Off-line approximate dynamic programming for the vehicle routing problem with a highly variable customer basis and stochastic demands

Mohsen Dastpak[a,b,c,*], Fausto Errico[a,b,c], Ola Jabali[c,d]

[a] *Department de génie de la construction, École de technologie supérieure, , Montréal, H3C 1K3, QC, Canada*
[b] *GERAD, , Montréal, H3T 1J4, QC, Canada*
[c] *CIRRELT, , Montréal, H3C 3J7, QC, Canada*
[d] *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, 20133, , Italy*

Abstract

We study a stochastic variant of the vehicle routing problem (VRP) arising in the context of domestic donor collection services. The problem we consider combines the following attributes. Customers requesting services are *variable*, in the sense that they are stochastic, but are not restricted to a predefined set. Furthermore, demand volumes are also stochastic and are observed upon visiting customers. The objective is to maximize the expected served demands while meeting vehicle capacity and time restrictions. We call this problem the VRP with a highly Variable Customer basis and Stochastic Demands (VRP-VCSD). We first propose a classical Markov Decision Process (MDP) formulation for the VRP-VCSD. The resulting model is, however, unusable due to the explosion in the dimension of the state and action spaces.

To solve the VRP-VCSD, we propose a number of methodological contributions aimed at reducing the state and action spaces. We first reformulate the MDP as an MDP with a consecutive action selection procedure. In this formulation, we enforce the treatment of a single vehicle (as opposed to multiple vehicles) at each decision epoch. We then introduce an observation function that selects a subset of the available information, which is deemed relevant for the considered vehicle in each epoch. We develop a Q-learning algorithm called QN-CO. In particular, we use a continuous state representation and incorporate a two-layer artificial neural network to approximate the Q values. Furthermore, we propose an aggregation strategy yielding a fixed-size output. Finally, we enhance our algorithm with Replay Memory and a Double Q Network.

We conduct a thorough computational analysis. Results show that QN-CO considerably outperforms five benchmark policies. Moreover, we show that QN-CO can compete with specialized methods developed for the particular case of the VRP-VCSD where customer locations and expected demands are known in advance.

Keywords: Stochastic Vehicle Routing Problem, Approximate Dynamic Programming, Q-learning

1. Introduction

Domestic Donor Collection Services (DDCSs) are services that collect household-based solid waste, such as furniture, kitchen tools, and spent textiles. DDCSs typically operate digital-based platforms (e.g., La Collecte Foundation (2023) and 211 of Greater Montréal (2023)) to facilitate their operations. We study the distribution planning problem arising in the context of DDCSs, which can be characterized as follows. First, similar to e-commerce applications, DDCSs deal with a very large basin of service requests, potentially comprising all residential addresses in a city. The number of customers to be served each day is, however,

*Corresponding author
Email address: mohsen.dastpak.1@ens.etsmtl.ca (Mohsen Dastpak)

limited and generally highly variable from one day to another. Second, DDCSs require donors to classify the size of their products from a list of preset categories. Therefore, while donors estimate the size of their products, the actual product sizes are only revealed upon the arrival of the collection service. Third, the majority of transported products are dry, and thus can be loaded in the same vehicle. Finally, DDCSs often deploy collection services with volunteers (Curran and Williams 2010). Therefore, it is likely that not all products may be collected on a given day.

Another essential feature of the application under study is a certain repetition of the information process and problem parameters. Donors' requests are registered daily, from the same basin of donors, in a given time period called the *registration period*. At the end of this period, pickup operations are planned, and vehicle operations begin afterward. Thus, in essence, the DDCSs planning problem is rather similar from one day to another, except that the set of donors requesting service changes. We refer to this feature by the term *variable customers*. We note that statistical information on donors' locations and demands are generally available in the form of historical data.

Our main objective is to develop a planning tool that can be adopted by DDCSs to program their collection activities. Given the nature of the application, this tool should capture the repetitive nature of the planning problem and should not require intensive daily computational effort. To achieve this, we introduce a variant of the vehicle routing problem (VRP) that is compatible with the DDCS characteristics. We call this the Vehicle Routing Problem with a highly Variable Customer basis and Stochastic Demands (VRP-VCSD). We assume that the set of customers is variable and that customers may request service from anywhere within a given service area. We model the evolution of the information process as follows. First, since the service requests are received during the registration period, which precedes the planning and the dispatching period, we assume that customer locations and expected demands are known at the beginning of the planning phase. Second, the actual customer demand is observed upon visiting the customer. The VRP-VCSD then consists in dispatching a homogeneous fleet of vehicles initially located at the depot to serve the demands of the realized set of customers. Vehicles are allowed to perform so-called *preventive restocking* (see, for example Louveaux and Salazar-González 2018), thus enabling the possibility to return to the depot before the vehicle capacity is filled. Furthermore, similar to Goodson et al. (2013, 2016), we assume that the fleet operates during a limited period of the day. Thus, some demand may remain unserved, and the objective is then to maximize the total served demand.

We first propose an MDP model for the VRP-VCSD. This is a classical MDP model, where the decision-maker has full information on the vehicles' and clients' statuses and establishes the routes for all vehicles. The resulting model is, however, intractable even for small problem instances. The main challenge comes from the multi-vehicle nature of the VRP-VCSD and the consequent explosion in the dimension of the state and action spaces. Therefore, to solve the VRP-VCSD, we propose a number of methodological contributions. We first reformulate the classical MDP in a manner that significantly reduces the action space. We establish this in two main steps. As a first step, we reformulate the VRP-VCSD as an MDP with a consecutive action selection procedure, we denote this formulation by MDP-C. At each decision epoch, this formulation considers decisions related to a single vehicle, as opposed to multiple vehicles. In particular, we impose a consecutive action selection strategy for the case where decisions need to be made for multiple vehicles at a given decision epoch. Since the MDP-C substantially reduces the decision space, we are ultimately able to use it for handling small sized instances. As a second step, we introduce an observation function in the MDP-C, yielding a third formulation which we denote by MDP-CO. The observation function selects a subset of the available information, which is deemed relevant for the considered vehicle in each epoch.

Two main categories exist for solving routing problems expressed in terms of MDPs: *off-line* and *on-line* algorithms (Ritzinger et al. 2016). In general, off-line methods invest most of the available computing time before operations start, and use the available information to provide an easy-to-use operational policy that can be repeatedly applied. On the contrary, on-line algorithms concentrate most of the computing effort during operations, thus taking advantage of the information revealed during operations. Given the repetitive nature of our application, and the limited computational resources typical of DDCSs, we turn our attention to off-line methods. To find the resulting operational policy, we resort to Reinforcement Learning, and in particular, we use a Q-learning algorithm (Watkins and Dayan 1992), which is an approximate and model-free form of the Value Iteration Algorithm for stochastic dynamic programming (Powell 2011). Q-learning

enables estimating the value of each state-action pair via simulation. In its simplest form, states and actions are discrete, and state-action values are represented as a lookup table.

We propose a Q-learning based algorithm for the MDP-CO, which we denote by QN-CO. In this algorithm, we use a continuous state representation and incorporate a two-layer artificial neural network to approximate the Q values. One of the main MDP-CO challenges is due to having variable-sized customer sets. To handle this, we propose an aggregation strategy based on a particular heatmap-style encoding technique yielding a fix-sized output vector. Furthermore, to improve the overall performance of our algorithm, we include Replay Memory (Mnih et al. 2015) and Double Q Network (Van Hasselt et al. 2016).

We conduct a thorough computational analysis. First, as no instances are available for the VRP-VCSD in the literature, we construct a new instance set. We then implement five benchmark policies to compare with the QN-CO. These include a random policy, a greedy policy, a hybrid greedy policy, an a priori policy benchmark, and a Q-Network for the MDP-C formulation denoted by QN-C. We show that the QN-CO considerably outperforms all considered benchmark policies. In particular, the comparison between the QN-CO and QN-C demonstrates the added value of including the observation function and the proposed aggregation strategies. To provide a comparison with existing literature, we trained our QN-CO on instances of the Vehicle Routing Problem with Stochastic Demand (VRPSD), which is a particular case of the VRP-VCSD where the customer set and expected demands are known in advance. Results show that even though our algorithm is not tailored to exploit the in-advance knowledge of customer locations, it is competitive with the current state-of-the-art algorithm of Goodson et al. (2016). Further computational analysis is then geared towards testing how the training phase of the QN-CO can be guided towards generalizations over different instances and problem dimensions, such as duration limits and stochastic variability. Finally, we show that the obtained policies can be easily deployed in an *on-line* algorithm, thus further improving the performance of the QN-CO.

To summarize, the contributions of the present work are as follows:

- We model the DDCSs by introducing the VRP-VCSD, which is a variant of the VRP with a variable customer basis and stochastic demand. For this new problem, we provide an MDP formulation based on a centralized decision-making framework.
- We reformulate the VRP-VCSD as an MDP with a consecutive action selection procedure. We then include a tailored observation function, which is coupled with a state space aggregation strategy.
- We solve the VRP-VCSD via a state-of-the-art Q-learning algorithm, featuring recent accelerating strategies such as Replay Memory and Double Q Networks. Furthermore, we introduce an easy-to-implement hybrid greedy policy which yields promising results.
- We perform extensive computational experiments showing that the QN-CO clearly outperforms five benchmarks for VRP-VCSD. Furthermore, when tested on instances of the VRPSD, QN-CO is competitive with state-of-the-art methods specifically designed for that problem. Moreover, we show that the training phase of the QN-CO can be generalized to tackle problem instances varying in terms of several dimensions, such as duration limits and stochastic variability. Finally, we show that the obtained off-line policies are suitable to be used as base policies in a rollout algorithm.

The rest of the paper is organized as follows. We provide a detailed literature review in Section 2. In Section 3, we describe the VRP-VCSD and provide its MDP formulation. In section 4, we describe our solution methodology, and present our computational results in Section 5. Finally, we present our conclusions in Section 6.

2. Literature Review

We first review optimization problems related to the VRP-VCSD in Section 2.1. We then review MDP-based solution methods for stochastic routing problems in Section 2.2.

2.1. Related problems

The VRP-VCSD belongs to the family of Stochastic VRPs (SVRPs), which have been extensively studied in the literature (see Oyola et al. (2017, 2018) for a comprehensive review). The VRP-VCSD accounts for two sources of uncertainty; the first concerns the set of customers requesting the daily service, and the second concerns their demands. The literature addressing customer uncertainty can be classified in two main streams. In the first stream, a planner receives requests up until the beginning of the operations period, e.g., 8:00 AM. As the operations begin, no more requests are accepted, and the observed set of customers is served (e.g., Lin et al. (2021)). We refer to this form of uncertainty as *static* customers. In the second stream, a planner may continue receiving requests during the operations period and decides whether to accept or reject them (see, for example Ulmer et al. 2017, Chen et al. 2022). We refer to this form of uncertainty as *dynamic* customers.

Among SVRPs with static customers, we further distinguish between two categories, according to the assumptions made about the set of potential customers. The first category assumes that the set of potential customers is fixed and known in advance (e.g., Gendreau et al. (1995), Waters (1989), Benton and Rossetti (1992), Haughton (2002)). In this setting, only a subset of customers is "present" each day. Given that the customer set is known in advance and does not change from one day to another, we denote this as the *fixed* customer assumption. It is worth noting that several works on stochastic versions of the Traveling Salesman Problem share this modeling framework, such as Voccia et al. (2013). Conversely, the second category of works assumes that customers may request service from any location in a given service area (e.g., Lin et al. (2021), Li et al. (2021a)). As previously mentioned, we refer to this feature as variable customers.

Works focusing on demand uncertainty generally assume that the set of customers requesting service is known in advance, i.e., fixed. Within this stream, we identify two main problem settings. In the first, vehicles must fully serve all customer demands. Due to the uncertainty of the latter, vehicles may have to return to the depot to reload and continue the service. In such cases, the objective is to minimize the expected total costs, which includes returns to the depot (e.g., Louveaux and Salazar-González (2018)). In the second stream, similarly to the VRP-VCSD, a deadline on the duration of the operations is considered. Mendoza et al. (2016) assume a soft deadline and penalize late services. Other authors do not allow late services, and in particular in Erera et al. (2010), additional vehicles might be dispatched if needed, while in Goodson et al. (2013, 2016) not all customers might be served, and the objective is to maximize the total demand served within the deadline.

The VRP-VCSD belongs to the category of SVRPs with static and variable customers, and additionally considers a second layer of uncertainty on customer demands. To the best of our knowledge, this problem setting has not been previously addressed in the literature.

2.2. Solution methods

MDP formulations are commonly used to model SVRPs. This section reviews MDP-based solution methods for the SVRP with stochastic customers and/or demands. In particular, we focus on Approximate Dynamic Programming (ADP), Reinforcement Learning (RL), and Multi-Agent Reinforcement Learning (MARL). For a more general discussion on these methods, the reader is referred to Soeffker et al. (2022). Table 1 summarizes the literature that is most relevant to our work. The first part of Table 1 reports works with a single-vehicle, whereas the second part reports works with multi-vehicles. The multi-vehicle nature of the VRP-VCSD requires considerably more complex methods. Thus, our review focuses on how large action and state spaces are addressed in multi-vehicle problems.

We classify MDP-based methods according to the approach used in making routing decisions. The first approach determines, at each *decision epoch*, the next location each vehicle should visit (see, for example Maxwell et al. 2010, Oda and Joe-Wong 2018, Chen et al. 2019). Conversely, at each decision epoch, the second approach maintains a temporary route of which only the first location is retained. These routes are dynamically updated at each decision epoch. We observe that in all reviewed papers, it is assumed that once a vehicle is assigned the next location, it will not be diverted. The first approach provides a simple and effective way to tackle routing problems where most of the customer set is not known in advance, such as in some dynamic settings (Oda and Joe-Wong 2018, Chen et al. 2019) or variable customers (Lin et al.

2021, Li et al. 2021a). Therefore, we adopt this approach in this paper. The second approach is more suited to a situation where a large portion of the customer locations are known in advance, such as Goodson et al. (2013, 2016), or, in some dynamic settings, where the new arriving customers need to be allocated in existing routes, such as Ulmer (2020), Joe and Lau (2020). We remark that solution methods may also combine these two strategies, as for example, Kullman et al. (2021) where a ride-hailing problem is addressed.

Reference	Problem	Obj. Func.	Uncert.		Fleet	TL	RD	Solution method (comp.)
			Cust.	Other				
Secomandi (2001)	SVRP	min TT	F	D	S	-	RB	RA (On.)
Novoa and Storer (2009)	SVRP	min TT	F	D	S	-	RB	RA (On.)
Peng et al. (2020)	SVRP	min C	V	-	S	-	N	PG + AM (Off.)
Brinkmann et al. (2019a)	SIRP	min No. SF	F	D	S	-	N	H (On.)
Ulmer et al. (2017)	DVRP	max No. SR	D	-	S	L	RB	RA + VFA (On.-Off.)
Nazari et al. (2018)	DVRP	max SD	D	-	S	TW	N	PG + AM (Off.)
Fan et al. (2006)	SVRP	min TT	F	D	M	-	✗	DP + RA (On.)
Goodson et al. (2013)	SVRP	max SD	F	D	M	L	RB	DP + RA (On.)
Goodson et al. (2016)	SVRP	max SD	F	D	M	L	RB	DP + RA (On.)
Lin et al. (2021)	SVRP	min C	V	-	M	TW	N	PG (Off.)
Li et al. (2021a)	SVRP	min TT	V	-	M[1]	TW	N	PG + AM (Off.)
Brinkmann et al. (2019b)	SIRP	min No. SF	F	D	M	-	N	H (On.)
Joe and Lau (2020)	DVRP	min C	D	-	M	TW	RB	SA + VFA (On.-Off.)
Chen et al. (2022)	SDD	max No. SR	D	-	M[1]	-	RB[2]	Q-learning + H (On-Off.)
Ulmer (2020)	SDD	max R	D	-	M	L, TW	RB	H + VFA (On.-Off.)
Li et al. (2021b)	PDVRP	min C	D	-	M	TW	RB	DP + AP + Q-learning (On.-Off.)
Kullman et al. (2021)	DARP	max P	D	-	M	L, TW	RB + N	DC + Q-learning (Off.)
Maxwell et al. (2010)	DRP	min No. DS	D	-	M	TW	N	API + Con (Off.)
Oda and Joe-Wong (2018)	DRP	min No. RR + IT	D	-	M	TW	N	DC + Q-learning (Off.)
Chen et al. (2019)	DRP	max R	D	-	M	TW	N	H + PG (On.-Off.)
Li et al. (2019)	DRP	max R	D	-	M	-	N	DC + Q-learning (Off.)
Our research	SVRP	max SD	V	D	M	L	N	Q-learning + Con + OF (Off.)

Table 1: MDP-based solution methods proposed for routing problems

[1]: Heterogeneous vehicles
[2]: The decision in the MDP formulation is to either reject or choose a vehicle/drone to assign that request. Then, a heuristic decides how to accommodate that request in the route.

- Problem: Stochastic VRP (SVRP), VRP with Dynamic Requests (DVRP), Dispatching and Re-positioning Problem (DRP), Dial-A-Ride Problem (DARP), Stochastic Inventory Routing Problem (SIRP), Same-Day Delivery (SDD), Pick-up and Delivery VRP (PDVRP)
- Obj. Func.: The Objective Function can be min of Travel Time (min TT), Number of Delayed Services (min No. DS), Costs (min C), Number of Rejected Requests (min No. RR), Number of Service Failures (min No. SF), and Idle Time (min IT), or max of Served Demand (max SD), Profit (max P), Revenue (max R), and Number of Served Requests (max No. SR)
- Uncert.:
  - Cust.: Customers can be Static and Fixed (F), Static and Variable (V), or Dynamic (D)
  - Other: Other sources of uncertainties can be Demand (D) or Travel Time (T)
- Fleet: Single vehicle (S), Multiple vehicles (M), and Unlimited vehicles (U)
- TL: Time limits including, a duration limit for the operation (L), or Time Windows for customers (TW)
- RD: The routing decision is Where to go next (N) or Route-based (RB)
- Solution method (comp.): Rollout Algorithm (RA), Approximate Policy Iteration (API), Simulated Annealing (SA), Value Function Approximation (VFA), Heuristics (H), Policy Gradient (PG), Attention Mechanism (AM), Decompose the problem into multiple single-vehicle problems (DP), Decentralize the action space (DC), Consecutive action selection (Con), Assignment Problem (AP), Use observation function (OF). comp.: indicates how/when the majority of computation is done, On-line (On) or Off-line (Off).

MDP formulations are generally solved by Dynamic Programming algorithms. However, these algorithms suffer from the so-called curse of dimensionality. In multi-vehicle SVRPs, typically featuring very large combinatorial state and decision spaces, this curse is further aggravated. Several approaches have been proposed to cope with these difficulties. Fan et al. (2006) and Goodson et al. (2013) decompose the problem into multiple single-vehicle problems. Fan et al. (2006) use a myopic heuristic technique to partition customers into m subsets, one for each available vehicle. Once the subsets are established, they are kept fixed during the execution of the algorithm. Goodson et al. (2013) refine this technique by dynamically re-partitioning customers at each decision epoch. While both approaches drastically reduce the action and state spaces, collaboration opportunities among vehicles are not explicitly enforced.

An alternative approach to reduce the curse of dimensionality is to decentralize the decision space. The term decentralize entails that the value function is computed for separated vehicle-specific action spaces rather than for the joint action space (OroojlooyJadid and Hajinezhad 2019). The simplest way to handle decentralization is to optimize the individual reward of each vehicle, regardless of the impact on the per-

formance of other vehicles. However, this is equivalent to ignoring cooperation among vehicles, which may lead vehicles to compete for the reward. For example, Oda and Joe-Wong (2018) study a dynamic fleet management problem for taxi dispatching and re-positioning where assignments of taxis to customers are determined to maximize the number of requests a given taxi can individually serve.

The task of achieving some level of collaboration in a decentralized framework is very challenging and it has been extensively studied in the literature of multi-agent systems (OroojlooyJadid and Hajinezhad 2019). One possible strategy is to solve, at each decision epoch, an auxiliary master problem accessing information from the value functions of each individual vehicle. For example, in the context of a bike-sharing system, Brinkmann et al. (2019b) first compute the number of expected unserved requests for each individual vehicle if assigned to each of the bike stations and then deploy a greedy algorithm to solve an assignment problem minimizing the total expected number of unserved requests. For a VRP with dynamic customers, where requests are dynamically assigned to vehicles, Joe and Lau (2020) propose an MARL method that develops an approximated value function to estimate the cost of each individual vehicle route and then adopt a Simulated Annealing algorithm to search for an assignment minimizing the expected total cost. An alternative strategy is to shape the individual rewards to account for the influence of the vehicle's actions on the overall system performance (Li et al. 2019, Chen et al. 2019, Kullman et al. 2021). This strategy implicitly favors vehicle coordination. For example, in an order dispatching problem, Li et al. (2019) model the reward function as a weighted sum of the prize collected by the vehicle, while accounting for potential future rewards from other vehicles.

In our work we consider an alternative strategy. In particular, similar to Maxwell et al. (2010) who investigated an ambulance relocation problem, we enforce the decision process to be consecutive, i.e., we determine the action for one vehicle at a time. This is consistent with the observation that several vehicles are unlikely to arrive at their destinations (and need to decide) at precisely the same moment. Although considering the consecutive action selection significantly reduces the action space, the state space remains challenging. Various techniques have been introduced to handle this issue. For example, Maxwell et al. (2010), Ulmer et al. (2018), Joe and Lau (2020) replace the actual state with a set of basis functions. Alternatively, Chen et al. (2019), Kullman et al. (2021) aggregate customers' information by a grid-like discretizion of the service region. Other works combine grid-like discretization with the assumption that vehicles cannot fully observe the current state. For example, in a ride-sharing application, Li et al. (2019) assume that vehicles can only access trip requests information located within a given range from their current position. The aggregation technique we propose in this paper is a combination of the previous ones. We aggregate customers' information using a grid-like discretization, we represent a subset of customers by basis functions, and we use a vehicle-specific *observation* function to return a simplified representation of the state. To the best of our knowledge, we are the first to integrate the concept of the observation function in an SVRP. We note that recently, new techniques involving the adoption of Deep Neural Networks have been proposed to aggregate the state (see Li et al. 2021a, Lin et al. 2021, for example) or extracting features from the system state as an image (Kullman et al. 2019). Although these techniques seem promising, their implementation is computationally very expensive and usually requires complex tuning procedures.

Two main methodologies have been adopted to solve SVRPs expressed by MDP formulations. Approximate Policy Iteration (API) methods, such as Policy Gradient, aim at directly developing the desired policy by iteratively improving an initial policy (e.g., Li et al. (2021a)). We observe that the on-line Rollout Algorithm, such as the one developed in Goodson et al. (2013), can be viewed as a single iteration of the Policy Iteration algorithm (Bertsekas 2013). On the other hand, Approximate Value Iteration (AVI) methods, such as Q-learning, compute a value function, from which it is straightforward to derive the corresponding policy (e.g., Li et al. (2021b), Chen et al. (2022)). We choose to implement a Q-learning algorithm.

To conclude, MDP-based methods are promising for SVRPs. To the best of our knowledge, no existing MDP-based method in the literature can be trivially adapted to the VRP-VCSD. Our work fills this gap by proposing an MDP-based method that incorporates two approaches to address the curses of dimensionality associated with the MDP formulation of our problem. First, we propose a consecutive decision-making procedure for the VRP-VCSD. This reduces the action space from the multiple vehicle action space to a single vehicle action space. Contrary to the literature that decomposes the problem into multiple single-vehicle problems, our consecutive decision-making procedure does not impose a rigid pre-assignment of customers to

vehicles. Therefore, our methodology maintains a higher level of collaboration between vehicles. Secondly, we reduce the state space by proposing a tailored observation function for our problem. Specifically, this addresses the challenge entailed by the variable number of customers. Finally, the combination of consecutive decision-making procedure and the observation function yields an MDP formulation that can be effectively handled by using a Q-learning algorithm.

3. The VRP-VCSD

In Section 3.1, we formally define the VRP-VCSD. In Section 3.2, we introduce the MDP formulation of the VRP-VCSD.

3.1. Problem definition

We consider a service area, where customers are located along with a depot denoted by l_0. A set $\mathcal{V} = \{1, \ldots, m\}$ of m homogeneous vehicles with capacity Q is initially placed at the depot. Customers may request service from any point in the service area. The set of customers requesting service $\mathcal{C}_0$ is revealed at the beginning of the operations period (e.g., every morning), with $n = |\mathcal{C}_0|$. Each customer $c \in \mathcal{C}_0$ is characterized by a location l_c, and a probabilistic demand whose expected value $\bar{d}_c$ is assumed to be known. Accordingly, each customer is defined by a tuple of $(l_c, \bar{d}_c)$. The customer's actual demand $\hat{d}_c$ is only realized upon the first visit. We assume that the customer's demand is splitable in the sense that if a vehicle is unable to fully serve a visited customer, it will serve the customer as much as possible. The rest of the demand may be later served by the same vehicle or another one. The time taken to travel between locations i and j is assumed to be deterministic and is denoted by $\tau_{i,j}$, for $i, j \in \{l_c | c \in \mathcal{C}_0\} \cup l_0$. Vehicles begin their trips at the depot, serve a certain subset of customers, and end their routes at the depot. Once a vehicle is assigned to a location to visit, it will not be diverted to another location. Furthermore, vehicles must complete their operations before a given duration limit L. Vehicles may return to the depot during their trip for restocking operations after leaving a customer and before visiting the next one, even if the vehicle capacity has not been filled yet. Although our problem is to pick up items, we use the term *restocking* to refer to such a policy as is typical in the literature (Louveaux and Salazar-González 2018). However, in our case, restocking entails emptying the vehicle capacity. We note that due the restrictions on the number of vehicles, their capacity, and the duration limit, it might not be possible to serve all demands during a working day. The objective is to maximize the expected total amount of served demand.

3.2. The MDP formulation

We now provide an MDP formulation of the VRP-VCSD. An MDP formulation generally consists of a *state* which, represents the current status of the system, a control *action*, and a function describing the transition of the system to a new state once an action is taken and exogenous information is revealed. In our MDP formulation, the *decision epoch* is any point in time at which either a vehicle visits its destination, or new information is revealed. This includes i) the beginning of the horizon, when the set of customers is revealed, and ii) during operations, when at least one vehicle arrives at a customer, or when a vehicle arrives at the depot. At each decision epoch k, the status of the system is represented by a state s_k as follows:

$$s_k = ([(l_c, h_c, \bar{d}_c, \hat{d}_c)]_{c \in \mathcal{C}_0}, [(l_v, a_v, q_v)]_{v \in \mathcal{V}}, t_k), \tag{1}$$

where the first two groups of components refer to the state of customers and vehicles, respectively, and the last component indicates the time. Note that although the customer and vehicle state components are naturally indexed by the decision epoch k, we have simplified the notation by omitting the subscript k. The state of each customer $c \in \mathcal{C}_0$ is represented by a tuple of $(l_c, h_c, \bar{d}_c, \hat{d}_c)$. The customer's availability h_c takes value 1 if the customer is available to be served and 0 once it is assigned to a vehicle or it has been completely served. If a customer c was assigned to a vehicle but could not be served completely, h_c is set to 1. For each vehicle $v \in \mathcal{V}$, the tuple of (l_v, a_v, q_v) forms its state, where l_v, a_v, and q_v are the current destination (i.e., the next location to visit), arrival time at destination, and the current available capacity, respectively. The time of the system at decision epoch k is denoted by t_k.

The action set is represented by an m-dimensional vector of $x_k = (x_k^1, ..., x_k^m)$, where x_k^v indicates the next location vehicle v travels to at decision epoch k. Let $\bar{\mathcal{V}}_k = \{v \in \mathcal{V} | a_v = t_k\}$ be the set of vehicles that arrive at their destinations and are available to take action at decision epoch k. We call $\bar{\mathcal{V}}_k$ the set of active vehicles. At decision epoch k, the action x_k belongs to the action space $A(s_k)$ defined as:

$$A(s_k) = \{x_k \in \{\mathcal{C}_0 \cup \{l_0\}\}^m : \tag{2}$$

$$x_k^v = l_v, \ \forall\, v \in \mathcal{V} \setminus \bar{\mathcal{V}}_k, \tag{3}$$

$$x_k^v = l_0, \ \forall\, \{v \in \bar{\mathcal{V}}_k | q_v = 0\}, \tag{4}$$

$$x_k^v \neq x_k^{v'}, \ \forall\, \{v, v' \in \mathcal{V} | v \neq v', x_k^{v'} \neq l_0\}, \tag{5}$$

$$x_k^v \in J(s_k, v) \cup \{l_0\}, \ \forall\, v \in \bar{\mathcal{V}}_k\}, \tag{6}$$

where $J(s_k, v)$ denotes the set of reachable customers by vehicle v that have not been completely served. Thus,

$$J(s_k, v) = \{c \in \mathcal{C}_0 | h_c = 1, \tau_{l_v, l_c} + \tau_{l_c, l_0} \leq L - t_k\}\,. \tag{7}$$

In the defined action space $A(s_k)$, Condition (3) enforces the fact that vehicles cannot be relocated when assigned to a destination. Condition (4) implies that vehicles with no available capacity should return to the depot to replenish. Condition (5) ensures that no two vehicles can choose the same destination unless the destination is the depot, while Condition (6) ensures that vehicles travel to reachable customers in $J(s_k, v)$ or to the depot.

The information about customers' presence and demand volumes is revealed in two phases. At the first decision epoch ($k = 0$), no customer is realized yet. Thus, $s_0 = ([], [(l_0, 0, Q)]_{v \in \mathcal{V}}, t_0)$. At this point, a dummy action $x_0 = (l_0)_{v \in \mathcal{V}}$ transfers the system to the next decision epoch ($k = 1$) where the system time and the vehicle states remain the same; however, the set of realized customer locations and expected demands populates the set of customers $\mathcal{C}_0$. Consequently, we have: $s_1 = ([(l_c, 1, \bar{d}_c, -1)]_{c \in \mathcal{C}_0}, [(l_0, 0, Q)]_{v \in \mathcal{V}}, t_1)$. For each realized customer $c \in \mathcal{C}_0$, we set $\hat{d}_c$ to -1 at the beginning and will update it upon its first visit. Also, h_c is initially set to one for all customers.

It is worth noting that the way the state s_0 is initialized highlights one of the main features of the VRP-VCSD. In this MDP formulation, we develop a policy accounting for all possible customer realizations, which can tackle realizations that have never been seen before. Accordingly, the system starts with no customers, and a random set of customers will be revealed in decision epoch 1. For $k \geq 1$, $\mathcal{C}_0$ is assumed to be fixed. Let w be a sample scenario of customer demands and $w_k \subset w$ be the set of realized demand volumes of those customers whose actual demands are observed at decision epoch k. In decision epoch $k \geq 1$, the vehicles in the active set $\bar{\mathcal{V}}_k$ take actions from the set $A(s_k)$ of possible actions in the state s_k, and the system transits to the next state $s_{k+1} = S^M(s_k, x_k, w_{k+1})$, where $S^M(.)$ is a two-step transition function. In the first step, the post-decision state s_k^x is the state that comes immediately after taking action x_k, but before revealing the exogenous information w_{k+1}, if it exists:

$$h_{x_k^v} = 0, \ \forall\, \{v \in \bar{\mathcal{V}}_k | x_k^v \neq l_0\}, \tag{8}$$

$$a_v = t_k + \tau_{l_v, l_{x_k^v}}, \ \forall\, v \in \bar{\mathcal{V}}_k, \tag{9}$$

$$l_v = l_{x_k^v}, \ \forall\, v \in \bar{\mathcal{V}}_k, \tag{10}$$

$$t_{k+1} = \min_{v \in \mathcal{V}} a_v. \tag{11}$$

Equation (8) flags the chosen customers as unavailable. The arrival time as well as the location of active vehicles are updated using Equations (9) and (10). Finally, Equation (11) establishes the time of the next decision epoch, where the exogenous information w_{k+1} will be revealed. All the other components are left unmodified. The second step of the transition function happens at the beginning of a new decision epoch $(k + 1)$ by transiting from the post-decision state s_k^x to the next state s_{k+1}. Let $\tilde{\mathcal{C}}_{k+1} = \{c \in \mathcal{C}_0 | \exists v \in$

$\bar{\mathcal{V}}_{k+1} \wedge l_c = l_v\}$ be the set of customers that are served at decision epoch $k+1$. If the actual demand of a customer in $\tilde{\mathcal{C}}_{k+1}$ is not yet realized ($\hat{d}_c = -1$), its value will be set to the observed demand w^c_{k+1}.

$$\hat{d}_c = w^c_{k+1}, \ \ \forall \{c \in \tilde{\mathcal{C}}_{k+1} | \hat{d}_c = -1\}. \tag{12}$$

Let η^{k+1}_v be the demand volume that vehicle $v \in \bar{\mathcal{V}}_{k+1}$ serves at decision epoch $k+1$, defined as:

$$\eta^{k+1}_v = \min\{\hat{d}_{c_v}, q_v\}, \ \ \forall \{v \in \bar{\mathcal{V}}_{k+1} | l_v \neq l_0\}, \tag{13}$$

where c_v refers to a customer $c \in \tilde{\mathcal{C}}_{k+1}$ that is served by vehicle v (i.e., $l_c = l_v$). The unserved demands of customers in $\tilde{\mathcal{C}}_{k+1}$ and the available capacity of vehicles in $\bar{\mathcal{V}}_{k+1}$ are then updated as follows:

$$\hat{d}_{c_v} = \hat{d}_{c_v} - \eta^{k+1}_v, \ \ \forall \{v \in \bar{\mathcal{V}}_{k+1} | l_v \neq l_0\}, \tag{14}$$

$$q_v = \begin{cases} q_v - \eta^{k+1}_v & \text{if } l_v \neq l_0, \\ Q & \text{otherwise.} \end{cases} \quad \forall\, v \in \bar{\mathcal{V}}_{k+1}. \tag{15}$$

Customers in $\tilde{\mathcal{C}}_{k+1}$ whose demands are not fully served will continue being available for service. Thus,

$$h_c = 1, \ \ \forall \{c \in \tilde{\mathcal{C}}_{k+1} | \hat{d}_c > 0\}. \tag{16}$$

These two steps are illustrated in Figure 1, which displays an example with four customers and two vehicles.

We define the terminal decision epoch K as the point in time at which all vehicles are located at the depot, and vehicles have no other choice than to stay at the depot due to the duration limit L (i.e., $A(s_K) = \{(l_0)_{v \in \mathcal{V}}\}$). The reward function $R(s_k, x_k, w)$ is defined as the demands served by taking action x_k in state s_k, and the exogenous information w is observed:

$$R(s_k, x_k, w) = \sum_{\{v \in \bar{\mathcal{V}}_k | x^v_k \neq l_0\}} \min\{q_v, d^w_{x^v_k}\}, \tag{17}$$

$$d^w_c = \begin{cases} w^c_{\bar{k}} & \text{if } \hat{d}_c = -1, \\ \hat{d}_c & \text{otherwise.} \end{cases} \quad \forall\, c \in \mathcal{C}_0, \tag{18}$$

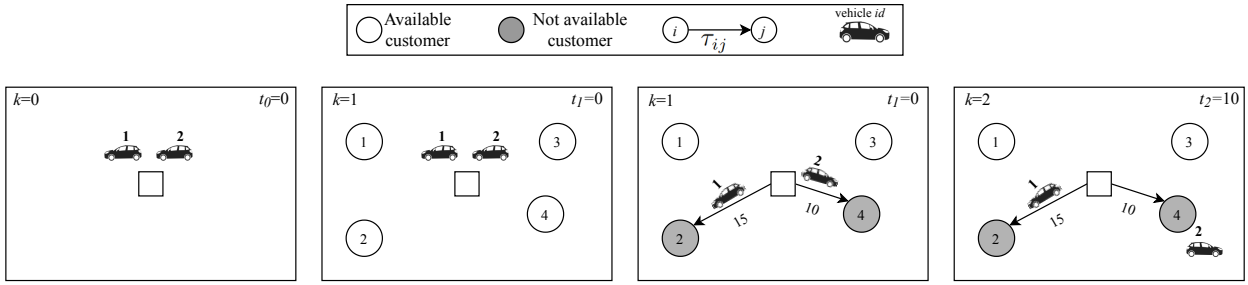
where $w^c_{\bar{k}}$ is the demand of customer c observed when the vehicle v visits the customer c at decision epoch $\bar{k}$. The decision epoch $\bar{k}$ corresponds to the point in time when the vehicle $v \in \bar{\mathcal{V}}_k$, which has selected customer x^v_k at the decision epoch k, arrives at its destination: $\bar{k} = \{k' \in [0, ..., K] | t_{k'} = t_k + \tau_{l_v, l_{x^v_k}}\}$. Thus, the collection of the reward is generally delayed.

The value of being in state s_k is defined as $V^\pi(s_k)$. We aim at finding a policy π that indicates the next location to visit for each vehicle in the active set $\bar{\mathcal{V}}_k$ in the feasible action space $A(s_k)$. The resulting policy selects actions such that the total served demand by all vehicles is maximized. The optimal policy π^* is obtained by solving the following equations:

$$V^\pi(s_k) = \max_{x_k \in A(s_k)} \mathbb{E}_w[R(s_k, x_k, w) + \gamma V^\pi(s_{k+1})], \tag{19}$$

$$\pi^*(s_k) = \arg \max_{x_k \in A(s_k)} \mathbb{E}_w[R(s_k, x_k, w) + \gamma V^\pi(s_{k+1})], \tag{20}$$

where, similar to Kullman et al. (2021), we consider the discount factor γ attributing greater importance to immediate rewards. Considering $\gamma < 1$ is necessary for ensuring the convergence of the value function for infinite-horizon MDPs (Puterman 2005). In the case of finite-horizon MDPs, such as in the VRP-VCSD, it is not necessary to set $\gamma < 1$, but doing so can significantly enhance learning performance (Powell 2011). In the terminal state s_K, there is no customer available to be served. Hence, $V^\pi(s_K) = 0$.



(a) No customer is realized.

(b) Customers are realized.

(c) Vehicles in $\bar{\mathcal{V}}_1 = \{1, 2\}$ choose action, $x_1 = (2, 4)$. The state s_1 transits to the post-decision state s_1^x.

(d) Vehicle 2 arrives at customer 4 and observes its actual demand w_2^4. The state s_1^x transits to s_2.

Figure 1: The transition function in the MDP formulation

4. Solution Method

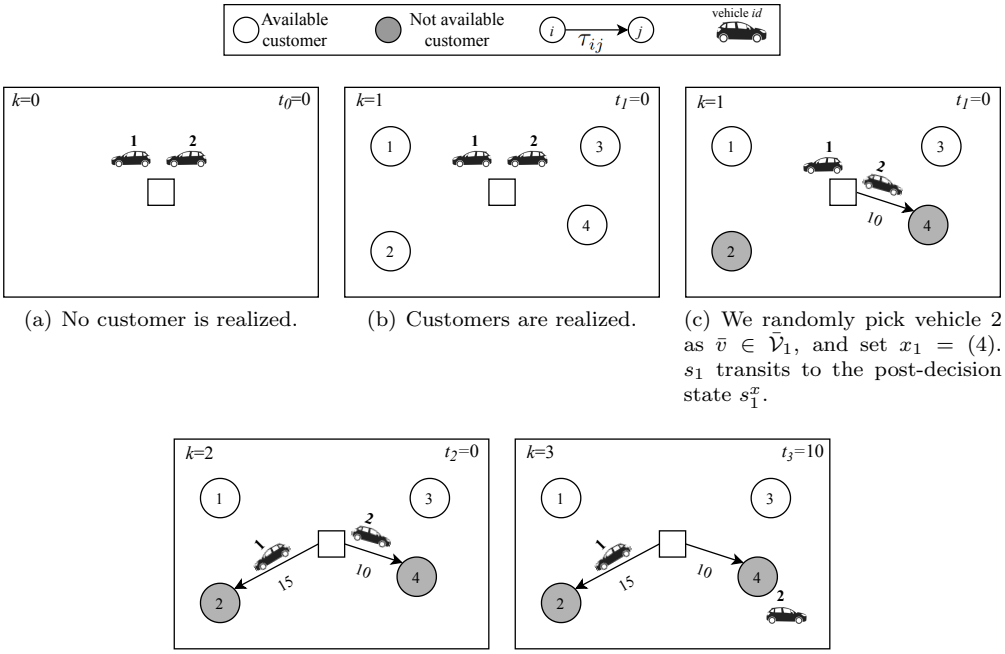
There are some challenges making the formulation presented in the previous section intractable for the VRP-VCSD. First, the dimension of the state representation is dependent on the cardinality of customers, which is stochastic. Consequently, the size of the vector s_k would be variable and dependent on the number of realized customers, preventing the use of many solution methods that assume a fixed-size state. For example, no linear regression method can be implemented to approximate the value function in Equation (19), when the size of the input (state vector) is variable. Furthermore, given the potentially large number of customers and the fact that the state of customers is represented by continuous variables, the state space is tremendously large. Moreover, managing multiple vehicles makes the size of the action space, defined in Equations (2)–(6), excessively large and handling such a large action set is computationally prohibitive. In this section, we describe two reformulations to cope with these challenges, as well as the proposed solution algorithm. The first reformulation is an MDP with a consecutive action selection procedure. It is denoted by MDP-C and presented in Section 4.1. Section 4.2 introduces the second reformulation, which leverages the structure of MDP-C and incorporates the concept of the observation function. This second reformulation is denoted by MDP-CO.

4.1. The MDP formulation with a consecutive action selection procedure

As previously mentioned, the action space $A(s_k)$ becomes excessively large when several active vehicles are considered at each decision epoch. To cope with this difficulty, we exploit a specific characteristic of the VRP-VCSD. The cardinality of $\bar{\mathcal{V}}_0$ at decision epoch $k = 0$ equals to m. However, for all subsequent epochs it is unlikely that several vehicles simultaneously arrive at their respective destinations.

In this section, similar to Maxwell et al. (2010), who investigated an ambulance relocation problem, we show how to reformulate the VRP-VCSD as an MDP with a consecutive action selection procedure. In this framework, only one vehicle decides on its next location at each decision epoch. In particular, when several vehicles require to be assigned to the next visit, as the fleet is homogeneous, a (random) order is imposed among them, therefore allocating each vehicle to a different decision epoch. For example, for the case of two vehicles ready to decide at decision epoch k, we allocate one vehicle to the current decision epoch k and the other to the decision epoch $k + 1$, where $t_{k+1} = t_k$.

Formally, let $\{v_{k_1}, \ldots, v_{k_y}\}$ be the set of vehicles that must be assigned to the next location at decision epoch k. We expand the count of decision epochs from k to $k + y - 1$ to accommodate the actions of the y vehicles. In particular, we randomly select the index $i \in \{1, \ldots, y\}$ and choose the vehicle $\bar{v} = v_{k_i}$ that will be active at the decision epoch k. We then form the corresponding singleton active set $\bar{\mathcal{V}}_k$, such that $\bar{v} \in \bar{\mathcal{V}}_k$ (i.e., $|\bar{\mathcal{V}}_k| = 1$). The procedure is then iterated for all remaining vehicles, until the generation of the last active set $\bar{\mathcal{V}}_{k+y-1}$.



(a) No customer is realized.

(b) Customers are realized.

(c) We randomly pick vehicle 2 as $\bar{v} \in \bar{\mathcal{V}}_1$, and set $x_1 = (4)$. s_1 transits to the post-decision state s_1^x.

(d) The state s_1^x transits to s_2. We set vehicle 1 as $\bar{v} \in \bar{\mathcal{V}}_2$. It takes action $x_2 = (2)$, and s_2 transits to s_2^x.

(e) Vehicle 2 arrives at customer 4 and observes its actual demand w_3^4. The state s_2^x transits to s_3.

Figure 2: The transition function in MDP-C formulation

With this formalism, the action space $A(s_k)$, can be explicitly expressed in terms of the active vehicle $\bar{v}$ as:

$$A(s_k, \bar{v}) = \{x_k \in \{J(s_k, \bar{v}) \cup \{l_0\}\} : \tag{21}$$

$$x_k = l_0 \quad \text{if } q_{\bar{v}} = 0, \tag{22}$$

$$x_k \neq l_0 \quad \text{if } J(s_k, \bar{v}) \neq \emptyset, l_{\bar{v}} = l_0\}, \tag{23}$$

where Condition (21) obliges x_k to be a customer from the set of feasible customers $J(s_k, \bar{v})$ or the depot, Condition (22) forces the active vehicle to travel to the depot when it has no remaining capacity, and Condition (23) does not allow the vehicle to stay at the depot if there are reachable customers to be served. We note that x_k has now dimension 1. By taking the action x_k from the set $A(s_k, \bar{v})$ of possible actions in the state s_k for $\bar{v}$, the system transits to the next state $s_{k+1} = S^M(s_k, x_k, w_{k+1})$, where $S^M(.)$ is a two-step transition function. The proposed formalism allows Equations (8–18) to be valid also for the MDP-C. Figure 2 illustrates the dynamics of the MDP-C for an example with four customers and two vehicles.

As for the MDP formulation, the value of being in state s_k is defined as $V^\pi(s_k)$. In the MDP-C, we aim at finding a policy π that indicates the next location to visit for the active vehicle $\bar{v}$ in the feasible action space $A(s_k, \bar{v})$. The resulting policy selects actions such that the total served demand by all vehicles is maximized. The optimal policy π^* is obtained by solving the following equations:

$$V^\pi(s_k) = \max_{x_k \in A(s_k, \bar{v})} \mathbb{E}_w[R(s_k, x_k, w) + \gamma V^\pi(s_{k+1})], \tag{24}$$

$$\pi^*(s_k) = \arg \max_{x_k \in A(s_k, \bar{v})} \mathbb{E}_w[R(s_k, x_k, w) + \gamma V^\pi(s_{k+1})]. \tag{25}$$

Compared to the MDP formulation, the MDP-C has two main advantages. First, since the decision is restricted to one vehicle only, the size of the action space is drastically reduced to $\underline{n}$, where $\underline{n}$ is the number

of available customers in the current decision epoch. Second, while preserving the global state of the system, MDP-C is such that it makes it possible to filter the information that is most relevant for the active vehicle. In Section 4.2, we take advantage of this feature and introduce the observation function, which substantially reduces the size of the state space and, as we will show, enables fixing the cardinality of the state vector.

4.2. The MDP formulation with a consecutive action selection procedure and an observation function

In this section, we introduce an observation function in the MDP-C. The primary purpose of the observation function is to address the challenges associated with the large state space by appropriately selecting the most relevant information for the active vehicle and by aggregating the remaining information. As we will show, the vehicle observation enables a significantly more compact state representation, when compared the one in Equation (1). Furthermore, unlike the representation in Equation (1), which depends on the variable number of customers, the cardinality of the vehicle observation can be fixed. To this purpose, we identify a subset of *target customers* $\mathcal{C}_{\bar{v}} \subset \mathcal{C}_0$ which we deem as more promising for the active vehicle. Therefore, instead of keeping detailed information on all customers, the observation function only maintains the comprehensive information of target customers and aggregates the rest. More formally, the observation function $O(s_k, \bar{v})$ takes as input the global state s_k and the current active vehicle $\bar{v}$, and returns $o_{k,\bar{v}}$, which is the state as observed by vehicle $\bar{v}$. The observation $o_{k,\bar{v}}$ is comprised of four parts:

$$o_{k,\bar{v}} = [F_k|H_k|G_k|t_k], \tag{26}$$

where F_k is the state of the target customers, H_k is an aggregated overview of all customers in the service area, G_k is the state of all vehicles, and t_k is the time at decision epoch k.

Considering our problem setting, we identify two desired characteristics for a target customer $c \in \mathcal{C}_{\bar{v}}$: requesting large demand volumes (d_c), and being close to the active vehicle ($\tau_{l_c,l_{\bar{v}}}$). Therefore, we define a new measure to identify $\tilde{n}$ target customers, where $\tilde{n}$ is an input. We choose $\tilde{n}$ customers with the highest $\rho_{c,\bar{v}} = \dfrac{min(\tilde{d}_c, q_{\bar{v}})}{\tau_{l_c,l_{\bar{v}}}}$ value, with $\tilde{d}_c = \bar{d}_c$ for $\hat{d}_c = -1$, and $\tilde{d}_c = \hat{d}_c$ otherwise. This measure normalizes the customers' servable demands by their distance to the active vehicle, and also accounts for the available capacity. The effectiveness of selecting target customers according to their score $\rho_{c,\bar{v}}$ was validated through preliminary computational tests.

In order to maintain information related to target customers, we represent each target customer c by using a set of features: $(l_c, \tau_{l_c,l_{\bar{v}}}, \tau_{l_c,l_0}, \tilde{d}_c, min(\tilde{d}_c, q_{\bar{v}}), \mu_c)$. In this feature set, l_c refers to the customer's location, $\tau_{l_c,l_{\bar{v}}}$ is the travel time between the customer and the active vehicle, τ_{l_c,l_0} is the travel time between the customer and the depot, $\tilde{d}_c$ is the customer's unserved demand (or expected demand, if not realized yet), $min(\tilde{d}_c, q_{\bar{v}})$ shows the portion of customer's demand that can be served by the active vehicle, and μ_c indicates whether the actual demand is realized or not. The vector of target customers is denoted as $F_k = [(l_c, \tau_{l_c,l_{\bar{v}}}, \tau_{l_c,l_0}, \tilde{d}_c, min(\tilde{d}_c, q_{\bar{v}}), \mu_c)]_{c\in\mathcal{C}_{\bar{v}}}$.

We propose an aggregated representation of all customers to provide the active vehicle with an overview of environment. The challenge here is that the stochastic customer set necessitates an aggregating technique that handles a variable number of inputs while delivering a fixed-size output. To this purpose, we propose a heatmap-style approach to encode all customers in the service area as a fixed-size vector H_k. In particular, we divide the service area into a set of square-shaped *partitions* P. Let $\mathcal{C}_p \subset \mathcal{C}_0$ be the subset of customers that are located in $p \in P$. We characterize each p by (ξ_p^c, ξ_p^d) representing the number of customers and the total expected demand in that partition, respectively. Thus, $\xi_p^c = |\mathcal{C}_p|, \quad \xi_p^d = \sum_{c\in\mathcal{C}_p} \tilde{d}_c$, and $H_k = [(\xi_p^c, \xi_p^d)]_{p\in P}$.

In addition to H_k and F_k, the state of the vehicles $G_k = [(l_v, a_v, q_v)]_{v\in\mathcal{V}}$ forms the third segment of the observation representation (Equation (26)). Therefore, the main difference between s_k and $o_{k,\bar{v}}$ is the customers' state, with a variable size, replaced by a fixed-size vector $[F_k, H_k]$. Compared to the state representation s_k, the observation representation $o_{k,\bar{v}}$ has two advantages: it overcomes the problem of the variable-sized state and reduces the dimension of the observation. In particular, the customers' state in s_k is of dimension $4 * |\mathcal{C}_0|$, while $[F_k, H_k]$ is of dimension $(6 * \tilde{n} + 2 * |P|)$. For example, assuming an average

number of customers of 50, the dimension of s_k is 200 on average. However, assuming $\tilde{n} = 10$ and $|P| = 25$, the dimension of $o_{k,\bar{v}}$ will be fixed at 110.

In addition to reducing and fixing the dimension of $o_{k,\bar{v}}$, using target customers reduces the dimension of the active vehicle's action space by limiting its actions to target customers. Therefore, in the definition of $J(s_k, \bar{v})$ for the MDP-CO, which is used to construct the action space of the active vehicle, we substitute $\mathcal{C}_0$ with $\mathcal{C}_{\bar{v}}$. As a result, the size of the action space is reduced to $|\mathcal{C}_{\bar{v}}| + 1$, where $|\mathcal{C}_{\bar{v}}| \leq \tilde{n}$.

By substituting the state s_k with the active vehicle's observation $o_{k,\bar{v}}$, we reformulate Equations (24) and (25) as follows:

$$V^{\pi}(o_{k,\bar{v}}) = \max_{x_k \in A(o_{k,\bar{v}},\bar{v})} \mathbb{E}_w[R(o_{k,\bar{v}}, x_k, w) + \gamma V^{\pi}(o_{k+1,\bar{v}'})], \tag{27}$$

$$\pi^*(o_{k,\bar{v}}) = \arg \max_{x_k \in A(o_{k,\bar{v}},\bar{v})} \mathbb{E}_w[R(o_{k,\bar{v}}, x_k, w) + \gamma V^{\pi}(o_{k+1,\bar{v}'})], \tag{28}$$

where $\bar{v}'$ refers to the active vehicle in decision epoch $k+1$, and $R(o_{k,\bar{v}}, x_k, w)$ and $A(o_{k,\bar{v}}, \bar{v})$ are defined as in Equations (17–18) and (21–23), respectively, by substituting $\mathcal{C}_0$ with $\mathcal{C}_{\bar{v}}$. We specify that the transition from the observed state $o_{k,\bar{v}}$ to $o_{k+1,\bar{v}'}$ follows the evolution of the global state from s_k to s_{k+1}.

4.3. Q-learning for MDP-CO

Although the MDP-CO formulation has considerably smaller state and action space than the MDP formulation, it still suffers from the so-called curse of dimensionality, which makes it difficult to solve by conventional stochastic dynamic programming. Several approximate methods have been proposed in the literature to overcome this challenge (see Powell 2011, for a comprehensive review). In this paper, we consider the Q-learning algorithm.

Q-learning is a model-free algorithm based on approximate value iteration in which an agent learns how to behave in order to maximize the collected reward through a sequence of interactions with the environment, resulting in a policy that satisfies Equation (20). Specifically, this method iteratively updates the value of making the action x_k in state s_k, known as the Q factor of that state-action pair ($Q(s_k, x_k)$). In fact, it is possible to rewrite the Bellman equation (defined in Equation (19)) in terms of Q factors as follows:

$$V(s_k) = \max_{x_k \in A(s_k)} Q(s_k, x_k), \tag{29}$$

where:

$$Q(s_k, x_k) = R(s_k, x_k, w) + \gamma \max_{x_{k+1} \in A(s_{k+1})} Q(s_{k+1}, x_{k+1}).$$

The reader is referred to Powell (2011) for a complete overview of the algorithm. Algorithm 1 illustrates the general Q-learning structure. This algorithm simulates the problem for several trials ($Trials\_MAX$) over different sample scenarios of stochastic components of the problem. The simulation follows the so-called ϵ-greedy policy, which enables random exploration of the action space. The level of exploration is typically larger in the initial phases of the algorithm when actions are mostly chosen randomly. As the algorithm proceeds, actions with higher estimated values are generally preferred. Q-learning exploits the interactions between the agent and the environment to update the Q values (the so-called *experiences*). An experience is defined as a tuple of (s_k, x_k, r_k, s_{k+1}), describing interaction of the agent with the environment consisting of the agent observation of the state s_k, its action x_k, the received reward r_k (in VRP-VCSD, $r_k = R(s_k, x_k, w)$) and the observation of the next state s_{k+1}. For each state-action pair experienced, we estimate their Q value as $\hat{Q}(s_k, x_k)$ using the following equation:

$$\hat{Q}(s_k, x_k) = r_k + \gamma \max_{x_{k+1} \in A(s_{k+1})} Q(s_{k+1}, x_{k+1}), \tag{30}$$

where γ is the discount factor. Then, we update the $Q(s_k, x_k)$ according to the following standard Q-learning updating equation (Powell 2011):

$$Q(s_k, x_k) \leftarrow Q(s_k, x_k) + \alpha[\hat{Q}(s_k, x_k) - Q(s_k, x_k)], \tag{31}$$

Algorithm 1: Generic Q-learning Algorithm

Initialize: set Q values to 0
while *trials* $<$ *Trials_MAX* **do**
  Choose a sample scenario w
  Observe state s_0
  $k \leftarrow 0$, *simulate* $\leftarrow$True
  while *simulate* **do**
    Take action: $x_k \leftarrow \begin{cases} \text{random action in } A(s_k) & \epsilon \\ \arg\max_{x' \in A(s_k)} Q(s_k, x') & 1-\epsilon \end{cases}$
    Transit to $s_{k+1} \leftarrow S^M(s_k, x_k, w_{k+1})$ and observe the reward r_k
    Estimate and update new Q values:
    - $\hat{Q}(s_k, x_k) \leftarrow \begin{cases} r_k & A(s_{k+1}) = \emptyset \\ r_k + \gamma \max_{x' \in A(s_{k+1})} Q(s_{k+1}, x') & \text{otherwise} \end{cases}$
    - $Q(s_k, x_k) \leftarrow Q(s_k, x_k) + \alpha[\hat{Q}(s_k, x_k) - Q(s_k, x_k)]$

    $k \leftarrow k+1$
    if $A(s_k) = \emptyset$ **then**
      simulate $\leftarrow$False
    end
  end
  Set $trials = trials + 1$
  Decay the exploration rate ϵ
end
return Q values

where α is the step size (learning rate).

Algorithm 1 returns a set of Q values that can be used as the decision policy. In particular, the resulting decision policy will be:

$$\pi^*(s_k) = \arg \max_{x_k \in A(s_k)} Q(s_k, x_k). \tag{32}$$

To implement the Q-learning on the MDP-CO formulation, we replace the reward r_k, the state s_k, and the action space $A(s_k)$ in Equation (30) by the active vehicle's reward function $R(o_{k,\bar{v}}, x_k, w)$, observation $o_{k,\bar{v}}$, and action space $A(o_{k,\bar{v}}, \bar{v})$, respectively, as follows:

$$Q(o_{k,\bar{v}}, x_k) = R(o_{k,\bar{v}}, x_k, w) + \gamma \max_{x_{k+1} \in A(o_{k+1,\bar{v}'}, \bar{v}')} Q(o_{k+1,\bar{v}'}, x_{k+1}). \tag{33}$$

We denote $\hat{Q}(o_{k,\bar{v}}, x_k)$ as the estimation of $Q(o_{k,\bar{v}}, x_k)$ and use it to update the Q factor of a given state-action pair. We do this by using the following equation, which is adapted from equation (31):

$$Q(o_{k,\bar{v}}, x_k) \leftarrow Q(o_{k,\bar{v}}, x_k) + \alpha * [\hat{Q}(o_{k,\bar{v}}, x_k) - Q(o_{k,\bar{v}}, x_k)]. \tag{34}$$

We implement the Q values in the form of an artificial neural network instead of traditional look-up tables. This method is called Q-network in the literature, and returns the approximate Q value for a given observation-action pair $(o_{k,\bar{v}}, x_k)$. For computational efficiency, it is desirable to design the Q-network in such a way that only one forward call is sufficient to return the Q values for all $(|\mathcal{C}_{\bar{v}}|+1)$ actions in $A(o_{k,\bar{v}}, \bar{v})$. To this purpose, we consider a neural network that returns $\tilde{n}+1$ Q values in every call. Notice that, under some circumstances, the number of target customers can be less than $\tilde{n}$, making the cardinality of $A(o_{k,\bar{v}}, \bar{v})$ variable, but restricted to $\tilde{n}+1$. Therefore, in the vector of $\tilde{n}+1$ Q values approximated by the Q-network, we associate the first $|\mathcal{C}_{\bar{v}}|$ values to the Q value of target customers $\mathcal{C}_{\bar{v}}$ and assign the last one to the depot. Q values not associated with target customers nor the depot will be set to 0. We note that the features in the observation function are dependent on the action set $A(o_{k,\bar{v}}, \bar{v})$. These are also known in the literature as action-dependent features (Powell and Ma 2011).

A key advantage of the proposed observation function is to enable the adoption of a relatively simple and general purpose Artificial Neural Network (ANN). Therefore, we adopt a simple neural network structure with two hidden layers (fully connected) and a ReLU activation function to showcase the effectiveness of our observation function and emphasize the importance of engineered features in the vehicle observation function. This network gets the observation representation ($o_{k,\bar{v}}$) as the input layer and returns approximate Q values in the output layer. We denote the size of the input and output layers by h^{in} and h^{out}, where $h^{in} = 6\tilde{n} + 2|P| + 3m + 1$ and $h^{out} = \tilde{n} + 1$. We note that the structure of the adopted neural network and the size of the hidden layers are chosen based on preliminary experiments. In particular, we set the size of the hidden layers to $\lfloor \frac{2}{3}(h^{in} - h^{out}) \rfloor + h^{out}$ and $\lfloor \frac{1}{3}(h^{in} - h^{out}) \rfloor + h^{out}$, respectively. Figure 5 in illustrates the architecture of the proposed artificial neural network. Denoting by θ the trainable weights of the proposed neural network, the Q value of an observation-action pair is represented as $Q(o_{k,\bar{v}}, x_k, \theta)$. It is worth noting that any equation involving $Q(o_{k,\bar{v}}, x_k)$ will also apply to $Q(o_{k,\bar{v}}, x_k, \theta)$.

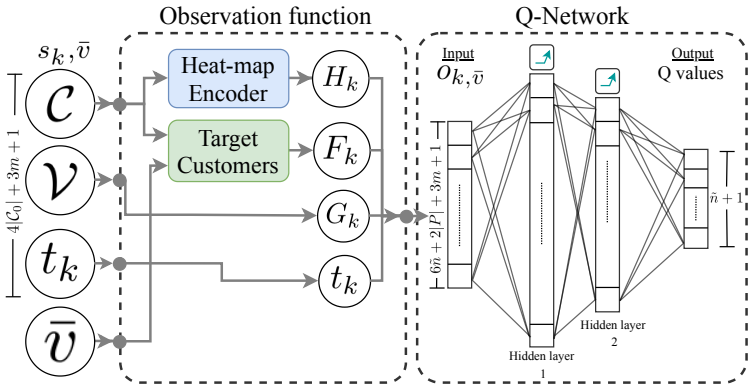


Figure 5: A schematic overview of QN-CO

To train the proposed Q-Network, we minimize the so-called loss function. We define this function based on the difference between the new estimation of the values $\hat{Q}(o_{k,\bar{v}}, x_k, \theta)$ and the current estimation $Q(o_{k,\bar{v}}, x_k, \theta)$, denoted as Δ (i.e., $\Delta = \hat{Q}(o_{k,\bar{v}}, x_k, \theta) - Q(o_{k,\bar{v}}, x_k, \theta)$). Among several loss functions used in the literature, we found that the Huber loss function performs better than the commonly used functions such as Mean Squared Error (MSE) and Mean Absolute Error (MAE). The Huber loss acts as an MSE for small Δ and as an MAE for larger Δ. The Huber loss is defined as:

$$Huber(\Delta) = \begin{cases} \frac{1}{2}\Delta^2 & \text{for } |\Delta| \leq \delta \\ \delta(|\Delta| - \frac{1}{2}\delta) & \text{otherwise,} \end{cases} \tag{35}$$

where δ is a control parameter. We obtained the best results by setting $\delta = 5$ and used the Adam optimizer (Kingma and Ba 2014) to update the network in order to minimize the loss function.

Algorithm 2 summarizes our QN-CO algorithm. Similar to what was suggested by Mnih et al. (2015), our preliminary results showed that using a target network (Van Hasselt et al. 2016) and a replay memory accelerates convergence. By adopting the target network (parameterized by $\bar{\theta}$ in Algorithm 2), the model learns from a stable target. The target network $\bar{\theta}$ is periodically copied from the primary network θ every β_d trials. In the replay memory, experiences are stored in a First-In-First-Out list, denoted as B, with a fixed size. At every decision epoch, with a probability of β_κ, a random batch of experiences ($\tilde{B}$) is sampled from this list to update the weights (θ) in the neural network. Several advantages have been mentioned for using the replay memory (Mnih et al. 2015) such as breaking the correlation between consecutive experiences and giving more chances to those experiences that happen more. The exploration rate (ϵ) and the learning rate (α) are linearly decayed during the training phase. The decaying rate is decided based on the length of the training process.

Algorithm 2: The QN-CO

Initialize the Q-network and the target Q-network with random weights $\theta, \bar{\theta}$
$B \leftarrow [\emptyset]$
while *trials* $<$ *Trials_MAX* **do**
  Choose a sample scenario w and observe a new set of customers $\mathcal{C}_0$
  $t_0 \leftarrow 0$, $k \leftarrow 0$, $\bar{v}_{prev} \leftarrow \emptyset$, Buff $\leftarrow [\emptyset]_{v \in \mathcal{V}}$, and *simulate* $\leftarrow True$
  while *simulate* **do**
    $\bar{v} \leftarrow$ a random $v \in \bar{\mathcal{V}}_k$
    if $k > 0$ **then**
      Transit from s^x_{k-1} to s_k and observe the reward r
      Buff$[\bar{v}_{prev}] \leftarrow (o_{k-1,\bar{v}_{prev}}, \bar{v}_{prev}, x_{k-1}, o_{k,\bar{v}}, \bar{v})$
      if Buff $[\bar{v}] \neq \emptyset$ **then**
        $(o, v, x, o', v') \leftarrow$ Buff$[\bar{v}]$
        Append (o, v, x, r, o', v') to B
      end
    end
    Observe $o_{k,\bar{v}} = O(s_k, \bar{v})$
    Take action: $x_k \leftarrow \begin{cases} \text{random action in } A(o_{k,\bar{v}}, \bar{v}) & rand[0,1] < \epsilon \\ \arg\max_{x'} Q(o_{k,\bar{v}}, x', \theta) & \text{otherwise} \end{cases}$
    Transit from s_k to s^x_k
    if $rand[0,1] < \beta_\kappa$ **then**
      Estimate and update new Q values:
      - Sample a batch of experiences $\tilde{B}$ from B
      - Compute the estimation
$$\hat{Q}(o_{k,v}, x, \theta) \leftarrow \begin{cases} r & A(o,v) = \{l_0\} \wedge l_v = l_0 \\ r + \gamma \max_{x'} Q(o', x', \bar{\theta}) & \text{otherwise} \end{cases}, \ \forall (o, v, x, r, s', v') \in \tilde{B}$$
      - Compute the loss function: $\Phi \leftarrow \mathbb{E}_{\tilde{B}} \left[Huber \left(Q\left(o, x, \theta\right) - \hat{Q}\left(o, x, \theta\right) \right) \right]$
      - Update the network weights θ: $\theta \leftarrow \theta - \alpha \nabla_\theta \Phi$
    end
    $k \leftarrow k + 1$, and $\bar{v}_{prev} \leftarrow \bar{v}$
    if $A(o_{k,\bar{v}}, \bar{v}) = \emptyset \wedge l_v = l_0, \forall v \in \mathcal{V}$ **then**
      simulate $\leftarrow$ False
    end
  end
  trials $\leftarrow$ *trials* $+ 1$
  Every β_d *trials*, $\bar{\theta} \leftarrow \theta$
  Decay the learning rate α and the exploration rate ϵ
end
return Q values

5. Experimental results

In this section, we present our computational experiments. Since no instance set is available in the literature, we construct a new set of instances for the VRP-VCSD. We design three main experiments. The first one, presented in Section 5.1, is aimed at evaluating the overall performance of the QN-CO algorithm. To this purpose, we implement five benchmarks, which we call Deterministic A Priori Policy (APP), Q-Network for the MDP-C formulation (QN-C), Random Policy (RP), Greedy Policy (GP), and Hybrid-Greedy Policy (HP). Given that APP requires solving a very complex deterministic optimization problem, we only test it on relatively small instances, and perform experiments on larger instances for the four other benchmarks. The purpose of the second experiment, presented in Section 5.2, is to compare the QN-CO with the existing literature. The challenge is that no direct comparison is possible. The closest problem in the literature we are aware of, is the VRPSD for which Goodson et al. (2016) developed the current state-of-the-art method. As mentioned earlier, the VRPSD can be seen as a particular case of the VRP-VCSD where both customer locations and expected demands are known in advance. Even though our algorithm does not take advantage of the specific characteristics of the problem, it turns out that our algorithm is competitive and sometimes outperforms the results in Goodson et al. (2016). The third experiment, in Section 5.3, can be seen as an extension of the second one, where we investigate the possibility of modifying the training phase of the QN-CO to develop a policy able to address a larger range of VRPSD instances aggregated over specific parameters, such as the duration limit and stochastic variability. Finally, an additional experiment providing a simple demonstration of the fact that the policies provided by our algorithm can be easily embedded as base policies in rollout algorithms, is presented in Appendix C. We note that depending on the size of the problem, training a policy according to Algorithm 2 takes 24 to 36 hours. Problems with fewer vehicles and customers or shorter duration limits, for example, demand less training time. The values of the hyper-parameters in Algorithm 2 and the computational environment are described in Appendix A.2. In Section 5.4, we present managerial insights based on the on our computational experiments.

5.1. Performance of QN-CO in VRP-VCSD

We describe the main features of the instance set in Section 5.1.1, while Appendix A.1.1 provides the details of the instance generation method. The benchmarks policies APP, QN-C, RP, GP, and HP are described in Section 5.1.2. We report the computational results in Section 5.1.3.

5.1.1. VRP-VCSD Instances

As custom when building computational experiments in learning-based methods, we need to provide two types of input data, one to be used in the training phase of the algorithm, the other in the testing phase. Elements of these two sets are samples of the problem at hand. Although training and testing sets are distinct, they are typically generated by sampling from the same distributions.

We define an instance i as the tuple $i = (\mathcal{A}, \Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}, m, Q, L)$, where $\mathcal{A}$ denotes the service area partitioned into a set of zones Z, while $\Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}$ denote the probability distribution functions of the number of customers n_z per zone $z \in Z$ (customer density), customer locations, expected demands, and actual demands, respectively. Furthermore, m, Q, L indicate the number of vehicles, their capacity, and the duration limit, respectively. We detail the instance generation procedure in Appendix A.1.1. In particular, we consider five different distributions for the customer density Ψ_{n_z} (Very Low, Low, Moderate, High, and Very High), and three different values of the vehicle capacity Q (25, 50, and 75), resulting in a set I of 15 instances. Furthermore, the distribution Ψ_{n_z} determines the values of m and L (Appendix A.1.1). For a given problem instance, we denote the expected total number of customers by $\bar{n}$. Table 2 reports the resulting values for $\bar{n}, m, L$ for each Ψ_{n_z}. We note that test instances and a code to generate training instances are available at https://github.com/moda707/vrpvcsd-instances. Given an instance i, a realization $\hat{i}$ (also referred to as a sample, or a scenario) is obtained by sampling from distributions $\Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}$. The set of all realizations $\hat{i}$ of i is denoted $\hat{I}(i)$.

Ψ_{n_z}	$\bar{n}$	m	L
Very Low (VL)	10	2	143.71
Low (L)	15	2	201.38
Moderate (M)	23	3	221.47
High (H)	53	7	195.54
Very High (VH)	83	11	187.29

Table 2: Values of $\bar{n}, m$, and L for each density level Ψ_{n_z}

5.1.2. Benchmarks

We now describe the benchmark policies. The APP assumes the knowledge of the customer locations and expected demands. It then constructs a set of routes, which are evaluated in the VRP-VCSD setting. To construct such routes, we formulate the deterministic counterpart of the VRP-VCSD (i.e., with realized customers and demands) by the MILP described in Appendix A.3. We find an initial set of routes by solving this formulation for a given set of customers with demands corresponding to their expected values. We then evaluate the performance of these routes by simulating the classical recourse policy (i.e., once the vehicle capacity is exceeded a return trip to the depot is performed and the capacity of the vehicle is restored) for 100 demand realizations. The MILP was solved by CPLEX 20.1 with a time limit of two hours. Unlike the QN-CO, which can be instantaneously executed for any set of customers (generated from a given distribution function), this benchmark requires a significant amount of time to find the fixed routes for each customer realization.

The QN-C benchmark method refers to the Q-learning algorithm implemented on the MDP-C formulation. We recall that the MDP-C refers to the MDP formulation where the consecutive action selection strategy is imposed but does not adopt the proposed observation function. In the MDP-C, the cardinality of the state is variable and dependent on the number of realized customers. Since the Q-learning algorithm requires a fixed-size vector for the state, we assume that the number of customers in the QN-C does not exceed a predetermined input parameter $\hat{n}$, where $\hat{n} > \bar{n}$. Then, instead of representing the state s_k based on the variable-size customer set $\mathcal{C}_0$, as described in Equation (1), we define it as follows:

$$s_k = ([(l_c, h_c, \bar{d}_c, \hat{d}_c)]_{c \in \mathcal{C}'_0}, [(l_v, a_v, q_v)]_{v \in \mathcal{V}}, t_k). \tag{36}$$

In this expression, $\mathcal{C}'_0 = \mathcal{C}_0 \cup \mathcal{C}_d$, where $\mathcal{C}_d$ is a set of $\hat{n} - |\mathcal{C}_0|$ dummy customers for which we set $l_c = 0,\ h_c = 0,\ \bar{d}_c = 0$, and $\hat{d}_c = 0$, for all $c \in \mathcal{C}_d$. To approximate Q factors in QN-C, we construct a Q-network by adopting the same procedure used for QN-CO (described in Section 4.3). In this case, however, the artificial neural network receives as input the global state, which is a vector of size $(4 * \hat{n} + 3 * m + 1)$ and returns a vector of size $\hat{n} + 1$, representing Q values for all customers and the depot. To ensure the feasibility of decisions, the Q value of customers $c \in \mathcal{C}_0 \setminus A(s_k, \bar{v})$ (i.e., infeasible customers) are masked to zero. We note that we keep all hyper-parameters and the training setup the same for both QN-C and QN-CO.

In the RP, an action is randomly picked from the action set. We compare with this policy since it is the policy that QN-CO starts with when the exploration rate in the ϵ-greedy method is set to one. Therefore, the improvement of our model over RP demonstrates the effect of learning.

The GP chooses the customer $c \in A(s_k, \bar{v})$ with the highest $\tilde{d}_c$ at each decision epoch. We recall that $\tilde{d}_c$ provides the expected or the remaining demand of customer c. If several customers have the same $\tilde{d}_c$, the closer one is selected. As GP selects customers according to the potential reward in terms of served demands, it is thus aligned with the VRP-VCSD objective.

Lastly, inspired by the procedure we proposed to select target customers in Section 4.2, HP chooses the customer $c \in A(s_k, \bar{v})$ with the highest $\rho_{c,\bar{v}}$ at each decision epoch, when the active vehicle is $\bar{v}$. This measure normalizes the demand of each customer by its distance from the active vehicle. Thus, a customer with a high demand that can be served with the shortest travel time will be prioritized.

RP, GP, HP, and QN are more comparable with our QN-CO. However, RP, GP, and HP cannot handle preemptive restocking. Therefore, the action set is redefined as:

$$A(s_k, \bar{v}) = \begin{cases} \{l_0\} & q_{\bar{v}} = 0 \ \ || \ \ J(s_k, \bar{v}) = \emptyset \\ J(s_k, \bar{v}) & \text{otherwise.} \end{cases} \tag{37}$$

5.1.3. Results and Discussion

As previously mentioned, the benchmark APP requires to solve a complex deterministic MILP for each customer realization. Given the two hours time limit we impose, the APP was only able to address instances with densities VL and L. Therefore, we organize our discussion into two parts. In the first, we present instances with densities VL and L, and compare QN-CO with all five benchmarks. In this case, we also restrict the vehicle capacity to 25 and 50, given that for $Q = 75$ almost all demands could be served. In the second part, we compare the performance of QN-CO with QN-C, RP, GP, and HP on instances with densities M, H, and VH.

The method to obtain the training and test sets is the same for each instance $i \in I$. In particular, the training set is obtained by sampling 3 million realizations $\hat{i}$ from $\hat{I}(i)$. We then trained the QN-CO and QN-C on these realizations. The exploration rate decayed linearly from 1.0 to 0.1 in the first one million trials. The learning rate also decreased linearly from 10^{-3} to 10^{-4} in the first two million trials. As detailed below, due to computational limitations, the cardinality of the test for the first part is smaller than the test set of the second part. Note that although the training and the test sets are sampled from the same distributions, they are disjoint.

For each instance with densities VL and L, the test set is obtained by generating ten customer realizations. For each customer realization, we then generate 100 demand realizations, resulting in 1000 scenarios. Table 3 shows the comparison based on average performance of QN-CO, APP, QN-C, RP, GP, and HP. Extended results are presented in Table B.22 in Appendix B. The first two columns in Table 3 indicate the characteristics of the instance, while columns n and $\mathbb{E}\sum d_c$ show the average number of customers and the average total expected demand. Column Opt. reports the number of customer realizations solved to optimality by the APP for each instance. The average total served demand by our proposed policy is reported in column QN-CO. Column $\%\mathbb{E}\sum d_c$ shows the portion of all demands served by QN-CO. Finally, the performance gap between QN-CO and the five benchmarks is reported in columns %APP, %QN-C, %RP, %GP, and %HP, where measure %X, is computed as: $\%\text{X} = \dfrac{\text{QN-CO} - X}{X} \times 100$.

The APP obtains optimal solutions in 17 out of 20 customer realizations on instances with the density VL. This number decreases to 4 out of 20 for instances with the density L. Therefore, the values in Table 3 refer to realizations for which APP obtained optimal routes. QN-CO outperforms APP by an average of 3.0%. This suggests that, when considering the densities VL and L, even by giving the APP sufficient time to solve the resulting deterministic problem, our method is still a better choice. Furthermore, QN-CO outperforms QN-C, RP, GP, and HP by, an average, 22.0%, 41.5%, 19.9%, and 18.4%. We note that such improvements increase for %QN-C and %RP, relatively remain stable for %GP, and decrease for %HP with the density. This can be explained by the fact that, as the number of customers increases, the size of the state and action set in QN-C becomes larger, resulting in a more complex problem. Also, the increase in the number of customers augments the probability that RP deviates from the optimal solution. A denser instance, in contrast, is more favorable for GP and HP. Intuitively, as instances become denser, the knapsack component of the problem becomes dominant, and greedy policies are known to behave well on these kind of problems.

According to Table 3, as the vehicle capacity becomes larger (from 25 to 50 in the density VL), the improvement over APP decreases (from 2.6% to 1.2%). To explain this finding, we remark that the vehicle capacity directly impacts the probability of route failure. When the capacity is larger, the routes are less vulnerable to failure caused by stochastic demands. Therefore, we argue that the advantage of our method over APP is more pronounced on instances with smaller capacity because our method can better handle route failures. Validating this finding for instances with customer density L is not easy because only a few could be solved optimally.

We now discuss the performance of QN-CO on instances with densities M, H and VH. The test set is obtained by sampling 500 customer realizations and 500 demand realizations for each instance $i \in I$ with densities M, H and VH, resulting in 250,000 scenarios. The aggregated results are reported in Table 4, where the first four columns indicate the characteristics of the instance. The expected total demand for each instance is reported in column $\mathbb{E}\sum d_c$. The results of QN-CO, along with the results of the four benchmarks, are reported in columns QN-CO, QN-C, RP, GP, and HP, respectively. QN-CO outperforms QN-C by an

Ψ_{n_z}	Q	n	Opt.	$\mathbb{E}\sum d_c$	QN-CO	$\%\mathbb{E}\sum d_c$	%APP	%QN-C	%RP	%GP	%HP
VL	25	10.4	17/20	105.0	68.9	66.3%	2.6%	22.7%	33.8%	26.0%	26.3%
	50				87.3	83.5%	1.2%	19.7%	44.3%	15.7%	14.4%
						Avg	**1.8%**	**20.9%**	**40.0%**	**19.9%**	**19.3%**
L	25	15.0	4/20	147.5	99.6	69.1%	2.3%	56.7%	56.9%	24.5%	13.7%
	50				122.0	84.2%	10.5%	16.7%	45.2%	18.2%	15.0%
						Avg	**8.5%**	**26.7%**	**48.1%**	**19.8%**	**14.7%**
						Total Avg	**3.0%**	**22.0%**	**41.5%**	**19.9%**	**18.4%**

Table 3: Results of QN-CO compared with APP, QN-C, RP, GP, and HP on small instances

Ψ_{n_z}, m, L	Q	$\mathbb{E}\sum d_c$	QN-CO	$\%\,\mathbb{E}\sum d_c$	QN-C	RP	GP	HP	%QN-C	%RP	%GP	%HP
M, 3, 221.5	25	230	159.1	69.2%	135.4	99.9	143.0	149.2	17.5%	59.2%	11.3%	6.7%
	50		193.1	83.9%	162.7	120.6	171.4	180.6	18.7%	60.2%	12.7%	6.9%
	75		210.1	91.4%	170.5	128.9	192.4	201.0	23.2%	63.0%	9.2%	4.5%
								Avg	**19.8%**	**60.8%**	**11.1%**	**6.0%**
H, 7, 195.5	25	530	360.9	68.1%	296.3	217.9	321.5	345.2%	21.8%	65.6%	12.3%	4.6%
	50		452.7	85.4%	361.4	264.5	417.9	419.2	25.3%	71.1%	8.3%	8.0%
	75		501.2	94.6%	418.1	269.6	460.0	487.2	19.9%	85.9%	8.9%	2.9%
								Avg	**22.3%**	**74.2%**	**9.8%**	**5.1%**
VH, 11, 187.3	25	830	552.5	66.6%	409.9	334.0	502.5	542.2	34.8%	65.4%	9.9%	1.9%
	50		703.0	84.7%	551.8	402.8	664.8	652.2	27.4%	74.6%	5.8%	7.8%
	75		778.7	93.8%	580.6	412.2	738.2	772.8	34.11	88.9%	5.5%	0.8%
								Avg	**32.1%**	**76.3%**	**7.0%**	**3.5%**
								Total Avg	**24.7%**	**70.4%**	**9.3%**	**4.9%**

Table 4: Results of QN-CO compared with QN-C, RP, and GP

average of 24.7%. This shows that the proposed observation function provides sufficient information for the active vehicle to decide. Although no specific pattern can be seen on changes of %QN-C with respect to the vehicle capacity, the improvement of our method over QN-C increases as the instances become denser. This can be explained by the fact that the increase in the number of customers makes the size of the state-action space in QN-C drastically larger. In contrast, the state-action space in our method is designed independent of the number of customers. In particular, this result shows that restricting the action set in QN-CO to a fixed number of target customers in order to reduce the action space, enhances the performance of our methodology.

QN-CO outperforms RP, GP, HP by 70.4%, 9.3%, and 4.9%, on average. The significant improvement over RP demonstrates the effect of the learning. According to Table 4, %GP and %HP fall from 11.1% and 6.0% to 7.0% and 3.5%, as the customer density increases. As previously stated, we argue that, as the customer density increases, the instance becomes relatively simpler for greedy policies. To demonstrate this behavior in our results, we analyze the sensitivity of GP to variations in customer density. To this purpose, we normalize the demand served by GP by $\mathbb{E}\sum d_c$. Notice that the value of $\mathbb{E}\sum d_c$ solely depends on the level of Ψ_{n_z}; hence, $\%\frac{\text{GP}}{\sum d_c}$ gives us a normalized measure that can be used to assess the performance of GP with respect to variations in customer density. This value can also be viewed as the ratio of demands served by GP. The right side of Table 5 reports the average values of $\%\frac{\text{GP}}{\sum d_c}$. We observe that, as the customer density rises, GP serves a larger ratio of demands. Specifically, the average ratio of demands served by GP increases from 73.4% to 76.5% when the customer density increases from M to VH. One intuitive explanation for this is that, as customers in the service area become denser, the problem for each vehicle will be more of a customer selection problem rather than a routing problem, where a long-term collective reward plays a decisive role. As a result, greedy policies, which ignore the routing component, perform better on more densely populated service areas. The left side of Table 5 reports average values of $\%\frac{\text{QN-CO}}{\mathbb{E}\sum d_c}$. We observe that the performance of the QN-CO remains approximately constant as the customer density varies. In particular, the value of $\%\frac{\text{QN-CO}}{\mathbb{E}\sum d_c}$, averaged across different values of Q, remains around $\approx 82\%$ for each level of customer density. This finding is noteworthy because it implies that increases in the problem size due to higher customer density Ψ_{n_z}, do not deteriorate the performance of QN-CO. Additionally, in line with the

earlier discussion, we may attribute this result to the fact that, unlike GP and HP, QN-CO accounts for routing.

$\%\frac{\text{QN-CO}}{\mathbb{E}\sum d_c}$		Q = 25	Q = 50	Q = 75	**Avg**
Ψ_{n_z}	M	69.2%	83.9%	91.4%	**81.5%**
	H	68.1%	85.4%	94.6%	**82.7%**
	VH	66.6%	84.7%	93.8%	**81.7%**
	Avg	**67.9%**	**84.7%**	**93.3%**	**81.9%**

$\%\frac{GP}{\mathbb{E}\sum d_c}$		Q = 25	Q = 50	Q = 75	**Avg**
Ψ_{n_z}	M	62.2%	74.5%	83.6%	**73.4%**
	H	60.7%	78.9%	86.8%	**75.4%**
	VH	60.6%	80.1%	88.9%	**76.5%**
	Avg	**61.1%**	**77.8%**	**86.5%**	**75.1%**

Table 5: The ratio of demands served by QN-CO and GP

Finally, Table 4 shows that the %GP and %HP are usually higher when Q is smaller. For example, considering instances with VH customer density, the %GP declines from 9.9% to 5.5% when the vehicle capacity increases from 25 to 75. A possible reason for this result is that, according to Table 5, as Q increases, the percentage of demands served by QN-CO and GP increases to serve almost all demands. Thus there is a lower margin of improvement in higher vehicle capacities, which reduces the potential difference between their performances.

5.2. Performance of QN-CO on VRPSD instances

The VRPSD can be seen as a particular version of the VRP-VCSD, where the set of customers, including their locations and expected demands, is known in advance. Given that the purpose of QN-CO is to handle variable customer sets, it does not assume the knowledge of customer locations, nor is designed to take advantage of such information. However, we now show that QN-CO can compete with the solution method proposed by Goodson et al. (2016), which is the best-performing benchmark specialized for VRPSDs with duration limits and multiple vehicles.

In Section 5.2.1, we define the instance set and describe the scenario generation procedure. For each instance, we train QN-CO on a training set comprised of 1.5 million scenarios of customer demand realizations sampled from distribution functions described in Section 5.2.1. We note that, since no specific distribution function is available for the customer locations of the instances studied in Goodson et al. (2016), a fixed set of customer locations is used for all training scenarios. Similar to the previous section, we decay the exploration and the learning rates linearly from 1.0 and 10^{-3} to 0.1 and 10^{-4} in the first 500K and one million trials. We describe the benchmark in Section 5.2.2 and discuss the results in Section 5.2.3.

5.2.1. VRPSD Instances

The VRPSD, as a particular version of the VRP-VCSD, assumes that the number of customers, their locations and their expected demands are given in advance. Therefore, the distribution functions $\Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}$ do not play a role in this section. The remaining instance-defining components $(\mathcal{A}, \Psi_{\hat{d}}, m, Q, L)$ are determined according to the procedure used in Goodson et al. (2016), which is described in Appendix A.1.2.

Goodson et al. (2016) conduct experiments on 216 instances. However, testing the QN-CO on the full set of instances is time-consuming and beyond the scope of this paper. Therefore, we choose a subset of relatively challenging instances. In particular, we focus on instances with $n = 75$ constructed from R101 Solomon instances. In these instances, the number of vehicles m is 11. We identify a VRPSD instance $i \in I$ as (L, Q, U) with $L \in \mathcal{L}$, $Q \in \mathcal{Q}$, and $U \in \mathcal{U}$. We denote U as the stochastic variability of the demands $\Psi_{\hat{d}}$. We define $\mathcal{L}, \mathcal{Q}$, and $\mathcal{U}$ as follows: $\mathcal{L} = \{\text{Short, Medium, Long}\}, \mathcal{Q} = \{25, 50, 75\}$, and $\mathcal{U} = \{\text{Low, Moderate, High}\}$. A total of 27 instances are tested.

5.2.2. Benchmarks

Goodson et al. (2016) proposed a rollout algorithm (RA) based on a restocking fixed-route policy. The RA is a form of forward-dynamic programming that can be viewed as a one-step policy iteration (Bertsekas 2013). At each decision epoch, they generate a set of fixed routes using a local search heuristic. Each fixed route is then evaluated using a reward-to-go function for a set of sample demand realizations. Finally, the fixed route with the highest reward-to-go is selected, and the next move on that route is chosen. To account

for preemptive actions for instances with up to 25 customers, they solve an auxiliary dynamic program to evaluate the reward-to-go of following a fixed route with and without performing a restocking move. For instances with more than 25 customers, they proposed a dynamic decomposition method to partition customers between vehicles, enabling them to handle instances of up to 100 customers. Goodson et al. (2016) obtain the initial fixed route by solving the VRPSD using a Simulated Annealing algorithm. They showed that the high-quality initial fixed route plays a vital role. According to their findings, starting with a high-quality initial solution outperforms the case of starting with a low-quality (randomly produced) fixed route by an average of 11.2%. We refer to this benchmark policy as GHQ when the algorithm starts with a high-quality fixed route, and refer to it as GLQ when the algorithm starts with a low-quality fixed route.

5.2.3. Results and Discussion

For each instance $i \in I$, we train QN-CO on a training set comprised of 1.5 million scenarios of customer demand realizations sampled from distribution functions described in Section 5.2.1. We recall that, since no specific distribution function is available for the customer locations of the instances studied in Goodson et al. (2016), a fixed set of customer locations is used for all training scenarios. The test set $\hat{I}(i)$ is obtained by sampling 500 demand realizations for each instance $i \in I$. The performance of our method is then compared with two benchmark policies, GLQ and GHQ. GLQ is analogous to our method in that, like QN-CO, it does not need computing an initial route for each instance. On the other hand, GHQ necessitates the computation of a high-quality initial route for each $i \in I$ instance.

The comparison is summarized in Table 6. In this table, the first three columns indicate the characteristics of the instance. The duration limits S, M, and L, respectively, refer to Short, Medium, and Long. The values L, M, and H for the stochastic variability are abbreviations for Low, Moderate, and High, respectively. The total served demand by QN-CO averaged over 500 realizations ($\hat{I}(i)$) for each instance $i \in I$ is reported as QN-CO. Column ($\%\mathbb{E}\sum d_c$) shows the ratio of demands that QN-CO serves. Note that the total expected demand for each instance is 1079. The results of the benchmarks are shown in columns GLQ and GHQ. Columns %GLQ and %GHQ display the percentage improvement of our method over GLQ and GHQ.

L	Q	U	QN-CO	$\% \mathbb{E}\sum d_c$	GLQ	GHQ	%GLQ	%GHQ
S	25	L	580.6	53.8%	534.6	595.9	8.6%	-2.6%
		M	556.3	51.6%	475.1	566.6	17.1%	-1.8%
		H	534.0	49.5%	479.8	527.5	11.3%	1.2%
	50	L	799.7	74.1%	738.3	834.0	8.3%	-4.1%
		M	781.0	72.4%	718.5	794.7	8.7%	-1.7%
		H	751.0	69.6%	704.9	760.4	6.5%	-1.2%
	75	L	934.0	86.6%	848.7	1011.4	10.1%	-7.7%
		M	902.4	83.6%	845.2	966.2	6.8%	-6.6%
		H	885.4	82.1%	839.3	913.5	5.5%	-3.1%
M	25	L	825.8	76.5%	785.6	842.9	5.1%	-2.0%
		M	800.3	74.2%	778.2	808.6	2.9%	-1.0%
		H	770.7	71.4%	747.3	776.2	3.1%	-0.7%
	50	L	1054.8	97.8%	1019.2	1077.5	3.5%	-2.1%
		M	1035.8	96.0%	1002.8	1067.0	3.3%	-2.9%
		H	1023.8	94.9%	977.1	1043.6	4.8%	-1.9%
	75	L	1074.5	99.6%	1058.3	1078.4	1.5%	-0.4%
		M	1072.8	99.4%	1052.1	1077.2	2.0%	-0.4%
		H	1069.7	99.1%	1044.4	1075.3	2.4%	-0.5%
L	25	L	1004.1	93.1%	974.4	1039.0	3.1%	-3.4%
		M	984.5	91.2%	959.9	998.6	2.6%	-1.4%
		H	971.3	90.0%	940.3	968.0	3.3%	0.3%
	50	L	1078.7	100.0%	1078.3	1078.3	0.0%	0.0%
		M	1080.3	100.1%	1076.3	1076.9	0.4%	0.3%
		H	1081.4	100.2%	1072.3	1075.3	0.9%	0.6%
	75	L	1079.7	100.1%	1078.4	1078.4	0.1%	0.1%
		M	1081.8	100.3%	1077.4	1077.5	0.4%	0.4%
		H	1082.7	100.3%	1075.5	1075.6	0.7%	0.7%
						Avg	**4.6%**	**-1.6%**

Table 6: Results of QN-CO compared with Goodson et al. (2016)

Avg Imp.	L			Q			U			Total
	S	M	L	25	50	75	L	M	H	
%GLQ	9.2%	3.2%	1.4%	6.3%	4.0%	3.3%	4.5%	4.9%	4.3%	4.6%
%GHQ	-3.1%	-1.4%	-0.3%	-1.3%	-1.5%	-1.9%	-2.5%	-1.7%	-0.5%	-1.6%

Table 7: Average improvement of QN-CO over GLQ and GHQ with respect to L, Q, and U

Overall, results show that QN-CO outperforms the GLQ on average by 4.6%, while GHQ outperforms QN-CO by 1.6%. To facilitate the discussion, the results are aggregated with respect to L, Q, and U in Table 7. Comparing the percentage improvements, averaged for each level of duration limit, demonstrates the considerable impact of the duration limit on the gap between our method and two benchmarks. We observe that, as the duration limit becomes longer, the gap between the performance of methods is less pronounced. In particular, when the duration limit changes from Short to Long, the absolute average value of %GLQ and %GHQ fall from 9.2% and 3.1% to 1.4% and 0.3%, respectively. This reduction in the gap may be explained by observing that, when the duration limit increases, it become less binding, and all methods are able to serve almost all demands. This logic is also supported by results reported in the fifth column ($\%\mathbb{E}\sum d_c$) of Table 6. The QN-CO serves almost all demands (at least 90.0%, except for instances with $(L, Q) = (\text{Medium}, 25)$) when the duration limit is Medium or Long. We now asses the impact of the vehicle capacity. Table 7 shows that the average %GHQ declines when vehicle capacity is reduced, whereas QN-CO widens the gap with GLQ. Since smaller vehicle capacities imply a larger number of restocking operations, this result may be interpreted as a sign that QN-CO successfully handles preventative restocking operations.

Regarding the impact of the stochastic variability, Table 7 shows an overall reduction in the absolute average gaps (%GLQ and %GHQ) as U increases from Low to High. This behavior can be explained by looking into the difference between GLQ and GHQ. Figure 6 illustrates the trend of served demands by QN-CO, GLQ, and GHQ, when the capacity is 50, in different levels of duration limit with respect to changes in stochastic variability. In this figure, results for different duration limits are identified by an indication appended to the policy name. For example, QN-CO-S refers to the results of QN-CO in instances with a Short duration limit. As shown in this figure, the increase in U narrows the gap between GLQ and GHQ when the duration limit is Low. This finding implies that as the stochastic variability increases, having a high-quality initial solution becomes less important. This figure also explains why the gap between GLQ and GHQ, and consequently %GLQ and %GHQ, do not follow a descending trend (as in instances with Short L) when U rises. The amount of served demands in instances with Medium and Long duration limit are very close to the total expected demand (i.e., 1079), making the performance of policies bounded. Hence the gap between GLQ and GHQ cannot follow the same trend as in instances with a Short duration limit. Another important finding is that while the performance of GHQ, when U increases from Low to High in instances with Short L, drops by 10.0% (from 595.9, 834.0, and 1011.4 to 527.5, 760.4, and 913.5, respectively for Small, Medium, and Large Q), the performance of our QN-CO decreases only by 6.4% (from 580.6, 799.7, and 934.0 to 534.0, 751.0, and 885.4, respectively for Small, Medium, and Large Q). It can be deduced that our method is relatively better at dealing with stochastic variability than GHQ.

A last comment concerns the computing times. Given that GLQ starts from a random set of routes, the initialization time is negligible. Conversely, GHQ necessitates the computation of a high-quality initial route for each instance $i \in I$. The computing times for establishing initial routes are not reported in Goodson et al. (2016). However, results shared with us by the authors, we conclude that the average CPU time dedicated to obtaining high-quality initial solutions for instances with 75 customers was 25,466 seconds (≈ 7 hours). Therefore, GHQ may not be suitable for situations where customers are variable and there is a relatively short time frame (less than 7 hours) between the realization of the customer set and the start of operations. In comparison, our method, once trained for 24 to 36 hours, can be used with no further computation as long as the instance (including the associated distribution functions and parameters such as L, Q, U) remains the same. Therefore, we can roughly claim that using our method for more than five ($\approx \frac{36}{7}$) operating days is computationally advantageous over the GHQ.

Summarizing, we observed that QN-CO and GLQ are analogous in the sense they both start with a random initial policy and solution. Results showed that the QN-CO method could outperform GLQ by 4.6% on average. On the other hand, the average gap between QN-CO and GHQ, where a high-quality

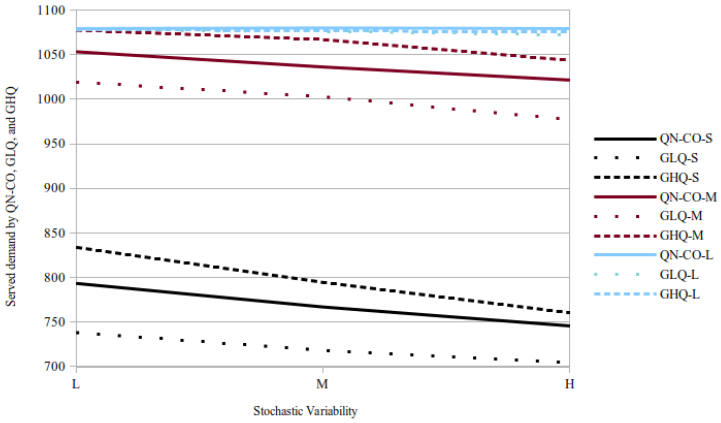


Figure 6: Performance of QN-CO, GLQ, and GHQ with respect to the stochastic variability

initial solution is required, is -1.6%. However, results showed that the advantages of starting from a high-quality initial solution become less pronounced for higher values of stochastic variability. Furthermore, obtaining high-quality initial solutions is computationally expensive, and QN-CO resulted computationally advantageous with respect to GHQ after five days of operations.

5.3. Generalized QN-CO

The previous section showed that when QN-CO is trained on VRPSD instances, it can compete with Goodson et al. (2016). However, it was necessary to train a new policy for each instance $i \in I$. In this experiment, we further explore the flexibility of the QN-CO in terms of its generalization capabilities. Specifically, we investigate the possibility to train one QN-CO model to solve a group of VRPSD instances I', where $I' \subset I$. This also has some practical advantages, as it would be beneficial for a manager to apply the same policy regardless of, for example, the duration limit.

We use the same instances described in Section 5.2.1, but we restrict our analysis to instances with $Q = 50$ (i.e., $I = \mathcal{L} \times \{50\} \times \mathcal{U}$). We carry out the experiment in three steps. In the first step, we aggregate instances with respect to the duration limit L. Accordingly, we train three policies, each for a $U \in \mathcal{U}$. More specifically, in instances aggregated over L, the duration limit follows a uniform distribution function to pick a value from $\mathcal{L}$. Similarly, we generalize the method over the stochastic variability U in the next step. In the last step, we generalize the QN-CO over both components. In this setting, a single policy is trained to solve all nine instances. To provide enough information to QN-CO, we append the value of the component on which we are aggregating to the observation representation. For example, when aggregating over the duration limit, we pass L as an extra component into Equation (26), resulting in $o_{k,\bar{v}} = [F_k|H_k|G_k|t_k|L]$.

Once we obtain a policy, we test it on all instances considered in the training phase. As in the previous experiments, the training and testing realization sets are disjoint. For the first step, Table 8 summarizes the results of the trained policies (QN-CO$_L$) and compares them with those trained in Section 5.2 (QN-CO) and the two benchmark policies, GLQ and GHQ. Detailed computational results are reported in Table B.23, in Appendix B. Table 8 reports the improvement percentage of the generalized policy over the benchmarks, averaged for each instance characteristic. Results show that the aggregation over the duration limit performs on average 2.3% and 3.7% worse than the individual policies (QN-CO) and the GHQ, respectively. However, it still outperforms the GLQ by 1.6%. It can also be seen that the gap between QN-CO$_L$ and QN-CO decreases when the duration limit gets longer (i.e., from -4.6% to -0.1%). Also, the aggregation works relatively better in higher stochastic variability.

Table 9 reports the results of the second step (QN-CO$_U$), where instances are aggregated with respect to the stochastic variability. Detailed results are provided in Table B.24 in Appendix B. Similar to the previous analysis, we compare the results with the individual (QN-CO) and benchmark (GLQ and GHQ)

Avg Imp.	L			U			Total
	S	M	L	L	M	H	
%QN-CO	-4.6%	-2.2%	-0.1%	-2.3%	-2.5%	-2.0%	-2.3%
%GLQ	2.9%	1.6%	0.3%	1.5%	1.4%	1.9%	1.6%
%GHQ	-6.8%	-4.4%	0.2%	-4.3%	-3.9%	-2.8%	-3.7%

Table 8: Average improvement of QN-CO$_L$ over QN-CO, GLQ, and GHQ

policies. Results show that our method can handle a broader range of instances at the expense of performing on average 0.7% worse than the individual policies. Interestingly, it can be seen that the generalized QN-CO performs as well as the individual policies in instances with Medium and Long duration limits. Although the generalized model has a gap of -2.2% with the GHQ, it still considerably outperforms the GLQ (3.2%).

Avg Imp.	L			U			Total
	S	M	L	L	M	H	
%QN-CO	-2.2%	0.0%	0.0%	-1.2%	-0.8%	-0.3%	-0.7%
%GLQ	5.4%	3.9%	0.4%	2.7%	3.3%	3.8%	3.2%
%GHQ	-4.5%	-2.3%	0.3%	-3.2%	-2.2%	-1.1%	-2.2%

Table 9: Average improvement of QN-CO$_U$ over QN-CO, GLQ, and GHQ

Finally, Table 10 reports the performance of the single policy trained on all nine instances (QN-CO$_{All}$). Detailed results are provided in Table B.25 in Appendix B. Similar to Tables 8 and 9, we compare the performance of the generalized QN-CO with the individual and benchmark policies. Results show that QN-CO$_{All}$ is able to solve a set of instances with varying values of L and U at the expense of a 2.1% reduction of the performance quality. Confronting these results with those obtained in Tables 8 and 9, we can deduce that most of the performance deterioration comes from the generalization over the time limit dimension L. This experiment also shows that, if suitably trained, the proposed algorithm can be adapted to handle problems with varying features such as, for example, different duration limits.

Avg Imp.	L			U			Total
	S	M	L	L	M	H	
%QN-CO	-4.3%	-1.9%	-0.2%	-2.7%	-2.2%	-1.4%	-2.1%
%GLQ	3.2%	1.9%	0.2%	1.0%	1.8%	2.5%	1.8%
%GHQ	-6.6%	-4.2%	0.1%	-4.7%	-3.6%	-2.3%	-3.5%

Table 10: Average improvement of QN-CO$_{All}$ over QN-CO, GLQ, and GHQ

5.4. Managerial insights

We proposed two models for the VRP-VCSD, i.e., MDP-C and MDP-CO. For MDP-C, we experimented with solution methods RP, GP, HP and QN-C, whereas for MDP-CO we experimented with QN-CO. From a modeling perspective, the MDP-CO is the more involved model out of the two, as it requires the definition and implementation of the observation function. However, this model allows using the QN-CO, which achieves the highest quality of results. If decision-makers are willing to trade solution quality with methodological simplifications, then we recommend using the MDP-C with HP. Notably, the HP method is fairly easy to implement but its results were inferior to those of QN-CO by 4.9%. More generally, the proposed QN-CO method outperforms all benchmarks at every level of density and vehicle capacity. However, it requires the training of the Q network, which entailed almost 36h of computing time. In contrast, the HP does not require any training time, and thus might be appealing for preliminary testing of realistic applications. We believe that the gain in profit achieved by the QN-CO justifies this training time, since realistic VRP-VCSD applications would solved daily.

6. Conclusions

Motivated by a distribution planning problem arising in domestic donor collection services, we introduced the VRP-VCSD. In this problem, customer locations and demands are stochastic, and a fleet of capacitated vehicles must complete the service within a specified time limit. The goal is to maximize the total served demand. We provided three MDP formulations for the problem. The first follows a traditional MDP formulation, while the second, denoted by MDP-C, imposes the consecutive action selection strategy, where a decision is made for a single vehicle only at each decision epoch. The MDP-C formulation enabled us to reduce the dimension of the action space, thus resulting in a more tractable formulation. In the third formulation, denoted by MDP-CO, we proposed an observation function that substitutes the variable-sized state vector with a smaller vector with a fix size, representing the observation of the vehicle for which a decision must be made. In particular, the proposed observation function addressed the variable-sized stochastic customer sets via a heatmap-style representation of the customer demands. More specifically, this approach partitions the stochastic set of customers geographically, and encodes it as a fixed-size vector. We then solved the resulting problem by a Q-learning algorithm. For this purpose, we developed a two-layer artificial neural network to approximate state-action Q factors. Computational results showed that our method significantly outperforms five benchmarks policies (APP, QN-C, RP, GP, and HP). Although APP provided better bounds than the other benchmarks, it could only handle instances with up to 15 customers. Comparing against QN-C, which implements a Q-learning algorithm on the MDP-C formulation, demonstrated the effectiveness of the proposed observation function. Our results demonstrate that, while our method outperforms HP by a considerable margin, the HP benchmark offers a simple method that provides fairly good performance. This may be appealing to practitioners who prioritize implementation simplicity over solution quality. It is worth noting that the strong performance of HP suggests that the adoption of target customers as state features in the observation function efficiently captures the pertinent state information. Furthermore, we evaluated the QN-CO on the VRPSD, which is a particular case of VRP-VCSD, and showed that our method could compete with the state-of-the-art solution method specialized for that problem. As a further investigation into the capabilities of our framework, we tested and showed that the QN-CO could be generalized over several problem components, such as the stochastic variability of demands. Finally, as shown in Appendix C, the obtained policies and value functions can be easily employed as a base policy and a reward-to-go estimator in on-line rollout methods.

As QN-CO can efficiently handle problems with stochastic customer sets, it is promising to investigate how it could be extended to tackle dynamic versions of the VRP, where customers arise dynamically during the execution period. Our method currently assigns only one vehicle at a time to each customer. However, future research may examine the added value of simultaneously assigning multiple vehicles to a customer. This could be done based on the remaining capacity of the vehicles and customer demand. Furthermore, it is worth investigating the effectiveness of the proposed method on real-world instances.

References

211 of Greater Montréal, 2023. Where can I donate? URL: https://www.211qc.ca/en/material-assistance-and-housing/used-articles-donation. accessed 14 May 2023.

Benton, W.C., Rossetti, M.D., 1992. The vehicle scheduling problem with intermittent customer demands. Computers & Operations Research 19, 521–531.

Bertsekas, D.P., 2013. Rollout algorithms for discrete optimization: A survey, in: Pardalos, P.M., Du, D.Z., Graham, R.L. (Eds.), Handbook of Combinatorial Optimization. Springer New York, New York, NY, pp. 2989–3013.

Brinkmann, J., Ulmer, M.W., Mattfeld, D.C., 2019a. Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. Computers & Operations Research 106, 260–279.

Brinkmann, J., Ulmer, M.W., Mattfeld, D.C., 2019b. The multi-vehicle stochastic-dynamic inventory routing problem for bike sharing systems. Business Research 2019 13:1 13, 69–92.

Chen, X., Ulmer, M.W., Thomas, B.W., 2022. Deep Q-learning for same-day delivery with vehicles and drones. European Journal of Operational Research 298, 939–952.

Chen, Y., Qian, Y., Yao, Y., et al., 2019. A case study in dynamic courier dispatching system, in: In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1395–1403.

Curran, A., Williams, I., 2010. The role of furniture and appliance re-use organisations in England and Wales. Resources, Conservation and Recycling 54, 692–703.

Erera, A.L., Morales, J.C., Savelsbergh, M., 2010. The vehicle routing problem with stochastic demand and duration constraints. Transportation Science 44, 474–492.

Fan, J., Wang, X., Ning, H., 2006. A multiple vehicles routing problem algorithm with stochastic demand, in: 6th World Congress on Intelligent Control and Automation, IEEE. pp. 1688–1692.

Gendreau, M., Laporte, G., Séguin, R., 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science 29, 143–155.

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. Operations Research 61, 138–154.

Goodson, J.C., Thomas, B.W., Ohlmann, J.W., 2016. Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. Transportation Science 50, 591–607.

Haughton, M.A., 2002. Route reoptimization's impact on delivery efficiency. Transportation Research Part E: Logistics and Transportation Review 38, 53–63.

Joe, W., Lau, H.C., 2020. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers, in: Proceedings of the international conference on automated planning and scheduling, pp. 394–402.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, International Conference on Learning Representations, ICLR. pp. 100–120.

Kullman, N.D., Cousineau, M., Goodson, J.C., et al., 2021. Dynamic ride-hailing with electric vehicles. Transportation Science , In press.

Kullman, N.D., Mendoza, J.E., Cousineau, M., Goodson, J.C., 2019. Atari-fying the vehicle routing problem with stochastic service requests. arXiv:1911.05922 .

La Collecte Foundation, 2023. Home Pick-up. URL: https://fondationlacollecte.ca/collecte-a-domicile/. accessed 14 May 2023.

Li, J., Ma, Y., Gao, R., et al., 2021a. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Transactions on Cybernetics , 1–14.

Li, M., Qin, Z., Jiao, Y., et al., 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning, in: The WWW Conference, ACM, New York, NY, USA. pp. 983–994.

Li, X., Luo, W., Yuan, M., et al., 2021b. Learning to optimize industry-scale dynamic pickup and delivery problems, in: Proceedings - International Conference on Data Engineering, IEEE. pp. 2511–2522.

Lin, B., Ghaddar, B., Nathwani, J., 2021. Deep reinforcement learning for the electric vehicle routing problem with time windows. IEEE Transactions on Intelligent Transportation Systems , 1–11.

Louveaux, F.V., Salazar-González, J.J., 2018. Exact approach for the vehicle routing problem with stochastic demands and preventive returns. Transportation Science 52, 1463–1478.

Maxwell, M.S., Restrepo, M., Henderson, S.G., Topaloglu, H., 2010. Approximate dynamic programming for ambulance redeployment. INFORMS Journal on Computing 22, 266–281.

Mendoza, J.E., Rousseau, L.M., Villegas, J.G., 2016. A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints. Journal of Heuristics 22, 539–566.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., et al., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533.

Nazari, M., Oroojlooy, A., Takáč, M., et al., 2018. Deep reinforcement learning for solving the vehicle routing problem. Advances in Neural Information Processing Systems 31, 9839–9849.

Novoa, C., Storer, R., 2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European Journal of Operational Research 196, 509–515.

Oda, T., Joe-Wong, C., 2018. Movi: A model-free approach to dynamic fleet management, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE. pp. 2708–2716.

OroojlooyJadid, A., Hajinezhad, D., 2019. A review of cooperative multi-agent deep reinforcement learning. arXiv preprint arXiv:1908.03963 .

Oyola, J., Arntzen, H., Woodruff, D.L., 2017. The stochastic vehicle routing problem, a literature review, Part II: solution methods. EURO Journal on Transportation and Logistics 6, 349–388.

Oyola, J., Arntzen, H., Woodruff, D.L., 2018. The stochastic vehicle routing problem, a literature review, part I: models. EURO Journal on Transportation and Logistics 7, 193–221.

Peng, B., Wang, J., et al., 2020. A deep reinforcement learning algorithm using dynamic attention for vehicle routing problems, in: Li, K., et al. (Eds.), AI Algorithms and Applications, Springer, Singapore. pp. 636–650.

Powell, W.B., 2011. Approximate dynamic programming. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., Hoboken, NJ, USA.

Powell, W.B., Ma, J., 2011. A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multidimensional continuous applications. Journal of Control Theory and Applications 9, 336–352.

Puterman, M.L., 2005. Markov decision processes : discrete stochastic dynamic programming. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc., Hoboken, NJ, USA.

Ritzinger, U., Puchinger, J., Hartl, R.F., 2016. A survey on dynamic and stochastic vehicle routing problems. International Journal of Production Research 54, 215–231.

Secomandi, N., 2001. A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49, 796–802.

Soeffker, N., Ulmer, M.W., Mattfeld, D.C., 2022. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. European Journal of Operational Research 298, 801–820.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265.

Ulmer, M.W., 2020. Dynamic pricing and routing for same-day delivery. Transportation Science 54, 1016–1033.

Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., et al., 2017. Offline-online approximate dynamic programming for dynamic vehicle routing with stochastic requests. Transportation Science 53, 1–35.

Ulmer, M.W., Mattfeld, D.C., Köster, F., 2018. Budgeting time for dynamic vehicle routing with stochastic customer requests. Transportation Science 52, 20–37.

Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 100–120.

Voccia, S.A., Campbell, A.M., Thomas, B.W., 2013. The probabilistic traveling salesman problem with time windows. EURO Journal on Transportation and Logistics 2, 89–107.

Waters, C.D.J., 1989. Vehicle-scheduling problems with uncertainty and omitted customers. The Journal of the Operational Research Society 40, 1099–1108.

Watkins, C.J.C.H., Dayan, P., 1992. Q-learning. Machine Learning 8, 279–292.

Appendix A. Experimental setup

Appendix A.1. Instance generation

This section provides details on the methods used to generate the VRP-VCSD and VRPSD instances in our experiments.

Appendix A.1.1. VRP-VCSD instances

We consider a squared-shape service area $\mathcal{A}$ of dimension 100×100 with a depot located at the center ($l_0 = (50, 50)$). Customers can potentially be located at any position inside the service area. However, in real-life applications, certain zones of the service area never generate requests, such as non-residential areas. Therefore, for a given service area partitioned into a set of 5×5 square-shaped zones Z, we randomly pick a subset $\mathcal{Z}$ of *active* zones, in which customers must be located. In our experiment, we consider 15 active zones out of 25 (i.e., 60% of the zones are active). Figure A.7 shows the layout of the considered region.

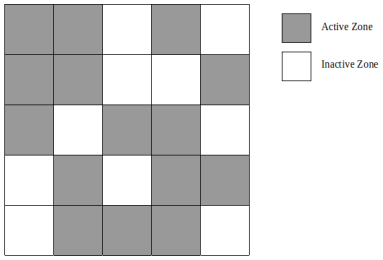


Figure A.7: The layout of active zones

To generate the customer locations and demands, we make the assumption that active zones are homogeneous in the sense that the customer density Ψ_{n_z} and distributions Ψ_l, $\Psi_{\bar{d}}$, $\Psi_{\hat{d}}$ do not depend on the specific zone. The customer density Ψ_{n_z} specifies the probability distribution of the number n_z of requests for each active zone. We consider five possible levels of customer density $\mathcal{D}$ ={Very Low, Low, Moderate, High, Very High}. Table A.11 presents the distribution function of n_z for each level of customer density. Additionally, this table provides the expected number of customers at each zone $\bar{n}_z$, corresponding to the given distribution function. The total expected number of customers $\bar{n}$ for a given Ψ_{n_z} is calculated as $\bar{n} = \mathbb{E}[\sum_{z\in\mathcal{Z}} n_z] = |\mathcal{Z}| * \bar{n}_z$, which is shown in the last column of Table A.11 for each level of Ψ_{n_z}. Once n_z is sampled according to the distribution Ψ_{n_z} for each zone $z \in \mathcal{Z}$, the customer locations are generated following the distribution Ψ_l. In this experiment, we choose Ψ_l to be the uniform distribution function over (x,y)-coordinates inside a given zone. The above process allows us to sample the initial set of customers $\mathcal{C}_0$.

Ψ_{n_z}	n_z Values	n_z Probabilities	$\bar{n}_z$	$\bar{n}$
Very Low	{0, 1, 2}	$\{0.5, 0.\bar{3}, 0.1\bar{6}\}$	$0.\bar{6}$	10
Low	{0, 1, 2}	$\{0.\bar{3}, 0.\bar{3}, 0.\bar{3}\}$	1.0	15
Moderate	{0, 1, 2, 3}	{0.1, 0.4, 0.4, 0.1}	1.5	23
High	{2, 3, 4, 5}	{0.1, 0.4, 0.4, 0.1}	3.5	53
Very High	{4, 5, 6, 7}	{0.1, 0.4, 0.4, 0.1}	5.5	83

Table A.11: Distribution functions of the number of customers at each zone n_z for each density level Ψ_{n_z}

Given a customer $c \in \mathcal{C}_0$, the expected demand $\bar{d}_c$ and actual demand $\hat{d}_c$ are sampled according to distributions $\Psi_{\bar{d}}$ and $\Psi_{\hat{d}}$, respectively. These distributions are chosen following the method proposed in Gendreau et al. (1995). In particular, $\Psi_{\bar{d}}$ is the discrete uniform distribution over the values $\{5, 10, 15\}$. Once the value $\bar{d}_c$ has been sampled, the actual demands follow the distribution $\Psi_{\hat{d}}$ which is the discrete uniform distribution on values $\{\bar{d}_c - 5, ..., \bar{d}_c, ..., \bar{d}_c + 5\}$. In order to avoid the no-demands case when $\bar{d}_c = 5$, we use the discrete uniform distribution function on values $\{\bar{d}_c - 4, ..., \bar{d}_c, ..., \bar{d}_c + 4\}$.

We determine the number of vehicles m and the time limit L uniquely as functions of the distribution Ψ_{n_z} by linking them through the concept of the filling rate. The filling rate $f = \frac{\mathbb{E}[\sum_{c\in\mathcal{C}_0} \bar{d}_c]}{m*Q}$ indicates the portion of all demands that can be served by all vehicles without replenishment. In the equation above, $\mathbb{E}[\sum_{c\in\mathcal{C}_0} \bar{d}_c]$ is the expected total demand and can be approximated by $\bar{n}_z * |\mathcal{Z}| * \bar{\bar{d}}$, where $\bar{\bar{d}}$ is the average expected demand of a customer. Since the distribution function $\Psi_{\bar{d}}$ assigns expected demands 5, 10, or 15 to customers uniformly (with a probability of $\frac{1}{3}$), the average expected demand of a customer ($\bar{\bar{d}}$) will be equal to 10 ($\frac{5+10+15}{3}$) in all instances. By fixing the filling rate and the vehicle capacity to 1 and 75, respectively, the number of vehicles for each level of customer density Ψ_{n_z} can then be obtained by the definition of the filling rate:

$$m = \left\lceil \frac{\mathbb{E}[\sum_{c\in\mathcal{C}_0} \bar{d}_c]}{f * Q} \right\rceil = \left\lceil \frac{\bar{n}_z * |\mathcal{Z}| * \bar{\bar{d}}}{f * Q} \right\rceil = \left\lceil \frac{\bar{n}_z * 15 * 10}{1 * 75} \right\rceil = \lceil 2 * \bar{n}_z \rceil. \tag{A.1}$$

In this way, the number of required vehicles for each class of customer density is 2, 2, 3, 7, and 11, respectively.

Notice that, according to the expression $\mathbb{E}[\sum_{c\in\mathcal{C}_0}\bar{d}_c]$, the expected total demand at each instance will be 100, 150, 230, 530, and 830 for Very Low, Low, Moderate, High and Very High customer density, respectively.

To determine the duration limit L for a given instance with the customer density distribution Ψ_{n_z} and its associated m, we use the average travel time of vehicles acquired by solving a set of 250,000 realizations of that instance via GP (described in Section 5.1.2). In this process, we assumed that $L = \infty$, $Q = 75$, and realizations are randomly sampled from distributions $\Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}$. To make the duration limit constraining, we let the duration limit be the average travel time of all vehicles multiplied by 0.75. The resulting duration limits for Very Low, Low, Moderate, High and Very High customer densities are 143.71, 201.38, 221.47, 195.54, and 187.29, respectively. To complete the description of the instance generation, we need to determine the values Q. According to Equation (A.1), the number of vehicles m is determined such that when $Q = 75$, vehicles can serve the total expected demand without replenishing at the depot. Therefore, for a given duration limit, while using smaller capacities (< 75) may increase the frequency of restocking operations, which results in serving fewer customers in a limited time, considering larger vehicle capacities (> 75) may not necessarily affect the performance of vehicles. Therefore, in this experiment, we consider three possible values $\mathcal{Q} = \{25, 50, 75\}$.

To summarize, the instance set I is obtained by varying the customer density distribution function $\Psi_{n_z} \in \mathcal{D} = \{\text{Very Low, Low, Moderate, High, Very High}\}$ and the vehicle capacity $Q \in \mathcal{Q} = \{25, 50, 75\}$, resulting in 15 different instances. Fixing the customer density distribution also uniquely determines the number of vehicles m and the time limit L. The rest of the instance-defining elements, namely the service area $\mathcal{A}$ and the distributions $\Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}$ are kept fixed throughout the experiment. In other words an instance in this experiment is completely identified by the pair (Ψ_{n_z}, Q). Finally, realizations $\hat{i}$ of an instance i are obtained by sampling from distributions $\Psi_{n_z}, \Psi_l, \Psi_{\bar{d}}, \Psi_{\hat{d}}$.

Appendix A.1.2. VRPSD instances

Goodson et al. (2013) used the *R101* and *C101* instances from Solomon (1987) and took the first n customers to generate the set $\mathcal{C}_0$ with $n = 25$, 50, 75, and 100. Concerning expected demands, the authors used the deterministic demands provided by the Solomon's instances. To determine the number of vehicles m and the duration limit L, the authors solved each instance as a classical VRP using a heuristic with a vehicle capacity of 100. The resulting number of routes determines m, while a set of three time limits are obtained by multiplying the length of the longest route by 0.75, 1.25, and 1.75, resulting in $\mathcal{L} = \{\text{Short, Medium, Long}\}$. The considered duration limits Short, Medium, and Long are 103.05, 171.75, and 240.45, respectively. They also varied the vehicle capacity Q to take a size in $\mathcal{Q} = \{25, 50, 75\}$.

Regarding the distribution functions of the customer demands, they characterized $\Psi_{\hat{d}}$ by an instance identifier U, denoted as the stochastic variability of the demands. The stochastic variability U may have a value in $\mathcal{U} = \{\text{Low, Moderate, High}\}$. In particular, $\Psi_{\hat{d}}$ takes the distribution function of $\{\bar{d}/2, \bar{d}, 3\bar{d}/2\}$ with probabilities $(0.05, 0.9, 0.05)$ for Low, $\{0, \bar{d}/2, \bar{d}, 3\bar{d}/2, 2\bar{d}\}$ with probabilities $(0.05, 0.15, 0.6, 0.15, 0.05)$ for Moderate, and $\{0, \bar{d}/2, \bar{d}, 3\bar{d}/2, 2\bar{d}\}$ with a uniform probability for High stochastic variability, respectively. For details, the reader may refer to Goodson et al. (2016). In consequence, the authors conduct experiments on 216 instances, comprised of eight different sets of customers (i.e., four levels for the number of customers $\{25, 50, 75, 100\}$ $\times$ two classes of instances {R101, C101}). In particular, for each set of customers with the same locations and expected demands, they did tests on 27 distinct instances.

Appendix A.2. Hyper-parameters and computing facility

The hyper-parameters in Algorithm 2 have been set as follows. We consider a minibatch of 32 experiences ($|\tilde{B}| = 32$) uniformly sampled from the replay memory with a size of 50000, where $\beta_t = 0.05$. We also replace the target network every 1000 trials ($\beta_d = 1000$). For all experiments, we set the discount factor γ to 0.999 (Kullman et al. 2021). The hyper-parameters related to the maximum number of trials ($Trials\_MAX$) and the decaying factor for the learning rate (α) and the exploration rate (ϵ) will be specified for each experiment later. In training the QN-C for a given instance, we set the maximum number of realized customers $\hat{n} = 1.2 * \bar{n}$, where $\bar{n}$ is the expected total number of customers for that instance. All the procedures have been implemented in python and executed on 3.6 GHz Intel Xeon processors with 64 GB RAM (no GPUs are used).

Appendix A.3. Deterministic Formulation

In this section, we present a mathematical formulation for the deterministic version of the VRP-VCSD. Unlike QN-CO, where both customers and demands are stochastic, this formulation assumes that the set of customers is known, and the actual demand is equal to the expected value. In order to provide an upper bound for the possible number of trips for a vehicle (E), we propose a heuristic algorithm, which is illustrated in Algorithm 3. This algorithm computes the total number of trips for a vehicle in the worst-case scenario when it visits customers one-per-trip from closest to farthest in terms of their distance to the depot. This strategy results in the highest number of trips.

Algorithm 3: Compute an upper bound for the number of trips of a vehicle E

```
C is the set of customers in C_0 ordered such that τ_0i ≤ τ_0j, ∀i < j;
E ← 0, i ← 0, l ← 0;
while l ≤ L do
    if l + τ_0i + τ_i0 > L then
        return E;
    else
        l ← l + τ_0i + τ_i0;
        E ← E + 1;
        if d_i ≥ Q then
            d_i ← d_i − Q;
        else
            i ← i + 1;
```

Notation	Description
N	set of nodes, $N = \mathcal{C}_0 \cup \{0\}$. Node 0 represents the depot.
i, j	index of nodes in N
$\mathcal{E}$	set of trips for a vehicle, $e \in \mathcal{E}$, $\mathcal{E} = \{0, 1, ..., E-1\}$
E	an upper bound for the number of trips for a vehicle
M	a big number
x_i^{ve}	the amount of node i's demand that is served by vehicle v in trip e
y_{ij}^{ve}	binary variable takes 1 if node j is immediately visited after node i by vehicle v in trip e
λ_i^{ve}	binary variable takes 1 if customer i is visited by vehicle v in trip e
q_i^{ve}	available capacity of vehicle v when it arrives at node i in trip e
t_i^{ve}	time at which the vehicle v arrives at node i in trip e
$\underline{t}^{ve}$	time at which the vehicle v leaves the depot for trip e

Table A.21: Notations of the deterministic formulation

$$\max \quad \sum_{i\in N}\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}} x_i^{ve} \tag{A.2}$$

$$\text{s.t.} \quad \sum_{j\in N} y_{ij}^{ve} \le 1; \qquad \forall i \in N, v \in \mathcal{V}, e \in \mathcal{E} \tag{A.3}$$

$$\sum_{j\in N} y_{ji}^{ve} \le 1; \qquad \forall i \in N, v \in \mathcal{V}, e \in \mathcal{E} \tag{A.4}$$

$$y_{ii}^{ve} = 0; \qquad \forall i \in N, v \in \mathcal{V}, e \in \mathcal{E} \tag{A.5}$$

$$\sum_{i\in N} y_{0i}^{ve} \le \sum_{i\in N} \lambda_i^{ve}; \qquad \forall v \in \mathcal{V}, e \in \mathcal{E} \tag{A.6}$$

$$\sum_{j\in N} y_{ji}^{ve} = \sum_{j\in N} y_{ij}^{ve} = \lambda_i^{ve}; \qquad \forall i \in N, v \in \mathcal{V}, e \in \mathcal{E} \tag{A.7}$$

$$\sum_{i\in N}\sum_{j\in N} y_{ij}^{ve} \le M\Big(\sum_{i\in N}\sum_{j\in N} y_{ij}^{v,e-1}\Big); \qquad \forall v \in \mathcal{V}, e \in \mathcal{E} \setminus 0 \tag{A.8}$$

$$\sum_{i\in N}\lambda_i^{ve} \le M(\sum_{i\in N}\lambda_i^{v,e-1}); \qquad \forall v\in\mathcal{V}, e\in\mathcal{E}\setminus 0 \qquad \text{(A.9)}$$

$$x_i^{ve} \le M\lambda_i^{ve}; \qquad \forall i\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.10)}$$

$$x_0^{ve} = 0; \qquad \forall v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.11)}$$

$$\sum_{v\in\mathcal{V}}\sum_{e\in\mathcal{E}} x_i^{ve} \le d_i; \qquad \forall i\in N \qquad \text{(A.12)}$$

$$x_i^{ve} \le q_i^{ve}; \qquad \forall i\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.13)}$$

$$\sum_{i\in N} x_i^{ve} \le Q; \qquad \forall v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.14)}$$

$$q_j^{ve} \le Q + M(1-y_{0j}^{ve}); \qquad \forall j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.15)}$$

$$q_j^{ve} \ge Q - M(1-y_{0j}^{ve}); \qquad \forall j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.16)}$$

$$q_j^{ve} \le (q_i^{ve} - x_i^{ve}) + M(1-y_{ij}^{ve}); \qquad \forall i,j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.17)}$$

$$q_j^{ve} \ge (q_i^{ve} - x_i^{ve}) - M(1-y_{ij}^{ve}); \qquad \forall i,j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.18)}$$

$$q_i^{ve} <= M\lambda_i^{ve}; \qquad \forall i\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.19)}$$

$$\underline{t}^{v0} = 0; \qquad \forall v\in\mathcal{V} \qquad \text{(A.20)}$$

$$\underline{t}^{v,e+1} = \underline{t}^{ve} + \sum_{i\in N}\sum_{j\in N}\tau_{ij}y_{ij}^{ve}; \qquad \forall v\in\mathcal{V}, e\in\mathcal{E}\setminus 0 \qquad \text{(A.21)}$$

$$t_j^{ve} \ge \underline{t}^{ve} + \tau_{0j} - M(1-y_{0j}^{ve}); \qquad \forall j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.22)}$$

$$t_j^{ve} \ge t_i^{ve} + \tau_{ij} - M(1-y_{ij}^{ve}); \qquad \forall i,j\in N\setminus 0, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.23)}$$

$$t_j^{ve} \le M\lambda_j^{ve}; \qquad \forall j\in N, v\in\mathcal{V}, e\in\mathcal{E} \qquad \text{(A.24)}$$

$$\underline{t}^{v,E-1} + \sum_{i\in N}\sum_{j\in N}\tau_{ij}y_{ij}^{v,E-1} \le L; \qquad \forall v\in\mathcal{V} \qquad \text{(A.25)}$$

$$0 \le x_i^{ve}, q_i^{ve} \le Q,\ 0 \le t_i^{ve},\ \underline{t}^{ve} \le L, y_{ij}^{ve}, \lambda_i^{ve} \in \{0,1\}; \qquad \forall i,j\in N, v\in\mathcal{V}, e\in\mathcal{E}$$

In this mathematical formulation, the objective function (A.2) maximizes the total amount of served demands. Constraints (A.3)–(A.5) ensure that each customer cannot be visited more than once on each trip. Constraint (A.6) implies that if a vehicle visits no customer during a trip, no traverse between nodes should be planned for that vehicle on that trip. On the other hand, constraint (A.7) states that if a customer is decided to be served, a route to visit that customer should be planned. Constraints (A.8) and (A.9) guarantee that trips of a vehicle should be planned consecutively. Constraints (A.10)–(A.14) restrict the amount a vehicle can serve when visiting a customer on a trip by customer's demand and vehicle capacity. Constraints (A.15)–(A.19) are flow balance constraints. Lastly, constraints (A.20)–(A.25) keep track of operation time and restrict it by the given duration limit.

Appendix B. Extended computational results

This section provides extended results of the first and third experiment reported in Tables B.22–B.25. Note that row Avg reports averages over instances solved to optimality by APP.

Ψ_{n_z}	Q	n	$\mathbb{E}\sum d_c$	QN-CO	$\%\mathbb{E}\sum d_c$	Opt. gap	APP	QN-C	RP	GP	HP	%APP	%QN-C	%RP	%GP	%HP
Very Low	25	12	125.0	74.6	59.7%	66.7%	-	58.4	53.3	55.5	52.8	-	27.7%	40.0%	34.3%	41.3%
		8	85.0	81.1	95.4%	0.0%	75.4	67.0	58.3	56.6	59.0	7.6%	21.1%	39.0%	43.2%	37.4%
		9	100.0	61.8	61.8%	0.0%	61.2	40.2	46.4	49.6	49.8	1.0%	53.5%	33.0%	24.5%	24.0%
		11	115.0	75.0	65.2%	0.0%	76.6	60.1	51.2	52.2	51.0	-2.1%	24.7%	46.6%	43.5%	47.0%
		10	90.0	65.5	72.7%	0.0%	61.0	58.6	49.1	56.0	53.8	7.3%	11.7%	33.4%	16.9%	21.7%
		11	95.0	52.8	55.5%	0.0%	53.0	47.0	39.5	44.8	45.4	-0.4%	12.2%	33.7%	17.8%	16.2%
		12	125.0	81.7	65.4%	38.9%	-	63.8	59.7	69.2	72.3	-	28.1%	37.0%	18.1%	13.0%
		10	95.0	58.1	61.2%	0.0%	55.8	54.4	44.5	51.4	52.5	4.1%	6.8%	30.6%	13.0%	10.7%
		9	115.0	74.0	64.4%	0.0%	73.5	59.8	61.4	60.2	58.4	0.7%	23.8%	20.5%	22.9%	26.8%
		12	105.0	64.9	61.8%	24.0%	-	42.9	48.3	52.7	53.4	-	51.2%	34.3%	23.1%	21.5%
											Avg	**2.6%**	**22.7%**	**33.8%**	**26.0%**	**26.3%**
	50	12	125.0	102.4	81.9%	0.0%	101.3	89.0	66.1	96.1	97.1	1.1%	17.7%	55.0%	6.5%	5.5%
		8	85.0	84.8	99.8%	0.0%	77.6	74.9	69.7	81.5	83.5	9.3%	13.2%	21.8%	4.1%	1.6%
		9	100.0	85.3	85.3%	0.0%	85.0	63.4	58.0	66.2	66.7	0.3%	34.5%	47.1%	28.8%	27.8%
		11	115.0	89.9	78.2%	0.0%	89.0	79.1	61.1	64.1	64.5	1.1%	13.7%	47.1%	40.2%	39.4%
		10	90.0	85.8	95.3%	0.0%	83.8	79.4	55.6	83.8	82.2	2.3%	8.1%	54.4%	2.3%	4.3%
		11	95.0	65.3	68.8%	0.0%	65.3	54.6	46.6	59.7	61.5	0.0%	19.6%	40.3%	9.4%	6.2%
		12	125.0	108.8	87.0%	0.0%	108.8	87.9	72.3	69.6	72.4	0.0%	23.8%	50.4%	56.3%	50.4%
		10	95.0	74.5	78.5%	0.0%	74.5	60.4	50.5	74.5	74.5	0.0%	23.4%	47.6%	0.0%	0%
		9	115.0	90.3	78.5%	0.0%	89.7	75.6	71.9	80.4	79.4	0.7%	19.45%	25.6%	12.4%	13.8%
		12	105.0	85.8	81.7%	0.0%	88.5	69.5	56.0	88.5	86.0	-3.0%	23.51%	53.3%	-3.0%	-0.3%
											Avg	**1.2%**	**19.7%**	**44.3%**	**15.7%**	**14.4%**
Low	25	14	145.0	105.7	72.9%	26.1%	-	65.6	67.5	87.7	91.4	-	61.1%	56.5%	20.5%	15.7%
		14	125.0	81.5	65.2%	38.9%	-	53.4	53.4	72.5	69.7	-	52.5%	52.7%	12.4%	16.9%
		18	165.0	112.9	68.4%	32.0%	-	91.1	71.0	101.1	101.0	-	23.9%	59.0%	11.6%	11.7%
		15	150.0	98.4	65.6%	42.9%	-	58.8	61.0	87.7	89.2	-	67.2%	61.2%	12.2%	10.3%
		10	100.0	82.7	82.7%	0.0%	80.8	55.5	52.7	66.4	72.7	2.3%	49.1%	56.9%	24.5%	13.7%
		17	180.0	92.8	51.5%	50.0%	-	64.2	53.4	90.4	91.2	-	44.5%	73.6%	2.7%	1.7%
		18	180.0	112.1	62.3%	50.0%	-	80.6	71.1	94.7	96.1	-	39.1%	57.6%	18.4%	16.6%
		18	175.0	101.3	57.9%	59.1%	-	79.1	61.6	87.9	87.4	-	28.1%	64.5%	15.3%	16.0%
		14	115.0	95.7	83.2%	9.5%	-	70.0	60.0	86.0	87.9	-	36.7%	59.6%	11.3%	8.9%
		12	140.0	113.2	80.9%	12.0%	-	84.5	76.5	99.0	103.5	-	34.0%	48.0%	14.3%	9.4%
											Avg	**2.3%**	**49.5%**	**56.9%**	**24.5%**	**13.7%**
	50	14	145.0	132.3	91.2%	7.4%	-	84.7	82.6	103.6	103.9	-	56.1%	60.2%	27.7%	27.3%
		14	125.0	103.6	82.9%	19.1%	-	93.8	65.9	99.5	99.3	-	10.4%	57.2%	4.1%	4.3%
		18	165.0	139.8	84.7%	10. 0%	-	104.7	88.1	111.6	111.4	-	33.5%	58.6%	25.3%	25.5%
		15	150.0	122.9	81.9%	20.0%	-	98.5	74.8	101.2	102.3	-	24.8%	64.3%	21.4%	20.1%
		10	100.0	96.2	96.2%	0.0%	84.4	90.9	69.2	85.9	89.3	13.9%	5.8%	39.1%	12.0%	7.7%
		17	180.0	117.6	65.3%	44.0%	-	93.1	78.1	105.1	104.2	-	26.4%	50.5%	11.9%	12.9%
		18	180.0	141.5	78.6%	24.1%	-	114.5	86.1	120.3	117.7	-	23.6%	64.4%	17.6%	20.2%
		18	175.0	119.4	68.3%	25.0%	-	93.5	77.7	112.6	116.9	-	27.8%	53.8%	6.1%	2.2%
		14	115.0	110.2	95.8%	0.0%	111.0	84.9	73.0	97.0	98.8	-0.7%	29.8%	51.1%	13.6%	11.6%
		12	140.0	136.2	97.3%	0.0%	115.1	118.9	93.7	105.4	108.3	18.3%	14.5%	45.4%	29.2%	25.8%
											Avg	**10.5%**	**16.7%**	**45.2%**	**18.2%**	**15.0%**
											Total Avg	**3.0%**	**22.0%**	**41.5%**	**19.9%**	**18.4%**

Table B.22: Extended results of QN-CO compared with APP, QN-C, RP, GP, and HP on small instances

Appendix C. On-line-off-line QN-CO

In the previous experiments, we investigated the performance of the trained Q-network as a policy. More specifically, for a given Q-network that provides Q values for observation-action pairs, the policy applied the action with the maximum Q value. This section investigates the use of the trained Q-network as a reward-to-go approximation function in an on-line RA. The approach presented in this experiment is similar to the solution method proposed in Ulmer et al. (2017), where an off-line trained value function is embedded into an RA in order to approximate the reward-to-go. In Appendix C.1, the proposed RA is explained in detail. We use instances and trained policies as illustrated in Section 5.1.1. Finally, we compare the performance of the proposed on-line-off-line QN-CO with the purely off-line method in Appendix C.2.

L	U	QN-CO$_L$	QN-CO	GLQ	GHQ	%QN-CO	%GLQ	%GHQ
	L	765.6	799.7	738.3	834.0	-4.25%	3.71%	-8.19%
S	M	737.1	781.0	718.5	794.7	-5.62%	2.60%	-7.24%
	H	722.4	751.0	704.9	760.4	-3.81%	2.48%	-5.00%
	L	1027.0	1054.8	1019.2	1077.5	-2.63%	0.77%	-4.69%
M	M	1017.7	1035.8	1002.8	1067.0	-1.75%	1.49%	-4.62%
	H	1002.8	1023.8	977.1	1043.6	-2.06%	2.63%	-3.91%
	L	1079.0	1078.7	1078.3	1078.3	0.02%	0.06%	0.06%
L	M	1078.2	1080.3	1076.3	1076.9	-0.19%	0.18%	0.12%
	H	1079.8	1081.4	1072.3	1075.3	-0.15%	0.70%	0.42%
					Avg	**-2.3%**	**1.6%**	**-3.7%**

Table B.23: Extended results of QN-CO$_L$

L	U	QN-CO$_U$	QN-CO	GLQ	GHQ	%QN-CO	%GLQ	%GHQ
	L	773.5	799.7	738.3	834.0	-3.3%	4.8%	-7.2%
S	M	760.2	781.0	718.5	794.7	-2.7%	5.8%	-4.3%
	H	745.3	751.0	704.9	760.4	-0.8%	5.7%	-2.0%
	L	1051.7	1054.8	1019.2	1077.5	-0.3%	3.2%	-2.4%
M	M	1038.6	1035.8	1002.8	1067.0	0.3%	3.6%	-2.7%
	H	1024.6	1023.8	977.1	1043.6	0.1%	4.9%	-1.8%
	L	1078.7	1078.7	1078.3	1078.3	0.0%	0.0%	0.0%
L	M	1080.3	1080.3	1076.3	1076.9	0.0%	0.4%	0.3%
	H	1080.3	1081.4	1072.3	1075.3	-0.1%	0.7%	0.5%
					Avg	**-0.7%**	**3.2%**	**-2.2%**

Table B.24: Extended results of QN-CO$_U$

L	U	QN-CO$_{All}$	QN-CO	GLQ	GHQ	%QN-CO	%GLQ	%GHQ
	L	755.9	799.7	738.3	834.0	-5.5%	2.4%	-9.4%
S	M	743.0	781.0	718.5	794.7	-4.9%	3.4%	-6.5%
	H	731.6	751.0	704.9	760.4	-2.6%	3.8%	-3.8%
	L	1026.5	1054.8	1019.2	1077.5	-2.7%	0.7%	-4.7%
M	M	1019.2	1035.8	1002.8	1067.0	-1.6%	1.6%	-4.5%
	H	1009.3	1023.8	977.1	1043.6	-1.4%	3.3%	-3.3%
	L	1078.1	1078.7	1078.3	1078.3	-0.1%	0.0%	0.0%
L	M	1078.7	1080.3	1076.3	1076.9	-0.1%	0.2%	0.2%
	H	1077.9	1081.4	1072.3	1075.3	-0.3%	0.5%	0.2%
					Avg	**-2.1%**	**1.8%**	**-3.5%**

Table B.25: Extended results of QN-CO$_{All}$

Appendix C.1. Proposed Rollout Algorithm

The Rollout Algorithm is an on-line method in the form of forward-dynamic programming that can be viewed as a one-step policy iteration (Bertsekas 2013). Powell (2011) showed that if we roll out a known heuristic policy (the so-called base policy) to approximate reward-to-go values, the resulting policy will be improved over the base policy. In practice, the rollout algorithm estimates the reward-to-go value of an available action at the current decision epoch by simulating the base policy after choosing that action. Interested readers may refer to a survey by Bertsekas (2013) for a detailed discussion about RAs.

Similar to what defined in Equation (19), we let $V^{\pi}(s_k)$ be the expected reward that can be obtained by starting from the state s_k and following the base policy π. Accordingly, we define the rollout policy Π as:

$$\Pi(s_k) = \arg \max_{x_k \in A(s_k)} \left[R(s_k, x_k) + \mathop{\mathbb{E}}_{w \in W} [\gamma V^{\pi}(s_{k+1})] \right]. \tag{C.1}$$

In our case, the base policy is provided by the Q values trained by the QN-CO method in the previous experiment. Therefore, we substitute the approximate value function $V^{\pi}(s_{k+1})$ according to Equation (C.1). Also, by following the MDP-CO formulation, we replace s_k, $A(s_k)$, and $R(s_k, x_k)$ with $o_{k,\bar{v}}$, $A(o_{k,\bar{v}}, \bar{v})$, and $R(o_{k,\bar{v}}, x_k)$, respectively. We rewrite the rollout policy Π as:

$$\Pi(o_{k,\bar{v}}) = \arg \max_{x_k \in A(o_{k,\bar{v}}, \bar{v})} \left[R(o_{k,\bar{v}}, x_k) + \mathop{\mathbb{E}}_{w \in W} [\gamma \max_{x' \in A(o_{k+1,\bar{v}'}, \bar{v}')} Q(o_{k+1,\bar{v}'}, x')] \right], \tag{C.2}$$

where $\bar{v}'$ represents the active vehicle in decision epoch $k+1$.

Note that, since the set of customers is realized at the beginning of the operation, the customer demands are the only source of uncertainty in the equation above. Therefore, the expectation function $\mathbb{E}(.)$ is computed over a finite set of sample scenarios for the realization of the stochastic demands (W). It is important to note that, according to Equation (17), which defines the reward function for QN-CO, the actual reward is realized with a delay. However, it is not feasible to wait to receive the actual reward through on-line methods, such as RA. Therefore, instead of using the reward function as in Equation (17), we define it as follows:

$$R(o_{k,\bar{v}}, x_k) = \begin{cases} \tilde{d}_c & x_k \text{ is a } c \in \mathcal{C}_0 \\ 0 & \text{otherwise} \end{cases}, \tag{C.3}$$

where $\tilde{d}_c$ is the unserved demand of customer c, if its actual demand is already realized; otherwise, it is the expected demand. It is important to note that Equation (C.3), which is defined based on expected demands, is only used to evaluate state-action pairs in the RA. The performance of the proposed method in computational results is measured using realized demands.

The RA proceeds at decision epoch k by simulating the system for each possible action $x_k \in A(s_k, \bar{v})$ for one step and observe the immediate reward $R(s_k, x_k)$ as well as the next observation $o_{k+1,\bar{v}'}$. In the next observation, the maximum Q value of available actions is considered as the approximate reward expected to be obtained onward, denoted as $\bar{f}$. The expected reward-to-go of each action is computed for a set of demand realizations (W). The action with the maximum $\bar{f}$ will be selected as the best action to take (x^*) at decision epoch k.

Given the delayed reward mechanism, we decided to simulate the problem for two steps for each action (instead of the traditional one-step) in our implementation. Therefore, the implemented Equation (C.2) has been modified as:

$$\Pi(o_{k,\bar{v}}) = \arg \max_{x_k \in A(o_{k,\bar{v}}, \bar{v})} \mathop{\mathbb{E}}_{w \in W} [R(o_{k,\bar{v}}, x_k) + \gamma R(o_{k+1,\bar{v}'}, x_{k+1}) + \gamma^2 \max_{x_{k+2} \in A(o_{k+2,\bar{v}''}, \bar{v}'')} Q(o_{k+2,\bar{v}''}, x_{k+2})], \tag{C.4}$$

where x_{k+1} is the action in decision epoch $k+1$ obtained according to the base policy π (i.e., $x_{k+1} = \arg\max_{x' \in A(o_{k+1,\bar{v}'}, \bar{v}')} Q(o_{k+1,\bar{v}'}, x')$) and $\bar{v}''$ is the active vehicle in decision epoch $k+2$. Algorithm 4 illustrates our proposed RA for each decision epoch k, when the active vehicle is $\bar{v}$, the global state is s_k, and W is a finite set of demand realizations.

Algorithm 4: Proposed Rollout Algorithm for decision epoch k

Initialize: set f^* to 0
$o_{k,\bar{v}} \leftarrow O(s_k, \bar{v})$
Construct the action space $A(o_{k,\bar{v}}, \bar{v})$
for $x_k \in A(o_{k,\bar{v}}, \bar{v})$ **do**
  $\bar{f} \leftarrow 0$
  Take action x_k and receive the reward $r_k = R(o_{k,\bar{v}}, x_k)$
  for $w \in W$ **do**
    State transition k to $k+1$:
    $s_{k+1} \leftarrow S^M(s_k, x_k, w)$
    $\bar{v}' \leftarrow \{v \in \mathcal{V} | a_v = \min_{v \in \mathcal{V}} a_v\}$
    $o_{k+1,\bar{v}'} \leftarrow O(s_{k+1}, \bar{v}')$
    Take action $x_{k+1} = \arg\max_{x'} Q(o_{k+1,\bar{v}'}, x')$ and receive $r'_k = R(o_{k+1,\bar{v}'}, x_{k+1})$
    state transition $k+1$ to $k+2$:
    $s_{k+2} \leftarrow S^M(s_{k+1}, x_{k+1}, w)$
    $\bar{v}'' \leftarrow \{v \in \mathcal{V} | a_v = \min_{v \in \mathcal{V}} a_v\}$
    $o_{k+2,\bar{v}''} \leftarrow O(s_{k+2}, \bar{v}'')$
    $f \leftarrow [r_k + \gamma r'_k + \gamma^2 \max_{x_{k+2} \in A(o_{k+2,\bar{v}''}, \bar{v}'')} Q(o_{k+2,\bar{v}''}, x_{k+2})]$
    $\bar{f} \leftarrow \bar{f} + f$
  end
  $\bar{f} \leftarrow \bar{f}/|W|$
  if $\bar{f} > f^*$ **then**
    $x^* \leftarrow x_k$
    $f^* \leftarrow \bar{f}$
  end
end
return x^*

Appendix C.2. Results for On-line-off-line QN-CO

In Section 5.1, we tested each trained policy for 250,000 realizations. However, because the rollout algorithm requires substantial on-line computing, testing across the same realization set would be computationally expensive. Therefore, we test the proposed on-line-off-line QN-CO for each instance $i \in I$, with $I = \mathcal{D} \times \mathcal{Q}$, on $\hat{I}(i)$ with 2,500 realizations (50 customer realizations $\times$ 50 demand realizations). Furthermore, we only focus on harder instances with $\mathcal{D} = \{$Moderate, High, Very High$\}$. For this experiment, we set the size of the sample scenarios for the rollout algorithm ($|W|$) to 50. Table C.26 reports the results

Ψ_{n_z}	Q	RA-QN-CO	QN-CO	%QN-CO
Moderate	25	164.3	161.3	1.9%
	50	202.7	196.9	3.0%
	75	217.1	211.9	2.5%
High	25	358.3	358.2	0.0%
	50	461.0	455.4	1.2%
	75	510.5	501.5	1.8%
Very High	25	551.7	551.7	0.0%
	50	708.7	701.4	1.0%
	75	790.3	776.7	1.8%
			Avg	**1.5%**

Table C.26: Results of on-line-off-line QN-CO compared with off-line QN-CO

of the on-line-off-line QN-CO (RA-QN-CO) and compares them with the performance of the off-line policy (QN-CO). The average improvement of the rollout algorithm over the off-line QN-CO is seen in column %QN-CO. According to the results, the proposed rollout outperforms the off-line version by an average of 1.5% in all cases. The average improvement of the rollout algorithm over the off-line QN-CO is shown in Table C.27. This table helps us to analyze how changes in customer density and vehicle capacity affect %QN-CO. Accordingly, as the customer density decreases from Very High to Moderate, the gap widens significantly. Specifically, the average improvement over the off-line QN-CO in instances with Very High

customer density is 0.9%, whereas this value in instances with Moderate density is 2.4%. It is essential to notice that, as discussed in Section 5.1, the importance of a good routing strategy is magnified in instances with a lower customer density. This finding shows that the on-line-off-line method improves the off-line policy in terms of routing aspects. Furthermore, the sensitivity analysis over the changes in the vehicle capacity also confirms this finding. According to Table C.27, the improvement percentage considerably increases from 0.6% to 2.0% when the vehicle capacity changes from 25 to 75. It is worth to notice that as the vehicle capacity becomes larger, vehicles are allowed to perform longer routes which highlights the necessity of an effective routing strategy. In Section 5.1, we demonstrated that the average improvement over GP

%QN-CO		Q			
		25	50	75	**Avg**
Ψ_{n_z}	Moderate	1.9%	3.0%	2.5%	**2.4%**
	High	0.0%	1.2%	1.8%	**1.0%**
	Very High	0.0%	1.0%	1.8%	**0.9%**
	Avg	**0.6%**	**1.7%**	**2.0%**	**1.5%**

Table C.27: Average improvement of on-line-off-line QN-CO over off-line QN-CO

drops from 11.2% to 7.9%, when the vehicle capacity increases from 25 to 75. However, this experiment shows that the on-line-off-line QN-CO partly compensates for this reduction.

In this experiment, the average computing time per decision epoch for Moderate, High, and Very High customer density is 0.10, 0.17, and 0.21 seconds, respectively. Although, as expected, the computation time grows as the number of customers increases, the on-line-off-line QN-CO can decide in 0.21 seconds, even in the most complex problems. Therefore, the proposed rollout algorithm can also be regarded as an efficient on-line method.

Appendix D. Table of notation

	Notation	Description
Problem identifiers	$\mathcal{V}$	set of vehicles
	$\mathcal{C}_0$	set of customers
	τ_{ij}	travel time between locations i and j
	m	number of vehicles $= \lvert\mathcal{V}\rvert$
	$n, \underline{n}, \bar{n}$	number of realized customers $= \lvert\mathcal{C}_0\rvert$, number of available customers, the average number of customers
	Q	capacity of the vehicle
	L	duration limit of the vehicle
	U	stochastic variability
MDP formulation	$l_c, \bar{d}_c, \hat{d}_c, h_c$	location, expected demand, actual demand, and availability of customer c
	l_v, a_v, q_v	location, arrival time, and the remaining capacity of vehicle v
	k, t_k	decision epoch and the time at decision epoch k
	$s_k, o_{k,\bar{v}}$	state and observation at decision epoch k
	F_k, H_k, G_k	state of target customers, heatmap aggregations, and state of vehicles
	x_k, x_k^v	all action of the vehicles and vehicle v's action at decision epoch k
	$\bar{v}, \bar{\mathcal{V}}_k$	the active vehicle and the set of active vehicles at decision epoch k
	η_v^k	demand served by vehicle v at decision epoch k
	X_k^{-v}	set of actions except for vehicle v
	$J(s_k, v)$	set of reachable customers to vehicle v
	$\tilde{C}_k$	set of customers being served
	w_k, w_k^c	realized demands and realized demand for customer c
	$p \in P, \mathcal{C}_p$	heatmap partitions and the set of customer in that partition
	$\mathcal{C}_{\bar{v}}$	set of target customers for vehicle $\bar{v}$
	$\tilde{n}$	hyper parameter to bound $\lvert\mathcal{C}_{\bar{v}}\rvert$
	$\rho_{c,\bar{v}}$	equals to $\frac{min(\bar{d}_c, q_{\bar{v}})}{\tau_{l_c, l_{\bar{v}}}}$ for customer c and vehicle $\bar{v}$
Q-learning	$\delta, \beta_d, \beta_t, \epsilon, \alpha$	hyper parameters for huber loss, target replacement, train probability, epsilon greedy, learning rate
	$\theta, \bar{\theta}$	primary and target neural networks to predict Q values
	Δ, Φ	temporal difference and loss function to train neural networks
	$B, \tilde{B}$	list and sample list of experiences
Instance generation	$Z, \mathcal{Z}$	a zone and set of zones to generate instances
	$I, \mathcal{I}$	an instance and the set of instances
	$\psi_{n_z}, \psi_l, \psi_{\bar{d}}, \psi_{\hat{d}}$	distribution functions for the number of customers in each zone, their coordinates, their expected demand, and their actual demand
	$\mathcal{L}$	set of duration limits
	$\mathcal{U}$	set of stochastic variabilities
	$\mathcal{Q}$	set of vehicle capacities

Table D.28: Table of notation