Transportation Research Part E 157 (2022) 102496

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Dynamic stochastic electric vehicle routing with safe reinforcement learning

Rafael Basso [a,*], Balázs Kulcsár [b], Ivan Sanchez-Diaz [c], Xiaobo Qu [d]

[a] Volvo Group Trucks Technology, Gothenburg, Sweden
[b] Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
[c] Technology Management and Economics, Chalmers University of Technology, Gothenburg, Sweden
[d] Architecture and Civil Engineering, Chalmers University of Technology, Gothenburg, Sweden

ARTICLE INFO

Keywords:
Reinforcement learning
Approximate dynamic programming
Electric vehicles
Energy consumption
Vehicle routing
Green logistics

ABSTRACT

Dynamic routing of electric commercial vehicles can be a challenging problem since besides the uncertainty of energy consumption there are also random customer requests. This paper introduces the Dynamic Stochastic Electric Vehicle Routing Problem (DS-EVRP). A Safe Reinforcement Learning method is proposed for solving the problem. The objective is to minimize expected energy consumption in a safe way, which means also minimizing the risk of battery depletion while en route by planning charging whenever necessary. The key idea is to learn offline about the stochastic customer requests and energy consumption using Monte Carlo simulations, to be able to plan the route predictively and safely online. The method is evaluated using simulations based on energy consumption data from a realistic traffic model for the city of Luxembourg and a high-fidelity vehicle model. The results indicate that it is possible to save energy at the same time maintaining reliability by planning the routes and charging in an anticipative way. The proposed method has the potential to improve transport operations with electric commercial vehicles capitalizing on their environmental benefits.

1. Introduction

With increasingly more interest in electric commercial vehicles, in part due to legislation, which drives vehicle manufacturers to develop such vehicles, but also due to public awareness and more transport companies investing in green transportation, it is important to properly integrate them into the current operations. These vehicles have several advantages compared with their ICE counterparts and can contribute to reduce local pollution and noise (Jochem et al., 2016). But despite the latest technology developments, electric vehicles are still limited by their battery capacities. Their batteries are heavy, big and expensive, which usually translates into a limited driving range. Furthermore, energy consumption depends on several uncertain factors, such as driving behavior and traffic conditions. Therefore, safe route planning becomes essential to ensure that the vehicle will not run out of energy along the route.

When considering more dynamic transport operations with customer requests that can arrive while the vehicle is already driving, and taking into account the uncertainty of energy consumption, the routing problem becomes even more challenging. This has been a particularly relevant issue recently, with the on-demand economy seen a rapid development during the pandemic and putting extra pressure on transport operators who now have to fulfill more requests and faster, often same day deliveries, while at the same time aiming at fulfilling emission reduction goals. One common example of such a scenario is dynamic pick-up, when the vehicle is

* Corresponding author.
E-mail address: rafael.basso@volvo.com (R. Basso).

https://doi.org/10.1016/j.tre.2021.102496
Received 27 January 2021; Received in revised form 20 September 2021; Accepted 26 September 2021
Available online 29 November 2021

supposed to collect packets during a day, but the requests arrive during the same day. In that case, every time a new request arrives the previously planned route is no longer valid. A new route needs to be planned in real-time taking into account the remaining battery capacity and the remaining customers to be visited, planning additional charging stops if necessary. To solve this problem it is necessary to predict future customer requests but also energy consumption. With historical data about customer requests it could be possible to predict dynamic customer requests. Energy consumption could be predicted using models available in the literature. Then these two predictive aspects need to be integrated into a single model to decide the best route.

In this paper we propose an anticipative method for dynamically routing a single electric commercial vehicle with reliable charge planning, considering stochastic energy consumption and dynamic customer requests. The safety aspect is also taken into account in order to avoid battery depletion while the vehicle is driving.

The method provides a reliable way to adopt electric commercial vehicles in transport operations with high levels of dynamism and stochasticity. To the best of our knowledge there is no published paper dealing with this problem. By using the proposed method it is possible to anticipate dynamic customer requests and predict energy consumption, therefore supporting the dispatcher and driver when planning their routes and charging. Additionally, the method can be used to react in real-time to unexpected events such as accidents and congestion by re-routing or planning extra charging stops. As a result, it could be possible to have vehicles with less battery capacity, making them cheaper, lighter and with higher payload. Consequently it does not only benefit the driver and transport company, but also has the potential to contribute to achieving environmental goals such as the ones proposed by the European Union (EC, 2011).

1.1. Literature review

The Vehicle Routing Problem (VRP) aims at finding the optimal set of routes for a number of vehicles to visit a number of customers. The problem has been studied since the fifties in many variations and several solution methods have been developed. For an overview we refer to Toth and Vigo (2014). The objective has been traditionally to find the shortest routes in terms of distance or travel time, starting at a depot, visiting all customers and returning to the same depot.

During recent years there has been some effort to add environmental aspects into the problem, such as the Pollution Routing Problem (PRP) (Bektaş and Laporte, 2011). Minimization of emissions, minimization of fuel consumption and minimization of energy consumption of electric vehicles (e.g. Erdoğan and Miller-Hooks, 2012; Schneider et al., 2014; Felipe et al., 2014; Tahami et al., 2020) have been studied. Recent reviews can be found in Lin et al. (2014), Demir et al. (2014), Pelletier et al. (2016), Bektaş et al. (2016) and Bektaş et al. (2019). Several different models for estimating energy/fuel/emissions have been investigated (Demir et al., 2011; Cuma and Koroglu, 2015), including some specific for electric vehicles such as Genikomsakis and Mitrentsis (2017), Qi et al. (2018) and Fiori and Marzano (2018). In Basso et al. (2019, 2021), an energy estimation model for electric vehicles is developed based on Bayesian machine learning. With that model it is possible to estimate link-level probabilistic energy consumption coefficients that can be used in routing problems. One of its advantages is that it is possible to estimate the probability distribution of energy consumption, not only the expected value. The model was validated with simulations and the results will be used in this paper.

Real-life traffic environments often present a series of uncertain factors such as traffic conditions and driving behavior. Additionally, some transport operations may include a certain degree of dynamism, with new customer requests arriving while the vehicles are already driving. These two elements make routing problems even more challenging. In Laporte et al. (1992) the VRP with stochastic travel and service times is introduced for the first time. A chance-constraint and two recourse models are presented. Later most problem formulations consider stochastic customer locations, stochastic demands or stochastic times (i.e. service and travel times). For an overview we refer to Toth and Vigo (2014) and Gendreau et al. (1996). Only a few publications consider the case of stochastic energy consumption, such as Basso et al. (2021) and Pelletier et al. (2019). Dynamic routing has also been examined over time, with surveys provided by Ritzinger et al. (2015), Gendreau et al. (2016), Psaraftis et al. (2016) and Oyola et al. (2018). In Asamer et al. (2016) the authors present a sensitivity analysis of energy demand for electric vehicles and quantify the impact of parameter uncertainty. In Sweda et al. (2017) heuristics are presented to adaptively find the route for an electric vehicle with uncertain charging station availability and waiting time. In Jafari and Boyles (2017) the authors consider stochastic travel times and availability of charging stations to find the shortest path for an electric vehicle. In Liao (2017) an online method is presented to solve the routing problem with stochastic travel times and customer requests, targeting minimization of emissions. In Eshtehadi et al. (2017) robust optimization is used to solve the PRP with uncertain travel time and customer demand. In Masmoudi et al. (2018) the authors solve the Dial-a-Ride Problem for electric vehicles with battery-swap stations using Evolutionary Variable Neighborhood Search algorithms. In Tang et al. (2019) a method for scheduling electric buses is proposed taking into account stochastic traffic conditions.

Reinforcement Learning (RL) (Sutton and Barto, 2018) and Approximate Dynamic Programming (ADP) (Powell, 2007) consist of a vast collection of techniques that can be used to solve many different kinds of problems. The key idea is that an agent, as it is called in the literature of the field, learns about the environment and is able to take successively better decisions. The agent can learn online or offline with simulations. The environment is usually modeled as a Markov Decision Process (MDP), since many algorithms use dynamic programming techniques by learning the cumulative reward or cost from a certain state until the end of the process. This is usually done in an approximate way, using Value Function Approximation methods, either learning post-decision states or state–action pairs. RL and ADP methods do not assume full knowledge of the model so they can be used in more complex MDPs and even model free. They have been successfully applied for solving dynamic and stochastic routing problems (Powell et al., 2012; Ulmer, 2017). In Secomandi (2000, 2001) a few different algorithms of ADP are developed and evaluated for the VRP with

stochastic demands. Those algorithms are further improved in Novoa and Storer (2009). In Adler and Mirchandani (2014) an ADP method is developed to solve online routing of electric vehicles and make battery reservations minimizing time. In Çimen and Soysal (2017) the authors propose a time-dependent PRP variant with stochastic speed and solve it using ADP. In Mao and Shen (2018) two variants of Q-Learning are used for solving routing problems in stochastic time-dependent road networks. In He et al. (2018) ADP is used for scheduling buses with stochastic trip times. In Ulmer et al. (2019) an offline–online ADP method is presented to solve the dynamic VRP with stochastic requests. In Liu et al. (2020) integrate Dijkstra's algorithm with inverse RL to plan routes for food delivery. RL with Neural Networks is not very commonly used for VRPs. In Nazari et al. (2018) RL is used with neural networks to solve the capacitated VRP. In Shi et al. (2019) the authors introduce an RL method with neural networks to solve the ride-hailing problem for electric vehicles. As can be seen in the reviews by Ritzinger et al. (2015) and Psaraftis et al. (2016), there is a strong focus on single-vehicle problems when using ADP/RL methods and modeling the dynamic problem as an MDP. In Fan et al. (2006) and Goodson et al. (2013) multiple vehicles are considered using decomposition schemes to assign customers to vehicles beforehand. Both solve the dynamic VRP with stochastic demands but with known customer locations. In Shi et al. (2019) and Liang et al. (2021) the authors use neural networks as approximators for the state values of each vehicle while formulating the fleet dispatch as a linear assignment problem. In Jin et al. (2019) the authors present a ride-hailing problem with two layers represented by Neural Networks. In Turan et al. (2020) a Deep RL method is proposed to minimize the total cost of a fleet of electric autonomous vehicles for ride hailing services, where the complete system state is approximated using neural networks. In Chen et al. (2021) the authors propose a Deep Q-Learning method to assign customers to vehicles and drones for same-day delivery.

One of the branches of the field is Safe Reinforcement Learning, which aims at minimizing cost or maximizing expected reward while keeping safety constraints when learning or after deployment. These methods are relevant for environments where failures can have a significant impact (e.g. accidents). These methods are also useful when the environment can potentially deviate from the training scenarios. An overview of the methods and review of recent literature is provided in Garcıa and Fernández (2015). In the context of electric vehicles, battery depletion is a risk with potentially high consequences such as extra costs for towing, delays, congestion and even traffic accidents. Therefore a safe approach can be used to minimize that risk.

To the best of our knowledge there is no published paper solving the electric vehicle routing with stochastic energy, dynamic customer requests and charge planning. It is also novel the use of Safe Reinforcement Learning as a solution method to plan the route predictively and ensure that the vehicle will not run out of energy while driving.

1.2. Contributions

The main contributions presented in this paper are:

- A model of the DS-EVRP as a Markov Decision Process;
- A Safe Reinforcement Learning solution method including:
  - A tailored Value Function Approximation;
  - A safe policy;
  - A training strategy;
- Realistic computational experiments to evaluate the proposed approach.

The Dynamic Stochastic Electric Vehicle Routing Problem is modeled as a Markov Decision Process (MDP). We propose a state representation, a set of valid actions and a state transition function. It considers stochastic customer requests and stochastic energy consumption. Like most related methods found in the literature, this paper focuses on a single vehicle.

The solution method for the problem is a Safe Reinforcement Learning approach based on Q-Learning but with some modifications. It aims at learning the cumulative energy cost and risk of failure (i.e. battery depletion) from a certain state when taking a certain action, and following the process until terminated (e.g. reaching back to the terminal). The first contribution of this article is a Value Function Approximation based on look-up tables indexed by a tailored reduced state representation to learn the value of state–action pairs. The second contribution is a chance-constrained policy with two layers of safety to ensure that the battery will not be depleted while en route. The third contribution is a training approach with a rollout heuristics to guide the learning to focus on the most promising actions directly when a state is first visited. Since to the best of our knowledge there are no solution methods for this problem in the literature, we present a deterministic online reoptimization method that is used as a benchmark in the experiments. Both the reoptimization method and the rollout for training the agent use a tabu-search heuristics that is shown to find routes very close to the optimal.

In order to evaluate the proposed method, a set of experiments is presented. A probabilistic energy consumption model is used based on Basso et al. (2021). In that paper a series of simulations are performed with a realistic traffic model for the city of Luxembourg (Codeca et al., 2015) and a high-fidelity vehicle model. Then machine learning is used to estimate the link-level energy consumption probability distributions for the city center. In this paper the energy consumption is sampled from those probability distributions. Customer requests are sampled from probabilities for each test instance. The reinforcement learning method is compared with the reoptimization benchmark in terms of reliability and energy consumption. Different levels of dynamism are investigated by allowing late new requests and by knowing less customers at the start. Results are also shown for scenarios with stochastic customer demand (i.e. payload weight). Runtime and memory usage are not in focus but an initial analysis is shown nonetheless. Finally, a discussion about the hyper-parameters and algorithm is presented.

The structure of the paper is as follows: Section 2 presents the problem formulation and MDP model; Section 3 presents the solution method; Section 4 describes the experiments and results; Section 5 presents conclusions and proposes potential future work. Appendix A describes the test instances provided in the supplementary material.

2. Problem formulation

The goal of the Dynamic Stochastic Electric Vehicle Routing Problem (DS-EVRP) is to dynamically find a route for a single electric vehicle to service customer requests that can arrive randomly during a certain period (e.g. a full or half working day). The vehicle departs from and returns to the same depot. There are two types of customer requests, (i) deterministic requests that are known before the vehicle leaves the depot and (ii) stochastic requests that are received after leaving the depot with a known probability. All customer locations are known. It is assumed that all requests are accepted and can be served by the vehicle. The amount of energy needed to drive a certain stretch of road is random with a known probability distribution. Since the vehicle is electric, the battery capacity is assumed to be limited and therefore there might be a need to dynamically plan for charging along the route.

The DS-EVRP is modeled as a Markov Decision Process (MDP). The problem can be visualized in a complete and directed graph, represented by $G = (\mathcal{V}, \mathcal{A})$ with $\mathcal{V} = \{0\} \cup C \cup \mathcal{F} = \{0, 1, 2, \dots, C + F\}$ as the set of nodes and $\mathcal{A}$ as the set of paths connecting each pair of nodes. The total number of customers is given by C and the total number of charging stations is given by F. The customer nodes are defined by $C = \{1, \dots, C\}$, the charging stations are $\mathcal{F} = \{C + 1, \dots, C + F\}$ and the depot is represented by node 0.

Each customer $c \in C$ has a demanded cargo weight u_c (kg). The total probability of sending a request is p_c and follows a binomial distribution with K trials, so the probability of a new request for epoch $k \leq K$ is given by $p_{ck} = 1 - \sqrt[K]{1 - p_c}$. The maximum payload (kg) for the vehicle is U and the empty vehicle (curb) weight (kg) is W. The vehicle has batteries with total capacity B (Wh) and minimum accepted battery level L (e.g. $L = 0.2B$, minimum 20% of the total capacity).

The state of the system at epoch k is given by $s_k = (i_k, b_k, u_k, \bar{C}_k, \tilde{C}_k)$, where:

- i_k is the vehicle node position, where $i_k \in \mathcal{V}$;
- b_k is the battery level, where $b_k \leq B$;
- u_k is the current payload, where $u_k \leq U$;
- $\bar{C}_k$ is the set of active requests, where $\bar{C}_k \subseteq C \setminus \tilde{C}_k$;
- $\tilde{C}_k$ is the set of visited customers, where $\tilde{C}_k \subseteq C$.

A decision epoch k is delimited by the time when the vehicle leaves the node i_k. In the initial state $s_0 = (0, B, U_0, \bar{C}_{early}, \emptyset)$ the vehicle is at the depot fully charged and with initial payload U_0 (e.g. zero for the pick-up case). The set of known (i.e. deterministic) requests before the vehicle departs is given by $\bar{C}_{early} \subseteq C$. It is assumed that the requested demands throughout the route will not exceed the total payload capacity U. Furthermore, new requests will only be received until epoch K and a visited customer cannot send a new request.

When the process is in state s_k at epoch k, the decision to be taken is which node will be visited next. The set of feasible actions is:

$$\bar{\mathcal{V}}(s_k) = \{j_k : j_k \in \mathcal{V}\}, \tag{1}$$

$$j_k \in \bar{C}_k \cup \mathcal{F} \quad \text{if } \bar{C}_k \neq \emptyset, \tag{2}$$

$$j_k \in \{0\} \cup \mathcal{F} \quad \text{if } \bar{C}_k = \emptyset. \tag{3}$$

Eq. (1) requires that j_k, the next node to be visited, is in the graph G. Constraint (2) requires a customer to be visited while there are still active requests. Constraint (3) makes the vehicle return to the depot when all requests have been served. Charging stops are allowed at any time.

When the process is in state s_k and decision j_k has been taken, a cost is accumulated e_{i_k,j_k}, which is the total energy consumed to drive from node i_k to node j_k. The energy consumption framework used in this paper is based on Basso et al. (2021). The probability distribution for the energy consumption to drive from node i to node j is known and given by:

$$e_{ij} = \begin{bmatrix} m & 1 \end{bmatrix} \mathbf{z}_{ij} \tag{4}$$

$$\sigma^2_{ij} = \begin{bmatrix} m & 1 \end{bmatrix} \mathbf{f}_{ij}. \tag{5}$$

where e_{ij} is the expected energy and σ^2_{ij} is the variance, following a Normal distribution. The total mass of the vehicle (curb weight + payload) is given by $m = W + u$. Coefficients $\mathbf{z}_{ij}$ and $\mathbf{f}_{ij}$ are calculated using the method described in Basso et al. (2021), where $\mathbf{z}_{ij} = \begin{bmatrix} \alpha_{ij} & \beta_{ij} \end{bmatrix}$ is an array with coefficients for the expected energy and $\mathbf{f}_{ij}$ is an array with coefficients for the variance. These energy coefficients embed information about topography, speed, traffic and the vehicle. For simplification, the expected energy consumption can be re-written as:

$$e_{ij} = \alpha_{ij}(W + u) + \beta_{ij}$$

which is the same as Eq. (4). Therefore e_{i_k,j_k} is the notation for a sample from probability distribution $\mathrm{N}(e_{ij}, \sigma^2_{ij})$ at epoch k.

From epoch k to epoch $k + 1$ there might be new customer requests represented by the set $\bar{\bar{C}}_{k+1}$. Each customer has a total probability p_c of requesting a visit during the allowed period (i.e. $0 < k \leq K$). The state transition from s_k to s_{k+1} is then given by:

$$i_{k+1} = j_k \tag{6}$$

$$b_{k+1} = b_k - e_{i_k,j_k} \tag{7}$$

$$u_{k+1} = u_k + u_j \tag{8}$$

$$\bar{C}_{k+1} = \bar{C}_k \cap \{j_k\} \cup \bar{\bar{C}}_{k+1} \tag{9}$$

$$\bar{C}_{k+1} = \bar{C}_k \cup \{j_k\}. \tag{10}$$

Eq. (6) assigns the new node location. Eq. (7) calculates the remaining battery level after arriving at node j_k. Eq. (8) increases the vehicle payload. If j_k is a charging station or the depot, then $u_j = 0$. Eq. (9) removes node j_k from the current requests, if j_k is a customer, and adds any new requests $\bar{\bar{C}}_{k+1}$ that might have arrived while driving from i_k to j_k. Eq. (10) adds node j_k to the list of visited customers, if j_k is a customer. If node j_k is a charging station, the battery is fully charged before leaving (i.e. $b_{k+1} = B$).

The MDP is terminated if all customers have been visited and the depot is reached, or if the battery is completely depleted (i.e. $b_{k+1} \leq 0$). An overview of the model is shown in Fig. 1.

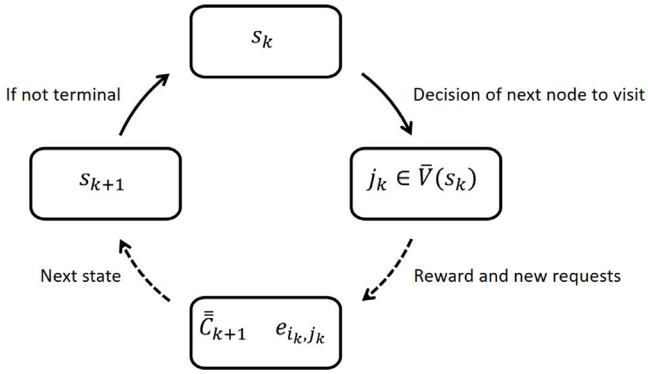

Fig. 1. MDP for the DS-EVRP.

As shown above, parts of the model are deterministic and parts are stochastic. In Fig. 1 the dashed lines represent stochastic transitions. Because the policy is deterministic, the actions are deterministic as well, since when the next node is decided it is assumed that the vehicle will drive there unless the battery is depleted on the way. On the other hand, the state transitions are stochastic with unknown probability distribution, since new requests are random. The rewards are stochastic since the energy consumption is random, but the probability distribution is assumed to be known. Furthermore, it is important to notice that this MDP has an infinite state space, since there are continuous variables in the state representation (i.e. battery level and payload). It is assumed that the current state is fully observable.

The objective of the problem is to minimize expected energy consumption and guarantee that the vehicle will be able to complete the route by planning charging when necessary. It is important to observe that these two objectives are somewhat conflicting. To increase safety it is usually necessary to add more charging stops, which in turn can increase energy consumption. Conversely, reducing the number of charging stops might reduce energy consumption but can potentially decrease safety. The objective function can be formulated as:

$$\min_{e} \mathbb{E} \sum_{k} e_{i_k,j_k} \tag{11}$$

subject to

$$\bar{C}_{end} = \varnothing \tag{12}$$

$$\bar{C}_{end} = \bigcup_{k} \bar{\bar{C}}_k \cup \bar{C}_{early} \tag{13}$$

$$Pr(b_k \leq L) \leq R \tag{14}$$

Eq. (11) minimizes the expected energy consumption for the complete route. Constraints (12) and (13) force the vehicle to visit all early requests in $\bar{C}_{early}$ and all dynamic requests in $\bar{\bar{C}}_k$ from all epochs k until the process is finished (i.e. $k = end$). The set of feasible actions given by $\bar{\mathcal{V}}(s_k)$ also constrain the problem to visit all customers with active requests before returning to the depot. Since e is stochastic, the battery level b is also stochastic, therefore the problem is chance-constrained. Eq. (14) implies that the risk of the battery being below the minimum L is less or equal to R (e.g. 5%) for the complete route.

The next section presents methods to try to achieve this objective.

3. Solution method

In order to find solutions for the problem described above, we present two different approaches. The first one is a reoptimization method. The second is a Reinforcement Learning method.

3.1. Deterministic reoptimization

In this section a deterministic online reoptimization method is presented to solve the MDP shown above. This approach adapts a tabu-search heuristics for the static EVRP to find the routes dynamically, and will be used as a benchmark for the comparisons in the computational experiments. It uses only the deterministic information known from state s_k at decision epoch k to take a decision j_k, considering only the current requests $\bar{C}_k$ to decide the route. To plan for charging and estimate the total energy consumption it uses the expected value of energy e as shown in Eq. (4).

At every epoch k, the decision of the best action j_k is made by running a tabu-search heuristics considering the current requests. First a route is constructed using a greedy policy from node i_k to visit all customer requests in $\bar{C}_k$ and back to the depot, choosing the nearest neighbors in terms of expected energy. Secondly, this initial route is improved using tabu-search with 2-opt moves. If the route is energy unfeasible (i.e. not enough battery capacity to drive all the way), then a charging station is inserted and the route is improved again. The best charging station is chosen. Since energy is stochastic, the method tries to keep the battery above the minimum limit L in order to increase reliability. The heuristics are described in details in Section 4.2. The solutions generated are shown to be very close to the optimal.

3.2. Reinforcement learning

This section introduces a reinforcement learning method to solve the MDP presented in Section 2. The approach is to try to anticipate future requests and predict energy consumption in order to take better decisions. As described, some aspects of the problem are known, some are unknown with known probabilities and some are unknown with unknown probabilities. Learning about all the unknowns can be difficult considering the curse of dimensionality (Powell, 2007). Furthermore, vehicle routing problems are well known for growing exponentially in size with increasing number of nodes (i.e. customers and charging stations).

The proposed method is based on Q-learning (Sutton and Barto, 2018), which is an algorithm that learns the value of taking an action from a certain state. This value is usually expressed as a reward and the most common approach is to maximize the reward from the current state until the process is terminated. There are similarities with Dynamic Programming and with extensive training the policy converges to a near-optimal solution. However, one of the advantages of Q-learning is that a complete MDP is not necessary and the algorithm can even run model free. One of the variants of Q-learning keeps a table with the reward for each state–action pair, called the Q-value. It is an off-policy algorithm, which means that it learns about the optimal policy by following an exploration policy (e.g. ϵ-greedy with some actions selected randomly). Fig. 2 shows an overview of how the agent interact with the environment, with some aspects that will be discussed in the following subsections.

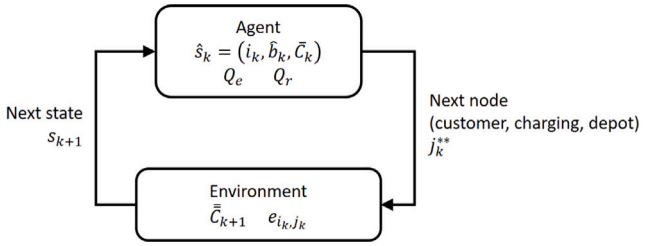

Fig. 2. Reinforcement Learning method overview.

For the problem discussed in this article, the MDP accumulates a cost (i.e. energy consumed) instead of a reward, which should therefore be minimized (i.e. instead of reward maximization). But it is also important to keep safety constraints, not letting the vehicle run out of energy while driving. Thus a Safe Reinforcement Learning approach will be used, by learning the expected cumulative energy cost and the risk from state s until the end of the route if action j_k is taken. These will be stored in two Q-tables, Q_e for expected energy and Q_r for the probability of failure (i.e. battery depletion). Following a certain policy π, they can be defined as:

$$Q_e(s_k, j_k) = \mathbb{E}(e_{i_k,j_k}) + Q_e(s_{k+1}, j_{k+1}) \tag{15}$$

$$Q_r(s_k, j_k) = \begin{cases} Q_r(s_{k+1}, j_{k+1}) & \text{if } s_{k+1} \text{ not terminal;} \\ \mathbb{P}(b_{k+1} \leq 0) & \text{otherwise.} \end{cases} \tag{16}$$

By selecting the next node to visit using Q_e and Q_r, it is possible to plan robust routes and keep safety constraints. The method will be presented in details in the following sections, focusing on its three main components: a value function approximation, a safe policy and a training strategy.

3.2.1. Value function approximation

The MDP presented in Section 2 has an infinite number of states, since both the battery b and the payload u are continuous variables. Furthermore, the number of combinations of $\bar{C}_k$ and $\tilde{C}_k$ grows very fast when increasing the number of customers. Therefore, in order to use Q-learning in an efficient way, the state needs to be discretized. With that it is possible to reduce storage space of the Q-tables and improve exploration. Hence, a value function approximation is presented, based on a reduced state representation.

To find a suitable reduced state representation $\hat{s}$, it is necessary to select which variables from s will be included and eventually discretize the continuous ones. A number of alternatives were evaluated. Since knowing the battery level is crucial for taking decisions (e.g. whether to charge or not), b is discretized into $\hat{b}$. The experiments will use 10 levels, from above 90% to 0% (i.e. 9 to 0). Furthermore, the current requests are necessary in order to know which actions are valid. Thus the reduced state representation is given by $\hat{s} = (i, \hat{b}, \vec{C})$. The payload weight u and the previously visited customers $\vec{C}_k$ were removed since they are less important for the solution method to work properly.

As a result any state s (infinite cardinality) in the MDP has a corresponding representation $\hat{s}$ (finite cardinality) that is used to index Q_e and Q_r. Therefore it is possible to get an approximation of the cumulative energy cost and risk, from state s if an action j is taken by reading $Q_e(\hat{s}, j)$ and $Q_r(\hat{s}, j)$ respectively. With that approximation the size of the Q_e and Q_r tables is significantly reduced and is finite. Despite the total number of reduced states still being quite large, the number of actual state–action pairs visited is significantly lower, as many are invalid or unreachable states (e.g. less than 100% battery at the start). This will be discussed in Section 4. Another advantage of this approximation is that similar states (e.g. small differences in battery level) will be merged, improving exploration. The complete state representation is used during the simulations to keep track of the system state and calculate energy consumption.

3.2.2. Policy

As the method is based on Q-learning, the target policy depends on Q_e and Q_r. The objective of the problem is to minimize energy consumption for the complete route, but also minimize the risk of running out of battery while on the road. As energy consumption has the same value over time, the problem does not include any discount factor. The value of $Q_r(\hat{s}, j)$ estimates the probability of failure (i.e. battery depletion) from state s until the end of the route if action j is taken. Therefore we propose a policy π^* that is greedy with respect to Q_e and chance-constrained with respect to Q_r, with a maximum accepted risk R (e.g. 5%). Taking into account the feasible actions from state s, the actions that satisfy the risk criteria are:

$$\mathcal{J}(s) = \{j : j \in \bar{\mathcal{V}}(s), Q_r(\hat{s}, j) \leq R\}, \tag{17}$$

The optimal action from state s is then decided using the policy:

$$j^* = \begin{cases} \underset{j \in \mathcal{J}(s)}{\arg\min}\, Q_e(\hat{s}, j) & \text{if } \mathcal{J}(s) \neq \varnothing; \\ \underset{j \in \bar{\mathcal{V}}(s)}{\arg\min}\, Q_r(\hat{s}, j) & \text{if } \mathcal{J}(s) = \varnothing. \end{cases} \tag{18}$$

If there are actions that satisfy the risk criteria, then the action with less expected energy consumption is chosen from $\mathcal{J}(s)$. If there are no actions that satisfy the risk criteria (i.e. $Q_r(\hat{s}, j) > R \quad \forall j \in \bar{\mathcal{V}}(s)$), then the action with less expected risk is chosen from all feasible actions $\bar{\mathcal{V}}(s)$.

As this is a probabilistic approach, there is still a risk of R for the vehicle running out of battery. Although it is possible to reduce the accepted risk R (e.g. $R = 1\%$), it might still happen that some tours will fail. That might not be acceptable for transport companies since it can incur high expenses (e.g. towing the vehicle, customer compensations, reputation, risk of accidents). Therefore we propose a second layer of safety with policy π^{**}.

This second layer of safety tries to keep it always feasible to visit a charging station from any node. It uses a two-step look-ahead to predict energy cost. The first step evaluates the energy cost from the current node i to the chosen destination j, as shown in Eq. (19). The second step evaluates the cost from node j to its nearest charging station (i.e. in terms of energy cost), as shown in Eq. (20).

$$e_{ij} = \alpha_{ij}(W + u) + \beta_{ij} \tag{19}$$

$$e_{jf} = \min_{f \in F} \alpha_{jf}(W + u + u_j) + \beta_{jf} \tag{20}$$

If the sum of those two costs is larger than the current battery level then the policy overrides the previous decision and chooses to charge the battery at the nearest charging station from the current node.

$$j^{**} = \begin{cases} j^* & \text{if } e_{ij} + e_{jf} \leq b; \\ \underset{f \in F}{\arg\min}\, \alpha_{if}(W + u) + \beta_{if} & \text{otherwise.} \end{cases} \tag{21}$$

In Section 4 a discussion about the implications of adding this second layer of safety to the policy will be presented, in terms of increased safety and increased energy cost.

3.2.3. Training

One of the biggest challenges of combinatorial optimization problems is that the number of possible solutions grows very quickly with the number of elements. For VRPs it can be computationally hard to find the optimal solution for problems with many customers to visit and even harder for problems which incorporate stochasticity. This becomes even more critical for the MDP presented in Section 2, since it has an infinite number of states due to the continuous values of battery and payload. As a consequence it is hard to train a Reinforcement Learning agent by visiting all feasible states many times. Even considering the reduced state representation, the total number of combinations grows very quickly with the number of customers. Therefore we propose a focused strategy to train the agent by finding good candidate actions directly on the first time a state is visited.

The training method is offline off-policy learning using ϵ-greedy as the exploration policy and the target policy π^{**}. Every episode simulated is a complete route from the depot, servicing all requests and returning to the depot, with charging stops if necessary. The episode is also terminated if the battery is depleted. The method is shown in Algorithm 1.

The basic idea is to simulate a complete episode (from Line 3 to 25) then update the cost and risk tables, Q_e and Q_r respectively (from Line 27 to 43). The number of times a state–action pair is visited is stored in $N(\hat{s}, j)$. The Q-table for expected energy is updated in line 36 and the Q-table for risk is updated in line 37, based on the average of all visits to the state–action pair. To keep track of the risk, the updates are done backwards, from the end of the route to the beginning.

The next action is decided from Line 8 to Line 14. If a state $\hat{s}$ has never been visited before, the action is decided using the same rollout heuristics as described in 3.1 and 4.2. Otherwise an ϵ-greedy policy is followed, but the random actions are chosen only from the set of feasible actions $\bar{\mathcal{V}}(s_k)$.

Function *step*, in Line 15, computes the transition from state s_k to the next state s_{k+1}. In that function a realization for the energy consumption e_{i_k,j_k} is sampled from the probability distribution given by Eqs. (4) and Eq. (5). Furthermore, new requests $\bar{\bar{C}}_{k+1}$ are sampled based on the probability p_{ck} for each customer c and added to the current requests in $\bar{C}_{k+1}$.

After the agent is trained offline, it can be used online to take decisions using policy π^{**} and the learned tables Q_e and Q_r. Since the method is based on look-up tables, very little computation is needed to take decisions online. Re-training may only be required if the set of customers or charging stations change, or if the probabilities for energy and customer requests change significantly.

4. Experiments

In order to evaluate the solution method, several numerical experiments are presented. The main objective is to demonstrate the energy savings and reliability comparing the reinforcement learning method with the reoptimization approach. Computational cost, runtime and memory usage are not the focus, but a simple assessment is shown nevertheless. An analysis of the impact of different levels of dynamism to the results is presented as well as a discussion about hyperparameter selection and the structure of the algorithm.

First the setup of the experiments is described. Secondly, an evaluation of the reliability of the reoptimization method is shown. Lastly, a detailed analysis of the reinforcement learning is presented.

4.1. Test environment

To evaluate the methods, 50 test instances were generated, half of them with 20 customers and half with 10 customers each. All customers are randomly located in the city center of Luxembourg with the same depot and two charging stations for all test instances, as shown in Fig. 3. The energy prediction model and the link-level energy coefficients are derived from the experiments in Basso et al. (2021).

The test instances were generated to reflect real scenarios for urban distribution of goods with trucks. The number of customers depends on the type of goods and the city as described in Holguín-Veras et al. (2013). Therefore, data from a company in Gothenburg was used to set those parameters based on Sanchez-Diaz et al. (2020). In those scenarios one vehicle typically delivers or picks-up between 10 and 20 packages, due to time and capacity constraints, of which around half can be dynamic.

To find the paths between all pairs of nodes, the Bellman–Ford algorithm was used in the same way as in Basso et al. (2021), then producing the energy coefficients $\mathbf{z}_{ij}$ and $\mathbf{f}_{ij}$ for each path (i, j). During the simulations both for training and evaluation, the energy is sampled from the probability distributions given by Eqs. (4) and (5). The test instances are provided in the supplementary material which is described in Appendix A.

Every customer has a weight demand, but the total payload for each instance does not exceed the maximum payload of the truck. The experiments focus on the pick-up case, so the payload weight is increased at every customer visit. Each customer has a certain probability of requesting a visit during an episode. They can send a new request up to epoch $K = 5$ or $K = 10$, for 10 and 20 customers instances respectively. About half the customer requests are known before the vehicle leaves the depot (i.e. $p_c = 1$), so there is a fixed $\bar{C}_{early}$ for each test instance, with $|\bar{C}_{early}| \approx 5$ for 10 customers and $|\bar{C}_{early}| \approx 10$ for 20 customers. The vehicle leaves the depot with the battery fully charged and recharges the battery completely when it stops at charging stations.

The vehicle model is an all-electric medium duty truck with two gears, similar to current truck models in the market (e.g. Volvo FL electric). Curb weight is 10 700 kg, including the battery and maximum payload is 16000 kg. The battery capacity and minimum battery are different for each experiment, therefore they are given in the descriptions below. For the value function approximation (i.e. Q_e and Q_r tables), the battery is discretized into ten levels, as described in Section 3.2.1 to produce the reduced state representation $\hat{s}$.

Each episode simulates a complete route for a single vehicle, starting at the depot with initial customer requests, new random requests while driving, finishing back at the depot after visiting all requests, unless the battery is depleted. The method to choose the route is either the reoptimization from Section 3.1 or the reinforcement learning from Section 3.2. The training for the reinforcement learning method is done by simulating 500 000 episodes, if not stated otherwise, and $\epsilon = 0.05$ without decay. Maximum risk accepted is $R = 0.1$ (i.e. 10% chance of failure). A discussion about these hyperparameters is presented later.

In order to acquire statistically significant data, the evaluation is done by simulating 20 000 episodes, both for the reoptimization and reinforcement learning. For the reoptimization approach, this adds runtime requirements, since for each episode there might be up to 10 events with new customer requests (for the case of 20 customers), which means that the route needs to be re-planned up to 10 times per episode. As shown in Section 4.2, for 20 customer requests with charging, on average it takes 0.69 s to find a

Algorithm 1 Reinforcement Learning

$Q_e(\hat{s}, j) = 0, Q_r(\hat{s}, j) = 0, N(\hat{s}, j) = 0 \quad \forall \hat{s} \in \hat{S}, j \in \mathcal{V}$
for all episodes **do**
  $k = 0$
  $s_k = (0, B, 0, \bar{C}_{early}, \varnothing)$
  $listE(k) = 0, listJ(k) = 0 \quad \forall k$
  $finished = 0$
  while $finished == 0$ **do**
    if $\sum_j N(\hat{s}, j) == 0$ **then**
      j_k =rollout(s_k)
    else if ϵ >rand **then**
      j_k =random($\bar{\mathcal{V}}(s_k)$)
    else
      $j_k = j^{**}$
    end if
    (s_{k+1}, e_{i_k,j_k}) =step(s_k, j_k)
    $listE(k) = e_{i_k,j_k}$
    $listJ(k) = j_k$
    $s_k = s_{k+1}$
    $k = k + 1$
    if $\bar{C}_k == \varnothing$ and $i_k == 0$ **then**
      $finished = 1$
    else if $b_k \leq 0$ **then**
      $finished = 1$
    end if
  end while
  $k = k - 1$
  $e = 0$
  if $b_{k+1} \leq 0$ **then**
    $r = 1$
  else
    $r = 0$
  end if
  while $k \geq 0$ **do**
    $e = e + listE(k)$
    $j = listJ(k)$
    $Q_e(\hat{s}, j) = (N(\hat{s}, j) * Q_e(\hat{s}, j) + e)/(N(\hat{s}, j) + 1)$
    $Q_r(\hat{s}, j) = (N(\hat{s}, j) * Q_r(\hat{s}, j) + r)/(N(\hat{s}, j) + 1)$
    $N(\hat{s}, j) = N(\hat{s}, j) + 1$
    if $j \neq j^{**}$ **then**
      $k = 0$
    end if
    $k = k - 1$
  end while
end for

route with the heuristics and 1087 s with a MILP solver. It took more than 21 h to run the reoptimization experiments for the 25 test instances with 20 customers and charging, 51 min per instance on average as discussed in Section 4.4.2. Therefore, using a MILP solver instead of the heuristics in the reoptimization approach would make the solution time impractical. Furthermore, the solutions computed by the heuristics are very close to optimal, as shown below.

4.2. Rollout heuristics

The algorithm shown below, based on tabu-search, tries to find a route to service a set of requests, considering the already visited customers in the route and a starting node.

First it builds an initial route with function *BuildRoute* from the current node visiting all customers with active requests. That first route is constructed using a greedy algorithm (i.e. nearest neighbor) with respect to energy consumption.

Next it tries to improve the initial route with 2-opt moves in function *ImproveRoute*. The function *routeCost* calculates the total energy cost for the route and checks if there is any battery violation (i.e. battery below minimum level L). If no improvement can

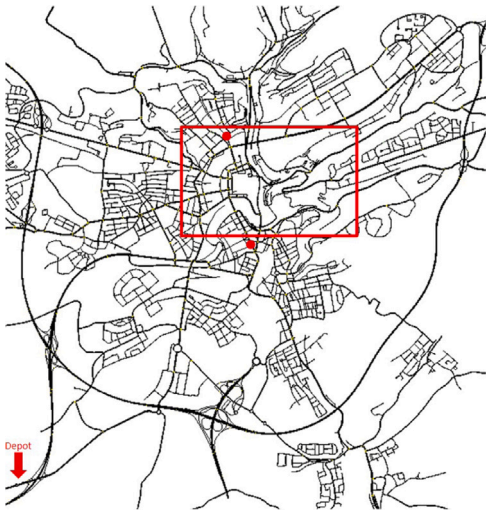

Fig. 3. Section of the map of Luxembourg with location of depot, customer area and charging stations.

be found, the algorithm tries again using a move generated by *TSMove*. That function uses the principle of tabu-search, generating a random 2-opt move that has not been performed before.

If the improved route violates the battery constraints, the algorithm tries to find the best charger.

In function *BetterRoute*, the battery constraints are prioritized over energy. A route with no battery violations is always preferred.

Below we present the comparison of the heuristics with an optimal solution, considering the test instances deterministic (i.e. all customers are visited). The CPLEX 12.9 mixed integer linear programming solver was used to solve the problem to optimality. The code was developed using Matlab 2017b and run in a computer with a 3.1 GHz dual-core i5 processor and 16 Gb RAM. Tables 1 and 2 show the results for 10 and 20 customers with battery $B = 200\,000$. Tables 3 and 4 show the results for 10 and 20 customers with battery $B = 20\,000$ and $B = 30\,000$, which means that at least one charge stop is needed. Columns *Energy* show the amount of energy required to complete the route in Wh. Columns *Runtime* show the total time needed to compute the solution with the different methods in seconds. Column *Diff* show the optimality gap in percentage for the heuristics. The solution time for CPLEX was limited to 30 min, therefore some instances have slightly better solutions by the heuristics.

As the results indicate, the gap from the heuristics to the optimal solution is small. At most it is 3.61% for an instance with 20 customers and charging. On average, for the 10 customers instances the gap is close to zero while for the 20 customers instances it is still quite small. The runtime difference however is significant. While the optimal solution reached to limit of half an hour for many instances, the longest time to solve with the heuristics was 1.1 s.

4.3. Deterministic reoptimization

In this section the goal is to evaluate how reliable the deterministic online reoptimization is. Since energy consumption is stochastic, but the method uses the expected energy to find the route, it is necessary to keep a certain margin with regards to the battery level. Therefore, in this set of experiments we evaluate the minimum level L by running the reoptimization method with the 20 customers instances. The battery capacity is $B = 30$ kWh, which means that in most cases at least one charging stop will be necessary to complete the route. Three different minimum levels are evaluated: $L = 0$, $L = 3000$ Wh and $L = 6000$ Wh.

Table 5 shows the results for simulating the deterministic online reoptimization method with minimum battery level of 20% ($L = 6000$ Wh), 10% ($L = 3000$ Wh) and 0 ($L = 0$ kWh). Column *Cost* shows the average energy cost (Wh) for each test instance for 20 000 simulated episodes. Column *Fail* shows how many times the battery got completely depleted while driving.

These results indicate that the only safe margin is 20%, which did not produce any battery depletion for any test instance. Therefore, the results for $L = 6000$ will be used as a benchmark for comparing with the reinforcement learning method below.

4.4. Reinforcement learning

In this section the goal is to evaluate how much energy can be saved keeping reliability using the reinforcement learning method compared with the deterministic online reoptimization approach. For the instances with 10 customers the battery is $B = 20$ kWh and for the instances with 20 customers the battery is $B = 30$ kWh. The minimum battery for the reoptimization is $L = 4$ kWh and $L = 6$ kWh for 10 and 20 customers respectively. For the reinforcement learning method $L = 0$. This means that in most episodes

Algorithm 2 Heuristics to find route

```
function ROLLOUT(route,current,requests)
    route = BUILDROUTE(route,current,requests)
    [route, cost, vio] = IMPROVEROUTE(route,current)
    if vio then
        for all chargers do
            tRoute = INSERTCHARGER(route,current,charger)
            [tRoute, tCost, tVio] = IMPROVEROUTE(tRoute,current)
            if BETTERROUTE(cost, vio, tCost, tVio) then
                vio = tVio
                cost = tCost
                route = tRoute
            end if
        end for
    end if
    return route, cost, vio
end function
function IMPROVEROUTE(route,current)
    [minCost, minVio] = ROUTECOST(route)
    minRoute = route
    n = length(route)
    updated = 1
    while updated do
        updated = 0
        for i = current + 1 : i < n − 1 do
            for j = i + 1 : j < n do
                tRoute = 2Opt(route, i, j)
                [tCost, tVio] = ROUTECOST(tRoute)
                if BETTERROUTE(minCost, minVio, tCost, tVio) then
                    minVio = tVio
                    minCost = tCost
                    minRoute = tRoute
                    updated = 1
                end if
            end for
        end for
        if updated then
            route = minRoute
            tsCount = 0
            bestVio = minVio
            bestCost = minCost
            bestRoute = minRoute
        else
            if tsCount < TSMAX then
                route = TSMove(bestRoute)
                tsCount = tsCount + 1
                updated = 1
            end if
        end if
    end while
    return bestRoute, bestCost, bestVio
end function
function BETTERROUTE(minCost, minVio, tCost, tVio)
    better = 0
    if tVio > 0 then
        if minVio > 0 and tVio < minVio then
            better = 1
        end if
```

```
    else
        if minVio > 0 or tCost < minCost then
            better = 1
        end if
    end if
    return better
end function
```

Table 1
Optimality gap for 10 customers without charging.

	Optimal		Heuristics		
	Energy	Runtime	Energy	Runtime	Diff
1	23 573	0.468	23 573	0.020	0.00%
2	28 304	1.393	28 304	0.017	0.00%
3	23 100	0.379	23 100	0.013	0.00%
4	26 737	0.506	26 737	0.010	0.00%
5	22 756	0.461	22 756	0.019	0.00%
6	26 067	3.502	26 067	0.008	0.00%
7	30 247	1.352	30 247	0.014	0.00%
8	24 124	1.113	24 124	0.015	0.00%
9	25 470	2.524	25 470	0.023	0.00%
10	28 187	1.941	28 187	0.032	0.00%
11	28 475	0.948	28 475	0.032	0.00%
12	22 918	2.643	23 266	0.012	1.52%
13	26 696	3.101	26 696	0.022	0.00%
14	26 761	0.304	26 761	0.011	0.00%
15	26 705	2.366	26 705	0.013	0.00%
16	25 394	1.184	25 394	0.012	0.00%
17	26 178	1.509	26 178	0.020	0.00%
18	25 726	1.617	25 726	0.025	0.00%
19	30 719	0.775	30 904	0.012	0.60%
20	27 326	1.713	27 326	0.027	0.00%
21	26 036	0.872	26 036	0.022	0.00%
22	26 526	0.204	26 526	0.023	0.00%
23	24 152	0.486	24 152	0.013	0.00%
24	26 555	0.415	26 831	0.017	1.04%
25	24 124	0.190	24 124	0.030	0.00%
Avg	26 114	1.279	26 147	0.019	0.13%
Min	22 756	0.190	22 756	0.008	0.00%
Max	30 719	3.502	30 904	0.032	1.52%

at least one charging stop will have to be planned. For the tests without charging, the battery is $B = 200$ kWh, which is enough for any route.

Tables 6 and 7 show the results of the experiments with 10 and 20 customers test instances respectively. Column *Reopt* shows the average energy cost (Wh) using the reoptimization method evaluated with 20 000 simulated episodes. Column *RL* shows the average energy consumption cost (Wh) using the reinforcement learning method evaluated with 20 000 simulated episodes, after training with 500 000 episodes. Column *Diff* shows the percentage difference, with a negative number indicating that the reinforcement learning method needs less energy on average. Additionally, although not shown in the tables, the average number of charging stops was slightly smaller for the reinforcement learning method.

As can be seen in the tables, it is possible to save energy for all instances. Using the reinforcement learning method there was not a single episode during evaluation where the battery was completely depleted. In summary, keeping reliability it was possible to save up to 12% energy. Another advantage is that after training offline, the method could be very fast to run online. These are very important results, since transport companies could benefit from better routing instead of investing in more expensive vehicles with larger batteries.

In order to analyze how well the method anticipates customer requests and predicts energy consumption, the Expected Value of Perfect Information (EVPI) is calculated for the instances with 20 customers and charging. After an episode is simulated and all dynamic and stochastic information was revealed (i.e. customer requests and energy consumption), the a posteriori route is computed with the heuristics presented in Section 4.2. Table 8 shows the results of the experiments. Columns *Cost* show the average energy cost for the a posteriori routes. Columns *Diff* show the average of the difference between the route generated by the reoptimization or RL method compared with the a posteriori route for each episode. As it can be seen, if the problem was deterministic with all information known a priori, it would be possible to save an additional 7% energy on average using the RL method. However, considering the stochastic and dynamic nature of the problem, this gap is reasonable.

Table 2
Optimality gap for 20 customers without charging.

	Optimal		Heuristics		
	Energy	Runtime	Energy	Runtime	Diff
1	34 710	1 804.091	35 144	0.282	1.25%
2	38 963	1 800.141	39 417	0.384	1.17%
3	34 087	167.636	34 432	0.293	1.01%
4	34 581	726.187	35 787	0.391	3.49%
5	32 871	1 295.417	33 268	0.154	1.21%
6	36 607	457.107	36 607	0.463	0.00%
7	31 360	66.037	32 158	0.370	2.55%
8	29 492	111.356	30 288	0.344	2.70%
9	31 580	184.494	31 776	0.292	0.62%
10	36 941	314.138	36 946	0.147	0.01%
11	32 152	1 800.071	32 487	0.194	1.04%
12	33 678	1 800.154	33 785	0.259	0.32%
13	32 095	39.121	33 145	0.191	3.27%
14	31 844	103.143	31 844	0.314	0.00%
15	35 146	1 800.098	35 174	0.173	0.08%
16	32 049	878.403	32 845	0.414	2.48%
17	30 490	1 800.000	30 689	0.178	0.65%
18	33 054	806.150	33 414	0.190	1.09%
19	32 368	353.039	33 053	0.360	2.12%
20	31 577	1 800.085	31 622	0.330	0.14%
21	36 252	323.001	36 563	0.224	0.86%
22	33 284	388.136	34 276	0.108	2.98%
23	29 370	492.257	29 736	0.220	1.25%
24	32 916	226.454	33 236	0.353	0.97%
25	30 113	10.038	30 113	0.293	0.00%
Avg	33 103	781.870	33 512	0.277	1.25%
Min	29 370	10.038	29 736	0.108	0.00%
Max	38 963	1 804.091	39 417	0.463	3.49%

Table 3
Optimality gap for 10 customers with charging.

	Optimal		Heuristics		
	Energy	Runtime	Energy	Runtime	Diff
1	23 573	1.126	23 573	0.051	0.00%
2	28 988	3.767	29 347	0.108	1.24%
3	23 100	1.056	23 100	0.046	0.00%
4	26 963	0.654	26 984	0.050	0.08%
5	22 992	0.833	22 992	0.060	0.00%
6	26 067	2.959	26 067	0.065	0.00%
7	30 598	1.915	30 598	0.083	0.00%
8	24 406	1.714	24 406	0.055	0.00%
9	25 796	1.773	25 796	0.059	0.00%
10	28 187	1.346	28 187	0.072	0.00%
11	29 371	1.506	30 429	0.066	3.60%
12	23 155	4.122	23 764	0.087	2.63%
13	27 083	2.087	27 083	0.062	0.00%
14	27 117	0.738	27 117	0.082	0.00%
15	26 705	3.464	26 705	0.055	0.00%
16	25 394	0.267	25 394	0.069	0.00%
17	26 762	2.281	26 762	0.061	0.00%
18	26 101	1.808	26 101	0.076	0.00%
19	30 719	1.950	30 719	0.107	0.00%
20	27 326	2.844	27 326	0.263	0.00%
21	26 422	4.071	26 422	0.059	0.00%
22	27 255	1.853	27 255	0.074	0.00%
23	24 496	1.132	24 496	0.058	0.00%
24	27 107	3.901	27 446	0.065	1.25%
25	24 863	0.940	24 862	0.085	0.00%
Avg	26 422	2.004	26 517	0.077	0.35%
Min	22 992	0.267	22 992	0.046	0.00%
Max	30 719	4.122	30 719	0.263	3.60%

Table 4
Optimality gap for 20 customers with charging.

	Optimal		Heuristics		
	Energy	Runtime	Energy	Runtime	Diff
1	34 811	1 800.297	35 333	0.912	1.50%
2	39 667	1 800.112	39 895	0.767	0.58%
3	34 785	1 686.601	35 779	0.457	2.86%
4	35 530	1 800.104	35 798	1.015	0.75%
5	33 244	1 538.557	33 530	0.431	0.86%
6	36 607	326.296	36 645	0.675	0.10%
7	31 672	271.001	32 101	0.550	1.36%
8	29 492	149.716	30 272	0.709	2.65%
9	31 580	139.269	31 699	0.724	0.38%
10	36 994	599.560	38 253	0.918	3.40%
11	32 196	1 800.099	32 334	0.494	0.43%
12	33 519	1 800.121	33 833	0.741	0.94%
13	32 448	98.022	33 190	0.921	2.29%
14	32 149	480.661	32 371	0.662	0.69%
15	35 578	1 800.105	35 528	0.609	−0.14%
16	32 719	1 800.108	32 922	1.103	0.62%
17	30 103	1 800.085	30 092	0.577	−0.04%
18	33 285	948.279	34 394	0.552	3.33%
19	32 368	419.321	32 985	0.485	1.91%
20	31 847	1 800.080	32 309	0.757	1.45%
21	36 362	381.956	37 461	0.712	3.02%
22	33 284	334.312	34 227	0.532	2.84%
23	29 697	1 800.070	30 084	0.543	1.30%
24	33 537	1 800.088	34 229	0.631	2.06%
25	30 113	7.720	31 200	0.698	3.61%
Avg	33 343	1 087.302	33 859	0.687	1.55%
Min	29 492	7.720	30 084	0.431	−0.14%
Max	39 667	1 800.297	39 895	1.103	3.61%

Table 5
Deterministic online reoptimization.

	L=6000		L=3000		L=0	
	Cost	Fails	Cost	Fails	Cost	Fails
1	36 327	0	36 316	8	36 295	622
2	33 319	0	33 354	3	33 412	1 265
3	31 137	0	31 096	10	30 699	2 583
4	28 699	0	28 737	7	28 324	1 895
5	31 667	0	31 574	8	31 494	937
6	31 273	0	31 585	7	31 568	935
7	26 999	0	26 937	14	26 643	919
8	28 003	0	28 062	5	27 724	1 629
9	32 213	0	32 817	12	32 998	958
10	31 131	0	31 226	6	31 171	1 927
11	29 230	0	30 128	9	29 773	3 340
12	30 733	0	30 643	3	30 344	1 560
13	32 918	0	32 957	3	33 031	1 178
14	29 967	0	29 947	14	29 758	1 552
15	31 307	0	31 076	7	31 028	2 820
16	32 076	0	32 319	12	32 360	1 268
17	30 739	0	31 007	10	31 058	2 173
18	30 241	0	30 529	8	30 491	2 108
19	32 510	0	32 404	3	32 603	896
20	32 549	0	32 660	3	32 873	1 396
21	31 259	0	31 336	9	31 539	1 028
22	29 678	0	29 885	5	29 997	1 975
23	24 961	0	24 837	3	24 821	131
24	32 396	0	32 417	9	32 591	1 809
25	31 475	0	31 620	0	31 833	2 036
Avg	30 912	0	31 019	7	30 977	1 558
Max	36 327	0	36 316	14	36 295	3 340
Min	24 961	0	24 837	0	24 821	131

Table 6
Reinforcement learning for 10 customers test instances.

	Without charging			With charging		
	Reopt	RL	Diff	Reopt	RL	Diff
1	22 782	22 197	−2.57%	22 693	22 032	−2.91%
2	26 939	25 901	−3.85%	30 409	27 080	−10.95%
3	16 510	16 128	−2.32%	16 630	16 123	−3.05%
4	19 735	18 491	−6.30%	19 954	18 783	−5.87%
5	21 397	19 604	−8.38%	21 073	19 741	−6.32%
6	20 594	19 515	−5.24%	20 708	19 602	−5.34%
7	27 419	25 851	−5.72%	27 102	25 700	−5.17%
8	22 343	21 194	−5.14%	23 055	21 657	−6.06%
9	18 415	17 785	−3.42%	18 786	18 080	−3.76%
10	25 885	24 108	−6.87%	26 775	24 334	−9.12%
11	24 864	23 857	−4.05%	26 623	24 711	−7.18%
12	22 307	21 938	−1.65%	23 150	22 531	−2.68%
13	26 112	25 309	−3.07%	25 904	25 695	−0.81%
14	24 267	23 387	−3.62%	24 623	23 389	−5.01%
15	25 184	24 502	−2.71%	25 419	24 467	−3.75%
16	21 655	21 152	−2.32%	21 511	20 947	−2.62%
17	22 521	21 786	−3.26%	23 166	22 476	−2.98%
18	23 474	22 931	−2.31%	24 073	23 382	−2.87%
19	28 665	28 051	−2.14%	30 231	28 295	−6.41%
20	23 346	21 515	−7.84%	23 221	22 176	−4.50%
21	22 121	21 323	−3.61%	22 641	21 754	−3.92%
22	24 730	24 066	−2.69%	25 599	24 982	−2.41%
23	23 053	21 980	−4.66%	23 263	21 888	−5.91%
24	20 151	19 102	−5.21%	20 407	19 618	−3.86%
25	24 754	23 652	−4.45%	26 212	24 760	−5.54%
Avg	23 169	22 213	−4.14%	23 729	22 568	−4.76%
Max	28 665	28 051	−8.38%	30 409	28 295	−10.95%
Min	16 510	16 128	−1.65%	16 630	16 123	−0.81%

Table 7
Reinforcement learning for 20 customers test instances.

	Without charging			With charging		
	Reopt	RL	Diff	Reopt	RL	Diff
1	36 239	32 338	−10.77%	36 284	31 922	−12.02%
2	33 000	31 054	−5.90%	33 389	31 303	−6.25%
3	30 412	29 689	−2.38%	31 161	29 481	−5.39%
4	28 479	27 361	−3.93%	28 741	27 210	−5.33%
5	31 214	28 134	−9.87%	31 641	28 741	−9.16%
6	31 463	29 570	−6.02%	31 257	29 724	−4.90%
7	26 839	26 036	−2.99%	27 008	25 831	−4.35%
8	27 991	26 079	−6.83%	28 041	26 464	−5.62%
9	32 821	30 375	−7.45%	32 242	30 499	−5.41%
10	31 125	29 426	−5.46%	31 177	29 619	−5.00%
11	29 853	28 421	−4.80%	29 238	28 016	−4.18%
12	30 039	28 907	−3.77%	30 732	29 586	−3.73%
13	32 898	30 993	−5.79%	32 931	31 291	−4.98%
14	29 738	28 285	−4.89%	30 006	28 765	−4.13%
15	30 861	30 131	−2.36%	31 360	30 475	−2.82%
16	32 566	31 102	−4.50%	32 069	31 144	−2.88%
17	30 920	29 007	−6.19%	30 795	29 015	−5.78%
18	30 283	29 338	−3.12%	30 243	29 171	−3.54%
19	32 955	31 626	−4.03%	32 479	31 388	−3.36%
20	32 578	30 058	−7.73%	32 580	30 364	−6.80%
21	31 386	30 050	−4.26%	31 252	29 962	−4.13%
22	30 134	28 710	−4.72%	29 711	28 834	−2.95%
23	24 826	24 073	−3.03%	24 968	24 107	−3.45%
24	31 887	31 323	−1.77%	32 375	31 014	−4.20%
25	31 785	29 549	−7.03%	31 476	29 587	−6.00%
Avg	30 892	29 265	−5.18%	30 926	29 341	−5.06%
Max	36 239	32 338	−10.77%	36 284	31 922	−12.02%
Min	24 826	24 073	−1.77%	24 968	24 107	−2.82%

Table 8
Expected value of perfect information for 20 customers with charging.

	Reopt		RL	
	Cost	Diff	Cost	Diff
1	30 119	−16.11%	29 744	−7.22%
2	28 472	−14.38%	27 718	−11.63%
3	27 371	−11.86%	27 091	−7.87%
4	25 506	−10.73%	25 193	−7.09%
5	26 505	−15.65%	26 077	−8.97%
6	27 523	−11.06%	27 415	−7.48%
7	24 102	−10.44%	23 855	−7.72%
8	24 942	−10.73%	25 051	−5.08%
9	27 840	−13.12%	27 715	−8.76%
10	29 115	−6.16%	28 630	−3.59%
11	26 248	−9.82%	26 073	−6.75%
12	27 878	−8.98%	27 017	−8.46%
13	29 183	−10.80%	28 707	−7.97%
14	26 348	−11.63%	26 014	−9.89%
15	28 881	−7.62%	28 447	−6.49%
16	30 034	−6.16%	29 612	−5.14%
17	26 405	−13.67%	26 252	−9.28%
18	27 389	−9.08%	27 010	−7.05%
19	30 696	−5.17%	30 355	−3.02%
20	29 129	−10.08%	28 786	−5.14%
21	29 293	−6.02%	28 938	−5.78%
22	27 650	−6.64%	27 544	−5.48%
23	22 281	−10.50%	22 016	−8.44%
24	29 269	−9.31%	28 821	−8.04%
25	28 150	−10.28%	28 130	−4.81%
Avg	27 613	−10.24%	27 289	−7.09%
Max	30 696	−16.11%	30 355	−11.63%
Min	22 281	−5.17%	22 016	−3.02%

4.4.1. Dynamism analysis

In this subsection an analysis of how different levels of dynamism affect the results is presented. To do that, the test instances with 20 customers and battery $B = 30$ kWh are re-run with different settings. The first test is to change the time limit for receiving new customer requests, allowing only early requests with $K = 5$ or allowing late requests with $K = 15$, which means that new requests can be received until visiting K nodes. The second test is with a different number of requests known from the beginning, with $|\bar{C}_{early}| \approx 5$ or $|\bar{C}_{early}| \approx 15$. The exact number of initial requests vary for different test instances. The other parameters are the same as before.

Table 9
Different levels of dynamism.

$\lvert\bar{C}_{early}\rvert$	10	10	10	5	15
K	10	5	15	10	10
Avg	−5.06%	−5.64%	−3.26%	−4.08%	−3.30%
Max	−12.02%	−10.14%	−10.82%	−9.54%	−5.97%
Min	−2.82%	−1.30%	−0.99%	−1.83%	−0.87%

Table 9 presents the results of the comparison between the reoptimization and reinforcement learning methods. When varying the amount of known customer requests from the beginning (i.e. $|\bar{C}_{early}|$), the results are slightly less favorable for both scenarios. With less known requests (i.e. $|\bar{C}_{early}| \approx 5$) the environment is more dynamic and it is also harder to predict ahead. With more known customers (i.e. $|\bar{C}_{early}| \approx 15$) the environment loses dynamism and it gets closer to the static version of the problem. When varying the time limit for new requests, the results move in either direction. With a more dynamic environment (i.e. $K = 15$) the results are slightly less favorable. One of the main reasons is that it is more difficult to find better actions and predict ahead when there can be changes so late during the route. It should be noticed that $K = 15$ is a quite late decision epoch. Considering that many routes will not have 15 total requests, they might receive new requests when there are just a few customers left to visit, which reduces the possibilities of improving the route.

The last test was performed to evaluate the impact of random weight for the customer requests. It was assumed that each customer has a demanded weight that follows a normal distribution with expected value u_c and standard deviation $0.1u_c$ (i.e. 10% of the mean). Table 10 shows the results for the experiments. As it is shown, the average savings is quite similar to the results shown in Table 7, where the requested demand is deterministic.

Table 10
Reinforcement learning for 20 customers with charging and random weight.

	Reopt	RL	Diff
1	53 840	31 754	−12.36%
2	48 324	31 266	−6.23%
3	42 880	29 555	−5.10%
4	40 077	27 237	−5.31%
5	46 274	28 722	−9.36%
6	47 127	30 003	−3.96%
7	40 082	25 777	−4.51%
8	40 137	26 326	−6.06%
9	47 299	30 758	−4.45%
10	48 192	29 668	−4.76%
11	38 891	28 043	−4.13%
12	42 181	29 637	−3.40%
13	46 166	31 263	−4.90%
14	43 054	28 946	−3.60%
15	44 703	30 420	−2.90%
16	42 832	31 349	−2.23%
17	45 774	29 472	−4.24%
18	42 115	29 726	−1.74%
19	41 841	31 777	−2.11%
20	45 331	30 415	−6.57%
21	43 837	29 792	−4.67%
22	41 885	28 861	−2.74%
23	35 583	24 429	−2.01%
24	42 731	31 408	−2.95%
25	40 792	29 600	−5.91%
Avg	43 678	29 448	−4.65%
Max	53 840	31 777	−12.36%
Min	35 583	24 429	−1.74%

4.4.2. Algorithm and hyper-parameters discussion

Different values for the hyper-parameters and different variations of the algorithm were tested. However, since the main focus of the paper is to show the reliability and energy savings of the algorithm, an extensive report of these experiments is refrained from. But a short discussion about the choice of hyper-parameters and algorithm is presented in this subsection.

Since combinatorial optimization problems can have a very large number of combinations even for a small set of inputs, using reinforcement learning can be challenging because the state–action space can be extensive. Therefore our solution method includes a rollout function to be used every time a state is visited for the first time. With that it is possible to get a good action to be taken from a state directly when that state is first visited. Consequently it is not necessary to evaluate all the possible actions from a state to find a good candidate. As a result, it is possible to use a small epsilon for exploration. The choices of $\epsilon = 0.05$ and $\epsilon = 0.1$ for 20 and 10 customers respectively, is because in this way on average one action will be random per episode, exploring around the regions with good candidate solutions initially found with the rollout heuristics. Another consequence is that it is not necessary to decay epsilon. With a larger epsilon with decay, the algorithm will explore outside the regions with good candidates, which could eventually produce even better results, but will also take considerably longer time.

For the instances with 20 customers and 2 charging stations, the total number of states, using the reduced state representation presented in 3.2.1 is slightly over 241 million. The number of state–action pairs is then a bit over 5.5 billion. However, on average, only around 890 thousand state–action pairs are visited during training with 500 thousand episodes. Table 11 shows the results for every instance. That means that on average only 0.3% of the states are actually visited and only about 0.02% of the state–actions are evaluated. That is mainly due to two reasons: the first is because there are many unreachable or invalid states; the second because the algorithm focuses on good solution candidates from the beginning.

Different variations of Algorithm 1 were also tried. The updates of Q_e and Q_r tables are done by averaging over time for a state–action pair. A few other approaches were tested, including Temporal Difference learning with different values of alpha and a moving average with different window lengths.

The risk of failure is managed by the policy with two layers as shown in Section 3.2.2. The first layer tries to keep the risk of battery depletion within a maximum R. The second layer tries to keep the nearest charging station reachable from any customer location. They are complementary, since the first helps during exploration to find good candidate actions by chance-constraining the risk, while the second can override the preferred action in order to guarantee safety. Removing the second layer produces slightly higher energy savings but there are some failures. If the first layer is removed there are no failures but the energy savings are reduced. Different levels of risk were tested but the best results were achieved by keeping it between $5\% \leq R \leq 10\%$ to maintain a good balance between the two layers of safety in the policy. A larger accepted risk increases the chance of the second layer being triggered, which increases the average energy cost. A smaller accepted risk becomes too restrictive and also increases the average energy cost.

Table 11
States and state–action pairs visited during training with 500 000 episodes.

	States	State–actions
1	916 905	1 185 262
2	570 438	752 203
3	520 798	680 771
4	385 481	531 307
5	788 581	1 023 170
6	540 408	722 163
7	560 552	750 294
8	538 015	713 170
9	1 192 212	1 508 462
10	691 211	912 817
11	458 616	639 037
12	827 253	1 085 228
13	809 451	1 074 588
14	738 704	997 174
15	612 099	826 111
16	792 716	1 047 321
17	861 451	1 111 264
18	387 920	537 679
19	731 495	999 865
20	912 723	1 154 258
21	404 824	554 731
22	784 143	1 028 691
23	528 140	697 421
24	640 545	848 024
25	630 087	865 354
Avg	672 991	889 855
Max	1 192 212	1 508 462
Min	385 481	531 307

As mentioned before, runtime performance was not the focus of the paper, but a short discussion is presented, although convergence to optimality is not included. The algorithm was implemented in Matlab 2017b. An ordinary computer with a 3.1 GHz dual-core i5 processor and 16 Gb RAM was used for the experiments. Training runtime for 500 thousand episodes was 126 min on average for the test instances with 20 customers and charging (i.e. $B = 30\,000$). For the validation of those instances, the average time for running 20 thousand episodes for the reoptimization method was 51 min while for the trained RL agent it was 33 s. However, training runtime for 100 thousand episodes was 45 min producing good savings in energy and keeping reliability. Therefore, by lowering the number of training episodes it was possible to still get good results and keep reliability but reducing considerably the amount of time required for training. Total runtime and energy savings for 50, 100, 250 and 500 thousand episodes are shown in Table 12. Fig. 4 illustrates how reducing the number of episodes reduces the total time without reducing much the energy savings. It is important to emphasize that the implementation was not focusing on speed and that there are several ways to make it faster. Therefore, with proper implementation it should be possible to use it in real transport operations. Furthermore, the online runtime difference between the reoptimization and the trained RL agent are substantial.

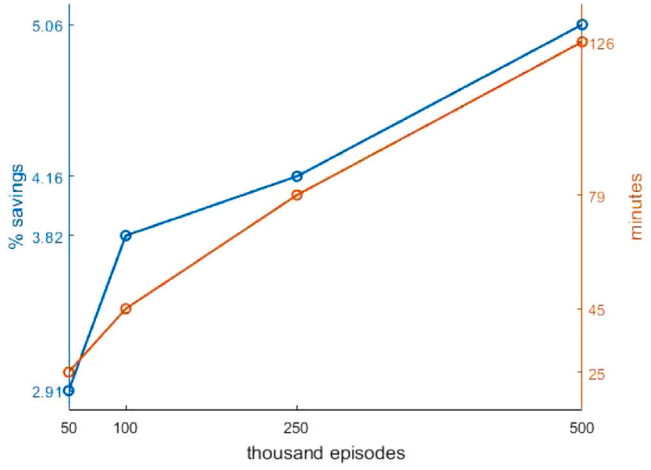

Fig. 4. Number of training episodes with average savings and runtime for the instances with 20 customers and charging.

Table 12
Number of training episodes with savings (%) and runtime (minutes) for the instances with 20 customers and charging.

	500k		250k		100k		50k	
	Savings	Time	Savings	Time	Savings	Time	Savings	Time
Avg	−5.06%	126	−4.16%	79	−3.82%	45	−2.91%	25
Max	−12.02%	377	−11.24%	234	−10.50%	131	−8.76%	62
Min	−2.82%	20	−1.72%	12	−1.03%	10	−0.27%	6

5. Conclusion

This paper introduces the Dynamic Stochastic Electric Vehicle Routing Problem (DS-EVRP) and models it as a Markov Decision Process. A solution method is proposed based on Safe Reinforcement Learning with the following contributions: (i) a Value Function Approximation with a reduced state representation to decrease the size of the Q-tables and improve exploration; (ii) a chance-constrained policy with two layers of safety to minimize energy consumption and avoid failures (i.e. battery depletion while in transit); (iii) a training approach with a rollout heuristics based on tabu-search in order to focus the exploration on the most relevant parts of the state–action space. Lastly, the paper presents a set of computational experiments to evaluate the solution method and analyze its properties. The main results can be summarized as follows:

- The reinforcement learning method could save on average 4.8% (up to 12%) energy by planning the route and charging anticipatively, compared with the deterministic online reoptimization approach;
- Although the reoptimization method can solve the problem, it is not as reliable as the reinforcement learning and can produce some failures (i.e. battery depletion) even when keeping a certain margin for the battery level;
- By using the two layer policy, the reinforcement learning method demonstrated more reliability than the reoptimization method, avoiding failures altogether;
- The method performed well for different levels of dynamism. The reinforcement learning algorithm performed better than the reoptimization method even when allowing late requests, knowing only a few customer requests before leaving the depot or with random customer demands (i.e. payload weight);
- The proposed training process can produce satisfactory results even with a low number of episodes, due to the rollout function and the Value Function Approximation. Therefore it could be tailored to be deployed in current transport operations;
- The online computation and memory usage could be suitable for onboard implementation, since the training is done offline and the reduced state representation decreases the used memory for the Q-tables. Furthermore, the trained RL agent has a significantly lower online runtime than the reoptimization method.

To the best of our knowledge there is no published paper considering the presented problem. With the proposed method it is possible to reliably integrate electric commercial vehicles into dynamic and stochastic transport operations. By anticipating dynamic customer requests and predicting energy consumption, this approach can help drivers and dispatchers to plan their routes in real-time, including charge planning. Additionally, it has the potential to support achieving environmental goals. Therefore, implementing this method opens an avenue for fulfilling the high demand on service levels from the on-demand economy while transitioning to a fossil free economy.

Possible future work includes adapting the method for large instances (e.g. 100 customers and above) with multiple vehicles. It would be interesting to investigate the use of Neural Networks either by using Deep Q-Learning or policy optimization methods. Since energy consumption is just one aspect in the total cost of transport operations, it could be interesting to analyze different objective functions for the problem, aiming at operational cost minimization, for instance by including time aspects and partial recharging. The same method could also be applied for time-windows, by estimating the risk of not meeting them. Additionally, it could be possible to integrate other sources of stochasticity, such as waiting times at charging stations or service times at customers.

Funding

This work was co-funded by Vinnova, Sweden through the project EL-FORT 2: Electric Fleet Optimization in Real-Time, by the FFI program Efficient and Connected Transport Systems.

Balázs Kulcsár and Ivan Sanchez-Diaz acknowledge the support of Transport Area of Advance within Chalmers University of Technology, Sweden.

Appendix A. Supplementary data

The supplementary material provided with this article contains all test instances used in the experiments. Each test instance is stored in a folder named *instance_10_1* to *instance_10_25* for the instances with 10 customers and *instance_20_1* to *instance_20_25* for the instances with 20 customers. Each folder contains 7 CSV files:

- *customers.csv*
- *matrixAlpha.csv*
- *matrixBeta.csv*
- *matrixDistance.csv*
- *matrixSigma1.csv*
- *matrixSigma2.csv*
- *matrixTime.csv*

The file *customers.csv* is a table with the first column being the weight demand (kg) for each customer and the second being the probability (%) of each customer sending a request during an episode.

The other files are matrices representing the graph $G = (\mathcal{V}, \mathcal{A})$, as described in Section 2. Each line and column represents a node in the graph. The first one is the depot, the following are the customers and the last two are the charging stations.

Files *matrixAlpha.csv* and *matrixBeta.csv* contain the coefficients for the expected energy for Eq. (4). Files *matrixSigma1.csv* and *matrixSigma2.csv* contain the coefficients for the energy variance for Eq. (5). Those coefficients are used to get the energy consumption (Wh) probability distribution for each path, as a function of total mass (i.e. curb weight + payload).

Files *matrixDistance.csv* and *matrixTime.csv* represent the distance (m) and time (seconds) for driving each path. They are not actually used in the experiments but are provided for a better visualization of the test instances.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.tre.2021.102496.

References

Adler, J.D., Mirchandani, P.B., 2014. Online routing and battery reservations for electric vehicles with swappable batteries. Transp. Res. B 70, 285–302.

Asamer, J., Graser, A., Heilmann, B., Ruthmair, M., 2016. Sensitivity analysis for energy demand estimation of electric vehicles. Transp. Res. D 46, 182–199.

Basso, R., Kulcsár, B., Egardt, B., Lindroth, P., Sanchez-Diaz, I., 2019. Energy consumption estimation integrated into the Electric Vehicle Routing Problem. Transp. Res. D 69, 141–167. http://dx.doi.org/10.1016/j.trd.2019.01.006.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., 2021. Electric vehicle routing problem with machine learning for energy prediction. Transp. Res. B 145, 24–55. http://dx.doi.org/10.1016/j.trb.2020.12.007.

Bektaş, T., Demir, E., Laporte, G., 2016. Green vehicle routing. In: Green Transportation Logistics. Springer, Cham, pp. 243–265.

Bektaş, T., Ehmke, J.F., Psaraftis, H.N., Puchinger, J., 2019. The role of operational research in green freight transportation. European J. Oper. Res. 274 (3), 807–823.

Bektaş, T., Laporte, G., 2011. The pollution-routing problem. Transp. Res. B 45 (8), 1232–1250.

Chen, X., Ulmer, M.W., Thomas, B.W., 2021. Deep Q-learning for same-day delivery with vehicles and drones. European J. Oper. Res..

Çimen, M., Soysal, M., 2017. Time-dependent green vehicle routing problem with stochastic vehicle speeds: An approximate dynamic programming algorithm. Transp. Res. D 54, 82–98.

Codeca, L., Frank, R., Engel, T., 2015. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In: 2015 IEEE Vehicular Networking Conference (VNC). IEEE, pp. 1–8. http://dx.doi.org/10.1109/VNC.2015.7385539.

Cuma, M.U., Koroglu, T., 2015. A comprehensive review on estimation strategies used in hybrid and battery electric vehicles. Renew. Sustain. Energy Rev. 42, 517–531.

Demir, E., Bektaş, T., Laporte, G., 2011. A comparative analysis of several vehicle emission models for road freight transportation. Transp. Res. D 16 (5), 347–357.

Demir, E., Bektaş, T., Laporte, G., 2014. A review of recent research on green road freight transportation. European J. Oper. Res. 237 (3), 775–793.

EC, 2011. Transport White Paper - Roadmap to a Single European Transport Area – Towards a Competitive and Resource Efficient Transport System. Technical Report, European Commission, Brussels, http://dx.doi.org/10.2832/30955.

Erdoğan, S., Miller-Hooks, E., 2012. A Green Vehicle routing problem. Transp. Res. E 48 (1), 100–114.

Eshtehadi, R., Fathian, M., Demir, E., 2017. Robust solutions to the pollution-routing problem with demand and travel time uncertainty. Transp. Res. D 51, 351–363.

Fan, J., Wang, X., Ning, H., 2006. A multiple vehicles routing problem algorithm with stochastic demand. In: Proceedings of the World Congress on Intelligent Control and Automation (WCICA), Vol. 1. pp. 1688–1692.

Felipe, A., Ortuño, M.T., Righini, G., Tirado, G., 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. Transp. Res. 71, 111–128.

Fiori, C., Marzano, V., 2018. Modelling energy consumption of electric freight vehicles in urban pickup/delivery operations: analysis and estimation on a real-world dataset. Transp. Res. D 65, 658–673.

Garcıa, J., Fernández, F., 2015. A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16 (1), 1437–1480.

Gendreau, M., Jabali, O., Rei, W., 2016. 50th Anniversary invited article future research directions in stochastic vehicle routing. Transp. Sci. 50 (4), 1163–1173.

Gendreau, M., Laporte, G., Séguin, R., 1996. Stochastic vehicle routing. European J. Oper. Res. 88 (1), 3–12.

Genikomsakis, K.N., Mitrentsis, G., 2017. A computationally efficient simulation model for estimating energy consumption of electric vehicles in the context of route planning applications. Transp. Res. D 50, 98–118.

Goodson, J.C., Ohlmann, J.W., Thomas, B.W., 2013. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. Oper. Res. 61 (1), 138–154.

He, F., Yang, J., Li, M., 2018. Vehicle scheduling under stochastic trip times: An approximate dynamic programming approach. Transp. Res. C 96, 144–159.

Holguín-Veras, J., Thorson, E., Wang, Q., Xu, N., González-Calderón, C., Sánchez-Díaz, I., Mitchell, J., 2013. Urban Freight Tour Models: State of the art and practice. In: Freight Transport Modelling. Emerald Group Publishing Limited, pp. 335–351.

Jafari, E., Boyles, S.D., 2017. Online charging and routing of electric vehicles in stochastic time-varying networks. Transp. Res. Rec.: J. Transp. Res. Board 2667 (1), 61–70.

Jin, J., Zhou, M., Zhang, W., Li, M., Guo, Z., Qin, Z., Jiao, Y., Tang, X., Wang, C., Wang, J., Wu, G., Ye, J., 2019. CoRide: Joint order dispatching and fleet management for multi-scale Ride-Hailing Platforms. In: International Conference on Information and Knowledge Management, Proceedings, Vol. 10. Association for Computing Machinery, pp. 1983–1992.
Jochem, P., Doll, C., Fichtner, W., 2016. External costs of electric vehicles. Transp. Res. D 42, 60–76.
Laporte, G., Louveaux, F., Mercure, H., 1992. The vehicle routing problem with stochastic travel times. Transp. Sci. 26 (3), 161.
Liang, Y., Ding, Z., Ding, T., Lee, W.J., 2021. Mobility-aware charging scheduling for shared on-demand electric vehicle fleet using deep reinforcement learning. IEEE Trans. Smart Grid 12 (2), 1380–1393.
Liao, T.-Y., 2017. On-line vehicle routing problems for carbon emissions reduction. Comput.-Aided Civ. Infrastruct. Eng. 32 (12), 1047–1063.
Lin, C., Choy, K.L., Ho, G.T., Chung, S.H., Lam, H.Y., 2014. Survey of Green Vehicle Routing Problem: Past and future trends. Expert Syst. Appl. 41 (4), 1118–1138.
Liu, S., Jiang, H., Chen, S., Ye, J., He, R., Sun, Z., 2020. Integrating Dijkstra's algorithm into deep inverse reinforcement learning for food delivery route planning. Transp. Res. E 142, 102070.
Mao, C., Shen, Z., 2018. A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. Transp. Res. C 93, 179–197.
Masmoudi, M.A., Hosny, M., Demir, E., Genikomsakis, K.N., Cheikhrouhou, N., 2018. The dial-a-ride problem with electric vehicles and battery swapping stations. Transp. Res. E 118, 392–420.
Nazari, M., Oroojlooy, A., Snyder, L., Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem. In: Advances in Neural Information Processing Systems. pp. 9839–9849.
Novoa, C., Storer, R., 2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European J. Oper. Res. 196 (2), 509–515.
Oyola, J., Arntzen, H., Woodruff, D.L., 2018. The stochastic vehicle routing problem, a literature review, part I: models. EURO J. Transp. Logist. 7 (3), 193–221.
Pelletier, S., Jabali, O., Laporte, G., 2016. Goods distribution with electric vehicles: Review and research perspectives. Transp. Sci. 50 (1), 3–22.
Pelletier, S., Jabali, O., Laporte, G., 2019. The electric vehicle routing problem with energy consumption uncertainty. Transp. Res. B 126, 225–255.
Powell, W.B., 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality, Vol. 703. John Wiley & Sons.
Powell, W.B., Simao, H.P., Bouzaiene-Ayari, B., 2012. Approximate dynamic programming in transportation and logistics: a unified framework. EURO J. Transp. Logist. 1 (3), 237–284.
Psaraftis, H.N., Wen, M., Kontovas, C.A., 2016. Dynamic vehicle routing problems: Three decades and counting. Networks 67 (1), 3–31.
Qi, X., Wu, G., Boriboonsomsin, K., Barth, M.J., 2018. Data-driven decomposition analysis and estimation of link-level electric vehicle energy consumption under real-world traffic conditions. Transp. Res. D 64, 36–52.
Ritzinger, U., Puchinger, J., Hartl, R.F., 2015. A survey on dynamic and stochastic vehicle routing problems. Int. J. Prod. Res. (MAY), 1–17.
Sanchez-Diaz, I., Palacios-Argüello, L., Levandi, A., Mardberg, J., Basso, R., 2020. A time-efficiency study of medium-duty trucks delivering in urban environments. Sustainability (Switzerland) 12 (1), 425.
Schneider, M., Stenger, A., Goeke, D., 2014. The electric vehicle routing problem with time windows and recharging stations. Transp. Sci. 48 (4), 500–520.
Secomandi, N., 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Comput. Oper. Res. 27 (11–12), 1201–1225.
Secomandi, N., 2001. A rollout policy for the vehicle routing problem with stochastic demands. Oper. Res. 49 (5), 796–802.
Shi, J., Gao, Y., Wang, W., Yu, N., Ioannou, P.A., 2019. Operating electric vehicle fleet for ride-hailing services with reinforcement learning. IEEE Trans. Intell. Transp. Syst. 1–13.
Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.
Sweda, T.M., Dolinskaya, I.S., Klabjan, D., 2017. Adaptive routing and recharging policies for electric vehicles. Transp. Sci. 51 (4), 1326–1348.
Tahami, H., Rabadi, G., Haouari, M., 2020. Exact approaches for routing capacitated electric vehicles. Transp. Res. E 144, 102126.
Tang, X., Lin, X., He, F., 2019. Robust scheduling strategies of electric buses under stochastic traffic conditions. Transp. Res. C 105, 163–182.
Toth, P., Vigo, D. (Eds.), 2014. Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics.
Turan, B., Pedarsani, R., Alizadeh, M., 2020. Dynamic pricing and fleet management for electric autonomous mobility on demand systems. Transp. Res. C 121, 102829.
Ulmer, M.W., 2017. Approximate Dynamic Programming for Dynamic Vehicle Routing, Vol. 61. Springer.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Hennig, M., 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. Transp. Sci. 53 (1), 185–202.