{\text{SD}_{\text{stop}}(A, B) \cdot \text{ERP}_{\text{norm}}(A, B)}{\text{ERP}_{\text{edit}}(A, B)},
\]

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

\[
\text{Performance} = \frac{1}{I} \sum_{i=1}^{I} \text{route}_{\text{score}},
\]

Evidently, a lower route score means that the predicted sequence coincides more with the realized sequence; 0 means they are exactly the same. The results for the baseline model on the test set containing 1,000 routes are given in Table 1. Also, we already present the performance of the final model, which is the culmination of the refinements detailed in the next section to make the approach practically more sound. Next to the baseline and the final model, we include two benchmarks. First, the Nearest Neighbor algorithm, which sequentially adds a connection from the last stop to the nearest unvisited stop. Such strategy is sometimes considered to model human route planning behavior (Graham, Joshi, and Pizlo 2000, Wiener, Ehbauer, and Mallot 2009). Second, for each
Table 1  Comparison of the methodology against benchmarks on the hold-out test set.

| Model       | SD\textsubscript{zone} | SD\textsubscript{stop} | ERP\textsubscript{norm} | ERP\textsubscript{edit} | ERP\textsubscript{ratio} | Perf.   |
|-------------|------------------------|------------------------|--------------------------|--------------------------|--------------------------|---------|
| Nearest Neighbor | 0.2441              | 0.0873                 | 201.6775                 | 145.0330                 | 1.3382                   | 0.1119   |
| Full TSP    | 0.1912                | 0.0650                 | 199.3882                 | 142.9720                 | 1.3330                   | 0.0826   |
| Baseline    | 0.1448                | 0.0437                 | 166.0363                 | 136.6430                 | 1.1390                   | 0.0512   |
| Final       | 0.0940                | 0.0335                 | 95.8804                  | 112.6360                 | 0.7389                   | 0.0299   |

Notes: The baseline refers to the model described in Section 3 while the final refers to the model described in Section 4. The performance metrics are described in Section 3.3.

instance in the test set the full TSP solution is generated, which is computed using Gurobi (a state-of-the-art solver).

Considering the performance scores, the baseline model already yields a solid performance of 0.0512, which is a 38% reduction on the full TSP performance. The final model boosts the score even further to 0.0299—a 42% reduction on the baseline performance and more than 64% reduction compared to the 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). But these sequences are not near the ones that are found by using a hierarchical breakdown implemented in the baseline and the final model. Finally, when considering individual route scores in Figure 3, we find that in the baseline model only 90 out of the 1,000 test routes (9%) receive a near-perfect score of below 0.01, while the final model succeeds to have 40% of the test routes to be near-perfectly predicted; and even gets 80% of them below the 0.05 threshold.

The experiments related to our methodology are run using an Intel Core i7-10850H 2.70 GHz CPU and 32 GB RAM. All the experiments satisfy the challenge time limits, namely 2 hours for the model build phase for a train set of 6,112 routes and 12 hours for the model apply phase for a test set of approximately 3,000 routes. More precisely, with a train set of 5,112 routes, it takes on average 1 minute for the model build phase, and with a test set of 1,000 routes, it takes on average 30 minutes for the the model apply phase. There is no significant difference in the run time between the baseline and the final model; for which the code is made publicly available (https://github.com/donato-maragno/Amazon-LMRRC).
4. Amendments and Experiments

In the previous section we presented the baseline approach as submitted to the challenge. The methodology accommodates several modifications, which further improve performance. We discuss and rationalize three directions: patching zones together, incorporation of other distance-based metrics, and last but not least several modifications to the cost matrix structure. Ultimately, we arrive at a model that has excellent predictive capability according to the performance indicators of Section 3.3. All the scores reported in this section are computed as the average of a five-fold cross-validation.

4.1. Connecting Within-Zone Sequences

As the zone sequence has been established, the next step is to connect the series of within-zone problems. On this local level, the goal is to have a connection between zones to better mimic the behavior of a driver when visiting a zone and moving to a next one. To do this, we distinguish three approaches that build on top of each other and are illustrated in Figure 4. Because of clarity of the exposition, we consider Euclidean distances in this figure, but in the implementation each OTSP is fed with the provided travel times.

Let the station be located in the bottom-left corner in each figure. In each of the cases the predetermined zone sequence is obeyed, but as seen the approaches result in different routes. We
I. Each OTSP starts at the stop closest to the final stop of the previous zone (baseline).

II. Include the last stop of the previous zone in each OTSP.

III. Add sense of direction in each OTSP by including a stop from next zone closest to the center (starred).

Figure 4 Three approaches to patch within-zone OTSPs (the zone sequence is Z₁, Z₂, and Z₃). In I, the densely dotted lines link the OTSP solutions (solid lines). In II, the dashed lines are also the result of an OTSP optimization. In III, both the dashed and loosely dotted lines are additional outputs of each optimization, but the dotted part has to be removed each time; except to the station (dash-dotted).

discuss the incorporation and the (dis)advantages of them below. They are readily implemented by modifying the TSP formulations.

I. Starting top-left in Figure 4, it shows the baseline approach of Section 3. Here, each OTSP is solved such that the first stop is the closest to the final stop of the previous zone. This baseline ensures that we have zone transitions (densely dotted) and that within a zone an optimal path, in terms of travel time, is followed. However, it results in outcomes wherein the last stop within one zone can be far from the first stop in a subsequent zone. As a result, the solution may result in excessive traveling, which will unlikely be the route followed by the driver, as drivers have good intuition of the global problem.

II. An improvement to the approach of I is to set the last stop of a previous zone to be the starting point of the OTSP for the current zone as shown in Figure 4; the dashed lines are now part of the OTSPs, starting from the station. The stop in the last zone is still set to be connected to the station. Although this intervention helps, it is still an inefficient outcome as the OTSP within a zone can finish at an inconvenient stop with regard to the stops of a next zone, as seen when traversing from Z₂ to Z₃.
III. To remedy the issues of II, we extend it by adding a stop from the subsequent zone, which are indicated by the dotted lines. Such a stop (the black circles in $Z_2$ and $Z_3$) is selected as the one closest (in terms of Euclidean distance) to the computed zone centers, which are starred in the figure. This continues until the station is used as additional ‘stop’ in the OTSP solution for the last zone. The resulting solution is the path without the dotted lines, which forms a logical sequence from a navigational point of view as it avoids path crossings, see Section 2.3.

To study the impact of patching, we incorporate the variants in the baseline model. The performance in terms of the metrics that are detailed in Section 3 are given in Table 2. As expected, we see that III outcompetes the other methods; note that in each case the zone sequence is the same since this amendment only affects the performance on the local level. So, incorporating a sense of direction by fixing the first and last nodes is adequate when predicting last-mile routing.

| Approach | SD$_{\text{zone}}$ | SD$_{\text{stop}}$ | ERP$_{\text{norm}}$ | ERP$_{\text{edit}}$ | ERP$_{\text{ratio}}$ | Perf. |
|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-------|
| I (baseline) | 0.1396 | 0.0432 | 167.0408 | 136.7003 | 1.1486 | 0.0509 |
| II | 0.1396 | 0.0422 | 166.8369 | 135.7301 | 1.1511 | 0.0499 |
| III | 0.1396 | 0.0400 | 164.9235 | 133.2536 | 1.1466 | 0.0473 |

Notes: The performance metrics are described in Section 3.3. The values in the table are obtained by averaging over the 5-fold cross-validation scores which differs from the scores on the test set, cf. Table 1.

4.2. Distance Metric

In the baseline we rely on the Euclidean distance metric in the distance matrix $D$ to determine the zone sequence in the global problem. Since, from a practical point of view, the implementation of straight-line distance is convenient as it relies on zone centers for which no travel times are available. However, it might be that changing this metric that proxies the distances between zone centers has an impact on the resulting zone sequence.

To study the impact of the choice of metric, we first replace it by the great-circle (Haversine) distance to incorporate the fact that the earth is a sphere. Another reasonable alternative is to rely on the Manhattan distance, which is defined as the sum of the absolute differences of their Cartesian coordinates, since it better captures the classical grid structure of roads. However, compared with
the Euclidean distance, experiments with the other metrics have shown no significant difference in terms of performance.

Another direction is to step down from using actual distances and instead use (expected) travel times as a link to measure ‘distance’ between zones. However, these times are not available as only travel times between stops within the route have been given. Since stops are assigned to zones, we can proceed with a two-step procedure to retrieve travel times between zones. First, we 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. This procedure outperforms the distance-based metrics indicating that travel times are key to imitate drivers’ behaviors.

| Metric       | SD\(_{zone}\) | SD\(_{stop}\) | ERP\(_{norm}\) | ERP\(_{edit}\) | ERP\(_{ratio}\) | Perf. |
|--------------|---------------|---------------|----------------|----------------|----------------|-------|
| Euclidean    | 0.1396        | 0.0400        | 164.9235       | 133.2536       | 1.1466         | 0.0473 |
| Haversine    | 0.1492        | 0.0416        | 168.0236       | 134.0533       | 1.1634         | 0.0497 |
| Manhattan    | 0.1406        | 0.0402        | 166.1590       | 133.4953       | 1.1540         | 0.0478 |
| Travel time  | 0.1354        | 0.0393        | 163.9405       | 132.8623       | 1.1390         | 0.0462 |

Notes: The performance metrics are described in Section 3.3. The values in the table are obtained averaging over the 5-fold cross-validation scores.

4.3. Structure of the Cost Matrix

As introduced in Section 3, the cost matrix consists of two elements. One is based on a distance related measure, while the other is based on historical information, capturing intuition and preference. This is initially done by invoking the count matrix $N$ to weigh down the distance if that zone transition has been frequently traversed, see Eq. (1). But the approach is versatile to accommodate different structures, so we can readily tune how heavily this weighting should be applied. So, a general form, for any $\omega \in (0, 1]$, reads:

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

(3)
wherein $\omega = 0.5$ returns a cost matrix equivalent to approach III in Section 4.1. Note that the cost of a self-loop is zero ($C_{ii} = 0$), but such loops are forbidden by the set of constraints in the TSP formulation, see Section 2.1.

Instead of imposing the multiplicative structure of above, an additive combination of distance and history models the situation that the driver weighs distance against history in a linear way. To do so, we translate the count matrix 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 $j$. As we minimize over a cost matrix, we reverse the transition matrix to $1 - P_{ij}$ in the cost matrix to arrive at the following linear combination with $\omega \in [0, 1]$ again being a weight parameter:

$$C_{ij} = \omega D_{ij} + (1 - \omega)(1 - P_{ij}), \quad (4)$$

and additionally we set the diagonal entries $C_{ii} = 0$. Finally, in both versions Eqs. (3) and (4), we also consider the replacement of the normalized Euclidean distance matrix $D$ by normalized travel times $T$, i.e., replace each element $D_{ij}$ by $T_{ij}$, as they have shown to be promising in Section 4.2. This results four versions, which are displayed in Figure 5. We observe that the additive cost structures outperform the multiplicative counterparts at any $\omega$ value with the exception when $\omega$ is nearly zero. Therefore we conclude that a linear combination of historical information and distance captures the implicit trade-off that a driver makes when planning a global route across zones. Although travel times show a considerable improvement in the multiplicative structure (for $\omega > 0.5$); the increase in performance for the additive structure is smaller and, as a result, the Euclidean metric can well be used as a proxy for travel time. Overall, we also observe that an $\omega$ value around 0.9 models the trade-off the best, but any value between 0.1 and 0.9 in Eq. (4) (dashed and solid lines) will also result in a performance of around 0.035. This observation underpins the robustness of the model as being insensitive to perturbations in $\omega$. In the next section we reconsider the pivotal role of the station, i.e., being the start and finish of any route.
4.4. Role of the Station

Taking the additive model, wherein we weigh the historical information against travel times, we further study the role of the station as the starting and ending point. In the cost matrix, the station is related to the first row (where it is the starting point) and column. In order to refine the role of the station by varying the corresponding $\omega$ values, we define the $\Omega$ matrix as

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

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-
sponding to the return to the station. The elements \( \Omega_{ij} \) from this matrix are put in the following refinement of Eq. (4) with travel times:

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

where, similarly to Eq. (4), the diagonal values are set to zero. With this generalized cost matrix we first set \( \omega_L = 1 \) as we anticipate that a driver is focused on starting at specific zones more than ending, so that, at the end of a route heading back to the station is the only concern, i.e., travel time. Having this parameter restrained, we study combinations of \( \omega_F \) and \( \omega_Z \) to obtain the contour plot in Figure 6. 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 from the station, historical information is weighted more important than zone-to-zone transitions during the route, which implicates that a driver is much more concerned about which zone to start the route with.

The choice of \( \omega_L = 1 \) is justified by the degradation in performance for values less than 1. This is shown in Figure 7 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 3.

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. Although this research originated as part of the Amazon Research Challenge, the proposed approach leads to several broadly applicable insights about delivery driving. The approach is based on first determining the zone sequence on a global level by solving a TSP over a cost matrix which is a weighted combination of historical information and distances between zones. Next, on the local level, adhering to the established zone sequence, the order of stops to be visited in a zone is found by solving an OTSP. Repeating this for all subsequent zones and patching them together generates a feasible route. The approach is convenient since it only learns on the zone level by simply tracking
Figure 6  A contour plot showing the performance across different weightings of $\omega_F$ (station-to-zone) and $\omega_Z$ (zone-to-zone) transitions, while keeping $\omega_L = 1$ (zone-to-station) in Eq. (5). The values are obtained averaging over the 5-fold cross-validation scores.

the zone transitions. But also 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 the newly realized historical routes of the same day. Furthermore, the approach lends itself to several amendments.

The key element in the approach is the computation of a cost matrix when establishing the zone sequence by means of a TSP formulation; it combines distance and historical information. The experiments show that a weighted combination with travel times provides better performance than any theoretical distance measure, e.g., Euclidean, but this difference is small. This fact shows that when determining the zone sequence an implicit trade-off is made between the driver’s zone-to-zone preferences and some measure of distance.

Studying the weight value by changing it from 0 to 1, where the setting of 1 corresponds to only taking travel time in consideration, uncovers that the exact value is not that critical. Over-
all, 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 tweaked in the cost matrix to further boost performance. Studying these variations is motivated by the logic that at the beginning of a route the driver evaluates where to start, whereas at the end, when all stops are visited, the only concern is returning to the station. Indeed the experiments confirm these hypotheses and we find that the travel time matrix should be weighted less in the station-to-zone transitions than in the zone-to-zone transitions, and from zone-to-station it is the only important factor.

On a detailed level, experiments with varying the method of connecting the zones together to form a full route reveal to have a big impact on the performance. Especially adjusting the OTSP of each zone to include the last stop of the preceding zone and a stop of the successive zone to add a sense of direction performs astonishingly well. In turn, this result substantiates that drivers subdivide the full stop sequence problem into the smaller within-zone problems. Moreover, it underpins the driver’s cognitive capability to solve the stop sequence to (near-)optimality ensuring smooth transitions to subsequent zones.
The approach only utilizes historical stop data (with zone ids) in making predictions, therefore an open question is whether the model can be further improved by using additional features to explain and close the final prediction gap. The additional data given as part of the challenge (package size, start time, quality of the route, et cetera) did not provide clear starting points for improvement. Although the zone ids were given, it is worthwhile to study the underlying segmentation, since it determines the breakdown of the problem into a global and a series of local problems and as such is pivotal to the performance of the approach.

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 approach can be employed in more (last-mile) delivery concepts, see for example Boyesen, Fedtke, and Schwerdfeger (2021), or generalized to other operational optimization problems, which are susceptible to human interference—a logical starting point is the allied Vehicle Routing Problem, e.g., Bräysy and Gendreau (2005a). From a predictive and prescriptive point of view, this research demonstrates that learning and integrating intuition in an optimization pipeline leverages its performance in practice.

Acknowledgments

This work was supported by the Dutch Scientific Council (NWO) through grant OCENW.GROOT.2019.015, Optimization for and with Machine Learning (OPTIMAL), and grant 024.002.003, Gravitation project NETWORKS.
References

Amazon and MIT Center for Transportation & Logistics, 2021 Amazon Last-Mile Routing Research Challenge. URL https://routingchallenge.mit.edu/, (accessed on 2021-20-11).

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, Fedtke 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.

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 2021-20-11).

Flood MM, 1956 The traveling-salesman problem. Operations Research 4(1):61–75.
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.

Janjevic M, Winkenbach M, 2020 Characterizing urban last-mile distribution strategies in mature and emerging e-commerce markets. Transportation Research Part A: Policy and Practice 133:164–196.

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.

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.

Lougee R, 2003 *The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community*. *IBM Journal of Research and Development* 47:57–66.

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.

Mitchell S, Consulting SM, Dunning I, 2011 *PuLP: A linear programming toolkit for Python*. (accessed on 2021-20-11).

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.

Simchi-Levi D, 2014 *OM forum—OM research: From problem-driven to data-driven research*. *Manufacturing & Service Operations Management* 16(1):2–10.

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 2021-20-11).

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.