Integrating driver behavior into last-mile delivery routing: Combining machine learning and optimization in a hybrid decision support framework

Peter Dieter[a], Matthew Caron[a], Guido Schryen[a]

[a]*Paderborn University, Warburger Str. 100, Paderborn, 33098, Germany*

Abstract

The overall quality of last-mile delivery in terms of operational costs and customer satisfaction is primarily affected by traditional logistics planning and the consideration and integration of driver knowledge and behavior. However, this integration has yet to be exploited. This phenomenon is mirrored in two largely separated research bodies on logistics planning and driver behavior. Bridging this gap by using and integrating historical data from actually driven tours into last-mile delivery planning is promising for research and practice. Still, it also leads to complex and large-scale routing problems, which require the development of an overall methodology that goes beyond classical optimization approaches as the needed approach requires a multi-stakeholder perspective, calls for a hybrid-analytical approach by incorporating tour prediction and prescription, and requires both data science and optimization methods. Accounting for these challenges, we suggest a hybrid decision support framework for the traveling salesman problem with time windows that combines machine learning techniques and conventional optimization methods and considers the deviation between suggested and predicted tours. We demonstrate the applicability of our framework in a case study that draws on real-world logistics data. Relying on a sensitivity analysis, we investigate and illustrate the trade-off between the level of deviation between predicted and suggested tours and tour costs. Our case study draws general managerial implications and recommendations that guide decision makers in building their decision support systems for last-mile delivery routing by instantiating our generic framework.

Keywords: Transportation, Driver behavior, Last-mile delivery, Machine learning

1. Introduction

By 2025, the number of packages delivered worldwide is expected to climb to 200 billion, compared to less than 90 billion in 2018 (Szczepanski et al., 2021). Coupled with this development is an increased demand for last-mile delivery operations, which is the most expensive part of the supply chain (Seghezzi et al., 2020). In addition, last-mile deliveries substantially impact the satisfaction of customers, who expect fast, flexible, and reliable deliveries (Vakulenko et al., 2019). Overall, planning last-mile logistics requires decision makers to consider both the minimization of costs and the maximization of customer satisfaction.

A common approach to address last-mile delivery problems is drawing on the well-known traveling salesman problem with time windows (TSPTW), for which the literature provides an abundance of mathematical models and algorithms (Gendreau et al., 1998; Ohlmann and Thomas, 2007; Baldacci et al., 2012). However, while TSPTW models and corresponding solutions mainly aim at optimizing traditional logistics criteria, such as the overall travel time (Ohlmann and Thomas, 2007) or the arrival time at the depot (Langevin et al., 1993; López-Ibáñez et al., 2013), they usually do not account for tacit driver knowledge and behavior, which may lead to deviations from

suggested tours. For example, a study by Li and Phillips (2019) on the realized driver tours of a large soft drinks company based in Mexico and the U.S. reveals that drivers did not follow the planned tour in three out of four deliveries. The findings of Samson and Sumi (2019) further support the idea that drivers tend to prioritize their familiarity with routes over the suggested tours provided by their navigation systems after conducting an investigation of the daily commuting habits of drivers in both Japan and the Philippines. However, it should be noted that drivers may deviate from prescribed routes for different reasons. Drivers may have gained knowledge of logistics conditions not included in tour prescription systems; for example, drivers may have collected experience with temporal traffic and parking conditions. When considering such conditions, driver deviations may result in tours that outperform tours prescribed by optimization systems in terms of key organizational metrics, such as tour length, time windows, and customer satisfaction. Hence, fostering this effect is crucial, as drivers' personal knowledge can have a positive impact on organizational success. Despite this, drivers may also choose to deviate from suggested tours based on individual personal preferences, resulting in tours that are inferior to prescribed tours, which should, undisputably, be avoided.

It seems reasonable to assume that both of the abovementioned effects are implicitly included in the set of driven tours recorded. From an empirical perspective, the extent of both effects can only be determined if additional information on why drivers selected their routes is available. However, as recorded data include historical tours driven by many drivers, we assume that (homogeneous) logistics knowledge is shared by many individual drivers and thus implicitly included in many tours. In contrast, individual preferences vary across drivers and are unlikely to lead to consistent deviations in driven tours. Thus, we assume that predictions based on patterns identified in a set of historical tours are more likely to be affected by driver knowledge rather than driver preferences. Therefore, considering predictions based on patterns found in historical tours when making prescriptions is advantageous for organizations. The closer a prescribed tour is to the predicted one, the more likely drivers will adopt it. However, when the deviation between the suggested and the driver's preferred route is significant, drivers are more likely to reject the prescribed tour in favor of their individually preferred routes. This driver behavior has the potential to adversely affect organizational objectives by reducing the overall efficiency and effectiveness of the delivery system. Although, in the absence of empirical data, one can only speculate that the incorporation of patterns from historical data will foster drivers' compliance with suggested tours, it seems reasonable to assume that this effect actually occurs in practice since the patterns are likely to mirror collective logistics knowledge of drivers, which may increase the trust of drivers in the quality of suggested tours and, in turn, the likelihood that drivers will adopt suggested tours.

However, research on driver knowledge and behavior in vehicle routing is scarce. Srinivas and Gajanand (2017) state that studies on driver behavior have constituted a separate body of research for many years and have yet to be integrated into routing prescription, despite the availability of historical data of actually driven tours. Exploiting and integrating such data into last-mile delivery problems lead to complex and large-scale routing problems, which require the development of methods that go beyond classical optimization approaches in different regards: First, the approach requires a multi-stakeholder perspective by considering both an organization's objective to identify cost-minimal tours and the driver's expectation that the suggested route seems reasonable based upon her/his knowledge and experience so that she/he actually adopts the suggested tour; second, it calls for a hybrid-analytical approach by incorporating tour prescription and tour prediction; and third, it requires a hybrid and non-conventional methodology by drawing on both operations research (OR) optimization methods and data science methods.

In this paper, we propose a novel generic methodological decision support framework that accounts for the challenge above. Rather than extending a TSPTW model to a (yet unknown)

"enriched model" that explicitly addresses the variety of reasons, including local conditions and uncertainties, why drivers may deviate from TSPTW solutions in practice (purely prescriptive approach), we adopt a two-step approach that first uses machine learning to predict the "real" route taken by a driver, which would have been found by an "enriched model", and then integrates this prediction into the TSPTW model by adding a constraint which limits the extent of deviation between a suggested tour and the predicted driver tour (predictive-prescriptive paradigm); thereby the abovementioned reasons of tour deviations by drivers are considered implicitly.

The underlying motivation of our hybrid approach is to combine the two epistemologically different approaches of tour prescription, which follows a classical, normative operations research approach using a mathematical model and an exact or heuristic approach, and tour prediction, which follows a behavioral and data-based approach. We constrain the "greediness" of the optimization algorithm by the prediction of the driver's route as a surrogate for enriching the (TSPTW) model with real-life constraints, which are generally challenging to incorporate into classical optimization models. The ultimate purpose of our approach is to suggest tours that are dominant over those produced by a simplistic and unrealistic TSPTW model in terms of both a) higher quality which is achieved under real-life conditions (e.g., in terms of total distances traveled) and b) higher level of acceptance by drivers, which, in turn, increases the likelihood that suggested tours are adopted by drivers. To the best of our knowledge, no behavior-oriented hybrid framework combining machine learning with conventional operations research methods has been suggested in the literature.

Our framework is generic in several regards and can be adapted to suit contextual needs. First, our framework allows using various machine learning methods for tour prediction and different (exact or heuristic) solution methods for prescribing solutions of TSPTW instances. Second, it allows the implementation of different deviation measures between a suggested and a predicted tour. Third, it allows tour planners to control the degree to which driver behavior is considered by adjusting the tour deviation limit.

We demonstrate the applicability of our framework in a case study that draws on real-world logistics data provided in the *Amazon Last-Mile Routing Research Challenge (ALMRRC)* (Merchan et al., 2022). In this case study, we develop a deep learning approach to predict driver tours and employ a variable neighborhood search (VNS) (Wei et al., 2015) to solve the resulting extended TSPTW instances. As a tour deviation measure, we use the Jaro distance (Jaro, 1989) and the longest common subsequence (LCSS) distance (Bergroth et al., 2000). We conduct a sensitivity analysis to investigate the trade-off between the level of deviation between the predicted (driver) tours and the suggested tours and tour quality in terms of costs.

The remaining structure of the paper is as follows: Section 2 gives an overview of related works. Section 3 provides the mathematical problem description. In Section 4, we expose the proposed framework. Section 5 exhibits our experimental case study using real-world data. Finally, we discuss our results and derive managerial implications in Section 6 and conclude the paper in Section 7.

2. Related work

2.1. Tour prescription

First, the literature on last-mile delivery routing provides an abundance of models and methods, and, as a result, it is out of the scope of this article to provide a complete overview of this field. For a review of the general class of vehicle routing problems (VRPs) and variants, see, for example, the works of Braekers et al. (2016); Bräysy and Gendreau (2005); Elshaer and Awad (2020); Toth and Vigo (2014). Yet, in this research, we focus on methods for solving the traveling salesman problem with time windows (TSPTW). The TSPTW consists of finding an optimal tour starting and ending at a given depot and visiting a set of nodes, with the restriction that each node must

be visited within a given time window. Exact approaches and heuristics have been developed to tackle the problem. Christofides et al. (1981) and Baker (1983) develop a branch-and-bound algorithm to solve the TSPTW with up to 50 nodes to optimality. Dumas et al. (1995) apply a dynamic programming approach by taking advantage of the time window constraints to reduce the state space significantly. Dash et al. (2012) partition the time windows into sub-windows and solve the problem with a branch-and-cut algorithm. Baldacci et al. (2012) introduce a dynamic programming algorithm that exploits tour relaxations. Pesant et al. (1998) presents a constraint logic programming model for the traveling salesman problem with time windows which yields a branch-and-bound algorithm.

Heuristics have been intensively applied to instances with large numbers of nodes (> 200) and wide time windows (Ohlmann and Thomas, 2007). Savelsbergh (1985) proposes a local search heuristic and proves that already finding a feasible solution to the TSPTW is an NP-hard problem. Gendreau et al. (1998) develop an insertion heuristic which consists of a construction and post-optimization phase. Calvo (2000) proposes an algorithm that first solves an assignment problem that produces multiple sub-tours, which are then inserted into the main tour using a greedy insertion procedure. A local search procedure is then applied to improve the initial solution. Carlton and Barnes (1996) suggest a tabu-search heuristic that considers infeasible solutions in the search by penalizing late arrivals. Ohlmann and Thomas (2007) extend this penalty approach by incorporating variable time window penalties in a simulated annealing heuristic, named compressed annealing. López-Ibáñez et al. (2013) adapt the compressed annealing approach to the TSPTW where the makespan is minimized. In a study by da Silva and Urrutia (2010), a feasible solution is generated by a variable neighborhood search (VNS) and then improved with a general variable neighborhood search (GVNS) heuristic.

2.2. Driver factors in tour prescription

Driver factors have been incorporated in prescriptive models for variants of the VRP. One direction of research focuses on consistent routing with the aim of increasing driver-customer familiarity. For this purpose, Groër et al. (2009) introduce the consistent vehicle routing problem (ConVRP). Goeke et al. (2019) develop the first exact method for the ConVRP. Kovacs et al. (2015) suggest a multi-objective problem that accounts for driver consistency and arrival time consistency next to the classical objective of minimizing tour costs. Luo et al. (2015) formulate the multi-period vehicle routing problem with time windows and limited visiting quota (MVRPTW-LVQ). This problem requires customers to be served by a limited number of vehicles over the planning horizon. In Ulmer et al. (2020), the authors assume that the customer service time decreases with an increasing number of visits of an individual driver. Another research direction considers historical customer sequences to construct tours. In a recent study, Quirion-Blais and Chen (2021) extend the objective function of a classical vehicle routing problem with time windows (VRPTW) formulation by a term that maximizes the number of historical customer chains in the solution. A construction heuristic is developed to generate a solution for the problem. The method is instantiated with driver data from an online retailer in China. This approach is based on reusing solutions of previously solved similar problems (Rochat and Taillard, 1995; Louis and Li, 2000; Tarantilis and Kiranoudis, 2002).

2.3. Route choice behavior and route prediction

Predictive models mainly consider trip-based route choices, i.e., a driver's decisions when navigating between one origin-destination pair. This type of problem is referred to as the "route choice problem" in the literature. Ben-Akiva et al. (1984) apply a two-stage decision process to model drivers' route choice behavior. In the first step, a set of alternative routes is generated. In the second stage, a multinomial logit (MNL) model is applied to model the drivers' choices. Bekhor et al.

(2006) evaluate different route choice set generation algorithms. Frejinger and Bierlaire (2007) capture the correlation between alternative route choice sets by introducing a subnetwork that simplifies the road network only containing easily identifiable and behaviorally relevant roads. In Frejinger et al. (2009), the authors assume that the choice set contains all paths connecting an origin-destination pair. A cost function assigns a probability to each link between the origin and destination. Higher probabilities are assigned to links with a low distance to the shortest path. In a random walk procedure, links are then added to the path successively, based on the probabilities of the respective link. Fosgerau et al. (2013) introduce recursive models where the route choice is modeled as the sequence of link choices, and no choice set generation is required. Dynamic programming is applied to trace the route with the highest expected utility, i.e., the route that is expected to be chosen. Zimmermann et al. (2017) show the method's advantages by predicting urban bike routes. Mai Anh et al. (2016) propose a decomposition method to reduce the computational complexity of the approach. Oyama and Hato (2017) argue that drivers' choice in networks with real-time traffic is often myopic, i.e., immediate utility is maximized.

2.4. Machine learning in vehicle routing

In recent years, machine learning methods have started making their appearance in the field of combinatorial optimization (CO). In fact, contemporary advancements in deep learning algorithms and architectures have kickstarted research in this OR-heavy discipline and, as a result, enabled the modeling of various vehicle routing problems, such as the TSP or the TSPTW. Machine learning has the benefit of replacing computationally heavy calculations with fast approximations while enabling the decomposition of a given problem into smaller yet more manageable learning tasks (Bengio et al., 2021).

Following the pioneering work by Gambardella and Dorigo (1995), researchers have relied heavily on reinforcement learning (RL) when it comes to solving CO problems, i.e., an approach aiming to maximize cumulative reward by which an agent learns from the interactions with a given environment (e.g., Nazari et al., 2018; Zhang et al., 2020; Li et al., 2021). Even though the algorithms and methods vary widely, most works rely on RL approaches' explorative, adaptive, and generalization capabilities. Moreover, given the defining characteristics of VRPs and their relatively straightforward reward function, RL is, by nature, particularly well suited for such observational tasks (Bello et al., 2017). In point of fact, Nazari et al. (2018) report that their RL approach can find near-optimal solutions for most VRP instances. In contrast, Zhang et al. (2020) expose that their proposed approach outperforms a conventional tabu-search heuristic when applied to a traveling salesman problem with time windows and rejection (TSPTWR). Moreover, most research emphasizes that RL methods are not only reliable but also inherently faster than the more conventional heuristics (e.g., Zhang et al., 2020; Li et al., 2021).

Still employing an RL methodology, various works aimed at solving CO problems further rely on the so-called pointer network architecture (e.g., Bello et al., 2017; Alharbi et al., 2021; Stohy et al., 2021). This approach, i.e., an encoder-decoder architecture capable of sorting sequences of variable lengths, differs from more traditional sequence-to-sequence approaches in which the weights produced by an attention mechanism, as proposed by Bahdanau et al. (2014), act as an indicator, or pointer, to the specific tokens in an input sequence building the ordered output sequence (Vinyals et al., 2015). Models trained using this highly flexible architecture are, according to the authors, even capable of generalizing on sequences longer than they were trained on (Vinyals et al., 2015).

Focusing on the underlying relationships found in geospatial data, researchers have, of late, started using graph neural networks to tackle CO problems (e.g., Li et al., 2018; Joshi et al., 2019; Hu et al., 2021). The natural characteristics of TSP datasets are particularly well suited for 2D representations and, as a result, for such graph-based approaches, i.e., vertex attributes (nodes),

edge attributes (links), and global attributes. In truth, Li et al. (2018) exhibit that their approach based on a graph convolutional network architecture can not only outperform modern deep learning approaches but that it can also perform neck and neck with highly optimized heuristic solvers. Moreover, Joshi et al. (2019) showed that their approach, also based on a graph convolutional network architecture, can dramatically reduce the optimality gap. Lastly, still leveraging the graph structure, Kool et al. (2018) reveal that their approach based on the graph attention network architecture (Veličković et al., 2017) can *"[...] significantly improve over recent learned heuristics"* and outperform *"[...] a wide range of baselines and getting results close to highly optimized and specialized algorithms"* (p.1).

3. Problem description

Our perspective on the problem to be solved includes the determination of tours which show high quality in terms of both a) total distances traveled (in order to save cost and time) and b) the extent of deviations from tours preferred by drivers (in order to achieve a high level of acceptance by drivers). As both goals may conflict, the problem at hand can be perceived as a bi-objective optimization problem, which opens up opportunities to draw on the broad set of modeling and solution approaches available in the multi-criteria field decision making. From those approaches, we decided to model goal a) with an objective function and goal b) with a hard constraint that guarantees that the solution deviates from the preferred route not more than a predefined level. The rationale of our approach lies in our perspective that the organizational goal of minimizing tour length should be focused on by embedding it in the objective function.

From a problem and model perspective, we extend the classical TSPTW with soft time windows; i.e., the violation of time windows is possible yet penalized in the objective function by an additional constraint that considers the deviation of the suggested tour from the predicted tour. We first present the mathematical formulation of our extended TSPTW and then illustrate the tour deviation constraint with an example.

3.1. Mathematical formulation

We draw on the TSPTW formulation of Ohlmann and Thomas (2007). Let $G = (N, A)$ be a finite graph, where $N = \{0, 1, ..., n\}$ is the finite set of nodes and $A = N \times N$ is the set of arcs connecting nodes. There exists an arc $(i, j) \in A$ for every $(i, j) \in N$. A tour is defined by the ordered sequence in which the n customers are visited and denoted by $T = \{p_0, p_1, ..., p_n, p_{n+1}\}$ where p_i denotes the index of the customer in the ith position of the tour. The depot is denoted by customer 0 and every tour begins and ends at the depot, i.e., $p_0 = p_{n+1} = 0$. Each of the remaining n customers is assigned to one position between p_1 and p_n including. For $j = 0, ..., n$, traversing the arc $a_j = (p_j, p_{j+1})$ comes with a cost $c(a_j)$. This cost of traversing the arc between customer p_j and customer p_{j+1} includes the service time at customer p_j and the time needed to travel from customer p_j to customer p_{j+1}. The objective is to minimize the total travel time. The time window of customer i is represented with $[e_i, l_i]$, where e_i is the time window start and l_i is the time window end. The depot does not have time windows. In the formulation by Ohlmann and Thomas (2007), late arrivals are allowed, and the vehicle needs to wait in case of early arrivals. As waiting at the customer's site in case of last-mile delivery routing is unreasonable, we penalize early arrivals with parameter λ_e and late arrivals with parameter λ_l. The variable A_{p_i} represents the arrival time at node i. We extend the TSPTW formulation by adding constraint (5): Let ϕ be a tour deviation measure, T any tour, T' a predicted tour, and δ an upper bound on the allowed deviation between tours T and T'. Then, the constraint ensures that feasible tours only deviate from the predicted tour up to a predefined level of deviation δ. We refer to the resulting, new optimization problem as

the *traveling salesman problem with time windows and deviation (TSPTW-Dev)*; a mathematical formulation of the TSPTW-Dev is presented below.

$$\min \sum_{i=0}^{n} c(a_i) + \sum_{i=1}^{n} [\lambda_l \cdot max(0, A_i - l_i) + \lambda_e \cdot max(0, e_i - A_i)]$$

$$A_{p_i} = A_{p_{i-1}} + c(a_{i-1}) \quad i = 1, ..., n \tag{1}$$

$$A_{p_0} = 0 \tag{2}$$

$$p_0 = 0 \tag{3}$$

$$p_{n+1} = 0 \tag{4}$$

$$\phi(T, T') \leq \delta \tag{5}$$

$$p_i \in \{1, 2, ..., n\} \quad i = 1, ..., n \tag{6}$$

$$p_i \neq p_j \quad i, j = 1, ..., n, \ i \neq j \tag{7}$$

$$A_{p_i} \geq 0 \quad i = 1, ..., n \tag{8}$$

3.2. Illustrative example of the tour deviation constraint

Figure 1 shows a visual example of the tour deviation constraint. Assume that we have a predicted tour T'. A feasible tour T to be suggested is a tour where the tour deviation between T and T' does not exceed δ. If δ equals 0, a suggested tour is not allowed to deviate from the predicted tour, i.e., $T = T'$. Assume that we choose a moderately low value for δ in our example depicted in Figure 1. In tour T, customers B and C of T' are swapped. This results in a feasible tour that slightly modifies the sequence of customers, and $\phi(T, T') \leq \delta$ holds. By contrast, the order of customers in tour T^* largely differs from that in tour T' with $\phi(T^*, T') > \delta$; therefore, tour T^* is infeasible.

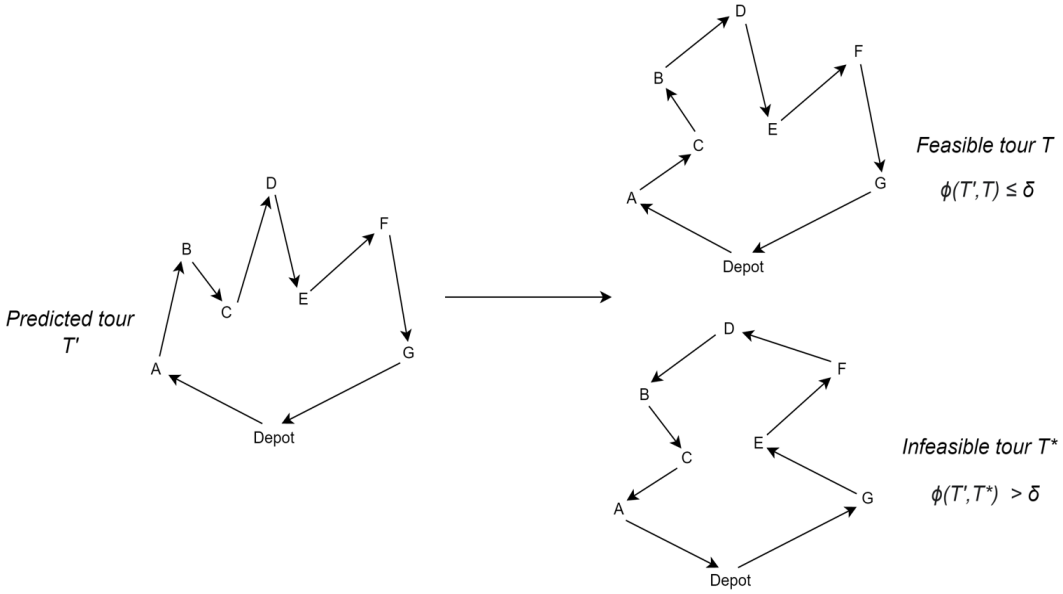


Figure 1: Illustrative example of the tour deviation constraint

4. Decision support framework

Based upon the mathematical formulation of the TSPTW-Dev problem suggested in the preceding section, we now approach the questions of how the prediction of tour T' and the prescription of tour $\overline{T}$ can be accomplished. Our framework provides some degrees of freedom: First, based on the set of historical tour data, an appropriate machine learning algorithm has to be selected. Second, an optimization procedure for solving problem instances needs to be selected based on runtime complexity and the "quality" of solutions when a (meta/mat)heuristic approach is applied. Finally, a tour deviation measure Φ, an upper bound δ, and penalties λ_e and λ_l for violating time windows need to be defined based upon the individual preferences of the decision maker.

The ultimate choices of decision makers need to cover issues of both tour prediction and tour prescription, which are covered in the *learning and configuration phase* of the suggested framework. It precedes the *application phase*, in which problem instances are built based on machine learning and solved using optimization techniques, ultimately leading to the suggested tours.

Accounting for the abovementioned algorithmic variety, it is hardly helpful to suggest one single bundle of machine learning and optimization techniques to predict tours, build TSPTW-Dev instances based upon this prediction, and solve these instances. It appears more suitable to designate a methodological framework (as a general methodology) that outlines the essential steps decision makers must undertake when confronted with TSPTW-Dev problem instances. Any concrete methodology implemented by decision makers can be considered an instantiation of the methodological pattern, which we refer to as a *decision support framework*. An overview of the framework can be retrieved from Figure 2. In the remainder of this section, we first describe the tour prediction methodology and machine learning approach to predict drivers' preferred tours before we unfold how the tour prescription methodology, in terms of optimization, draws on the predicted tour to develop tour suggestions.

4.1. Tour prediction

As exposed, our hybrid decision support framework necessitates a predictive element based on machine learning capable of forecasting how a given set of delivery stops would be visited by a given or an arbitrary driver, i.e., T'. In this case, a tour T' is predicted using either pairs of coordinates, distance or travel-time matrices, or a combination of all three, and, optionally, features related to diverse behavioral and environmental factors, e.g., driver preferences, weather forecasts, road works, or traffic conditions.

Following the traditional machine learning pipeline depicted in Figure 2, the goal in this initial learning and configuration phase is to train a model, based on historical data, capable of predicting the most-likely sequence of delivery stops. Given the generic nature of the presented approach, any architecture capable of either making discrete classifications (sequential) or sorting variable-sized sequences, such as pointer networks (Vinyals et al., 2015), can be employed. This training stage is crucial to the success of the presented framework and should be thoroughly assessed. In fact, our framework emphasizes that a predefined performance threshold, which should be defined by domain experts using a task-specific test set, should first be met before moving on to the application phase, in which the trained model can be deployed to predict tours (T') on unseen data. Such a performance threshold might involve a comparison to a heuristic approach or state-of-the-art machine learning approaches.

4.2. Tour prescription

Based upon the mathematical problem formulation suggested above, decision makers need to configure and instantiate the optimization model in several regards. In order to build the tour deviation constraint (5), i.e., $\phi(T, T') \leq \delta$, one needs to determine a tour deviation measure ϕ, which

reflects the similarity between a predicted tour T' and any tour T. Several measures have been suggested in the literature for strings and sequences, including the Levenshtein distance (Levenshtein, 1966), the Jaro distance (Jaro, 1989), and the longest common subsequence (LCSS) distance (Bergroth et al., 2000). Such measures are helpful in our context when tours are represented as strings or sequences of nodes visited in the order they appear in the string or sequence. Assuming without loss of generality that low values of $\phi(T', T)$ indicate high similarities between tours T' and T, possible deviation values need to be defined in terms of an upper bound δ, which indicates the extent with which a suggested tour may deviate from a predicted tour, thereby indicating the level of importance with which (predicted) preferences of drivers are considered during the development of tour suggestions.

Further parameters λ_e and λ_l need to be specified which penalize time window violations (early and late arrivals, respectively) in the objective function. In one extreme scenario, decision makers require all time windows to be adhered to, leading to substantial values of λ_e and λ_l, which prevent any violations of time windows. In many other scenarios, time windows may be violated and are considered "soft"; then, the effect of violations on the objective function can be set by fixing appropriate λ_e and λ_l values. In this regard, the suggested framework as well as the mathematical model are flexible.

Having defined the value of λ_e and λ_l, a TSPTW model instance can be built using problem data from an instance database. Adding the tour deviation constraint, based upon the predicted tour T', the chosen deviation measure ϕ , and the upper bound of deviation δ, finally yields the TSPTW-Dev model instance I to be solved by an optimization algorithm, leading to the suggested tour $\overline{T}$.

Solving instance I can be accomplished by drawing on a plethora of optimization algorithms, including meta-heuristic approaches. Often, the selected optimization algorithm needs to be configured in various ways. For example, many meta-heuristics, such as tabu-search, or variable neighborhood search, require the definition of one or more problem-specific local neighborhoods, which are used in the search process controlled by the meta-heuristic. The application of the configured optimization algorithm on the problem instance I finally results in the suggested tour.

5. Case study

In this section, we demonstrate the suggested decision support framework through its application on the problem case of the *Amazon Last-Mile Routing Research Challenge (ALMRRC)* (Merchan et al., 2022). For tour prediction, we choose a recursive logit-modeling approach in combination with a feedforward neural network; for tour prescription, we draw on a variable neighborhood search (VNS) meta-heuristic to solve TSPTW-Dev instances. In our computational experiments, we use an Intel Core i9 processor with 3.6 GHz and 64 GB RAM. The neural network is implemented with Pytorch (Paszke et al., 2019), and the VNS is implemented in Python.

In the first part of this section, we describe our instantiation of the framework in detail, addressing the used dataset, the logit-modeling and neural network approach, the VNS meta-heuristic, and selected sequence deviation measures for measuring tour deviations. In the second part, we present the results of our computational evaluation.

5.1. Framework instantiation

5.1.1. Data

We draw on real-world data provided in the *ALMRRC* (Merchan et al., 2022). The dataset consists of 6,112 historical driver tours in a last-mile delivery context and was collected between July and August 2018 in five metropolitan areas in the U.S. The dataset consists of realized driver

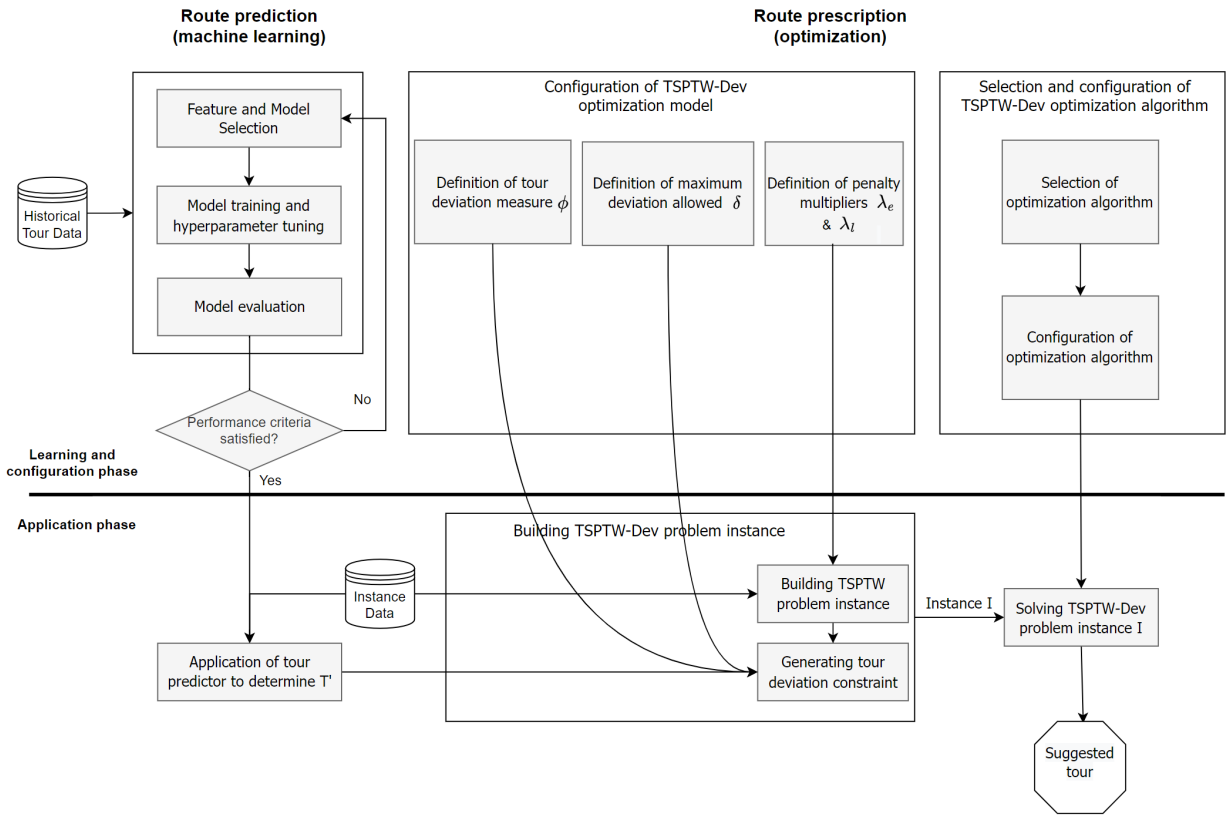


Figure 2: Decision support framework

tours; it does not include the tours as initially planned. In each tour, 148 customers are served on average with a standard deviation of 31. A route took, on average, 8.1 hours, of which 3.6 hours were spent on transit and 4.5 hours on service. Beyond driver tours, the dataset also includes travel times between stops, the latitude and longitude of each stop, and a delivery time window, which has an average length of nine hours and a standard deviation of 2.7 hours. For further information on the dataset, we refer to Merchan et al. (2022).

Amazon decomposes the set of customers by using customer clusters: customers belonging to the same cluster need to be served in a sequence before customers of another cluster are visited. On average, the clusters have 7.3 customers. In our formulation of the TSPTW-Dev described in Section 3, this requirement can be considered by adding a large constant M to the costs of an arc which connects customers in distinct clusters, leading to the adherence of customer clusters in any optimal tour.

5.1.2. Driver tour prediction

Different than the route choice problem presented in the related work section of this paper, in our case, a sequence of stops needs to be predicted. We draw on the myopic link choice model by Oyama and Hato (2017). In every iteration, a selection probability is assigned to every unvisited destination. We assume that, in each iteration, the driver chooses the destination with the highest probability. This process is repeated until no unvisited destination exists. We divide the prediction model into two phases: In the first phase, hereafter referred to as *cluster phase*, the sequence of customer clusters is predicted. In the second phase, hereafter referred to as *customer phase*, the sequence of customers in these clusters is predicted. The features applied in the cluster phase are presented in Table 1.

Table 1: Features to predict next cluster chosen by driver

No.	**Feature**
1	Average travel time and geographical distance from current cluster to candidate cluster
2	Average travel time and geographical distance from the candidate cluster to not yet visited clusters
3	Average travel time and geographical distance from current cluster to the depot
4	One-hot encoded vector representing opening hours
5	Percentage of customers already visited

The one-hot encoded time window vector consists of 24 binary entries, each representing if a cluster can be served in that hour (1) or not (0). In each cluster, we define the opening time as the time between the latest start and the earliest closing of each individual customer in that cluster. In case no time window is specified, the vector solely consists of 1's.; i.e., customers in that cluster can be served at any time. The features applied in the customer phase are presented in Table 2.

Table 2: Features to predict next customer chosen by driver

No.	Feature
1	Travel time and geographical distance from the current stop to the candidate stop
2	Average travel time and geographical distance from the candidate stop to not yet visited stops in current cluster
3	Average travel time and geographical distance from the candidate stop to not yet visited stops in next cluster
4	Travel time and geographical distance from the candidate stop to its closest stop in next cluster

In both phases, a feedforward neural network is applied as prediction architecture, and min-max normalization is applied to scale all features in each iteration, as suggested for neural approaches by Basheer and Hajmeer (2000). For the sake of simplicity, the feedforward neural network contains three hidden layers and rectified linear (ReLU) activation functions. The output layer consists of a sigmoid function, as we deal with multiple binary predictions in every decision iteration, enabling the selection of the destination with the highest probability. As a loss function, we apply the Binary Cross Entropy loss, which takes the following form:

$$BCELoss = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i) \tag{9}$$

We tune the hyperparameters of the network, such as the hidden layer sizes, learning rate, and batch size, with the ASHA algorithm (Li et al., 2020) in both phases separately. The hyperparameters can be retrieved from Appendix A.

5.1.3. Variable neighborhood search

To solve TSPTW-Dev instances, we draw on the VNS meta-heuristic, which has been successfully applied to routing problems, including the VRPTW (Zhang et al., 2021). In particular, we use an approach for the capacitated vehicle routing problem with two-dimensional loading constraints suggested by Wei et al. (2015).

Our VNS is presented in pseudocode formulation in Algorithm 1. As the VNS is an improving (meta-)heuristic, it requires having an initial tour solution S as input; for S, we use the tour T' as predicted through machine learning (see Figure 2). Prior to executing the search, the VNS also needs to define a set of H different neighborhood structures NS_h ($h = 1, \ldots, H$); a definition of $H = 3$ neighborhood structures is provided below. In an outer loop, the VNS conducts an iterative search for improving tour solutions, terminating upon the execution of *maxNonImp* consecutive iterations without any improvement; in our pretests, no improvement has been made after approximately 30 iterations, and, therefore, we set the value of $maxNonImp$ to 30. For the current tour solution S, in an inner loop iterating over all neighborhood structures NS_h ($h = 1, \ldots, H$), the VNS proceeds as follows: it randomly generates a neighbored solution S' from the current neighborhood h. The tour solution S' is then improved with a local search procedure, resulting in tour solution S''. The local search procedure consists of applying the full enumeration of the three local search operators described below. If the new solution S'' is better than our best-known solution S^*, the VNS updates the best solution and our current solution S before restarting the inner loop with neighborhood

$h = 1$. Otherwise, the next neighborhood $h+1$ is applied analogously. If all neighborhood structures have been applied without improving the solution, the VNS diversifies the current best solution S^*, and a new iteration of the outer loop starts.

Algorithm 1 Variable Neighborhood Search

```
Input: S
Define a set of neighborhood structures NS_h (h = 1, ..., H)
S* = S, nonImp = 0
while nonImp < maxNonImp do
    nonImp = nonImp + 1
    h = 0
    while h < H do
        h = h + 1
        Generate a random neighboring S' of S using NS_h
        S'' = LocalSearch(S')
        if S'' is better than S* then
            Set S = S'', S* = S''
            h = 0, nonImp = 0
        end if
    end while
    S = Diversify(S*)
end while
return S*
```

We employ the three neighborhood structures and local search operators as suggested by Zhang et al. (2021). While all neighborhood structures employ the reordering of customer clusters, in the local search procedure, the order of clusters remains unchanged, but the sequence of customers within the clusters is adjusted. The neighborhood structures and local search operators are visualized in Figure 3. The neighborhood structures are defined through tour solutions that result from *cluster relocate*, where a cluster is relocated to a different position, *cluster swap*, which exchanges the position of two clusters, and *cluster 2-opt*, in which a sequence of consecutive clusters is reversed.

The three local search operators work based upon the same principles but are applied to customers in a single cluster rather than to clusters themselves: *Relocate* moves a customer to a different position in the same cluster, *swap* exchanges the position of two customers within one cluster, and *2-opt* reverses the sequence of consecutive customers within one cluster. Before the local search starts, the order of operators is shuffled. The search then evaluates all possible moves (in every cluster) applying the first operator. If an improvement is made, it is accepted immediately, the order of operators is shuffled, and the search procedure restarts. If no improvement is made in any of the clusters based on the current operator, the next operator is applied to all clusters. The local search terminates when all possible moves of all three operators do not yield a better solution in any of the clusters.

The diversification procedure follows a ruin-reconstruct approach. In the ruin phase, a number of clusters are removed from the tour and added to a pool. In the reconstruct phase, the removed clusters in the pool are re-inserted into the tour. The insertion is accomplished by iterating over a) all clusters in the pool and b) all possible positions in the tour and then choosing the cluster-position pair, which results in the lowest additional costs per added stop. The inserted cluster is then removed from the pool, and the next iteration begins. The procedure finishes when the pool of

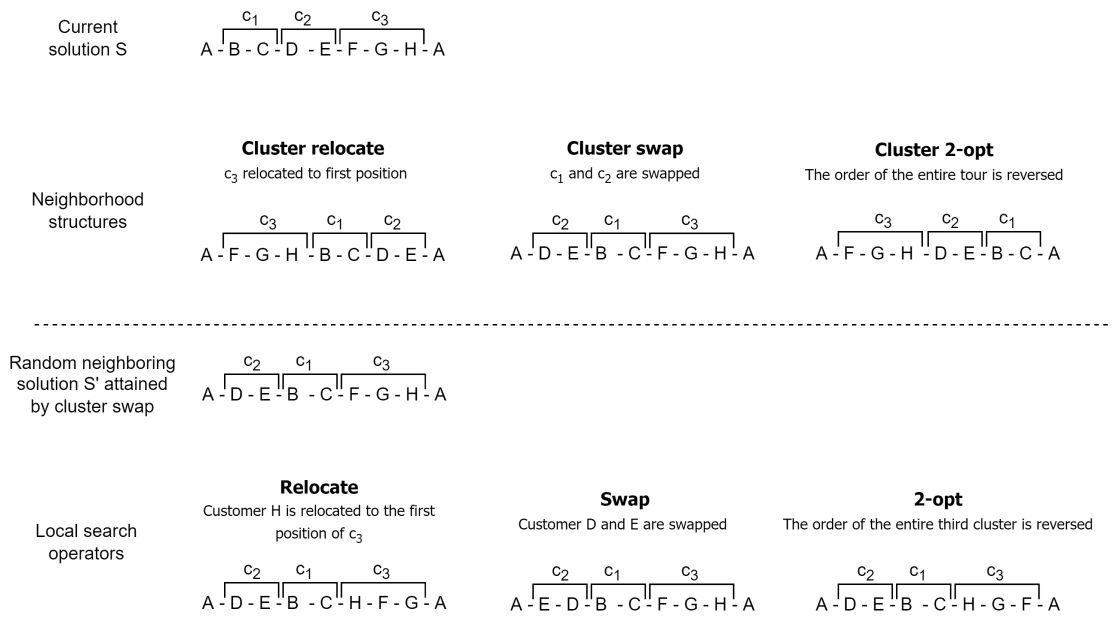


Figure 3: Neighborhood Structure and Local Search Operators

removed clusters is empty. In the implementation of Wei et al. (2015), the degree of destruction (in terms of the number of clusters removed) depends on the number of iterations without improvement $nonImp$, and is set to $min\{0.5 \cdot N, 0.1 \cdot N + nonImp\}$, where N is the total number of stops in a complete tour. In our implementation, the degree of destruction additionally depends on the parameter value of the deviation limit δ. Based on pretests, in our implementation, we set the number of removed clusters to $\lfloor min\{min\{\delta, 0.5\} \cdot N, 0.05 \cdot N + nonImp\}\rfloor$, where N is the total number of clusters and $\delta \in [0, 1]$. Our setting implies that the number of clusters removed in the ruin phase and re-inserted in the reconstruct phase is small when both $nonImp$ and δ are small and when not more than 50% of the clusters are removed.

5.1.4. Sequence deviation measures

In order to measure the deviation between two tours, we apply, as briefly exposed earlier, two alternative deviation measures Φ: the Jaro distance (Jaro, 1989) and the LCSS distance (Bergroth et al., 2000; Sevaux et al., 2005). The Jaro distance is defined as the inverse of the Jaro similarity (Jaro, 1989), a metric that measures the similarity between two sequences. The metric has been developed for record linkage, which deals with linking records of the same entity within data sources. We choose to apply this metric for two reasons: First, in contrast to other measures, such as the Levenshtein distance (Levenshtein, 1966), the Jaro distance not only considers the number of edits but also the relative positions of the stops. Second, in the record-linkage literature, good results have been obtained based upon this measure (Cohen et al., 2003). Hence, for the following case study, the Jaro distance between two tours t_1 and t_2 with an identical number of stops can be defined as

$$JaroDistance(t_1, t_2) = 1 - \frac{1}{3}(\frac{2m}{n} + \frac{m - trans}{m}),$$

where n is the number of stops, m is the number of matching stops between the tours and $trans$ is the number of needed transpositions to convert the matching stops of one tour into the other.

A stop is considered matching, if its position in the first and second tour does not differ by more than $(\frac{n}{2} - 1)$. The measure is bounded between $[0, 1]$, where 0 indicates the presence of two identical tours. Consider the tour $t_1 = A - B - C - D - E - F - G - H - A$ and a second tour $t_2 = A - G - H - D - E - F - B - C - A$, in which the customers pairs B-C and G-H have been swapped. Stop G is at the 7th position in t_1 and at the second position in t_2. As $|7 - 2| \geq \frac{9}{2} - 1$, stop G is not considered matching. The same procedure applies to stops H, B and C. Stops $A - D - E - F - A$, in turn, are considered matching, so $m = 5$. As the order in which these matching stops appear in both tours is identical, no transpositions are needed. The Jaro distance between these two tours is therefore given by:

$$JaroDistance(t_1, t_2) = 1 - \frac{1}{3}(\frac{2 \cdot 5}{9} + \frac{5 - 0}{5}) = 0.296.$$

The longest common subsequence distance is based on the longest common subsequence (LCSS), which is widely used to measure the closeness of sequences in various disciplines (see Bergroth et al. (2000)). The LCSS is defined as the maximum number of identical symbols in two strings, preserving the symbol order (Bergroth et al., 2000). The LCSS distance is defined as n minus the length of the LCSS, divided by $(n - 1)$ (Sevaux et al., 2005), where n is the length of the sequence:

$$d_{LCSS}(t_1, t_2) = \frac{n - |LCSS(t_1, t_2)|}{n - 1}.$$

Similar to the Jaro distance, the LCSS distance is bounded between [0,1]. Reconsider the tours t_1 and t_2 from the previous example. The longest subsequence that is shared by both tours is $A - D - E - F - A$ with a length of 5. The LCSS distance between the two tours is therefore

$$d_{LCSS}(t_1, t_2) = \frac{9 - 5}{9 - 1} = 0.5.$$

We remark that both the Jaro and LCSS deviation measures do not consider the geographical distance of customers, i.e., the distance between two customers is disregarded when transposing one (sub-) sequence to another. A driver might experience that a swap of customers might change his tour more significantly when customers are further away from each other, compared to a swap of customers who are close to each other. However, to the best of our knowledge, sequence deviation measures that consider the distance between customer stops do not exist in the literature. Future research could develop such measures, e.g., through weighting transpositions by the absolute change in tour length the transposition would cause. In the scope of this paper, we rely on the above mentioned deviation measures from literature, that have been proven useful in various disciplines (Bergroth et al., 2000; Cohen et al., 2003).

5.2. Computational study

5.2.1. Tour prediction results

In this section, we look at our machine learning model's predictive accuracy and the Jaro distance between predicted and observed tours. For our computational study, we split our data into a training set and a test set comprised of 5,112 and 1,000 samples, respectively. The confusion matrices depicted in Table 3 and 4 summarize the prediction results of both the cluster and customer predictions. We benchmark the predictions with a nearest neighbor strategy in which the closest customer in the same cluster as the current customer is visited next. If all customers of a cluster have been visited, the nearest customer, disregarding the cluster, is visited next. Note that the number of false positives and false negatives are equal, as a wrong prediction logically leads to both a false positive and a false negative. The data is unbalanced for the following reason: Given that

there are n clusters in the first iteration, the driver can choose from n clusters, among which only one is finally chosen. In the second iteration, the driver can choose from $(n-1)$ clusters among, again, only one is chosen, and so on. Therefore, there are more "not chosen" clusters than chosen clusters. The same logic applies to the customer phase.

Table 3: Confusion matrices cluster phase

(a) Neural network

	Predicted	
Actual	Chosen	Not chosen
Chosen	14,141	3,854
Not chosen	3,854	182,646

(b) Nearest neighbor

	Predicted	
Actual	Chosen	Not chosen
Chosen	12,882	5,113
Not chosen	5,113	181,217

Table 4: Confusion matrices customer phase

(a) Neural network

	Predicted	
Actual	Chosen	Not chosen
Chosen	99,592	25,485
Not chosen	25,485	463,917

(b) Nearest neighbor

	Predicted	
Actual	Chosen	Not chosen
Chosen	94,457	30,620
Not chosen	30,620	458,782

In our case, it is especially interesting to look at the sensitivity of the predictions, i.e., the probability that we correctly predict the chosen cluster/customer. In the cluster phase, the sensitivity, or recall, is 0.786 for the neural approach, compared to 0.716 for the nearest neighbor strategy. In the customer phase, the neural approach yields a sensitivity of 0.794 compared to 0.755 for the nearest neighbor approach. The neural approach clearly outperforms the nearest neighbor strategy in both phases, showing that the selected features have predictive power and are, as a result, explanatory of driver behavior. Our approach slightly outperforms the best neural network approach submitted to the *ALMRRC*, i.e., an accuracy of 77% on predicting the next chosen customer (Huang et al., 2021). This benchmark serves as our predefined performance threshold and proves the acceptability and usability of the proposed model.

Table 5: Tour prediction results

(a) Predicted tours

	Mean	Median	Std
Jaro	0.306	0.274	0.151
LCSS	0.703	0.716	0.144

(b) Nearest neighbor tours

	Mean	Median	Std
Jaro	0.317	0.312	0.126
LCSS	0.729	0.741	0.119

Figures B.5 and B.6, which can be found in the Appendix, show the distribution of the Jaro and LCSS distances between predicted tours and driver tours in the test set, respectively. Summary statistics are presented in Table 5. The mean and median Jaro distances are 0.306 and 0.274, respectively. The standard deviation is 0.151. The mean and median LCSS distances amount to 0.703 and 0.716, respectively. The standard deviation is 0.144. Figures B.7 and B.8 show the distribution of the Jaro and LCSS distances between the nearest neighbor tours and driver tours in the test set. The mean and median Jaro distance is 0.317 and 0.312, respectively. The standard

deviation is 0.126. The mean and median LCSS distance is 0.729 and 0.741, respectively. The standard deviation is 0.119. These results confirm that the neural network approach outperforms the nearest neighbor approach.

5.2.2. Tour prescription results

As the decision support framework shown in Figure 2 indicates, the instantiations of TSPTW-Dev model instances allow but also require the ex-ante determination of the tour deviation measure Φ, the maximum deviation allowed δ, and penalty multipliers λ_e and λ_l. In our casestudy, we assume that early and late arrivals lead to similar extents of customer dissatisfaction (e.g., caused by failed deliveries) and thus, we apply equal values for λ_e and λ_l by setting them to a single value λ. In our experiments, we used and combined various measures and parameter values of δ and λ. For δ, we choose the set of values $\{0, 0.04, \ldots, 0.96, 1\}$ as both the Jaro distance and the normalized LCSS distance are bounded between 0 and 1. Regarding λ, Guo and Mak (2004) choose the values of 1, 10 and 100. We extend this set by choosing values of $\{0, 0.5, 1, 5, 10, 20, 50, 100\}$.

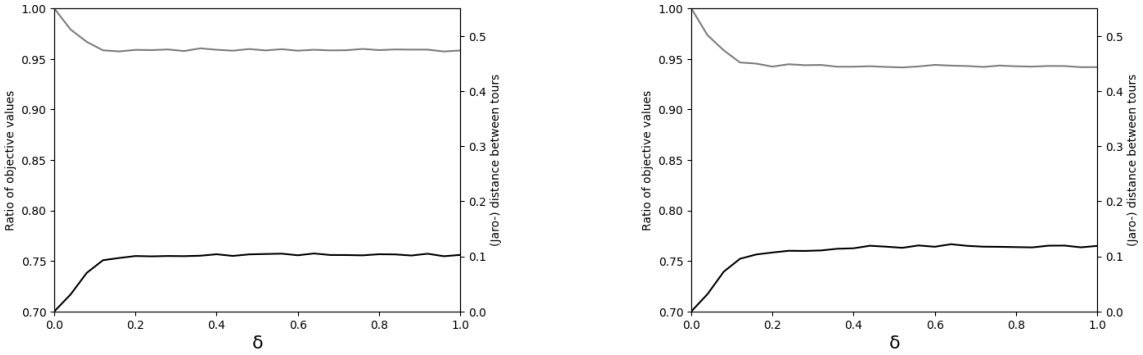
Figure 4 shows, for the Jaro distance, how the objective values of tours suggested by our VNS meta-heuristic relate to the objective values of driven tours as predicted by our machine learning algorithm. The grey line represents the ratio between the suggested tour's objective value and the predicted tour's objective value. All ratios are averaged over the test data of $1,000$ historical driver tours. The black line represents the distance between the suggested and predicted tours. Analogously, the averaged (Jaro) distances between suggested and predicted tours are plotted. The plots for the normalized LCSS distance, as well as detailed values of our results, can be retrieved from Figure C.9 and Tables C.7-C.10 in Appendix C.

6. Discussion

6.1. Analysis of computational results

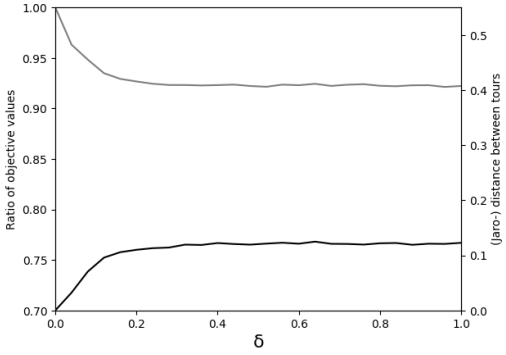
The parameterization of the TSPTW-Dev model allows controlling the extent to which the violations of time windows are penalized and to which feasible tours may deviate from (predicted) driven tours. For the various selected values of parameters λ and δ, we thus analyze the relative improvements of the quality of suggested tours over driven tours (referred to as "the ratio") and similarities between both types of tours using deviation measures. While the former metric refers to an organization's objective to minimize tour costs, the latter addresses drivers' adoption of suggested tours. Our evaluation of both metrics across sets of parameter values allows for determining the sensitivity of the parameters.

Figure 4 reveals that, across all values of λ, the ratios decrease, and the Jaro distances increase with increasing values of δ. The decrease of ratios – and thereby the improvement of tour qualities – is reasonable as larger values of δ allow larger deviations between suggested tours and driven tours, which, in turn, corresponds to larger feasible regions (see constraint (5) in the mathematical formulation of the TSPTW-Dev). The results show a steep decrease of ratios for relatively small values of λ, which indicates that even low deviations from driven tours improve tour quality. However, the curve of ratios immediately flattens, for all values of λ, when δ exceeds $\delta^* \approx 0.15$. This effect means that any further improvements of tour qualities can only hardly be achieved when tours may substantially deviate from driven tours, i.e., the shadow price of constraint (5) gets close to zero when $\delta > \delta^* \approx 0.15$. Apparently, the qualities (in terms of costs) of driven tours, as predicted by the employed machine learning approach, are already reasonably good. Interestingly, δ^* does not depend on how strong time window violations are penalized, i.e., δ^* does not depend on λ. However, the extent of tour improvements substantially depends on λ: for the minimum value $\lambda = 0$, time window violations are not penalized, and the original TSPTW degenerates to the TSP,

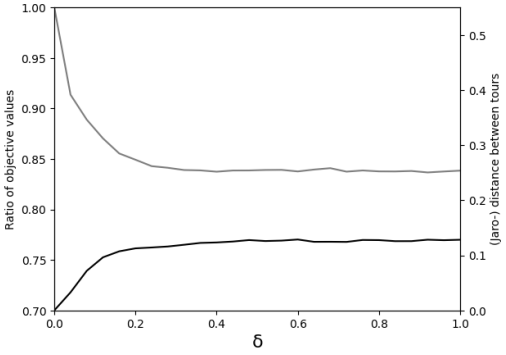


(a) $\lambda = 0$

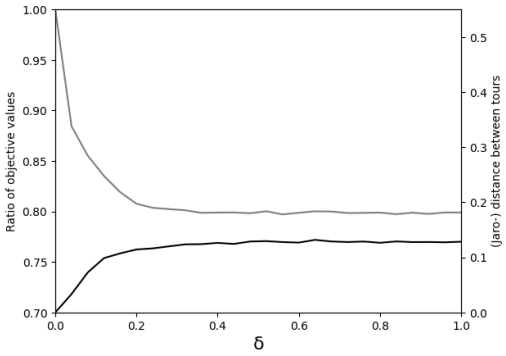
(b) $\lambda = 0.5$



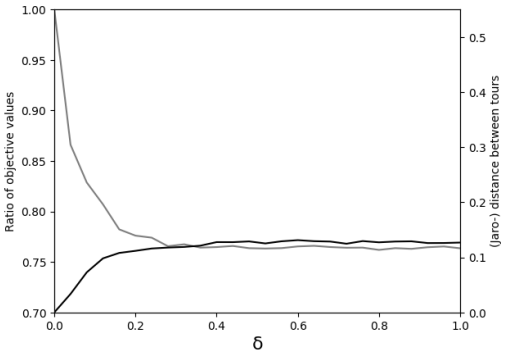
(c) $\lambda = 1$



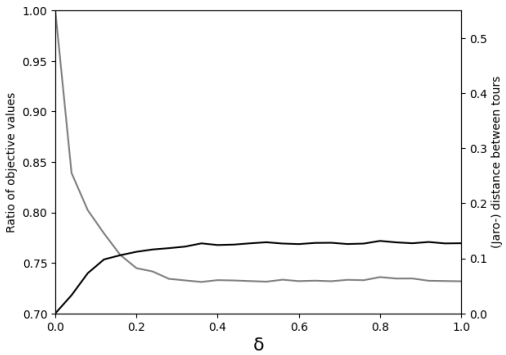
(d) $\lambda = 5$



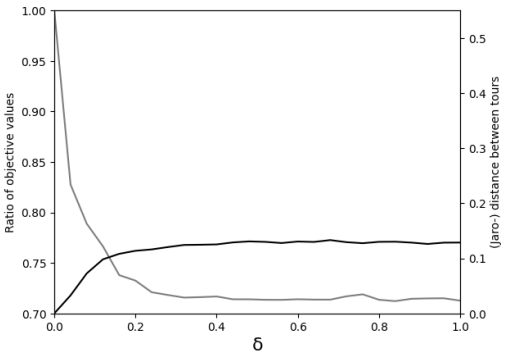
(e) $\lambda = 10$



(f) $\lambda = 20$



(g) $\lambda = 50$



(h) $\lambda = 100$

Figure 4: Relationships between suggested tours and (predicted) driven tours - Jaro distance

which is easier to solve. Thus, it is not surprising that the driven tours are of high quality and can be improved by the VNS meta-heuristic only slightly by up to 4%. With increasing values of λ, the relevance and impact of time window violations also increase, and obtaining good tour solutions becomes more challenging as time windows need to be considered in tour planning to an increasing extent. For example, for $\lambda = 100$, time window constraints become almost "hard constraints", and applying the VNS meta-heuristic leads to an improvement of tour quality of about 28%. Overall, our computational results indicate that, in particular, for high values of λ, applying optimization is substantially beneficial.

The development of (Jaro) distance values can be explained similarly. With increasing values of δ, for all values of λ, deviations of feasible tours from driven tours may become more significant, resulting in increasing (Jaro) distance values. Unsurprisingly, the distance curve starts flattening at $\delta^* \approx 0.15$, the value at which the ratio curve starts flattening. This phenomenon is reasonable to expect as increasing the allowed extent of deviations from driven tours beyond δ^* does not result in yielding tours of increasing quality so that the distances between tours suggested by the VNS meta-heuristic and driven tours can be expected not to increase either. However, and in contrast to the development of ratios, the (Jaro distance) values to which the distance curves "converge" do not depend on the value of λ, i.e., the extent to which time window violations are penalized do not affect the maximum level of dissimilarity between suggested tours and driven tours.

Substituting the Jaro distance with the normalized LCSS distance results in similar observations, as Figure C.9 shows. The effects described above hold again, but δ^* is different as the dissimilarity between tours is measured differently. In particular, for all values of λ, the extent of improvements of tour qualities through the application of the VNS meta-heuristic are very similar to those achieved when using the Jaro distance. Overall, our results are robust against the selected deviation measure.

6.2. Managerial implications

With our case study, we not only demonstrate the instantiation and application of the suggested decision support framework, but we also point to the general domain and algorithmic issues that need to be considered by organizational decision makers when implementing a decision support system for last-mile logistics.

While our framework does not limit the application of machine learning and optimization algorithms, it leaves algorithm selection and configuration to the decision maker. The choice may depend on the algorithmic experience, the available tools, and the prediction and prescription quality of the algorithms when applied to the tour data at hand. The portfolio of results for different parameter values λ and δ provides insights into the quantitative effects that selected penalties of time window violations and maximum deviations from predicted tours have on quality improvements over and on actual deviations of suggested tours from predicted tours. Although we do not expect the selected deviation measure to impact the trade-off between tour improvements and deviations significantly, we recommend that decision makers include variations of measures in their experiments. Of course, the selected deviation measures affect the value of δ^*.

Decision makers need to determine the extent to which time window violations are penalized (fixing the value of λ). In some cases, the importance of avoiding time window violations may be evident and exogenously given; for example, when time windows do not play an essential role, λ may be set to zero, and when time windows need to be adhered to, λ may be set to a value which is prohibitively large to allow any time window violations.

In other cases where the parameter λ can be set to any value in a given range, the results of computational experiments may advise decision makers on how to parameterize λ.

One key decision that needs to be made prior to implementing a decision support system is related to the abovementioned trade-off between tour improvements and deviations from predicted

tours. On the one hand, allowing large deviations from predicted tours may improve tour quality, thereby reducing tour costs. On the other hand, such large deviations may reduce the probability that drivers accept and actually follow the tour suggestion, thereby marginalizing the impact of tour suggestions on actually driven tours. The ultimate decision on how to address this trade-off (by parameterizing δ) depends on the decision maker's preferences. Assuming that the curve of tour quality improvements starts flattening at δ^*, it is reasonable to set $\delta \leq \delta^*$ for two reasons: first, substantial tour improvements may not be expected for $\delta > \delta^*$ but (at least slight) increases of tour deviations may occur; second, with increasing values of δ, the solution space increases, which often reduces the efficiency of optimization algorithms. Conducting extensive computation experiments supports decision makers in fixing the value of δ according to their preferences.

7. Conclusions

From an organizational perspective, logistics providers usually draw on optimization approaches to plan last-mile operations, aiming at minimizing travel times/ distances or makespans. At the same time, drivers may derive from tour suggestions based upon their knowledge of temporal traffic and parking conditions, customer preferences, and safety conditions, for example. Literature on both organizational and behavioral issues of vehicle routing exists, but they are largely disconnected. With our study, we aim to bridge this gap by proposing a hybrid decision support framework that enables the simple integration of driver behavior into the optimization process. This framework integrates a machine learning approach to analyze and predict driver behavior based on historical tour data with an optimization approach that suggests tours by considering the predicted driven routes. From a modeling perspective, we suggest a mathematical formulation of a modified TSPTW model, the TSPTW-Dev model. From an algorithmic perspective, our framework is generic and allows the use of various machine learning and optimization approaches.

We illustrate the instantiation and application of the suggested framework by implementing a deep learning approach to predict driver tours and a VNS meta-heuristic to suggest tours, drawing on real-world tour data. We conduct extensive computation results and a sensitivity analysis to demonstrate the effects of various parameter values of the TSPTW-Dev. We use our insights gained in the computational case study to derive managerial implications beyond our analysis and target all logistics providers that strive to exploit historical data for planning their last-mile logistics.

With our computational study, we quantify the effects of applying an optimization approach to the predicted tour; that is, our analysis relates the objective value of a suggested tour to the objective value of the predicted tour. Decision makers may also be interested in quantifying the (empirical) effects of considering predicted tours (and thus driver knowledge) in the prescription of tours. In order to analyze this effect and thus the empirical value of our suggested framework, an empirical study would need to track and compare actually driven tours under two different conditions: while condition (1) assumes that drivers receive suggested tours based on an optimization approach that does not consider predicted tours, condition (2) assumes that predicted tours are included in tour suggestions as implemented with our framework. Conducting this empirical analysis would allow determining whether and to what extent actually driven tours based upon condition (2) are superior (e.g., in terms of tour characteristics, customer satisfaction, adoption of tour suggestions by drivers) over actually driven tours based upon condition (1) due to the incorporation of driver knowledge. However, this empirical analysis needs to be conducted in an empirical follow-up study that goes far beyond our computational study.

Our research has several limitations which open avenues for future research. First, we have modeled the partly conflicting goals of tour cost minimization and closeness to predicted tours by considering the latter as a constraint. We have yet to analyze other multi-objective optimization

approaches, such as using multiple-objective definitions. Future research should investigate such approaches and compare their appropriateness with our approach. Second, we have implemented a single deep learning approach and a single meta-heuristic. Future work may develop and evaluate other machine learning and optimization approaches. Third, driver-related features and environmental factors, such as road and traffic conditions, should also be considered for the training and optimization steps. Fourth, we have instantiated and evaluated the framework in a single case study. In future work, other data sets should be investigated. These investigations may disclose relationships between the structure and volume of tour data and the appropriateness of selected machine learning and optimization algorithms. Finally, the empirical value of our suggested framework, as discussed in the preceding paragraph, should be investigated.

References

Alharbi, M.G., Stohy, A., Elhenawy, M., Masoud, M., El-Wahed Khalifa, H.A., 2021. Solving traveling salesman problem with time windows using hybrid pointer networks with time features. Sustainability 13, 12906.

Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

Baker, E.K., 1983. An exact algorithm for the time-constrained traveling salesman problem. Operations Research 31, 938–945. URL: http://www.jstor.org/stable/170895.

Baldacci, R., Mingozzi, A., Roberti, R., 2012. New state-space relaxations for solving the traveling salesman problem with time windows. INFORMS Journal on Computing 24, 356–371. URL: https://doi.org/10.1287/ijoc.1110.0456, doi:10.1287/ijoc.1110.0456, arXiv:https://doi.org/10.1287/ijoc.1110.0456.

Basheer, I., Hajmeer, M., 2000. Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods 43, 3–31. URL: https://www.sciencedirect.com/science/article/pii/S0167701200002013, doi:https://doi.org/10.1016/S0167-7012(00)00201-3. neural Computting in Micrbiology.

Bekhor, S., Ben-Akiva, M.E., Ramming, M.S., 2006. Evaluation of choice set generation algorithms for route choice models. Annals of Operations Research 144, 235–247. URL: https://doi.org/10.1007/s10479-006-0009-8, doi:10.1007/s10479-006-0009-8.

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2017. Neural combinatorial optimization with reinforcement learning. arxiv (jan. 2017).

Ben-Akiva, M., Bergman, M., Daly, A., Ramaswamy, R., 1984. Modelling inter urban route choice behaviour, in: Proceedings of the 9th International Symposium on Transportation and Traffic Theory, VNU Press, Utrecht. pp. 299–330.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research 290, 405–421.

Bergroth, L., Hakonen, H., Raita, T., 2000. A survey of longest common subsequence algorithms, in: Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000, pp. 39–48. doi:10.1109/SPIRE.2000.878178.

Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I., 2016. The vehicle routing problem: State of the art classification and review. Computers & Industrial Engineering 99, 300–313.

Bräysy, O., Gendreau, M., 2005. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. Transportation science 39, 104–118.

Calvo, R., 2000. A new heuristic for the traveling salesman problem with time windows. Transportation Science 34, 113–124. doi:10.1287/trsc.34.1.113.12284.

Carlton, W.B., Barnes, J.W., 1996. Solving the traveling-salesman problem with time windows using tabu search. IIE Transactions 28, 617–629. URL: https://doi.org/10.1080/15458830.1996.11770707, doi:10.1080/15458830.1996.11770707, arXiv:https://doi.org/10.1080/15458830.1996.11770707.

Christofides, N., Mingozzi, A., Toth, P., 1981. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming 20, 255–282. URL: https://doi.org/10.1007/BF01589353, doi:10.1007/BF01589353.

Cohen, W.W., Ravikumar, P., Fienberg, S.E., et al., 2003. A comparison of string distance metrics for name-matching tasks., in: IIWeb, Citeseer. pp. 73–78.

da Silva, R.F., Urrutia, S., 2010. A general vns heuristic for the traveling salesman problem with time windows. Discrete Optimization 7, 203–211. URL: https://www.sciencedirect.com/science/article/pii/S1572528610000289, doi:https://doi.org/10.1016/j.disopt.2010.04.002.

Dash, S., Günlük, O., Lodi, A., Tramontani, A., 2012. A time bucket formulation for the traveling salesman problem with time windows. INFORMS Journal on Computing 24, 132–147. URL: https://doi.org/10.1287/ijoc.1100.0432, doi:10.1287/ijoc.1100.0432, arXiv:https://doi.org/10.1287/ijoc.1100.0432.

Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M.M., 1995. An optimal algorithm for the traveling salesman problem with time windows. Operations Research 43, 367–371. URL: https://doi.org/10.1287/opre.43.2.367, doi:10.1287/opre.43.2.367, arXiv:https://doi.org/10.1287/opre.43.2.367.

Elshaer, R., Awad, H., 2020. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering 140, 106242.

Fosgerau, M., Frejinger, E., Karlstrom, A., 2013. A link based network route choice model with unrestricted choice set. Transportation Research Part B: Methodological 56, 70–80. URL: https://www.sciencedirect.com/science/article/pii/S0191261513001276, doi:https://doi.org/10.1016/j.trb.2013.07.012.

Frejinger, E., Bierlaire, M., 2007. Capturing correlation with subnetworks in route choice models. Transportation Research Part B: Methodological 41, 363–378. URL: https://www.sciencedirect.com/science/article/pii/S0191261506000762, doi:https://doi.org/10.1016/j.trb.2006.06.003.

Frejinger, E., Bierlaire, M., Ben-Akiva, M., 2009. Sampling of alternatives for route choice modeling. Transportation Research Part B: Methodological 43, 984–994. URL: https://www.sciencedirect.com/science/article/pii/S0191261509000381, doi:https://doi.org/10.1016/j.trb.2009.03.001.

Gambardella, L.M., Dorigo, M., 1995. Ant-q: A reinforcement learning approach to the traveling salesman problem, in: Machine learning proceedings 1995. Elsevier, pp. 252–260.

Gendreau, M., Hertz, A., Laporte, G., Stan, M., 1998. A generalized insertion heuristic for the traveling salesman problem with time windows. Operations Research 46, 330–335. URL: https://pubsonline.informs.org/doi/abs/10.1287/opre.46.3.330, doi:10.1287/opre.46.3.330, arXiv:https://pubsonline.informs.org/doi/pdf/10.1287/opre.46.3.330.

Goeke, D., Roberti, R., Schneider, M., 2019. Exact and heuristic solution of the consistent vehicle-routing problem. Transportation Science 53, 1023–1042. URL: https://doi.org/10.1287/trsc.2018.0864, doi:10.1287/trsc.2018.0864, arXiv:https://doi.org/10.1287/trsc.2018.0864.

Groër, C., Golden, B., Wasil, E., 2009. The consistent vehicle routing problem. Manufacturing & Service Operations Management 11, 630–643. URL: https://doi.org/10.1287/msom.1080.0243, doi:10.1287/msom.1080.0243, arXiv:https://doi.org/10.1287/msom.1080.0243.

Guo, Z., Mak, K., 2004. A heuristic algorithm for the stochastic vehicle routing problems with soft time windows, in: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), pp. 1449–1456 Vol.2. doi:10.1109/CEC.2004.1331067.

Hu, Y., Zhang, Z., Yao, Y., Huyan, X., Zhou, X., Lee, W.S., 2021. A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs. Engineering Applications of Artificial Intelligence 97, 104061.

Huang, S., Rebello, R., Zhu, Q., 2021. A perspective on driver's preferences for route planning, in: Winkenbach, M., Parks, S., Noszek, J. (Eds.), Technical Proceedings of the 2021 Amazon Last Mile Routing Research Challenge, MIT Libraries. pp. XVIII.1 – XVIII.7.

Jaro, M.A., 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association 84, 414–420. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478785, doi:10.1080/01621459.1989.10478785, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1989.10478785.

Joshi, C.K., Laurent, T., Bresson, X., 2019. An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227 .

Kool, W., Van Hoof, H., Welling, M., 2018. Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475 .

Kovacs, A.A., Parragh, S.N., Hartl, R.F., 2015. The multi-objective generalized consistent vehi-

cle routing problem. European Journal of Operational Research 247, 441–458. URL: https://www.sciencedirect.com/science/article/pii/S0377221715005445, doi:https://doi.org/10.1016/j.ejor.2015.06.030.

Langevin, A., Desrochers, M., Desrosiers, J., Gélinas, S., Soumis, F., 1993. A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. Networks 23, 631–640. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230230706, doi:https://doi.org/10.1002/net.3230230706, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.3230230706.

Levenshtein, V., 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707.

Li, K., Zhang, T., Wang, R.W.Y., Han, Y., 2021. Deep reinforcement learning for combinatorial optimization: Covering salesman problems. arXiv preprint arXiv:2102.05875 .

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-tzur, J., Hardt, M., Recht, B., Talwalkar, A., 2020. A system for massively parallel hyperparameter tuning, in: Dhillon, I., Papailiopoulos, D., Sze, V. (Eds.), Proceedings of Machine Learning and Systems, pp. 230–246. URL: https://proceedings.mlsys.org/paper/2020/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf.

Li, Y., Phillips, W., 2019. MIT Research: Learning From Route Plan Deviations in Last-Mile Delivery. https://www.scmr.com/article/learning_from_route_plan_deviations_in_last_mile_delivery.

Li, Z., Chen, Q., Koltun, V., 2018. Combinatorial optimization with graph convolutional networks and guided tree search. arXiv preprint arXiv:1810.10659 .

Louis, S.J., Li, G., 2000. Case injected genetic algorithms for traveling salesman problems. Information Sciences 122, 201–225. URL: https://www.sciencedirect.com/science/article/pii/S0020025599001243, doi:https://doi.org/10.1016/S0020-0255(99)00124-3.

Luo, Z., Qin, H., Che, C., Lim, A., 2015. On service consistency in multi-period vehicle routing. European Journal of Operational Research 243, 731–744. URL: https://www.sciencedirect.com/science/article/pii/S0377221714010200, doi:https://doi.org/10.1016/j.ejor.2014.12.019.

López-Ibáñez, M., Blum, C., Ohlmann, J., Thomas, B., 2013. The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. Applied Soft Computing 13, 3806–3815. doi:10.1016/j.asoc.2013.05.009.

Mai Anh, T., Bastin, F., Frejinger, E., 2016. A decomposition method for estimating recursive logit based route choice models. EURO Journal on Transportation and Logistics 7. doi:10.1007/s13676-016-0102-3.

Merchan, D., Arora, J., Pachon, J., Konduri, K., Winkenbach, M., Parks, S., Noszek, J., 2022. 2021 amazon last mile routing research challenge: Data set. Transportation Science URL: https://pubsonline.informs.org/doi/10.1287/trsc.2022.1173.

Nazari, M., Oroojlooy, A., Snyder, L.V., Takáč, M., 2018. Reinforcement learning for solving the vehicle routing problem. arXiv preprint arXiv:1802.04240 .

Ohlmann, J., Thomas, B., 2007. A compressed-annealing heuristic for the traveling salesman problem with time windows. INFORMS Journal on Computing 19, 80–90. doi:10.1287/ijoc.1050.0145.

Oyama, Y., Hato, E., 2017. A discounted recursive logit model for dynamic gridlock network analysis. Transportation Research Part C: Emerging Technologies 85, 509–527. URL: https://www.sciencedirect.com/science/article/pii/S0968090X17302747, doi:https://doi.org/10.1016/j.trc.2017.10.001.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pesant, G., Gendreau, M., Potvin, J.Y., Rousseau, J.M., 1998. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. Transportation Science 32, 12–29.

Quirion-Blais, O., Chen, L., 2021. A case-based reasoning approach to solve the vehicle routing problem with time windows and drivers' experience. Omega 102, 102340. URL: https://www.sciencedirect.com/science/article/pii/S0305048320306940, doi:https://doi.org/10.1016/j.omega.2020.102340.

Rochat, Y., Taillard, É.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167. URL: https://doi.org/10.1007/BF02430370, doi:10.1007/

BF02430370.
Samson, B.P.V., Sumi, Y., 2019. Exploring factors that influence connected drivers to (not) use or follow recommended optimal routes, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. p. 1–14. URL: https://doi.org/10.1145/3290605.3300601, doi:10.1145/3290605.3300601.
Savelsbergh, M.W.P., 1985. Local search in routing problems with time windows. Annals of Operations Research 4, 285–305. URL: https://doi.org/10.1007/BF02022044, doi:10.1007/BF02022044.
Seghezzi, A., Mangiaracina, R., Tumino, A., Perego, A., 2020. 'pony express' crowdsourcing logistics for last-mile delivery in b2c e-commerce: an economic analysis. International Journal of Logistics Research and Applications 0, 1–17. URL: https://doi.org/10.1080/13675567.2020.1766428, doi:10.1080/13675567.2020.1766428, arXiv:https://doi.org/10.1080/13675567.2020.1766428.
Sevaux, M., Sörensen, K., et al., 2005. Permutation distance measures for memetic algorithms with population management, in: Proceedings of 6th Metaheuristics International Conference (MIC'05), Citeseer. pp. 832–838.
Srinivas, S.S., Gajanand, M.S., 2017. Vehicle routing problem and driver behaviour: a review and framework for analysis. Transport Reviews 37, 590–611. URL: https://doi.org/10.1080/01441647.2016.1273276, doi:10.1080/01441647.2016.1273276, arXiv:https://doi.org/10.1080/01441647.2016.1273276.
Stohy, A., Abdelhakam, H.T., Ali, S., Elhenawy, M., Hassan, A.A., Masoud, M., Glaser, S., Rakotonirainy, A., 2021. Hybrid pointer networks for traveling salesman problems optimization. arXiv preprint arXiv:2110.03104 .
Szczepanski, K.v., Wagener, C., Mooney, T., McDaniel, L., Mathias, O., Sharp, L., 2021. Only an ecosystem can solve last-mile gridlock in package delivery. https://www.bcg.com/publications/2021/solving-the-package-delivery-system-problems-with-a-new-ecosystem.
Tarantilis, C., Kiranoudis, C., 2002. Boneroute: An adaptive memory-based method for effective fleet management. Annals of Operations Research 115, 227–241. doi:10.1023/A:1021157406318.
Toth, P., Vigo, D., 2014. Vehicle routing: problems, methods, and applications. 2nd ed., SIAM - Society for Industrial and Applied Mathematics.
Ulmer, M., Nowak, M., Mattfeld, D., Kaminski, B., 2020. Binary driver-customer familiarity in service routing. European Journal of Operational Research 286, 477–493. URL: https://www.sciencedirect.com/science/article/pii/S0377221720302514, doi:https://doi.org/10.1016/j.ejor.2020.03.037.
Vakulenko, Y., Shams, P., Hellström, D., Hjort, K., 2019. Online retail experience and customer satisfaction: the mediating role of last mile delivery. The International Review of Retail, Distribution and Consumer Research 29, 306–320. URL: https://doi.org/10.1080/09593969.2019.1598466, doi:10.1080/09593969.2019.1598466, arXiv:https://doi.org/10.1080/09593969.2019.1598466.
Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 .
Vinyals, O., Fortunato, M., Jaitly, N., 2015. Pointer networks. arXiv preprint arXiv:1506.03134 .
Wei, L., Zhang, Z., Zhang, D., Lim, A., 2015. A variable neighborhood search for the capacitated vehicle routing problem with two-dimensional loading constraints. European Journal of Operational Research 243, 798–814. doi:10.1016/j.ejor.2014.12.048.
Zhang, R., Prokhorchuk, A., Dauwels, J., 2020. Deep reinforcement learning for traveling salesman problem with time windows and rejections, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.
Zhang, Y., Zhang, Z., Lim, A., Sim, M., 2021. Robust data-driven vehicle routing with time windows. Operations Research 69, 469–485. URL: https://doi.org/10.1287/opre.2020.2043, doi:10.1287/opre.2020.2043, arXiv:https://doi.org/10.1287/opre.2020.2043.
Zimmermann, M., Mai, T., Frejinger, E., 2017. Bike route choice modeling using GPS data without choice sets of paths. Transportation Research Part C: Emerging Technologies 75, 183–196. URL: https://www.sciencedirect.com/science/article/pii/S0968090X16302637, doi:https://doi.org/10.1016/j.trc.2016.12.009.

Appendix A. Hyperparameters of the feedforward neural network

Hyperparameters of the feedforward neural network used to predict driver tours, determined by the ASHA algorithm Li et al. (2020).

Table A.6: Hyperparameters of feedforward neural network

Phase	Hidden layer size one	Hidden layer size two	Hidden layer size three	Learning rate	Batch size
Cluster	128	64	16	0.00113	32
Customer	64	32	8	0.000589	4

Appendix B. Detailed figures of prediction results

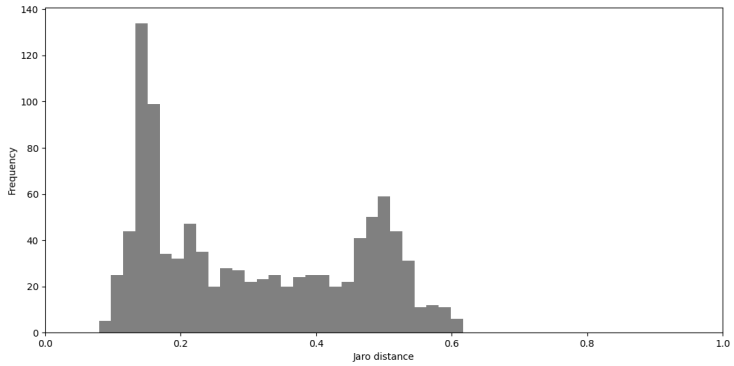


Figure B.5: Distribution of Jaro distances to historical driver tours - Neural Network

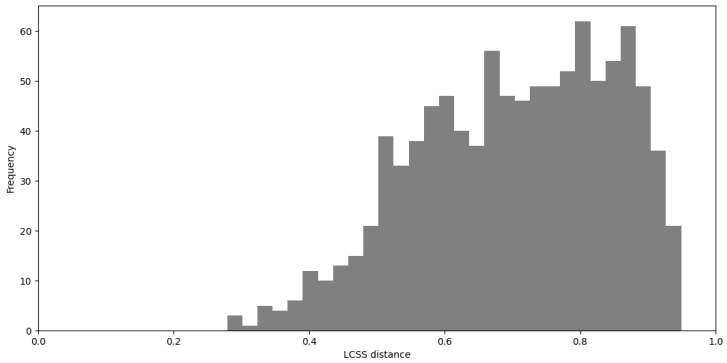


Figure B.6: Distribution of LCSS distances to historical driver tours - Neural Network

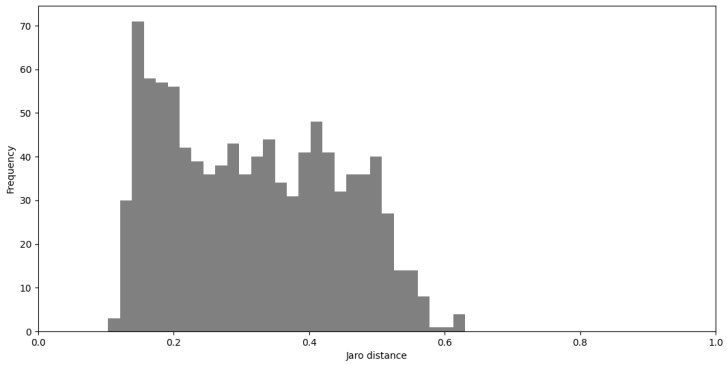


Figure B.7: Distribution of Jaro distances to historical driver tours - Nearest Neighbor

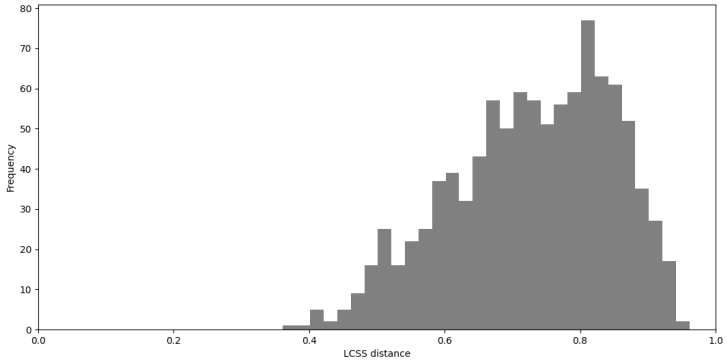
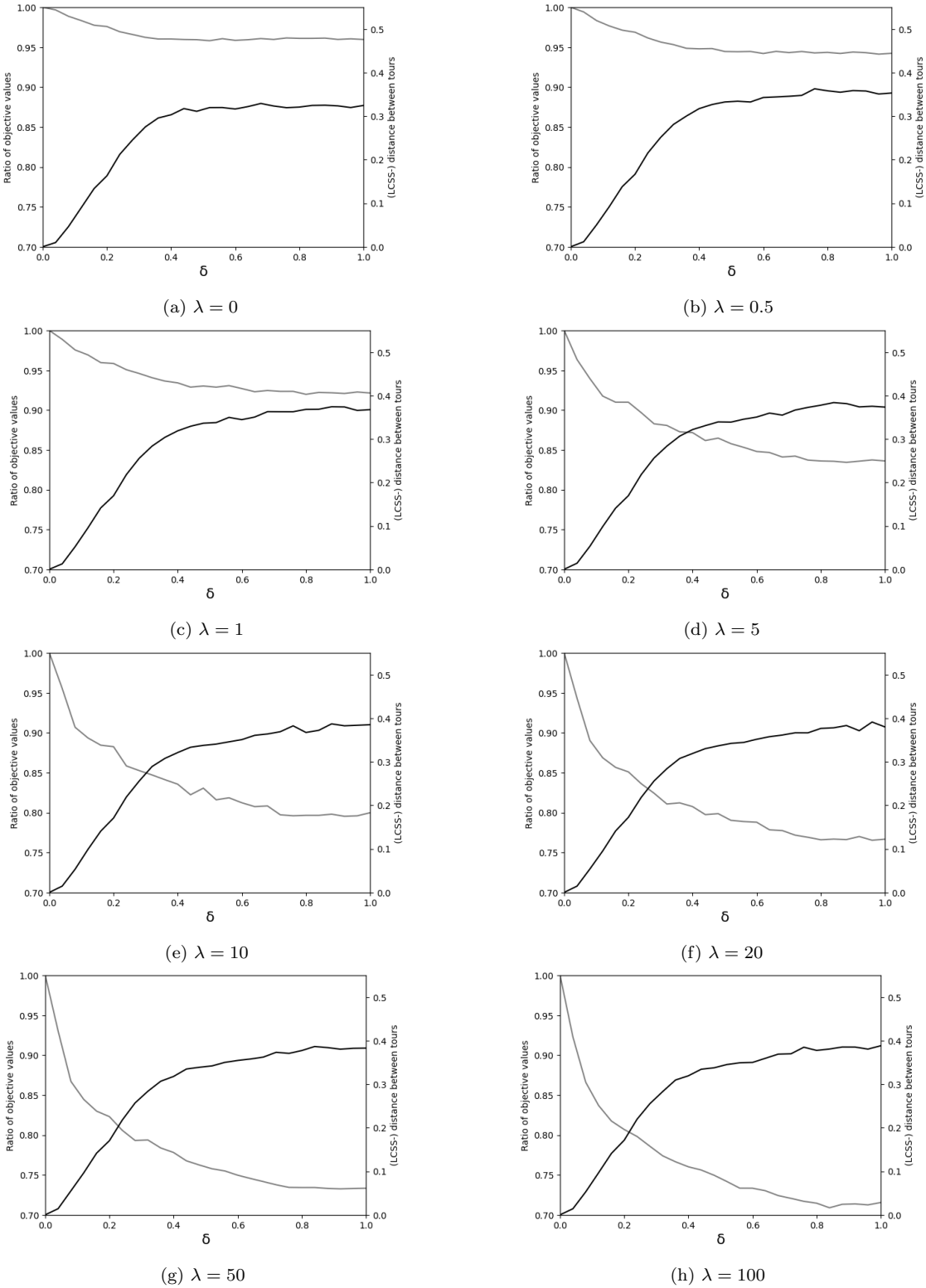


Figure B.8: Distribution of LCSS distances to historical driver tours - Nearest Neighbor

Appendix C. Detailed figures of the application of VNS meta-heuristic



(a) $\lambda = 0$

(b) $\lambda = 0.5$

(c) $\lambda = 1$

(d) $\lambda = 5$

(e) $\lambda = 10$

(f) $\lambda = 20$

(g) $\lambda = 50$

(h) $\lambda = 100$

Figure C.9: Relationships between suggested tours and (predicted) driven tours - normalized LCSS distance

Table C.7: Objective value change using Jaro distances

λ / δ	0	0.5	1	5	10	20	50	100
0.00	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.04	0.979	0.973	0.963	0.914	0.884	0.866	0.839	0.828
0.08	0.967	0.958	0.948	0.889	0.855	0.829	0.802	0.789
0.12	0.958	0.946	0.935	0.870	0.835	0.807	0.779	0.766
0.16	0.957	0.945	0.929	0.855	0.819	0.782	0.758	0.738
0.20	0.959	0.942	0.927	0.849	0.808	0.776	0.745	0.732
0.24	0.959	0.945	0.924	0.843	0.804	0.774	0.742	0.721
0.28	0.959	0.944	0.923	0.841	0.802	0.766	0.734	0.718
0.32	0.958	0.944	0.923	0.839	0.801	0.767	0.733	0.716
0.36	0.960	0.942	0.923	0.839	0.799	0.764	0.731	0.716
0.40	0.959	0.942	0.923	0.837	0.799	0.765	0.733	0.717
0.44	0.958	0.943	0.924	0.839	0.799	0.766	0.733	0.714
0.48	0.960	0.942	0.922	0.839	0.798	0.764	0.732	0.714
0.52	0.958	0.942	0.921	0.839	0.800	0.763	0.731	0.713
0.56	0.960	0.943	0.924	0.839	0.797	0.764	0.733	0.713
0.60	0.958	0.944	0.923	0.838	0.799	0.765	0.732	0.714
0.64	0.959	0.943	0.924	0.839	0.800	0.766	0.732	0.714
0.68	0.958	0.943	0.922	0.841	0.800	0.765	0.732	0.714
0.72	0.959	0.942	0.923	0.837	0.798	0.764	0.733	0.717
0.76	0.960	0.943	0.924	0.839	0.799	0.764	0.733	0.719
0.80	0.959	0.943	0.922	0.838	0.799	0.762	0.736	0.714
0.84	0.959	0.942	0.922	0.838	0.797	0.764	0.735	0.712
0.88	0.959	0.943	0.923	0.838	0.799	0.763	0.735	0.714
0.92	0.959	0.943	0.923	0.837	0.798	0.765	0.732	0.715
0.96	0.957	0.942	0.921	0.838	0.799	0.765	0.732	0.715
1.00	0.958	0.942	0.922	0.838	0.799	0.763	0.732	0.713

Table C.8: Jaro distances

λ / δ	0	0.5	1	5	10	20	50	100
0.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.04	0.031	0.031	0.032	0.033	0.033	0.033	0.033	0.033
0.08	0.070	0.072	0.070	0.072	0.073	0.073	0.073	0.073
0.12	0.093	0.096	0.096	0.096	0.099	0.098	0.098	0.098
0.16	0.097	0.103	0.106	0.107	0.107	0.108	0.106	0.108
0.20	0.101	0.107	0.110	0.113	0.114	0.112	0.112	0.114
0.24	0.100	0.110	0.113	0.114	0.116	0.116	0.116	0.116
0.28	0.101	0.110	0.114	0.116	0.120	0.118	0.118	0.120
0.32	0.100	0.111	0.119	0.119	0.124	0.119	0.121	0.124
0.36	0.101	0.114	0.119	0.122	0.124	0.121	0.127	0.125
0.40	0.104	0.115	0.122	0.123	0.126	0.128	0.124	0.125
0.44	0.101	0.119	0.121	0.125	0.124	0.128	0.125	0.129
0.48	0.103	0.118	0.119	0.128	0.129	0.129	0.127	0.131
0.52	0.104	0.115	0.121	0.126	0.129	0.125	0.129	0.130
0.56	0.105	0.120	0.123	0.127	0.128	0.129	0.127	0.128
0.60	0.102	0.117	0.121	0.129	0.127	0.131	0.126	0.130
0.64	0.105	0.122	0.125	0.125	0.132	0.129	0.128	0.130
0.68	0.102	0.119	0.121	0.125	0.129	0.129	0.128	0.133
0.72	0.102	0.118	0.121	0.124	0.128	0.125	0.126	0.129
0.76	0.102	0.117	0.119	0.128	0.129	0.129	0.127	0.127
0.80	0.104	0.117	0.122	0.128	0.126	0.127	0.132	0.130
0.84	0.103	0.116	0.122	0.126	0.129	0.129	0.129	0.130
0.88	0.101	0.119	0.119	0.126	0.128	0.129	0.127	0.129
0.92	0.105	0.119	0.121	0.128	0.128	0.126	0.130	0.126
0.96	0.100	0.116	0.121	0.127	0.127	0.126	0.127	0.128
1.00	0.103	0.119	0.123	0.128	0.128	0.127	0.127	0.129

Table C.9: Objective value change using LCSS distances

λ / δ	0	0.5	1	5	10	20	50	100
0.00	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.04	0.997	0.994	0.989	0.963	0.955	0.943	0.930	0.923
0.08	0.989	0.983	0.976	0.940	0.907	0.890	0.867	0.866
0.12	0.983	0.977	0.969	0.918	0.894	0.869	0.845	0.837
0.16	0.978	0.971	0.960	0.910	0.884	0.857	0.830	0.817
0.20	0.976	0.969	0.959	0.910	0.883	0.851	0.823	0.807
0.24	0.969	0.962	0.951	0.897	0.858	0.836	0.806	0.798
0.28	0.966	0.956	0.946	0.883	0.853	0.824	0.793	0.786
0.32	0.962	0.953	0.941	0.881	0.847	0.811	0.794	0.774
0.36	0.960	0.949	0.936	0.873	0.841	0.812	0.784	0.766
0.40	0.960	0.948	0.934	0.872	0.836	0.808	0.778	0.760
0.44	0.960	0.948	0.929	0.862	0.822	0.797	0.768	0.756
0.48	0.959	0.945	0.930	0.865	0.831	0.799	0.762	0.750
0.52	0.958	0.944	0.929	0.858	0.816	0.790	0.758	0.742
0.56	0.961	0.945	0.931	0.853	0.819	0.789	0.755	0.733
0.60	0.959	0.942	0.927	0.848	0.812	0.788	0.750	0.733
0.64	0.959	0.945	0.923	0.847	0.807	0.778	0.745	0.730
0.68	0.961	0.943	0.925	0.841	0.808	0.777	0.742	0.724
0.72	0.960	0.945	0.924	0.842	0.797	0.772	0.738	0.721
0.76	0.962	0.943	0.924	0.837	0.796	0.769	0.734	0.717
0.80	0.961	0.944	0.920	0.836	0.797	0.766	0.734	0.715
0.84	0.961	0.942	0.922	0.836	0.797	0.767	0.734	0.709
0.88	0.961	0.944	0.922	0.834	0.798	0.766	0.733	0.713
0.92	0.960	0.943	0.921	0.836	0.795	0.770	0.733	0.714
0.96	0.961	0.941	0.923	0.837	0.796	0.765	0.733	0.712
1.00	0.960	0.942	0.921	0.836	0.800	0.767	0.733	0.715

Table C.10: LCSS distances

λ / δ	0	0.5	1	5	10	20	50	100
0.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.04	0.010	0.011	0.013	0.014	0.014	0.014	0.014	0.014
0.08	0.046	0.050	0.052	0.053	0.053	0.054	0.055	0.053
0.12	0.090	0.092	0.095	0.098	0.099	0.095	0.096	0.097
0.16	0.134	0.138	0.141	0.141	0.141	0.141	0.142	0.141
0.20	0.163	0.166	0.169	0.169	0.171	0.172	0.170	0.171
0.24	0.212	0.215	0.218	0.218	0.219	0.218	0.218	0.220
0.28	0.245	0.251	0.256	0.256	0.256	0.256	0.257	0.255
0.32	0.275	0.281	0.284	0.284	0.289	0.284	0.284	0.283
0.36	0.296	0.300	0.304	0.307	0.308	0.308	0.307	0.310
0.40	0.303	0.317	0.319	0.322	0.321	0.319	0.318	0.319
0.44	0.317	0.327	0.329	0.331	0.333	0.330	0.335	0.334
0.48	0.311	0.333	0.336	0.339	0.338	0.337	0.339	0.338
0.52	0.320	0.334	0.338	0.339	0.341	0.342	0.342	0.345
0.56	0.320	0.332	0.350	0.346	0.346	0.345	0.350	0.349
0.60	0.316	0.343	0.345	0.351	0.351	0.352	0.355	0.350
0.64	0.322	0.344	0.351	0.360	0.361	0.358	0.358	0.360
0.68	0.329	0.346	0.363	0.355	0.364	0.361	0.362	0.369
0.72	0.323	0.348	0.363	0.367	0.369	0.367	0.373	0.370
0.76	0.319	0.363	0.363	0.373	0.383	0.367	0.371	0.385
0.80	0.321	0.358	0.368	0.378	0.367	0.377	0.377	0.378
0.84	0.325	0.355	0.369	0.384	0.372	0.378	0.387	0.381
0.88	0.325	0.359	0.374	0.382	0.387	0.383	0.384	0.385
0.92	0.324	0.358	0.374	0.374	0.383	0.371	0.381	0.385
0.96	0.320	0.351	0.366	0.376	0.384	0.391	0.382	0.381
1.00	0.325	0.353	0.368	0.374	0.385	0.380	0.383	0.388