Constraints (2023) 28:363–396
https://doi.org/10.1007/s10601-023-09363-2

Learn and route: learning implicit preferences for vehicle routing

Rocsildes Canoy[1] · **Víctor Bucarey**[2] · **Jayanta Mandi**[1] · **Tias Guns**[3]

Accepted: 8 September 2023 / Published online: 11 October 2023

Abstract
We investigate a *learning* decision support system for vehicle routing, where the routing engine learns implicit preferences that human planners have when manually creating route plans (or *routings*). The goal is to use these learned *subjective* preferences on top of the distance-based *objective* criterion in vehicle routing systems. This is an alternative to the practice of distinctively formulating a custom vehicle routing problem (VRP) for every company with its own routing requirements. Instead, we assume the presence of past vehicle routing solutions over similar sets of customers, and learn to make similar choices. The learning approach is based on the concept of learning a Markov model, which corresponds to a probabilistic transition matrix, rather than a deterministic distance matrix. This nevertheless allows us to use existing arc routing VRP software in creating the actual routings, and to optimize over both distances and preferences at the same time. For the learning, we explore different schemes to construct the probabilistic transition matrix that can co-evolve with changing preferences over time. Our results on randomly generated instances and on a use-case with a small transportation company show that our method is able to generate results that are close to the manually created solutions, without needing to characterize all constraints and sub-objectives explicitly. Even in the case of changes in the customer sets, our approach is able to find solutions that are closer to the actual routings than when using only distances, and hence, solutions that require fewer manual changes when transformed into practical routings.

Keywords Preference learning · Vehicle routing · Markov models · Transition probabilities

✉ Rocsildes Canoy
rocsildes.canoy@vub.be

Víctor Bucarey
victor.bucarey@uoh.cl

Jayanta Mandi
jayanta.mandi@vub.be

Tias Guns
tias.guns@kuleuven.be

[1] Data Analytics Laboratory, Vrije Universiteit Brussel, Brussels, Belgium

[2] Institute of Engineering Sciences, Universidad de O'Higgins, Rancagua, Chile

[3] KUL Institute for AI, Katholieke Universiteit Leuven, Leuven, Belgium

1 Introduction

Vehicle routing problems (VRP) at small or medium-sized enterprises (SME) are constrained by the limited number of vehicles, the capacity of each delivery vehicle, and the scheduling horizon within which all deliveries have to be made. The objective, often implicitly, can include a wide range of company goals including reducing operational costs, minimizing fuel consumption and carbon emissions, as well as optimizing driver familiarity with the routes and maximizing fairness by assigning tours of similar length and/or time duration to the drivers. Daily plans are often created in a route optimization software that is capable of producing plans that are optimal in terms of route length and travel time. We have observed, however, that in practice, route planners usually modify the result given by the software, or simply pull out, modify, and reuse an old plan that has been used and known to work in the past. The planners, by performing these modifications, are essentially optimizing with their own set of objectives and personal preferences.

These preferences are often subjective and sometimes delicate to formalize in constraints. Some examples are: where the best places for lunch breaks are, which stops are best served earlier or later, and which drivers (tours) are more flexible in receiving more stops. Failure to capture such *human* aspects is often a source of frustration for both drivers and planners, and a cause for reluctance to use the optimisation software. Furthermore, planners may find it easier or more effective to manually change a previous solution than to provide or update the detailed information in the system. Being able to automatically capture such preferences without the need to formalize them can hence lead to a wider acceptance and better use of optimisation systems.

The goal of this research is to learn the preferences of the planners (and implicitly the drivers) when choosing one option over another. Hence, the goal is to build an 'intelligent' system that can effectively reuse all of the knowledge and effort that have been put into creating previous routings; much like how human planners use prior experience. We focus on techniques from the domain of artificial intelligence that can learn from historical data, and that can be used to manage and recommend similar routes as used in the past. This is in contrast to the current practice of optimizing a VRP instance separately and independently each day.

To learn from historical data, we take inspiration from various machine learning papers on route prediction for a single vehicle: Markov models developed from historical data have been applied to driver turn prediction [28], prediction of the remainder of the route by looking at the previous road segments taken by the driver [44], and predicting individual road choices given the origin and destination [39]. These studies have produced positive and encouraging results for their respective tasks. Hence, in this work, we investigate the use of Markov models for predicting the route choices for an entire fleet, and how to use these choices to create *preference-aware* solutions to the VRP.

With a Markov model, route optimization can be done by maximizing the product of the probabilities of the sequential actions taken by the vehicles, which corresponds to maximizing the sum of log likelihoods over the route arcs. In the case of a first order Markov model, a key property of our approach is that it can use *any existing VRP solution method* to find the maximum likelihood solution, as it corresponds to the sum of log likelihoods over the individual arcs. This is a promising novel approach to the vehicle routing problem.

Our proposed first order Markov model approach has been published in a conference proceeding [10]. In this article, we extend this methodology by developing the following new contributions:

- We provide a more general formalization of the methods for learning and optimizing the preferences, and detail both a first order *and a second order* Markov model.
- We introduce a new and superior weighing technique called the exponential weighing scheme and examine how it handles data drift, i.e., when there is either a large reduction or a sudden increase in the number of stops.
- We present an extended study on distance-based, preference-based, and combined optimisation.
- We test our approach on synthetic data and compare the results against the recently published inverse optimization approach introduced in [13]. Results show that our approach is not only as good as the inverse optimization approach in terms of quality of the learning, but is also more efficient in terms of computation time.

This manuscript is organized as follows: in Section 2, we provide the relevant work related to this article. We discuss in Section 3 routing models that maximize the probability of the learned preferences. Later in the section, we introduce the algorithms to learn the transition matrix from historical data. The comparison of the different construction schemes and the experimental results on synthetic and actual company data are shown in Section 4. Finally, our conclusions and future research directions are presented in Section 5.

2 Related work

The first mathematical formulation and algorithmic approach to solving the Vehicle Routing Problem (VRP) appeared in the paper by [16] which aimed to find an optimal routing for a fleet of gasoline delivery trucks. Since its introduction, the VRP has become one of the most studied combinatorial optimization problems. Faced on a daily basis by distributors and logistics companies worldwide, the problem has attracted a lot of attention due to its significant economic importance.

A large part of the research effort concerning the VRP has focused on its classical version—the *Capacitated* Vehicle Routing Problem (CVRP). The presumption is that the algorithms developed for CVRP could be extended and applied to more complicated real-world cases [29]. Due to the recent development of new and more efficient optimisation methods, research interest has shifted towards realistic VRP variants known as Rich VRP [9, 21]. These problems deal with realistic, and sometimes multi-objective, optimisation functions, uncertainty, and a wide variety of real-life constraints related to time and distance factors, inventory and scheduling, environmental and energy concerns, driver experience, etc. [32]. The VRP becomes increasingly complex as additional sub-objectives and constraints are introduced. The inclusion of preferences, for example, necessitates the difficult, if not impossible, task of formalizing the route planners' knowledge and choice preferences explicitly in terms of constraints and weights. In most cases, it is much easier to get examples and historical solutions rather than to extract explicit decision rules from the planners, as observed by Potvin et al. in the case of vehicle dispatching [35]. One approach is to use learning techniques, particularly learning by examples, to reproduce the planners' decision behavior. To this end, we develop a new method that learns from previous solutions by using a Markov model, which can simplify the problem formulation phase of vehicle routing by eliminating the need to characterize all preference constraints and sub-objectives explicitly.

Learning from historical solutions has been investigated before within the context of constraint programming. For example, in the paper of Beldiceanu and Simonis on constraint seeker [5] and model seeker [6], and Picard-Cantin et al. on learning constraint parameters

from data [34], where a Markov chain is used, but for individual constraints. In this respect, our goal is not to learn constraint instantiations, but rather to learn choice preferences, e.g., as part of the objective. Related to the latter is the work on Constructive Preference Elicitation [20], although this elicitation technique actively queries the user, as does constraint acquisition [8].

Our motivation for Markov models is that they have been previously used in route prediction of individual vehicles. Ashbrook and Starner [4] presented a Markov model to predict the next stop of a moving agent equipped with a wearable GPS receiver. They estimated transition probabilities as frequencies and considered Markov models that depend not only on the actual stop but also on past stops; that is, high order Markov chains. Personalized route prediction has been used in transportation systems that provide drivers with real-time traffic information and in intelligent vehicle systems for optimizing energy efficiency in hybrid vehicles [17]. Ye et al. [44] introduced a route prediction method that can accurately predict an entire route early in the trip. The method is based on Hidden Markov Models (HMM) trained from the drivers' past history. A route prediction algorithm that predicts a driving route for a given pair of origin and destination was presented by Wang et al. [39]. Also based on the first order Markov model, the algorithm uses a probability transition matrix that was constructed to represent the knowledge of the driver's preferred links and routes. An algorithm for driver turn prediction using a Markov model is developed in Krumm [28]. Trained from the driver's historical data, the model makes a probabilistic prediction based on a short sequence of just-driven road segments. Experimental results showed that by looking at the most recent 10 segments into the past, the model can effectively predict the next segment with about 90% accuracy. While related to our current work, we do not assume that a partial route is given. Moreover, in our case, we are interested in optimizing the routing across *multiple vehicles* rather than making predictions for a single vehicle.

Several algorithms were deployed to direct users through paths between origin and destination. These were done either through an enumeration of paths [42], or by adapting weights on edges and computing shortest paths [18, 23, 30]. Although similar to this second category, where weights represent probabilities, our work differs in that instead of planning single paths (i.e., for an origin-destination pair), our methods use preferences to create a routing plan for several vehicles. Also, we are adapting the classical CVRP mixed integer formulation in order to generate high probability routes.

The importance of preference-based learning can be realized from the study of Ceikute and Jensen [11], which finds that local drivers often prefer paths that are not optimal in terms of travel time or cost. This is because local drivers favor certain routes based on their personal preferences. Chang et al. [12] and Letchner et al. [30] proposed a framework to recommend personalized routes to the driver by taking into account which roads he or she is familiar with. Yang et al. estimated a driver's preferences by comparing the paths used by the driver to the skyline paths [41]. Yang and Yang [43] introduced *PathRank*, a machine learning model trained with historical data, to rank candidate paths based on the driver's preferences. Again, the above-mentioned are mostly concerned with point-to-point 'shortest path' travel where the source and destination pairs are known. Our work, on the other hand, focuses on the VRP: recommending a multi-vehicle solution over shared stops.

Our work aligns with the ongoing line of research focused on directly solving combinatorial optimization problems, such as CVRPs, using neural networks. Recently, the use of deep learning and reinforcement learning to solve various combinatorial optimization problems has received increasing attention. However, it is important to note that our work does not aim to solve CVRP problems efficiently; but to learn the preferences, which we transform as the cost of the CVRP. While we acknowledge that these neural network models can be adapted, for both learning the cost of the edges and solving the CVRP, their successful

implementation requires a substantial amount of training data. Due to the limited availability of a large training dataset, using such models are beyond the scope of our research. Nevertheless, in the paragraph that follows, the readers can find a concise overview of notable works that address the problem of solving combinatorial optimization problems using neural networks.

The Pointer Network proposed in Vinyals et al. [38] makes use of sequence-to-sequence models to solve a TSP. Here, an attention model is used to learn the optimal order of cities to visit within a supervised learning framework. In Nazari et al. [33] the pointer network is extended to tackle VRPs by making the model invariant with respect to the input sequence. The work in Bello et al. [7] also tackles TSPs, where the pointer network architecture Vinyals et al. [38] is used to determine the next city to visit. However, in this work the negative of the tour length is considered as the reward and the network is trained via a policy gradient algorithm Williams [40] without supervised solutions. The sequence-to-sequence model in the pointer network is replaced with the Graph Attention Network Veličković et al. [37] in Kool et al. [27]; Deudon et al. [19] and the model is trained via policy gradient algorithm.

A recent article [2] showed an application of routing with preferences using *patient preferences* over the drivers/vehicles to create route visits for caregivers, where the problem is formulated as a bi-criteria optimization problem. In contrast, our work is concerned in creating routings while taking into account the implicit *planner and driver preferences* over the set of customers. More recently, an inverse optimization formulation that learns a cost matrix from previous solutions was introduced in Chen et al. [13]. As with our approach, when route optimization is done using this new learned cost matrix instead of the original distance matrix, resulting solutions are found to be as good as those created by the experts. However, our methodology is able to capture preferences as effectively as the inverse optimization approach while being considerably more efficient in terms of computational time.

3 Formalisation

In this section we introduce the probabilistic model underlying our methodology to learn the drivers' preferences. Furthermore, we adapt the classical formulation of the CVRP to compute the route with the highest probability.

3.1 Routing, probabilities and Markov models

We begin with a set of stops $V = \{0, 1, \ldots, n\}$, where 0 represents the depot and the other stops represent the locations of the customers, and a fully-connected directed graph $G = (V, A)$. We call a **routing** $\mathbf{x}$ of V with m homogeneous vehicles, a set of at most m tours in G with each tour starting from and ending at the depot 0 and such that each node in $V\backslash\{0\}$ is visited exactly once.

We denote the set of all possible routings of m vehicles over V as $\mathcal{X}_V^m$, or simply $\mathcal{X}$ when it is clear from the context. A typical element s of $\mathcal{X}_V^m$ can be represented as a sequence

$$s = (0, s_{11}, s_{12}, \ldots, s_{1p-1}, s_{1p}, 0, 0, s_{21}, s_{22}, \ldots, s_{2q-1}, s_{2q}, 0, \ldots, 0, s_{m1}, \ldots s_{mr}, 0),$$

where p, q and r represent the number of stops visited in routes 1, 2 and m, respectively. Our goal is to determine a probabilistic model **Pr** such that $\mathbf{Pr}(\mathbf{x})$ is the probability of the vehicles following the routing $\mathbf{x} \in \mathcal{X}$.

A Markov model is defined over a single sequence of events, and so to be able to use Markov models for $\mathbf{Pr}(\mathbf{x})$, we first daisy-chain the set of routes $\mathbf{x}$. This is done by connecting the routes into one long sequence:

$$\begin{aligned}
&0 \longrightarrow s_{11} \longrightarrow s_{12} \longrightarrow \ldots \longrightarrow s_{1p-1} \longrightarrow s_{1p} \longrightarrow 0 \longrightarrow \\
&0 \longrightarrow s_{21} \longrightarrow s_{22} \longrightarrow \ldots \longrightarrow s_{2q-1} \longrightarrow s_{2q} \longrightarrow 0 \longrightarrow \\
&\vdots \\
&0 \longrightarrow s_{m1} \longrightarrow s_{m2} \longrightarrow \ldots \longrightarrow s_{mr-1} \longrightarrow s_{mr} \longrightarrow 0,
\end{aligned}$$

which we then simplify by replacing the $(0 \rightarrow 0)$ connections by 0. Given this sequence interpretation of $\mathbf{x}$, we can decompose the joint probability $\mathbf{Pr}(\mathbf{x})$ as the probability of each element conditional on the elements before it in the sequence:

$$\begin{aligned}
\mathbf{Pr}(\mathbf{x}) &= \mathbf{Pr}(0, s_{11}, s_{12}, s_{13}, \ldots, 0, s_{21}, \ldots) \\
&= \mathbf{Pr}(X_0 = 0)\, \mathbf{Pr}(X_1 = s_{11} \mid X_0 = 0)\, \mathbf{Pr}(X_2 = s_{12} \mid X_0 = 0, X_1 = s_{11}) \\
&\qquad \mathbf{Pr}(X_3 = s_{13} \mid X_0 = 0, X_1 = s_{11}, X_2 = s_{12}) \cdots .
\end{aligned}$$

Estimating all conditional probabilities $\mathbf{Pr}(s_{ij} \mid 0, \ldots, s_{ij-2}, s_{ij-1})$ is difficult because of the large number of such possible probabilities and their sparsity in the data. A common approach is to use a Markovian approximation of the probabilistic model [3], which states that the probability of an event depends only on the state of the previous event. The main advantage is that, with this approach, the model can be described with much fewer parameters to learn.

We consider a *first order* Markov chain $\{\mathbf{X_t}\}_{\mathbf{t \geq 0}}$ over V with transition probabilities between states as:

$$\mathbf{Pr}\left(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0\right) \approx \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right).$$

The joint probability $\mathbf{Pr}(\mathbf{x})$ therefore becomes:

$$\mathbf{Pr}(\mathbf{x}) \approx \mathbf{Pr}(X_0 = 0)\, \mathbf{Pr}(X_1 = s_{11} \mid X_0 = 0)\, \mathbf{Pr}(X_2 = s_{12} \mid X_1 = s_{11})\, \mathbf{Pr}(X_3 = s_{13} \mid X_2 = s_{12}) \cdots .$$

On one hand, the interpretation of this model's assumption is straightforward: a driver's decision to go from a stop i to another stop j depends only on the current position (current stop), and does not consider the stops visited before that. With this assumption, the Markov model can be seen as a probability distribution over the set of arcs A. On the other hand, this model is in fact an approximation: in practice, drivers and planners also take the stops preceding the current stop, and potentially even the stops after it, into account when making their decision. In either case, the current stop X_t is the key deciding factor. A finer-grained approximation is to consider a higher order, in particular a *second order*, Markov chain. In this case, the state of an event contains both the current stop and the stop before it. By extending the history of previously visited arcs, we get a better approximation of the probabilistic model over $\mathcal{X}$. For the second order model, the conditional probabilities are approximated as follows:

$$\mathbf{Pr}\left(X_{t+1} = k \mid X_t = j, X_{t-1} = i, \ldots, X_0 = i_0\right) \approx \mathbf{Pr}\left(X_{t+1} = k \mid X_{t-1} = i,\ X_t = j\right).$$

The trade-off is that the number of parameters to estimate increases from $O(n^2)$ to $O(n^3)$. Later in this section, we will investigate the different ways of estimating these conditional probabilities from the historical routings.

3.2 Maximum likelihood routing

Given a (learned) probability distribution **Pr** over a set of stops V, the goal is to find the *maximum likelihood* routing, that is, the routing with the highest joint probability. Let $\mathcal{X}$ be the set of all possible routings for a given number m of vehicles and a set of stops V. The goal of finding the maximum likelihood routing then becomes:

$$\max_{\mathbf{x}\in\mathcal{X}} \mathbf{Pr}(\mathbf{x}). \tag{1}$$

Naturally, the set of all possible routings $\mathcal{X}$ is not given explicitly. However, we can define it implicitly as a set of constraints over decision variables, as common in optimisation approaches to vehicle routing. We first look into formulating the constraints, and then the objective function corresponding to (1).

Constraints

Note that while the probability distribution **Pr** is defined over all possible routings, we are free to impose additional constraints over $\mathcal{X}$ when searching for the maximum likelihood routing. In effect, this will be a constrained maximum likelihood inference problem.

In this paper, we will use a standard Capacitated Vehicle Routing Problem (CVRP) formulation by means of Mixed Integer Programming (MIP) [25]. To do so, we represent the routing $\mathcal{X}$ by the binary vector $\mathbf{x} \in \{0, 1\}^{|A|}$ where A is the set of all possible edges between V, that is, the arc set of the complete graph $G = (V, A)$. Each component of a vector $\mathbf{x}$, namely x_{ij}, takes the value 1 if there exists a transition from i to j in sequence $\mathbf{x}$, and 0 otherwise. The CVRP routing problem with m homogeneous vehicles, each with capacity Q, and a given demand q_i for every stop $i \in V$, can then be represented by the following standard constraints [25]:

$$\sum_{j\in V,\, j\neq i} x_{ij} = 1 \qquad i \in V\backslash\{0\} \tag{2}$$

$$\sum_{i\in V,\, i\neq j} x_{ij} = 1 \qquad j \in V\backslash\{0\} \tag{3}$$

$$\sum_{j=1}^{n} x_{0j} \leq m \tag{4}$$

$$\text{if } x_{ij} = 1 \Rightarrow u_i + q_j = u_j \qquad (i, j) \in A : j \neq 0,\ i \neq 0 \tag{5}$$

$$q_i \leq u_i \leq Q \qquad i \in V\backslash\{0\} \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j) \in A, \tag{7}$$

where constraints (2) and (3) impose that every stop other than the depot must be visited by exactly one vehicle and that exactly one vehicle must leave from each node. Constraint (4) limits the number of routes to the size of the fleet, m. In constraint (5), u_j denotes the cumulative vehicle load at node j. The constraint plays a dual role—it prevents the formation of subtours, i.e., cycling routes that do not pass through the depot, and together with constraint (6), it ensures that the vehicle capacity is not exceeded. While the model does not make explicit which stop belongs to which route, this information can be reconstructed from the active arcs x_{ij} in the solution.

Objective function

For a first order Markov chain approximation, the joint probability over a daisy-chained sequence of stops $\mathbf{x} = \{0, s_1, s_2, s_3, \ldots s_n, 0\}$ decomposes into the following:

$$\begin{aligned} \mathbf{Pr}(\mathbf{x}) &\approx \mathbf{Pr}(X_0 = 0)\,\mathbf{Pr}(X_1 = s_1 \mid X_0 = 0) \cdots \mathbf{Pr}(X_{n+1} = 0 \mid X_n = s_n) \\ &= \mathbf{Pr}(X_0 = 0) \prod_{(i \to j) \in \mathbf{x}} \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right). \end{aligned} \tag{8}$$

In our routing setting, by construction, the first stop is always the depot. Hence, we know that $\mathbf{Pr}(X_0 = 0) = 1$. In order to transform the above into an objective function over decision variables x_{ij} we remark that $x_{ij} = 1 \Leftrightarrow (i \to j) \in \mathbf{x}$. We can now derive the following:

$$\begin{aligned} \arg\max_{\mathbf{x}} \mathbf{Pr}(\mathbf{x}) &\approx \arg\max_{\mathbf{x}} \mathbf{Pr}(X_0 = 0) \prod_{(i \to j) \in \mathbf{x}} \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right) \\ &= \arg\max_{\mathbf{x}} \prod_{(i \to j) \in \mathbf{x}} \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right) \\ &= \arg\max_{\mathbf{x}} \sum_{(i \to j) \in \mathbf{x}} \log \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right) \\ &= \arg\max_{x} \sum_{(i,j) \mid x_{ij} = 1} \log \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right) x_{ij} \\ &= \arg\min_{x} \sum_{(i,j) \in A} -\log \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right) x_{ij}. \end{aligned} \tag{9}$$

If we define $\hat{p}_{ij} = -\log \mathbf{Pr}\left(X_{t+1} = j \mid X_t = i\right)$, then we obtain the following **standard CVRP formulation**:

$$\begin{aligned} &\min_{x} \sum_{(i,j) \in A} \hat{p}_{ij} x_{ij} \\ &\text{s.t.} \quad \text{Constraints } (2) - (7). \end{aligned} \tag{10}$$

Hence, using $\hat{p}$ as the cost vector in the traditional VRP setting enables us to use any existing VRP solver to find the maximum likelihood routing of a given first order Markov model. We now consider the case of a **second order Markov chain**:

$$\begin{aligned} \mathbf{Pr}(\mathbf{x}) &\approx \mathbf{Pr}(X_0 = 0)\,\mathbf{Pr}(X_1 = s_1 \mid X_0 = 0)\,\mathbf{Pr}(X_2 = s_2 \mid X_0 = 0, X_1 = s_1) \cdots \\ &\quad \mathbf{Pr}(X_{n+1} = 0 \mid X_{n-1} = s_{n-1}, X_n = s_n) \\ &= \mathbf{Pr}(X_0 = 0)\,\mathbf{Pr}(X_1 = s_1 \mid X_0 = 0) \prod_{(i \to j \to k) \in \mathbf{x}} \mathbf{Pr}\left(X_{t+1} = k \mid X_{t-1} = i, X_t = j\right). \end{aligned} \tag{11}$$

To construct a corresponding objective function over decision variables x_{ij}, we have to be more careful than in the first order case. More specifically, the second order Markov model includes the probabilities of transitions over the different vehicles, such as $(s_{1p} \to 0 \to s_{21})$. However, vehicles (and hence routes) are assumed to be homogeneous and therefore interchangeable. Hence, the ordering of the routes when constructing the daisy-chain is arbitrary. The last stop of one route should therefore not have any influence on the first stop of another route. We will hence ignore all transitions of the kind $(s_{1p} \to 0 \to s_{21})$ and instead use the first order transition probability from the depot: $(0 \to s_{21})$. These need to be

estimated for the first transition $\mathbf{Pr}(X_1 = s_1 \mid X_0 = 0)$ anyway. This leads to the following derivation:

$$\begin{aligned}
&\arg\max_{\mathbf{x}} \ \mathbf{Pr}(X_0 = 0)\,\mathbf{Pr}(X_1 = s_1 \mid X_0 = 0) \prod_{(i \to j \to k) \in \mathbf{x}} \mathbf{Pr}\left(X_{t+1} = k \mid X_{t-1} = i, X_t = j\right) \qquad (12)\\
&\quad \approx \arg\max_{\mathbf{x}} \ \mathbf{Pr}(X_0 = 0) \prod_{(0 \to i) \in \mathbf{x}} \mathbf{Pr}(X_{t+1} = i \mid X_t = 0) \prod_{\substack{(i \to j \to k) \in \mathbf{x} \\ j \neq 0}} \mathbf{Pr}\left(X_{t+1} = k \mid X_{t-1} = i, X_t = j\right)\\
&\quad = \arg\min_{x} \sum_{(0,i) \in A} -\log \mathbf{Pr}(X_{t+1} = i \mid X_t = 0)\, x_{0i}\\
&\qquad\qquad + \sum_{\substack{(i,j) \in A,\,(j,k) \in A \\ j \neq 0}} -\log \mathbf{Pr}\left(X_{t+1} = k \mid X_{t-1} = i, X_t = j\right)\, x_{ij}\, x_{jk}.
\end{aligned}$$

If we now define $\hat{p}_{ijk} = -\log \mathbf{Pr}(X_{t+1} = k \mid X_{t-1} = i, X_t = j)$, and $\hat{p}_{ij} = -\log \mathbf{Pr}(X_{t+1} = j \mid X_t = i)$ as before, then we obtain the following CVRP formulation:

$$\begin{aligned}
&\min_{x} \sum_{(0,i) \in A} \hat{p}_{0i} x_{0i} + \sum_{\substack{(i,j) \in A \\ j \neq 0}} \sum_{\substack{(j,k) \in A \\ j \neq 0}} \hat{p}_{ijk} x_{ij} x_{jk} \qquad (13)\\
&\text{s.t.} \quad \text{Constraints } (2) - (7).
\end{aligned}$$

Note how we are still using the same x_{ij} decision variables. The objective function, however, is now a quadratic function and hence the problem becomes a Mixed Integer Quadratic Problem (MIQP). There are well-known techniques to linearize the quadratic term and obtain an MIP by introducing additional constraints and decision variables, e.g., by the McCormick inequalities [31]. Many mathematical programming solvers for MIQP also exist. However, it is reasonable to expect that optimizing with this objective function of the second order Markov chain will be computationally harder than optimizing with the linear objective function of the first order model.

We now explain how to learn the transition probability matrix from historical solutions (Section 3.3), followed by different ways of weighing the instances (Section 3.4). Finally, in Section 3.5 we discuss how to combine a learned probability matrix with a distance-based probability matrix.

3.3 Learning the transition probability matrix

We now outline the main approach that we propose for estimating the transition probability matrix from historical solutions. We assume given a dataset $\mathcal{H} = \{(V^t, m^t, z^t, \mathbf{x}^t)\}$ of instances with each instance a tuple where t is a timestamp (e.g., a specific day in case of daily vehicle routing), V^t is the set of stops served by the solution at timestamp t, m^t is the number of vehicles used, $\mathbf{x}^t$ is the solution created by an expert or the actual driven routes extracted from an on-board system, and z^t are additional problem parameters such as the demand of every stop, or some other known constraint parameters.

Note that V^t, as well as m^t and z^t, can change from instance to instance. The approach we propose assumes that while V^t indeed changes from day to day, there will always be some overlap with other days. In general, the set of stops is assumed to be composed of *regular* and occasional or *ad hoc* stops. These assumptions are necessary to learn preferences between pairs of stops. Even in cases where there is no actual repetition of stops, it may still be

possible to extract patterns that capture the expert's preferences. Rather than focusing solely on individual stops, the analysis can be expanded to include stops with shared characteristics, such as geographic proximity or similar service time intervals (in the case of time windows). By examining stops that exhibit similar properties, valuable insights can be gained into the expert's preferences and underlying patterns.

Probability estimation

The basic idea of our approach is to estimate all the conditional probabilities $\mathbf{Pr}(X_{n+1} = j \mid X_n = i)$ over the set of all stops in the data: $V^{all} = \bigcup_t V^t$. The conditional probability of a vehicle moving from stop i to stop j is:

$$\mathbf{Pr}(X_{n+1} = j \mid X_n = i) = \frac{\mathbf{Pr}(X_{n+1} = j,\ X_n = i)}{\mathbf{Pr}(X_n = i)},$$

with $\mathbf{Pr}(X_n = i) = \sum_k \mathbf{Pr}(X_{n+1} = k,\ X_n = i)$.

To compute this, we will count how often two stops follow each other in the data. We denote by $[\![\cdot]\!]$ the Iverson bracket which returns the value 1 if the statement inside the bracket evaluates to *true*, and 0 otherwise. We define the frequency of a transition $(i \rightarrow j)$ in dataset $\mathcal{H}$ as:

$$f_{ij} = \sum_t [\![(i \rightarrow j) \in \mathbf{x}^t]\!]. \tag{14}$$

Now we can empirically estimate the conditional probabilities from the data as follows:

$$\mathbf{Pr}(X_{n+1} = s_j \mid X_n = s_i) \ = \ \frac{f_{ij}}{\sum_k f_{ik}}. \tag{15}$$

Laplace smoothing

To account for the fact that the number of samples may be small, and some f_{ij} may be zero, we can smooth the probabilities using the Laplace smoothing technique [14, 26, 44]. The term smoothing describes methods that adjust the maximum likelihood estimate of probabilities by setting low probabilities upward and high probabilities downward. In particular, Laplace smoothing adds a non-negative term to each event (in this case to each transition), which reduces the impact of data sparseness arising in the process of building the transition matrix. As a result of smoothing, these arcs are given a small, non-negative probability, thereby eliminating the zeros in the resulting transition matrix. The removal of zeros is also desirable from a computational perspective as it deletes the occurrence of "log(0)" during maximum likelihood computation.

Conceptually, with λ as the smoothing parameter ($\lambda = 0$ corresponds to no smoothing), we add λ observations to each event. The conditional probability estimation therefore becomes:

$$\mathbf{Pr}(X_{n+1} = s_j \mid X_n = s_i) \ = \ \frac{f_{ij} + \lambda}{\sum_k (f_{ik} + \lambda)}. \tag{16}$$

First-order estimation algorithm

Algorithm 1 shows the algorithm for estimating the first order probability transition matrix. The dimension of the matrix, that is, the size of the total set of unique stops, is determined in

Algorithm 1 Estimating a first order transition matrix from historical instances.

Input: A dataset $\mathcal{H} = \{(V^t, m^t, z^t, \mathbf{x}^t)\}$ where we will assume that the timestamps t have been replaced by integer values $1 \ldots |\mathcal{H}|$ in a way that respects the ordering of the timestamps; a weight w_t per data instance, where the default value is $w_t = 1$; and the Laplace smoothing parameter $\lambda \geq 0$.

1. Determine the total set of stops $V^{all} = \bigcup_t V^t$ and let $\mu = |V^{all}|$.

2. For each $(V^t, m^t, z^t, \mathbf{x}^t) \in \mathcal{H}$, construct an adjacency matrix $\mathbf{A}^t_{\mu \times \mu} = [a^t_{ij}]$, where

$$a^t_{ij} = [\![(s_i, s_j) \in \mathbf{x}^t]\!]. \tag{17}$$

3. Build the arc transition frequency matrix $\mathbf{F}_{\mu \times \mu}$ with the weights w_t and the adjacency matrices constructed in Step 2:

$$\mathbf{F} = \sum_t w_t \mathbf{A}^t. \tag{18}$$

4. Apply the Laplace smoothing technique to $\mathbf{F}_{\mu \times \mu} = [f_{ij}]$ to get the final probability estimates $\hat{\mathbf{P}}_{\mu \times \mu} = [\hat{p}_{ij}]$:

$$\hat{p}_{ij} = \frac{f_{ij} + \lambda}{\sum_k (f_{ik} + \lambda)}. \tag{19}$$

Output: Transition matrix $\hat{\mathbf{P}}_{\mu \times \mu} = [\hat{p}_{ij}]$, where $\hat{p}_{ij} = \mathbf{Pr}(X_{n+1} = s_j \mid X_n = s_i)$.

Step 1. In Step 2, an adjacency matrix is constructed for each historical instance. A frequency matrix is constructed in Step 3 by computing the (weighted) sum of all the adjacency matrices (18). By default, $w_t = 1$ for all instances; more advanced schemes will be discussed in Section 3.4. Finally, during normalisation in Step 4, Laplace smoothing is applied if $\lambda > 0$.

Second-order estimation algorithm

The second order transition matrix can be analogously constructed by building a three-dimensional matrix $\mathbf{A}^t_{\mu \times \mu \times \mu} = [\, a^t_{ijk} \,]$ in Step 2, where $a^t_{ijk} = [\![(s_i, s_j, s_k) \in \mathbf{x}^t]\!]$. Laplace smoothing is then applied by dividing each element of the frequency matrix $\mathbf{F}_{\mu \times \mu \times \mu} = [\, f_{ijk} := \sum_t [\![(i \to j), (j \to k) \in \mathbf{x}^t]\!] \,]$ with the row sum to obtain the transition matrix $\hat{\mathbf{P}}_{\mu \times \mu \times \mu} = [\, \hat{p}_{ijk} \,]$, where

$$\hat{p}_{ijk} = \mathbf{Pr}(X_{n+1} = s_k \mid X_{n-1} = s_i, X_n = s_j) = \frac{f_{ijk} + \lambda}{\sum_l (f_{ijl} + \lambda)}.$$

We also estimate the conditional probabilities of leaving from the depot:

$$\hat{p}_{0i} = \frac{f_{0j} + \lambda}{\sum_k (f_{0k} + \lambda)}.$$

With these estimates, we can use the above-defined optimisation formulation to find the maximum likelihood VRP solution.

3.4 Every day is different

The estimation method described above assumes that each instance is equally important, and that the VRP solution of each instance is independently drawn from the same distribution. However, we know that this is not entirely true: human planners learn from day to day, with more recent experiences typically closer to mind. Furthermore, because the set of stops changes each day, the similarity of the current instance to previous instances may also change how choices on the current instance will be made.

We identify two types of *context* that change the importance of previous instances—the temporal context and the similarity context. The temporal context is known in machine learning as *concept drift* [22]. The concept of similarity is central in many machine learning and data mining approaches. We now discuss two modifications of the above transition probability estimation algorithm that will take these contexts into account.

To this end, we assume that we are currently at timestamp T. Also, we are given a set of stops V^T, number of vehicles m^T, and other parameters z^T. That is, we have a new unseen tuple (V^T, m^T, z^T) for which we are to determine the corresponding $\mathbf{x}^T$. We also have a set of historical instances until timestamp T that we denote with $\mathcal{H} = \{(V^t, m^t, z^t, \mathbf{x}^t)\}_{t=1}^{T-1}$.

To take the temporal and similarity aspects into account, we will define a *prior* on each historical training instance, based on the current tuple (V^T, m^T, z^T). More specifically, we use a *weighing* scheme where we define a weight w_t for each historical instance in $\mathcal{H}$ based on the properties of the current tuple.

Table 1 provides an overview of the three types of prior: uniform, time-based, and similarity-based, with other possible variations within each type.

Time-based weighing

In machine learning, it is well known that temporally ordered data can have *concept drift* [22], that is, the underlying distribution can change over time.

To account for this, we can use a time-based weighing scheme where older instances are given smaller weights, and newer instances larger ones. Assuming the timestamps are (replaced by) integer values in a way that respects the same ordering, we can weigh the instances as:

$$w_t = \frac{t}{T}. \tag{20}$$

This assumes a *linearly* increasing importance of instances, with the oldest (first) instance having weight $1/T$ and the latest instance weight $(T-1)/T$. We can increase the importance of the newer instances by squaring the weights: $w_t = (t/T)^2$ or more generally, $w_t = (t/T)^a$ for some value a.

Exponential smoothing

A further amplification of the importance of more recent instances can be done by considering an exponential weighing scheme, a popular approach in time series analysis and forecasting [15, 24].

In principle, exponential smoothing uses a weighted average of the most recent observations and the forecasted data. In general, let $\hat{f}_T$ be the estimated data up to, but not including time period T. Then, using the data f_T of the current timestamp, the following gives the

Table 1 An overview of the proposed weighing schemes

Name	Weights	Squared weights	Exponential Weights (EXP)
Uniform (UNIF)	$w_t = 1$	—	
Time-based (TIME)	$w_t = t/T$	$w_t = (t/T)^2$	$w_t = \alpha(1-\alpha)^{T-t}$
Similarity-based (SIMI)	$w_t = J(V^t, V^T)$	$w_t = J(V^t, V^T)^2$	

forecast for the next time period:

$$\hat{f}_{T+1} = \alpha f_T + (1-\alpha)\hat{f}_T, \tag{21}$$

where the smoothing parameter $\alpha \in (0, 1)$ is a weight assigned to the most recent data. Furthermore, the expansion of (21) yields

$$\hat{f}_{T+1} = \sum_{t=1}^{T} \alpha(1-\alpha)^{T-t} \; f_{T-t}. \tag{22}$$

This weighing scheme is called exponential smoothing because the weights in (22) decline exponentially with time.

We can apply the same exponential smoothing on the frequency matrices, resulting in $\mathbf{F} = \sum_t \alpha(1-\alpha)^{T-t}\, \mathbf{A}^t$ and hence:

$$w_t = \alpha(1-\alpha)^{T-t}. \tag{23}$$

Similarity-based weighing

The stops in each instance typically vary, and the presence or absence of different stops can lead to different decision behaviors. To account for this, we can use the similarity between the set of stops of the current instance and the set of stops of each historical instance as a prior. The goal is to assign larger weights to training instances that are more similar to the test instance, and smaller weights if they are less similar.

The similarity of two stop sets can be measured using the Jaccard similarity coefficient. The Jaccard similarity of two sets is defined as the size of the intersection divided by the size of the union of the two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{24}$$

for two non-empty sets A and B. The Jaccard similarity coefficient is always a value between 0 (no overlapping elements) and 1 (exactly the same elements).

Hence, we can use the following distance-based Jaccard similarity weighing:

$$w_t = J(V^t, V^T). \tag{25}$$

To further amplify the differences in the weights, we can also use the squared Jaccard coefficient $w_t = J(V^t, V^T)^2$ or in general, $w_t = J(V^t, V^T)^a$.

3.5 Mixing distances and preferences

We have discussed learning preferences from historical solutions and how to optimize over them, leading to the most likely VRP solution in expectation. However, this uses *only* the historical preferences but never reasons at the level of distances traveled (kilometers). Using only the probability matrix can hence lead to purely *mimicking* the human behaviour, instead of reasoning and optimizing over both preferences and the impact on the driven kilometers.

To face this issue, we define a *distance-based probability* distribution based on the *softmax* function on the distances between stops. The larger the distance between stops i and j, the lower is the probability of that transition. This probability is defined as:

$$\hat{d}_{ij} \;=\; \mathbf{Pr}_{dist}(X_{n+1}{=}s_j \mid X_n{=}s_i) \;=\; \frac{e^{-d_{ij}}}{\sum_k e^{-d_{ik}}}. \tag{26}$$

The main goal is to have a comparable measure between the driver's preferences and the cost of driving long distances. In Appendix A, we show that this definition of distance-based probabilities would produce, under some mild conditions, the same set of solutions as the standard CVRP formulation with the objective of minimizing the total distance.

Combining transition probability matrices

Given transition probability matrices $\hat{\mathbf{P}} = [\hat{p}_{ij}]$ and $\hat{\mathbf{D}} = [\hat{d}_{ij}]$, we can take the convex combination as follows:

$$\hat{c}_{ij} = \beta \hat{p}_{ij} + (1 - \beta)\hat{d}_{ij}. \tag{27}$$

Taking $\beta = 1$ corresponds to using purely the history-based transition probabilities, while $\beta = 0$ will use only distance-based probabilities, with values in between resulting in a combination of the two probabilities.

Note that this approach places no conditions on how the history-based transition probability matrix $\hat{\mathbf{P}} = [\hat{p}_{ij}]$ is computed, and hence is compatible with Laplace smoothing and weighing during the construction of $\hat{\mathbf{P}}$.

4 Experiments

We evaluate the performance of our approach using two sets of data. We first present our numerical results on data consisting of synthetic networks generated from benchmark CVRP instances of VRPLIB. This is followed by a case study where we apply our proposed algorithms on actual company data.

The numerical experiments were performed using Python 3.7.4 and the CPLEX 12.9 solver with the default setting, on a Lenovo ThinkPad X1 Carbon with an Intel Core i7 processor running at 1.8GHz with 16GB RAM.

In our experiments, each VRP is solved exactly using CPLEX and the MIP formulation in (10). We use an exact formulation in order to isolate the sources of errors during the training process. By solving each VRP to optimality, we are able to isolate the errors coming from the learned cost vectors from those that come from computing a heuristic (sub-optimal) solution.

4.1 Experiments on synthetic data

Generation of random instances

As described in Chen et al. [13], synthetic networks with 5, 10, 15, 30 and 50 vertices are randomly generated. Instances of 5, 10 and 15 vertices are generated from a single instance of VRPLIB containing 20 vertices; instances of 30 and 50 are from a VRPLIB instance with 75 vertices. We use the smaller instances to directly compare our approaches with the inverse optimization approach in Chen et al. [13], and the larger ones to test the scalability of our approaches on synthetic data.

The number of instances generated for each network size is 1000, except for $|N| = 5$ where we generate 2000 instances. Each set of instances is split into a training set T and a test set T', as shown in Tables 2 and 3.

The expert solution for each instance is obtained by optimizing with respect to a *preference cost matrix* $\mathbf{D}' = [d'_{ij}]$, which represents the preferences of the expert. From the absolute

Table 2 Markov models (UNIF, SIMI) against the inverse optimization approach

$\|N\|$	T	T'	Inverse optimization			Markovian (UNIF)			Markov (SIMI)		
			Runtime (s)	Edit dist θ	Sol error $e_2(\%)$	Runtime (s)	Edit dist θ	Sol error $e_2(\%)$	Runtime (s)	Edit dist θ	Sol error $e_2(\%)$
5	100	100	38	**1.22**	**7.16**	1.89	2.14	7.24	**1.72**	2.04	7.57
5	500	100	225.6	**0.76**	5.01	**1.93**	1.67	5.08	1.95	1.54	**4.76**
5	1000	100	606.1	**0.62**	4.46	**1.81**	1.46	3.99	1.95	1.41	**3.56**
5	1500	100	1144.1	**0.64**	4.36	2.13	1.43	4.04	**2.11**	1.38	**3.63**
5	1900	100	1865.9	**0.58**	4.29	2.11	1.46	4.15	**1.88**	1.41	**3.87**
10	10	100	15.4	**1.77**	**7.42**	3.57	5.62	8.70	**3.31**	5.42	9.04
10	40	100	54.2	**2.06**	**4.8**	3.59	4.80	6.08	**3.55**	4.97	6.62
10	100	100	165.1	**1.9**	**3.81**	**3.46**	4.97	5.32	3.57	4.76	5.23
10	500	100	1992.7	**1.14**	3.08	3.58	3.44	**2.53**	**3.48**	3.95	3.13
10	900	100	5887.6	**1.01**	2.75	3.66	3.21	2.27	**3.59**	3.41	**2.25**
15	10	100	29.4	**4.51**	**5.85**	**6.47**	5.80	9.64	6.62	5.82	9.21
15	40	100	155.9	**5.09**	10.96	6.23	5.66	**8.07**	**6.06**	5.71	8.08
15	100	100	592.8	5.37	13.36	6.73	**5.22**	**7.95**	**6.33**	**5.22**	8.02
15	500	100	8357.7	8.04	25	6.01	**5.21**	**6.90**	**5.79**	**5.21**	**6.90**
15	900	100	24791.4	7.32	25.03	5.89	**5.22**	7.08	**5.86**	**5.22**	**7.06**

In each row, values in bold represent the best-performing results for the respective network size $|N|$

Table 3 Markov models (UNIF, SIMI) on larger instances

$\|N\|$	T	T'	Markovian (UNIF)			Markov (SIMI)		
			Runtime (s)	Edit dist θ	Sol error $e_2(\%)$	Runtime (s)	Edit dist θ	Sol error $e_2(\%)$
30	10	100	**218**	15.96	22.47	220	**15.79**	**21.52**
30	40	100	**206**	**15.51**	**18.52**	274	15.77	20.13
30	100	100	**125**	**14.53**	**17.55**	130	15.09	19.10
30	500	100	**113**	**12.77**	**12.85**	**113**	13.06	14.47
30	900	100	120	**13.21**	**13.37**	**104**	13.29	15.54
50	10	100	2181	**34.07**	26.90	**2139**	34.18	**25.08**
50	40	100	2150	**31.76**	**24.73**	**2074**	31.82	25.63
50	100	100	**2627**	**30.96**	**22.19**	2667	31.69	23.08
50	500	100	3034	29.70	22.35	**3015**	**29.36**	**21.81**
50	900	100	4561	29.86	20.95	**2915**	**29.62**	**20.53**

In each row, values in bold represent the best-performing results for the respective network size $|N|$

distance matrix $\mathbf{D} = [d_{ij}]$ consisting of the pairwise distances of all available nodes, $\mathbf{D}'$ was derived as follows: i) A matrix of epsilons $\mathbf{E} = [\epsilon_{ij}]$ was first created, where each ϵ_{ij} is a random real number from the uniform distribution $U(0.8, 1.2)$; ii) Then $\mathbf{D}' = \mathbf{D} \odot \mathbf{E}$, where $\odot$ represents the Hadamard (element-wise) product of two matrices.

For $|N| = 5$ and 10, a single vehicle is used during optimization; two vehicles are used for $|N| = 15$, and three vehicles for $|N| = 30$ and 50. For each network size, the same 100 instances are used to test the model performance.

Evaluation metrics

In order to allow a direct comparison with the results in Chen et al. [13], we use exactly the same metrics used in the paper, namely:

i) Average edit distance (θ). The edit distance $\bar{d}$ between two routings is based on the *Levenshtein distance,* as described in Sörensen [36]. It is defined as the minimal number of edit operations (substitutions, insertions, deletions) required to transform one routing into the other. The average edit distance θ between our solutions $\mathbf{x}^{t'}$ on the test instances and the expert's solutions $\mathbf{x}^{*t'}$, $t' = 1, \ldots, T'$, is computed as

$$\theta = \frac{\sum_{t'=1}^{T'} \bar{d}\,(\mathbf{x}^{t'}, \mathbf{x}^{*t'})}{T'}.$$

ii) Average solution error (e_2). Solution error measures the suboptimality of our solutions $\mathbf{x}^{t'}$ when compared to the expert's solutions $\mathbf{x}^{*t'}$. The average is calculated as

$$e_2 = \frac{1}{T'} \sum_{t'=1}^{T'} \frac{D' \cdot \mathbf{x}^{t'} - D' \cdot \mathbf{x}^{*t'}}{D' \cdot \mathbf{x}^{*t'}}.$$

Results on synthetic data

We compare our Markovian approach with the uniform (UNIF) and similarity-based (SIMI) weighing schemes against the inverse optimization approach. UNIF and SIMI are the logical

choices in this experiment, as the randomly generated instances do not have a temporal structure. In contrast, time-based (TIME) and the exponential (EXP) weighing schemes require a chronological order of the instances representing the evolution of the decision-making process. This experiment is shown in Table 2. Results show that the Markovian approaches are faster in terms of computational time, with the difference becoming more dramatic as the size of the training set increases. While the routes obtained through inverse optimization have slightly lower edit distances, this advantage diminishes in larger instances ($N = 15$). As for solution error, we also observe a dominance of the Markovian approaches when the size of the training set increases. We remark that our Markovian approach is generally able to learn in larger instances, increasing the applicability of the learning in larger, more realistic data sets.

We also applied our SIMI and UNIF Markovian approaches on larger synthetic instances to examine their scalability. The results of this experiment, summarized in Table 3, reveal no clear dominance between the two weighing schemes. It is worth noting that learning preferences within a VRP structure inherits the inherent difficulties of solving the NP-hard VRP problem. Nevertheless, despite this challenge, our approach demonstrates its capability to learn with instances of 50 nodes and over a training set of 900 instances, requiring five times less computational time compared to the inverse optimization approach on instances with only 15 nodes (as shown in the last line of Table 2). These findings highlight the scalability of our approach and the potential of our methodology to address larger-scale vehicle routing problems.

4.2 Case study: results on actual company data

Description of the data

The historical data used in these experiments consist of daily routings collected within a span of nine months. The routings were generated and modified by the route planners to align with their local knowledge and preferences. As these are the actual routings used by the company in their operations, it is possible that the drivers also made adjustments to the routes. Each data is a numbered instance and the entire data is ordered by time.

Data instances are grouped by day-of-week including Saturday and Sunday. This grouping mimics the operational characteristic of the company. The entire data set is composed of 201 instances, equivalent to an average of 29 instances per weekday. Capacity demand estimates for each stop were provided by the company. In all case study experiments, capacity demands were taken into consideration. (See Appendix B for an experiment that examines how our approach performs without the capacity demands.)

Data visualization

Figure 1 shows the number of stops served per weekday during the entire experimental time period. A *concept drift* is clearly discernible starting Week 53, where a decrease in stop set size occurs. An average of 9 vehicles servicing 35 stops are used per instance in the data before drift, and 6 vehicles (25 stops) for the 73 instances after drift. This observation has prompted us to make explicit whenever we are using all the data from the *entire* period, or only data from the period *before the drift*.

The number of times each of the customer stops has appeared in the historical data is shown in Fig. 2, which exhibits a mix of regular and ad hoc (occasional) stops. At the extremes,

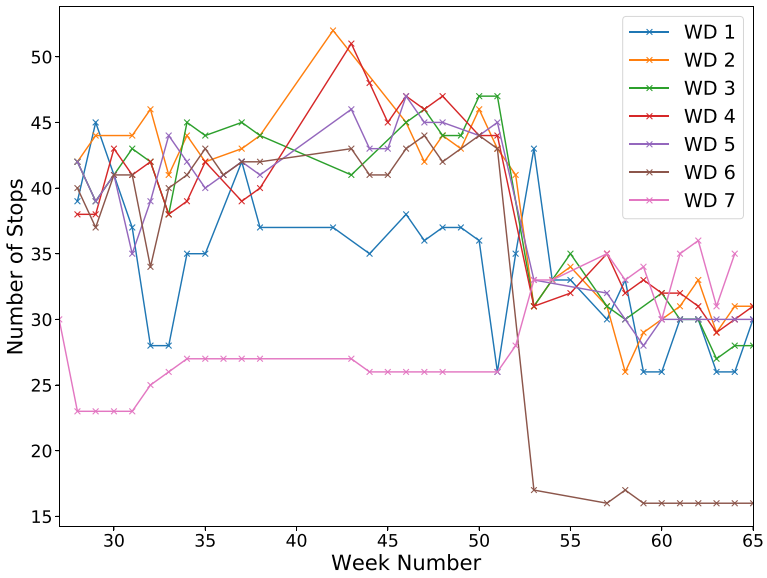


Fig. 1 No. of stops by weekday (WD)

14 out of the 73 unique stops (19.2%) have been serviced more than 195 (out of 201) times, while 30 (41.1%) have appeared in only ten or less instances.

Evaluation methodology

In a traditional machine learning setup, the dataset is split into a training set and a test set. The training set is used for training, and the test set for evaluation. This is called batch evaluation, as all test instances are evaluated in one batch. The best resulting model is then deployed. However, our data has a temporal aspect to it, namely the routing is performed everyday. Hence, each day one additional training instance becomes available, allowing us

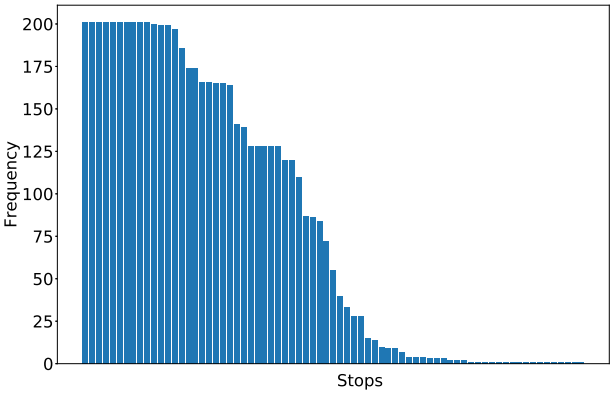


Fig. 2 Frequency of stops across data

Table 4 Initial training and test set sizes after 75% − 25% split

WD	Before drift		Entire data	
	Train	Test	Train	Test
1	14	5	23	7
2	12	5	21	7
3	11	5	19	7
4	13	5	22	7
5	14	5	22	7
6	14	5	22	7
7	15	5	23	7
Total	93	35	152	49

to *incrementally* grow the data and learn from it. Indeed, human planners also learn from prior experience and expand their knowledge day by day.

As this is the most sensible way in which such a system would be used, the system has to be evaluated in this way, i.e., by incremental evaluation (See Ade and Deshmukh [1] for more details). The incremental evaluation procedure is depicted in Algorithm 2 and the breakdown of the entire data set after the initial train-test split is shown in Table 4.

Algorithm 2 Training and testing with an incrementally increasing training set.

Input: A dataset $\mathcal{H} = \{(V^t, m^t, z^t, \mathbf{x}^t)\}$ with timestamps t as in Algorithm 1.
1. Start from an initial η training instances, e.g., $\eta = \lfloor 0.75\,|\mathcal{H}| \rfloor$ for a 75% − 25% initial split.
2. For $\sigma = \eta + 1, \ldots, |\mathcal{H}| - 1$ do:
   2.1. Build the probability transition matrix $\hat{\mathbf{P}}^\sigma$ on $\mathcal{H}^\sigma = \{(V^t, m^t, z^t, \mathbf{x}^t)\}_{t<\sigma}$ using Algorithm 1.
   2.2. Add to $\hat{\mathbf{P}}^\sigma$ all stops in V^σ that are not in $\mathcal{H}^\sigma$, with uniform probability.
   2.3. Solve CVRP (10) using $\hat{\mathbf{P}}^\sigma$ as in Section 3.2.
   2.4. Evaluate the CVRP solution against $\mathbf{x}^\sigma$.

On top of the evaluation measures introduced in the previous subsection, when making a comparison of the prediction accuracy of the proposed schemes, we evaluate the performance using two evaluation measures based on two properties of a VRP solution, namely the grouping of stops into routes (*Route Difference*) and the resulting chosen arcs (*Arc Difference*).

Route Difference (RD) indicates the percentage of stops that were incorrectly assigned to a different route. Intuitively, RD may be interpreted as an estimate of how many moves between routes are necessary when modifying the predicted solution to match the actual grouping of stops into routes. To compute route difference, a pairwise comparison of the routes contained in the predicted and test solution is made. The pair of routes with the smallest difference in stops is greedily selected without replacement. Incorrectly assigned stops are counted and the total number is taken as RD. The percentage is taken by dividing RD by the total number of stops in the whole routing.

Arc Difference (AD) measures the percentage of arcs traveled in the actual solution but not in the predicted solution. AD is calculated by taking the set difference of the arc sets of the test and predicted solutions, then dividing the value by the total number of arcs in the routing. Correspondingly, AD gives an estimate of the total number of modifications needed to *correct* the predicted solution.

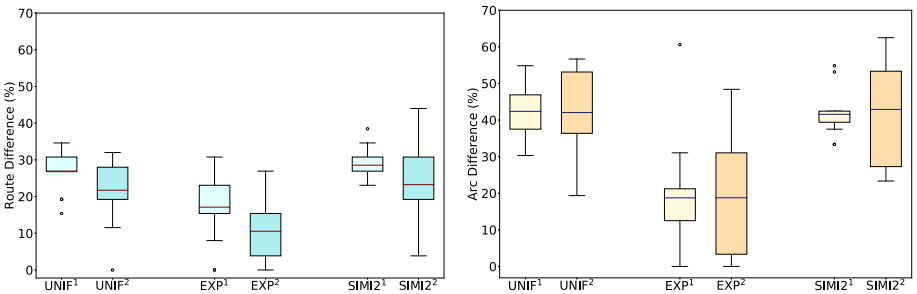


Fig. 3 First order (^{1}) versus second order (^{2}) model (data from entire period)

Experimental setup

The Laplace smoothing parameter (λ), convex combination parameter (β) and exponential smoothing parameter (α) take default values $\lambda = 1$, $\beta = 1$, and $\alpha = 0.7$. (See Appendix C for experiments on the parameter sensitivity of λ and α.)

Following the weighing scheme overview in Table 1, we denote uniform weighing by UNIF, TIME for time-based, EXP for exponential time-based weighing and SIMI for similarity-based weights. TIME2 and SIMI2 indicate squared weights. Furthermore, whenever necessary, we use superscripts (^{1}) and (^{2}) to distinguish when a weighing scheme is applied using the first or the second order model, respectively. For instance, UNIF^1 indicates uniform weighing using the first order model.

4.2.1 First order versus second order model

In this experiment, we tested the performance of our models on a subset, composed of three weekdays, of all data from the entire period using three schemes (UNIF, EXP, SIMI2)—one from each type of prior: uniform, time-based and similarity-based. From the results shown in (Fig. 3), we see that the second order model consistently produced better results in terms of route difference. However, this is not the case for arc difference, where aside from there being almost no change in prediction accuracy, the variance also increased. We assert that more data points are necessary in order to better evaluate the second order model.

Looking at the computation times (Table 5), we see a considerable disparity between the two models. While schemes using the first order formulation result to subsecond computation times, using the fastest scheme in the second order model (EXP^2) takes more than 2000 seconds for *each* test instance to be solved to optimality. We can hence remark that the slight

Table 5 Average runtimes for the different weighing schemes

Weighing scheme	UNIF^1	UNIF^2	EXP^1	EXP^2	SIMI2^1	SIMI2^2
Avg Runtime (s)	0.1016	4355.75	0.1455	2055.33	0.0922	9101.41

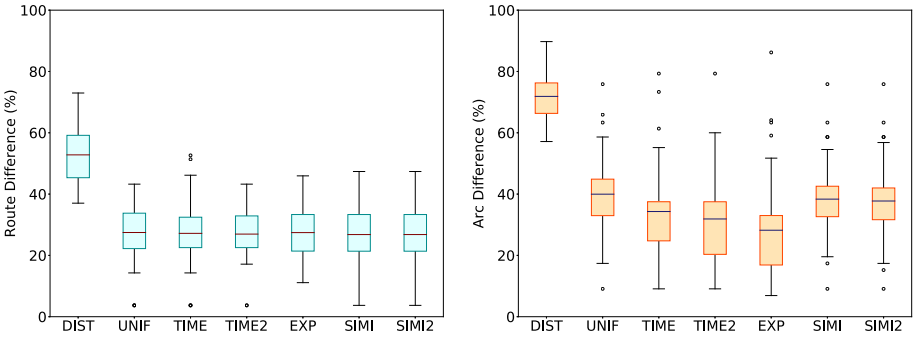


Fig. 4 Route and arc difference (period before drift)

improvement in route difference is, for practical purposes, reduced by the substantially longer running time of the second order model. Subsequently, we observe that while a higher-order model may offer a closer representation, it comes with increased computational complexity and data requirements without providing significant empirical gains, compared to the first-order model. Considering the substantial computational burden and the comparable solution quality, we believe that the first-order model is more practical for real-world applications.

For the succeeding experiments, we therefore focus our attention on further investigating the effects of the first order formulation.

4.2.2 Evaluation of weighing schemes in the first order model

In the next two experiments (Figs. 4 and 5), we do a wider comparison by investigating the performance of all our proposed weighing schemes (see Table 1)—UNIF, TIME, SIMI, and EXP, and TIME2 and SIMI2—in the first order model.

Figure 4 is on data before the drift (up to week 53 in Fig. 1). It shows that all the proposed schemes produced better estimates than DIST. There appears to be no significant difference in the results in terms of route difference. For arc difference, however, using time-based weighing (TIME, TIME2, EXP) considerably improved the solutions given by UNIF. Among all schemes, EXP gave the most accurate predictions. Hence, it can be deduced that more recent routings are more relevant when making choices in this case.

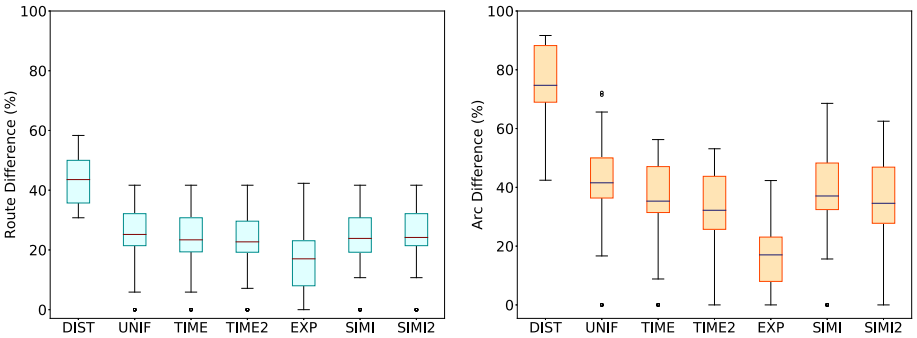


Fig. 5 Route and arc difference (entire period)

Results on data from the entire period (Fig. 5) exhibit a slightly different behavior. As before, all the schemes outperformed DIST. Notably in terms of arc difference, both the time-based and similarity-based schemes significantly outperformed UNIF. Also, in most cases, the schemes with the squared weights (TIME2, SIMI2) performed better than their counterparts (TIME, SIMI). As before, EXP gave the most accurate predictions among all schemes.

4.2.3 How the weighing schemes handle concept drift

The two succeeding experiments were conducted to observe the performances of the proposed schemes during a concept drift. Our motivation is that we want to determine how the prediction quality of each scheme evolves when there is a sudden change in data structure.

We consider two scenarios. The first case is when there is an abrupt drop in the number of stops. Observe that this is the case immediately after week 53 in Fig. 1. For this scenario, we trim our data set such that our 13 test instances, i.e., instances in the *tail* end of the data set, consist of 3 instances before the drift and 10 instances after the drift. As before, the probability matrix is trained on all data older than the ones contained in the test set. We plot the prediction accuracy of each scheme on each of the test instances.

The results of the first scenario are shown in Fig. 6, where instance 0 denotes the first instance after the drift. Naturally, in both route and arc difference, we observe a sharp rise in *error* percentage at instance 0. Especially with route difference, the error remained high for all the schemes (except EXP) until about instance 2 or 3, after which we see some improvement particularly for TIME2 and SIMI2. EXP, on the other hand, readjusted immediately after instance 0 and clearly outperformed the other schemes in terms of prediction accuracy and its ability to adjust to changes in data structure.

For the second scenario, we consider the case where there is a sudden rise in the number of stops. In order to use the same company data that we have, we simulate this case by reversing the time stamps of all the data instances. With the data in reverse order, we are able to simulate a drift where there is an increase in the number of stops, e.g., new stops that have not been seen before. As with the first case, we trim the data and select the 13 test instances *after* reordering the data. Compared to the first case, here all the schemes seemed to *adjust* more rapidly (Fig. 7) after instance 0. The increase in route and arc difference at instance 0 is unsurprisingly greater than in the previous scenario, as the schemes adjust with the new stops that were not seen before.

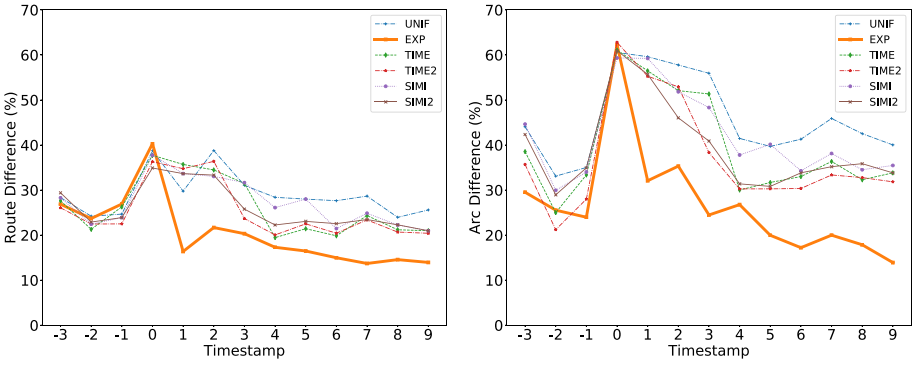


Fig. 6 Route and arc difference during concept drift (drop in number of stops)

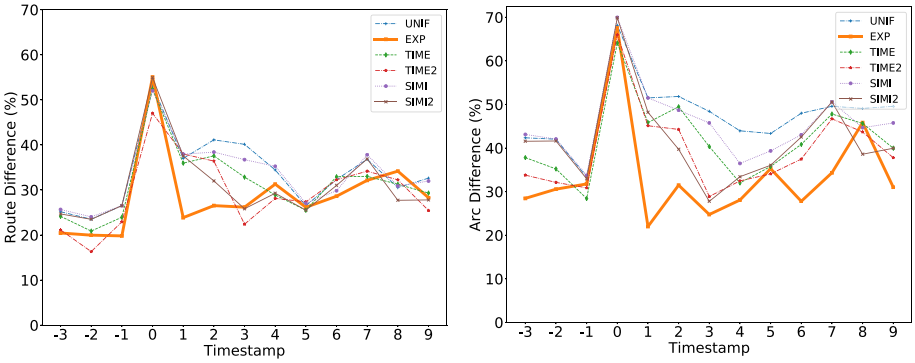


Fig. 7 Route and arc difference during concept drift (rise in number of stops)

The above experiments show that generally, all schemes manage to stabilize after the initial rise in error due to structural change. In both scenarios, TIME2 and SIMI2 restabilize faster than their counterparts, TIME and SIMI. Among all the schemes, UNIF clearly performed the worst, while EXP was the fastest to adjust.

4.2.4 Addition of distance-based probabilities

Up to this point, we have investigated the performance of our model using transition matrices made of only probabilities learned from the historical solutions. We expect, however, to gain further improvement when the learned transition probability matrix is combined with a distance-based probability matrix, the construction of which is described in Section 3.5. For the next experiment, we investigate how the final solution is affected by the different ways of mixing, i.e., varying values of the β parameter in (27).

Figure 8 shows the result for different β values, using EXP with the default value on data from the entire period. When combined with the learned probability matrix, we see that even with small values of β, we already get better results than using distances alone, especially in terms of arc difference.

While the boxplots alone seem to indicate that larger β values lead to small but consistent improvements in RD and AD, we must warn that the arc and route differences with respect to the actual solution tell only one side of the story. In the accompanying table below the figures (Table 6), we show the average RD/AD as well as the average total distance driven. Looking

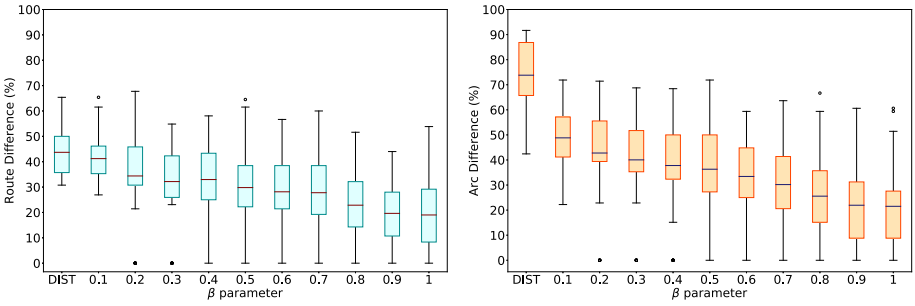


Fig. 8 Route and arc difference for varying values of β (entire period)

Table 6 EXP results for varying values of β (entire period)

		β parameter										
	(DIST)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	(Actual Sol)
RD	43.70	41.22	34.39	32.17	32.97	29.80	28.14	27.78	22.86	19.64	19.01	0
AD	73.81	48.82	42.77	40.04	37.78	36.31	33.43	30.19	25.57	21.95	21.52	0
Avg Total Dist (km)	395.42	396.67	400.31	403.79	403.32	403.80	406.44	409.50	414.46	418.87	427.41	412.13
Avg Runtime (s)	5958.66	223.66	162.60	102.54	15.47	2.26	0.73	0.42	0.22	0.09	0.11	-
Avg Edit Dist (θ)	24.00	19.60	16.76	15.96	15.06	14.00	12.96	12.53	10.22	8.92	8.49	0

closely at the kilometers, we see that while higher values of β generally lead to lower percent differences, they also increase the number of kilometers and do so beyond the number of kilometers of the actual solution. Using a β value of 0.8 shows that on average, this leads to a good RD/AD accuracy as well as an average number of kilometers roughly equal to that of the actual solutions. For solutions that optimize *both* the preferences and the total distance, we hence recommend a rather large β value, such as $\beta = 0.8$, i.e., a mix of 80%-preference and 20%-distance probabilities. Alternatively, we propose generating a set of diverse routes using different β values. By providing this range of options, we let the domain expert choose the route that best aligns with their priorities, whether their focus is on minimizing the number of kilometers or optimizing their preferences. This approach allows for a more personalized selection process, tailored to individual preferences and requirements.

4.2.5 Detailed example

A visual illustration of the way the routes are predicted by the transition matrix can be observed in Figs. 9, 10, 11, 12, 13, 14. Note that these figures are not drawn to scale. The main focus of the visualizations should be on understanding the sequence of stops and the overall routing rather than the precise distances depicted. Figure 9 shows a map of 24 stops to visit using three delivery vehicles. The depot is denoted by 0. Also shown in Fig. 9 is the solution obtained by using the standard distance-based approach. The actual routing used by the company is shown in Fig. 10. Notice that except for the slight similarity in the way the stops are clustered into routes, the two routings appear to be almost completely different.

From the historical data of the same weekday as the actual solution, we construct a transition probability matrix. The first order matrix learned with the simplest scheme (UNIF) is visualized in Fig. 11, where darker arcs indicate higher probabilities. It can be observed that the image shows a clear structure, with distinct connections, e.g., to the furthest stops, but also a higher variability in the denser regions and near the depot. Figure 12 shows the first order solution constructed with the probability matrix of Fig. 11. We can see that the solution

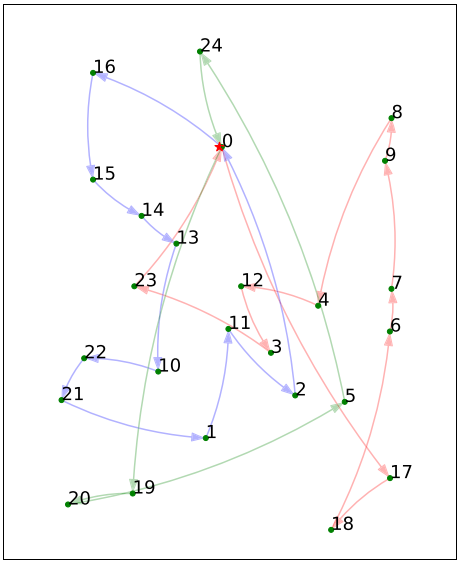
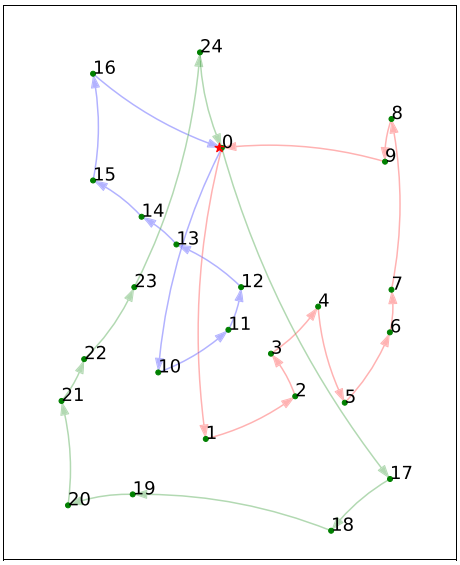


Fig. 9 Distance-based solution

Fig. 10 Human-made solution



was able to capture key structural parts while making trade-offs elsewhere to come up with a global solution, e.g., making a connection from stop 11 to 16. The actual human-made solution, on the other hand, was made with a number of distinct choices which cannot be easily predicted just by looking at the probability matrix map, e.g., the direct connections from 0 to 17, and 23 to 24.

The solution obtained by adding distance-based probabilities to the first order model is displayed in Fig. 13. It can be observed that the solution is able to keep the general structure of the human-made solution, while also avoiding the longer connections (e.g., 11 to 16, and 15 to 22) made by the purely preference-based first order model.

Figure 14 shows the solution when optimising with the second order model, which improved the route difference of the first order solution (RD in Table 7). Also from the

Fig. 11 Learned first order probabilities

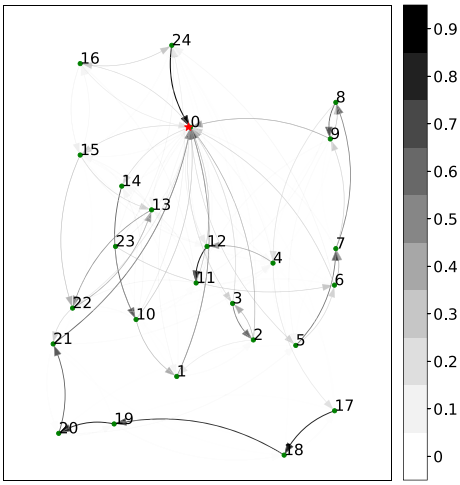
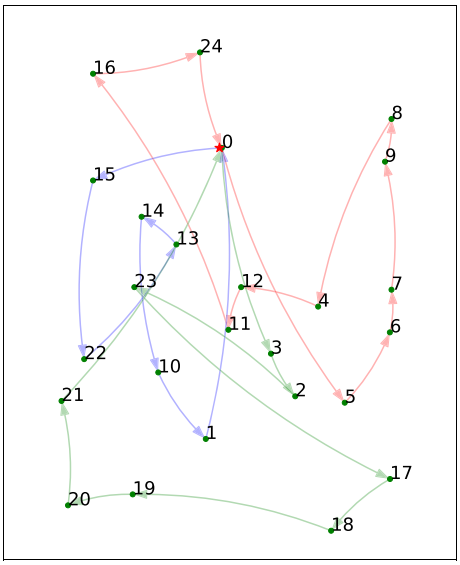


Fig. 12 First order solution



detailed table of results, we can see that in this particular example, the first order with β model was able to most closely capture the general structure of the human-made solution. Obviously, this is not always the case (see Fig. 8 in Section 4.2.4). Nevertheless, looking at the total distances displayed in Table 7, here we observe that adding distance-based probabilities not only preserved the structure but also offered a solution with fewer kilometers. Hence, it does not merely mimic the human planners but is also able to take preferences into account *while* still optimizing the total distance.

Fig. 13 First order solution with $\beta = 0.8$

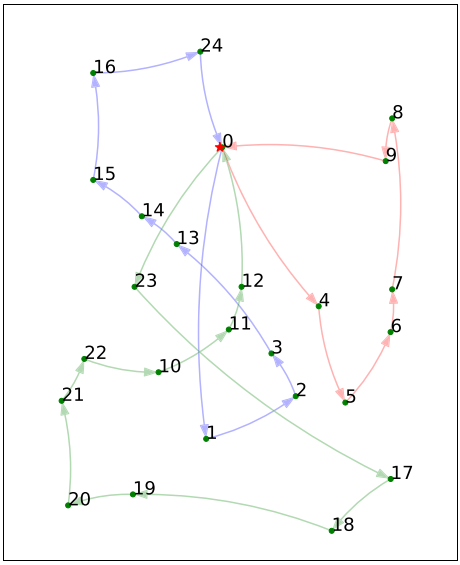
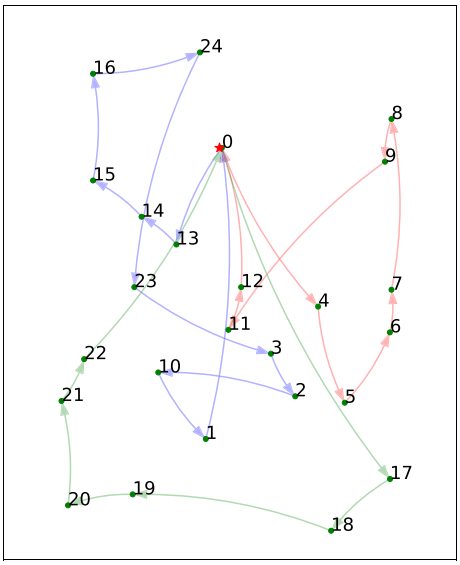


Fig. 14 Second order solution



5 Concluding remarks

One of the crucial first steps in solving vehicle routing problems is explicitly formulating the problem objectives and constraints. Oftentimes in practice, the optimized routing takes into account not only time and distance-related factors, but also a multitude of other concerns, which remain a tacit knowledge of the route planners. Specifying each sub-objective and constraint may be an intractable task. Moreover, as we have observed in practice, computed solutions seldom fully satisfy the wishes of the route planners and all involved stakeholders.

This paper studied the potential of learning the preferences of the route planners from historical solutions in solving VRP. Specifically, we presented an approach to solving the VRP which does not require a full explicit problem characterization. Inspired by existing research on exploiting Markov models to predict individual route choices, we developed an approach that learns a probabilistic preference model based on historical solutions. This probabilistic model can subsequently be optimized to obtain the most *likely* routing for an entire fleet. We have shown how this approach is capable of learning implicit preferences from real data, resulting in more accurate solutions than using distances alone. The algorithm performs well even without capacity demands, confirming its ability to learn patterns in the underlying solution structure.

Table 7 Detailed results for example routing

	DIST	First order	First order ($\beta = 0.8$)	Second order	Actual (Human-made)
RD	36.00	32.00	28.00	28.00	0
AD	81.48	70.37	25.93	72.63	0
Total Dist (km)	337.12	352.71	346.02	359.17	353.32
Runtime (s)	9938.91	0.2642	0.2580	21606.17	-

Our work extends beyond a first order Markov model and we have provided a more general formalization, which includes a second order Markov model, for learning and optimizing preferences. We have seen that the second order model improves the accuracy in terms route difference. This improvement, however, does not translate to arc difference. Furthermore, the second order model is expensive in terms of computational burden. Here, we assert that a richer database is necessary to better evaluate the performance of higher order models.

We also presented an approach which combines distance-based and preference-based probabilities. This approach has the advantage of being robust to changing customer sets and computationally fast due to the sparsity of the resulting cost matrix. By mixing the two probabilities, we obtained more robust results without sacrificing computational times. Other objectives may also be combined with our approach, as user and expert preferences can come in various forms. For example, route planners may have preferences over the order in which stops are visited. Customers may prefer to be serviced at a particular time of day. Meanwhile, drivers may have preferences of a more "global" scope, such as shorter working hours or equally distributed travel distances among drivers. We remark that the focus of our proposed Markovian approach is more towards learning "local" preferences, i.e., on the sequencing of stops. We argue, however, that the approach can be readily combined with other sub-objectives and constraints to deal with the global preferences. For example, a parameter can be added to the model to measure fairness among drivers. Fairness can be measured in terms of travel distance or time; the weight of the parameter may be learned from the historical solutions or adjusted to find a compromise between, e.g., preferences, the overall route length, and balancing of the routes.

Results on real data have been encouraging. Validation on other real-life data, however, should be considered for generalization, as other data may have more (or less) structural assumptions. Our proposed first order approach could be plugged into existing VRP software today, but as with all predictive techniques there should be human oversight to avoid unwanted bias. It is important to note that, although our approach effectively incorporates the standard hard constraints of the CVRP, there may be additional domain-specific hard constraints that our proposed model cannot explicitly learn. Future research should focus on addressing these constraints to further enhance the applicability and effectiveness of the proposed methodology. Furthermore, exploring advanced techniques and alternative modeling approaches will be crucial in order to reduce the observed route and arc differences between routes during evaluation, as well as improve the overall performance of the proposed methodology. This will allow for a more comprehensive and adaptable solution to real-world vehicle routing problems.

Future work on the routing side will involve applications to richer VRP models, e.g., problems involving time windows, multiple deliveries, etc. On the learning side, the use of richer predictive models such as neural networks or higher order Markov models can be considered. Also, using separate learned or contextual models per vehicle or per driver is worth investigating. Due to the hardness of solving the VRP exactly, the use of more advanced exact methods and of heuristics necessary to deal with larger instances open an opportunity for research in this topic. While our experimental results demonstrate the effectiveness of our methods on synthetic and real data, further investigation is needed to evaluate their performance on larger instances. This will require new techniques to isolate the errors coming from the process of learning the cost vectors from those that result from the generation of sub-optimal solutions. Finally, extending the technique so that the user can be actively queried and learned from during construction is an interesting direction, e.g., to further reduce the amount of user modifications needed on the predicted solutions.

Appendix

A Distance-based probabilities

Here, we prove that minimizing the absolute distances in the standard CVRP is equivalent to minimizing the distance-based probabilities of (26) under some mild conditions. We write the solution using the distance-based probabilities as:

$$\arg\min_x \sum_{(i,j)\in A} \hat{d}_{ij} x_{ij} = \arg\min_x \sum_{(i,j)\in A} -\log\left(\frac{e^{-d_{ij}}}{\sum_k e^{-d_{ik}}}\right) x_{ij} = \arg\min_x \sum_{(i,j)\in A} -\left(\log e^{-d_{ij}} - \log \sum_k e^{-d_{ik}}\right) x_{ij} \tag{28}$$

$$= \arg\min_x \sum_{(i,j)\in A} d_{ij} x_{ij} + \sum_{i\in V} \sum_{j:(i,j)\in A} \log\left(\sum_k e^{-d_{ik}}\right) x_{ij}$$

$$= \arg\min_x \sum_{(i,j)\in A} d_{ij} x_{ij} + \sum_{i\in V\setminus\{0\}} \log\left(\sum_k e^{-d_{ik}}\right) \sum_{j:(i,j)\in A} x_{ij} + \log\left(\sum_k e^{-d_{0k}}\right) \sum_{j:(0,j)\in A} x_{0j}.$$

The first term in the final expression corresponds to the classical VRP where the total distance is minimized. Given that $\sum_{j:(i,j)\in A} x_{ij} = 1$ (2), the second term is always a constant. Hence, it does not play any role in the optimization above. The third term depends on the number of vehicles used in the routing $\mathbf{x}$.

Whenever the optimal solution utilizes all the vehicles, we have $\sum_{j:(0,j)\in A} x_{0j} = m$ (where m is the number of vehicles). As the third term also becomes a constant, (28) simply becomes $\arg\min_x \sum_{(i,j)\in A} d_{ij} x_{ij}$, which is the same as the original CVRP objective function.

From the operational point of view, in the company setting in which this work is applied, the number of available drivers each day is fixed. Allocating a route to each of the drivers does not entail any additional cost. Therefore, we can assume that the equality constraint is always active.

Otherwise, we can always scale up the softmax function with a parameter $\theta > 0$ in order to obtain $g(\theta) = \sum_k e^{-\theta d_{0k}} = 1$. In fact, $g(\cdot)$ is a continuous function such that $g(0) = |V|$ and $\lim_{\theta\to+\infty} g(\theta) = 0$. Hence, there exists a value $\theta^* > 0$ with the desired property. Thus, scaling up by θ^* makes the third term in (28) equal to 0, and the solution using the distance-based probabilities is always equivalent to the CVRP solution.

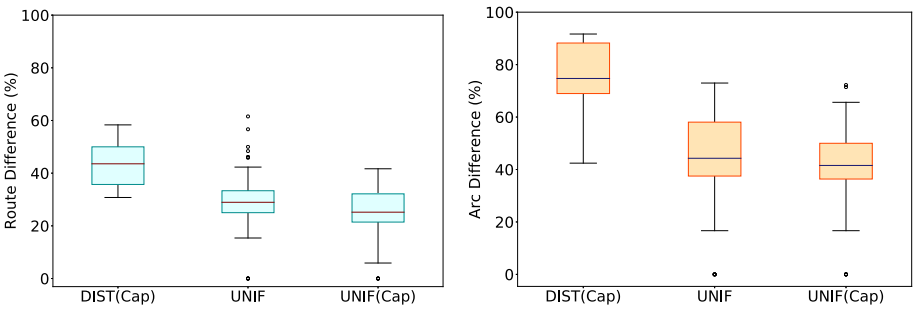


Fig. 15 UNIF without and with capacity (Cap) constraints (data from entire period)

Table 8 Average runtimes for UNIF without and with capacity constraints

Weighing scheme	DIST(Cap)	UNIF	UNIF(Cap)
Avg Runtime (s)	5958.66	0.1135	0.9730

B UNIF without and with capacity constraints

This experiment is done to compare the prediction accuracy of UNIF without and with the capacity (Cap) demand estimates (Fig. 15). The motivation is to investigate how UNIF will perform even without the capacity constraints. When evaluating without the capacity estimates, in order to keep the subtour elimination constraint (5), each q_i is assigned a value of 1 while using the number of stops as a fictive bound on the vehicle capacities, i.e., $Q = n$. As a baseline, we include the solution (DIST) obtained by solving the standard distance-based CVRP to near-optimality (5% optimality gap). The experiment is done on data from the entire period.

Results show that DIST is consistently outperformed by UNIF. Moreover, in both route and arc difference, we always notice some improvement when capacity estimates are taken into account. Remarkably, when using the transition probability matrices, we see that we can solve the VRP even *without* the capacity constraints and still get meaningful results. This shows the ability of the method to learn the structure underlying the problem just from the solutions.

As for the computation time (Table 8), DIST needed an average of almost 6000 seconds to solve *each* instance despite the imposed near-optimality relaxation. In contrast, both UNIF and UNIF(Cap) only took less than a second on average to obtain the optimal solutions, with UNIF(Cap) taking slightly more time due to the additional capacity constraints. Additionally, we observed that the learned probability matrices are much more sparse (containing more 0 or near-0 values) than the distance matrices.

C Parameter sensitivity

Laplace parameter λ To understand the effects of varying the Laplace parameter λ (Fig. 16), we performed an experiment on data from the entire period. Capacity demands were taken into account and EXP was selected on the basis of the results from the previous experiments.

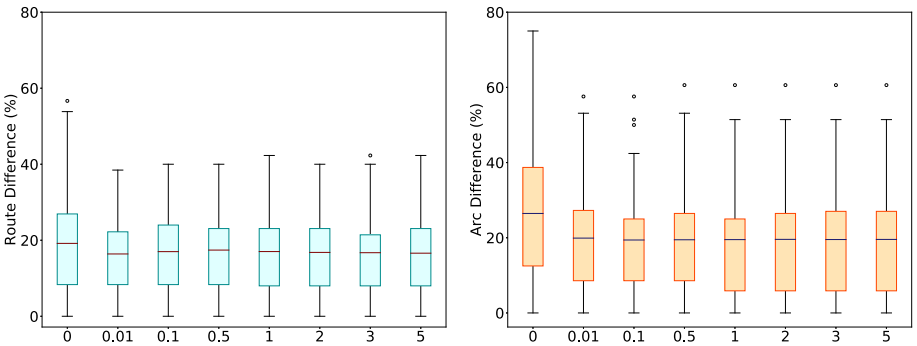


Fig. 16 Route and arc difference for values of Laplace parameter λ (entire period)

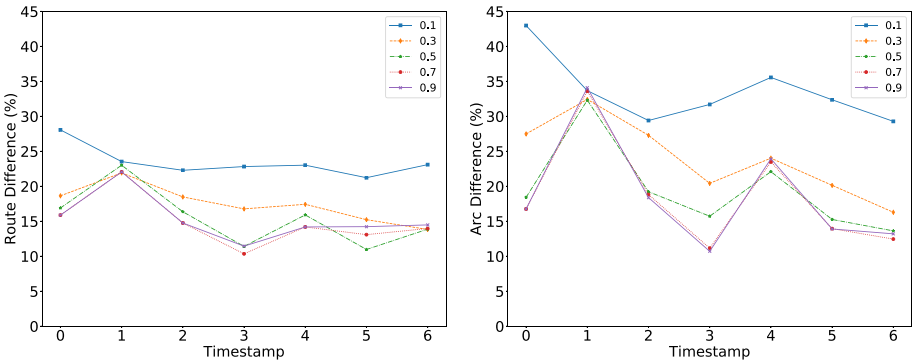


Fig. 17 Route and arc difference for varying values of EXP parameter (entire period)

It is interesting to note that EXP worked well even with no smoothing ($\lambda = 0$). It can also be observed that EXP produced slightly improved results when smoothing is applied. In general, however, we see that on our data, Laplace smoothing has very little effect.

EXP parameter α We examined the behavior of the model for different values between 0 and 1 of the EXP parameter α. Instead of looking at the average percent differences as we did for the preceding experiment on the Laplace parameter, here we opted to inspect more closely the evolution of the prediction across different time periods. In Fig. 17, each point on the graph represents the average route or arc difference when the model is tested on the test instances from the corresponding time period 0, 1, ..., or 6, with 0 being the oldest, and 6 the newest time period. The results of the experiment show that as α increases, prediction accuracy also increases. Accuracy appears to stabilize at $\alpha = 0.7$, which incidentally coincides with our choice of the parameter's default value.

Funding Víctor Bucarey was funded by the ANID Fondecyt Iniciacion grant no 11220864. This research also received partial funding from the FWO Flanders project grant FWO-S007318N (Data-driven logistics), the European Research Council (ERC H2020, Grant agreement No. 101002802, CHAT-Opt), and the Institute for the Encouragement of Scientific Research & Innovation of Brussels (Innoviris, 2021-RECONCILE).

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Ade, R. R., & Deshmukh, P. R. (2013). Methods for incremental learning: a survey. *International Journal of Data Mining & Knowledge Management Process, 3*(4), 119.
2. Ait Haddadene, S. R., Labadie, N., & Prodhon, C. (2019). Bicriteria Vehicle Routing Problem with Preferences and Timing Constraints in Home Health Care Services. *Algorithms, 12*(8), 152.
3. Ames, C. (1989). The Markov process as a compositional model: A survey and tutorial. *Leonardo, 4*, 175–187.
4. Ashbrook, D., & Starner, T. (2003). Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing, 7*(5), 275–286.
5. Beldiceanu, N., & Simonis, H. (2011). "A constraint seeker: Finding and ranking global constraints from examples." In: International conference on principles and practice of constraint programming. Springer, pp. 12–26
6. Beldiceanu, N., & Simonis, H. (2012). "A model seeker: Extracting global constraint models from positive examples." In: International conference on principles and practice of constraint programming. Springer, pp. 141–157
7. Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). Neural combinatorial optimization with reinforcement learning. https://openreview.net/forum?id=rJY3vK9eg
8. Bessiere, C., Koriche, F., Lazaar, N., & O'Sullivan, B. (2017). Constraint acquisition. *Artificial Intelligence, 244*, 315–342.
9. Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., & Juan, A. A. (2015). Rich vehicle routing problem: Survey. *ACM Computing Surveys (CSUR), 47*(2), 32.
10. Canoy, R., & Guns, T. (2019). "Vehicle routing by learning from historical solutions." In: International conference on principles and practice of constraint programming. Springer, pp. 54–70
11. Ceikute, V., & Jensen, C. S. (2013). "Routing service quality-local driver behavior versus routing services." In: 2013 IEEE 14th international conference on mobile data management. Vol. 1. IEEE, pp. 97–106
12. Chang, K.-P., Wei, L.-Y., Yeh, M.-Y., & Peng, W.-C. (2011). "Discovering personalized routes from trajectories." In: Proceedings of the 3rd ACM SIGSPATIAL international workshop on location-based social networks, pp. 33–40
13. Chen, L., Chen, Y., & Langevin, A. (2021). An inverse optimization approach for a capacitated vehicle routing problem. *European Journal of Operational Research, 295*(3), 1087–1098.
14. Chen, S. F., & Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language, 13*(4), 359–394.
15. Cox, D. R. (1961). Prediction by exponentially weighted moving averages and related methods. *Journal of the Royal Statistical Society: Series B (Methodological), 23*(2), 414–422.
16. Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science, 6*(1), 80–91.
17. Deguchi, Y., Kuroda, K., Shouji, M., & Kawabe, T. (2004). *HEV charge/discharge control system based on navigation information*. SAE Technical Paper: Tech. rep.
18. Delling, D., Goldberg, A. V., Goldszmidt, M., Krumm, J., Talwar, K., & Werneck, R. F. (2015). "Navigation made personal: Inferring driving preferences from gps traces." In: Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information Systems, pp. 1–9
19. Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., & Rousseau, L.-M. (2018). "Learning heuristics for the tsp by policy gradient." In: Integration of constraint programming, artificial intelligence, and operations research: 15th international conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15. Springer, pp. 170–181
20. Dragone, P., Teso, S., & Passerini, A. (2018). Constructive preference elicitation. *Frontiers in Robotics and AI, 4*, 71.
21. Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research, 5*(1–2), 47–63.
22. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46*(4), 44.
23. Guo, C., Yang, B., Hu, J., Jensen, C. S., & Chen, L. (2020). Context-aware, preference-based vehicle routing. *The VLDB Journal, 29*, 1149–1170.
24. Harrison, P. J. (1967). Exponential smoothing and short-term sales forecasting. *Management Science, 13*(11), 821–842.
25. Irnich, S., Toth, P., & Vigo, D. (2014). "Chapter 1: The family of vehicle routing problems." In: Vehicle routing: problems, methods, and applications, second edition. SIAM, pp. 1–33
26. Johnson, W. E. (1932). Probability: The deductive and inductive problems. *Mind, 41*(164), 409–423.
27. Kool, W., Van Hoof, H., & Welling, M. (2019). "Attention, Learn to Solve Routing Problems!" In: International conference on learning representations. https://openreview.net/forum?id=ByxBFsRqYm

28. Krumm, J. (2008). "A Markov Model for Driver Turn Prediction." In: SAE 2008 world congress. Lloyd L. Withrow Distinguished Speaker Award
29. Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL), 54*(8), 811–819.
30. Letchner, J., Krumm, J., & Horvitz, E. (2006). "Trip router with individualized preferences (trip): Incorporating personalization into route planning." In: AAAI, pp. 1795–1800
31. McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I-Convex underestimating problems. *Mathematical Programming, 10*(1), 147–175.
32. Mor, A., & Speranza, M. G. (2020). "Vehicle routing problems over time: a survey." In: 4OR, pp. 1–21
33. Nazari M, Oroojlooy A, Snyder L, & Takác M. (2018). "Reinforcement Learning for Solving the Vehicle Routing Problem." In: Advances in Neural Information Processing Systems. Ed. by Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf
34. Picard-Cantin, É., Bouchard, M., Quimper, C.-G., & Sweeney, J. (2016). "Learning parameters for the sequence constraint from solutions." In: International conference on principles and practice of constraint programming. Springer, pp. 405–420
35. Potvin, J.-Y., Dufour, G., & Rousseau, J.-M. (1993). Learning vehicle dispatching with linear programming models. *Computers & Operations Research, 20*(4), 371–380.
36. Sörensen, K. (2007). Distance measures based on the edit distance for permutation-type representations. *Journal of Heuristics, 13*(1), 35–47.
37. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). "Graph Attention Networks." In: International conference on learning representations. https://openreview.net/forum?id=rJXMpikCZ
38. Vinyals, O., Fortunato, M., & Jaitly, N. (2015). "Pointer Networks." In: Advances in neural information processing systems. Ed. by Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., & Garnett, R. Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf
39. Wang, X., Ma, Y., Di, J., Murphey, Y. L., Qiu, S., Kristinsson, J., Meyer, J., Tseng, F., & Feldkamp, T. (2015). Building efficient probability transition matrix using machine learning from big data for personalized route prediction. *Procedia Computer Science, 53*, 284–291.
40. Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement Learning,* 5–32.
41. Yang, B., Guo, C., Jensen, C. S., Kaul, M., & Shang, S. (2014). "Stochastic skyline route planning under time-varying uncertainty." In: 2014 IEEE 30th international conference on data engineering. IEEE, pp. 136–147
42. Yang, B., Guo, C., Yu, M., & Jensen, C. S. (2015). Toward personalized, context-aware routing. *The VLDB Journal, 24*(2), 297–318.
43. Yang, S. B., & Yang, B. (2019). "PathRank: A Multi-Task Learning Framework to Rank Paths in Spatial Networks." arXiv preprint arXiv:1907.04028
44. Ye, N., Wang, Z.-Q., Malekian, R., Lin, Q., & Wang, R.-C. (2015). A method for driving route predictions based on hidden Markov model. *Mathematical Problems in Engineering, 2015*