

# Reinforcement Learning Approaches for the Orienteering Problem with Stochastic and Dynamic Release Dates

Yuanyuan Li, Claudia Archetti, Ivana Ljubić\*

IDS Department, ESSEC Business School, Cergy, France,  
 yuanyuan.li@essec.edu archetti.archetti@essec.edu ivana.ljubic@essec.edu

In this paper, we study a sequential decision making problem faced by e-commerce carriers related to when to send out a vehicle from the central depot to serve customer requests, and in which order to provide the service, under the assumption that the time at which parcels arrive at the depot is stochastic and dynamic. The objective is to maximize the number of parcels that can be delivered during the service hours. We propose two reinforcement learning approaches for solving this problem, one based on a policy function approximation (PFA) and the second on a value function approximation (VFA). Both methods are combined with a look-ahead strategy, in which future release dates are sampled in a Monte-Carlo fashion and a tailored batch approach is used to approximate the value of future states. Our PFA and VFA make a good use of branch-and-cut-based exact methods to improve the quality of decisions. We also establish sufficient conditions for partial characterization of optimal policy and integrate them into PFA/VFA. In an empirical study based on 720 benchmark instances, we conduct a competitive analysis using upper bounds with perfect information and we show that PFA and VFA greatly outperform two alternative myopic approaches. Overall, PFA provides best solutions, while VFA (which benefits from a two-stage stochastic optimization model) achieves a better tradeoff between solution quality and computing time.

*Key words:* Reinforcement Learning, Two-Stage Stochastic Optimization, Branch-and-Cut, Markov

Decision Process, Orienteering Problem

---

\* Corresponding author

## 1. Introduction

E-commerce markets are booming at remarkable rates. Due to an unprecedented series of lockdowns, billions of people stay at home to prevent the spread of the virus. It is reported that the e-commerce revenue saw a 10% increase in Europe in 2020 due to the pandemic <sup>1</sup>. At the same time, the expectations of customers in terms of service quality are also increasing. When questioned about the features they would like to obtain from the delivery of online purchases, almost half (48%) of shoppers mentioned speed, 42% reduction of the cost of delivery and 41.6% more reliable information about delivery time<sup>2</sup>. By providing an evidence of the clash between SF Express and Alibaba, Cui et al. (2020) show that delivering expensive or less-discounted products in a reliable and timely manner brings better customer experience and thus higher profits. When it comes to the specific challenges in improving customers' satisfactions on delivery, one of the crucial features is related to the last-mile delivery leg (Archetti and Bertazzi (2021)). An important feature of last-mile distribution is that its operations start as soon as parcels are delivered to the final logistic center, typically a city distribution center (CDC). Given the short delivery times requested by the customers, delivery operations typically need to start before all parcels expected to arrive during the day are available at the CDC. This raises a question: should the dispatcher wait for more or all parcels to be delivered at the CDC or should he/she start the delivery as soon as there is any available parcel and vehicle?

In this paper, we focus on the sequential decision making problem related to when to deliver parcels and which parcels to deliver under the assumption that the time at which parcels become available at the CDC (called *release dates*) is stochastic and dynamic. We introduce a new problem called the *Orienteering Problem with Stochastic and Dynamic Release Dates* (DOP-rd). We assume that a single and uncapacitated vehicle is serving customer requests with no time window, so that the vehicle should finish its service before the deadline  $T_E$ . The release dates are considered to be uncertain, stochastically and dynamically updated during the sequential decision making process. The objective is to maximize the number of requests served within  $T_E$ .

This paper presents the following main contribution to the literature:

1. The DOP-rd shares similarities with the problem studied in Archetti et al. (2020). The main differences rely on the objective function and deadline. In fact, in Archetti et al. (2020) the objective is minimizing the travel time while ensuring that all parcels are delivered, and no deadline is considered. Instead, in the DOP-rd the objective is maximizing the number of parcels delivered within the deadline. Recent studies highlight the practical importance of focusing on the maximum number of delivered parcels (Voccia et al. (2019), Hong et al. (2018), Ulmer and Thomas (2018), Behrend and Meisel (2018)), as typically, the vehicles have to return to the depot during the working hours and some of the parcels may be left to be distributed the next day or by a third-party carrier. This is a fundamental change in the problem setting, that allows us to develop ad-hoc procedures, like the one for approximating the value of future states presented in Section 5.1.
2. We model the problem as a Markov Decision Process (MDP) and we propose a method to approximate the value of future states based on Monte-Carlo simulation (for scenario generation), and a batch approach that uses continuous approximation from Daganzo (1984) to estimate the length of future routes. At each decision epoch, the updated information is embedded both in scenario generation and in the approximation approach, thus leading to a reinforcement learning approach.
3. We propose two novel hybrid approaches to derive a decision-making policy in the (approximated) decision space. The first is a policy function approximation through a consensus function (PFA) while the second is a look-ahead policy with value function approximation (VFA). The former one relies on exact solutions of a series of deterministic Integer Linear Programs (ILPs), the latter one uses a two-stage stochastic programming ILP model instead. These ILPs are solved using state-of-the-art branch-and-cut techniques. The obtained solutions are then used to decide when to leave the depot and which customers to serve, while maximizing the expected number of parcels delivered. In addition, we propose a partial characterization of optimal policy, the conditions of which are checked before applying PFA/VFA.

4. In an extensive computational study, we compare PFA and VFA against two myopic benchmark approaches. We also provide results of a competitive analysis, in which the upper bounds are calculated by solving to optimality the problem under assumption of having perfect information.
5. The results show the benefit of implementing partial characterization of optimal policy for both approaches. Also, PFA outperforms other methods in solution quality while VFA achieves a better tradeoff between solution quality and computing time.

Most of the existing literature dealing with MDP relies on heuristic look-ahead policies. In this work, we approximate the future states in each decision epoch, and employ exact methods to determine the best action associated with the projection of future states. The aim of this work is to empirically demonstrate the advantage of embedding exact methods inside reinforcement learning framework, to improve sequential decision making processes.

The paper is organized as follows. Section 2 presents a literature review by focusing on the differences with the current work. Section 3 contains the problem description and formulation as a Markov Decision Process. Section 4 presents upper bounds to solution values and a partial characterization of the optimal policy. The solution approaches are introduced in Section 5 while Section 6 is devoted to computational experiments and results. Finally, we draw conclusions in Section 7 and present more detailed results in Appendix 8.

## 2. Literature Review

We first focus on the literature related to routing problems with stochastic and dynamic features. Then, we move to problems with release dates and we highlight differences in problem settings and solution approaches compared to the current work.

The literature on stochastic and dynamic routing problems is wide. We focus on pioneering papers and surveys. Psaraftis (1988) introduce the dynamic traveling salesman problem with the dynamics lying on the customer demands. Bertsimas and Van Ryzin (1991) generalize the contribution of the former paper and propose a mathematical model for the stochastic and dynamic vehicle

routing problems (VRP). Other interesting extensions of stochastic or dynamic settings appear in Savelsbergh and Sol (1998), Bent and Van Hentenryck (2004), Secomandi (2001), Secomandi and Margot (2009), Goodson et al. (2013), Subramanyam et al. (2021). A recent contribution (Liu et al. (2021)) captures the uncertainty of service time in the last-mile delivery services: with the aim of improving the on-time delivery performance, a framework integrating travel-time predictors with order-assignment optimization is proposed, which aims at improving the on-time delivery performance. As for surveys on stochastic and dynamic VRPs, we refer to Ritzinger et al. (2016), Oyola et al. (2018), Soeffker et al. (2022).

The previous studies mainly focus on the uncertainties in terms of demands and travel time. The works on other stochastic and dynamic aspects such as release dates appeared just recently. The routing problem with release date was firstly introduced in Cattaruzza et al. (2016), where the authors define it as a problem where each parcel is associated with a release date indicating the time at which the parcel is available at depot. Release dates are supposed to be known beforehand. The problem is a multi trip vehicle routing problem with release dates and the authors solve it through a population-based algorithm. In Archetti et al. (2015) the Traveling Salesman Problem with release dates (TSP-rd) is considered, i.e., the problem with a single uncapacitated vehicle. The authors analyze the complexity of the problem on two variants - minimizing the total distance traveled with a constraint on the maximum deadline and minimizing the total time needed to complete the distribution without deadline. Later, Reyes et al. (2018) extend the results found in Archetti et al. (2015) by considering service guarantees on customer-specific delivery deadlines. Shelbourne et al. (2017) consider the VRP with release and due dates where a due date indicates the latest time the order should be delivered to the customer. The objective function is defined as a convex combination of the operational costs and customer service level. The authors propose a path-relinking algorithm to address the problem. More recently, Archetti et al. (2018) propose a mathematical programming formulation and present a heuristic approach based on an iterated local search (ILS) for solving the TSP-rd.

All contributions mentioned above study routing problems with deterministic release dates. Recently, there have been several studies related to applications in same-day delivery services, where release dates are stochastic. Among them, Voccia et al. (2019) study a multi-vehicle dynamic pickup and delivery problem with time constraints where requests arrive dynamically from online customers. Each request is associated with a deadline or a time window. The objective is to maximize the expected number of requests that can be delivered on time. The authors first model the problem as an MDP, then, based on sample-scenario planning, they propose a heuristic solution approach. Rather than defining delivery windows at the receiver side, Van Heeswijk et al. (2019) consider dispatch windows at the consolidation center. As in Voccia et al. (2019), an MDP is defined. To overcome “the three curses of dimensionality”, the authors propose an approximated dynamic programming (ADP) algorithm including an ILP. In contrast, Archetti et al. (2020) study a single-vehicle problem with no deadline and where the objective is to deliver all parcels as fast as possible. They propose a reoptimization approach which heuristically reduces the number of decision epochs, furthermore, two models are defined to select an action at each reoptimization epoch. While both models aim to minimize the total completion time, the main difference between the two is that the first model is a stochastic mixed-integer program model which considers the full stochastic information on the release dates while the second one is a deterministic model which uses a point estimation for the stochastic release dates.

Based on the work of Archetti et al. (2020), our paper studies the DOP-rd considering a single and uncapacitated vehicle serving customer requests with no time window. We assume that the vehicle should finish its service before the deadline  $T_E$  and our objective is to maximize the number of requests served within  $T_E$ . This is consistent with recent works studying applications in same-day delivery services (see, e.g., Voccia et al. (2019)). The main differences between the current work and Voccia et al. (2019) are the following: in the latter paper, customer locations are unknown and multiple vehicles are available. As for solution approaches, the ones proposed in the current work are innovative not only in the way future information is approximated, but also on how policies are derived through VFA and PFA and in the way the ILP models are constructed.

Another contribution which is very close to the current study is Van Heeswijk et al. (2019). Again, the problem is formulated as an MDP but decision epochs are pre-defined time instants separated by equidistant intervals. A similar assumption is made in Klapp et al. (2018) where these equidistant time intervals are called ‘waves’. The objective function is to minimize the expected vehicle operating costs and penalties for unserved requests. The main differences between the current work and the one of Van Heeswijk et al. (2019) is the fact that in the latter paper multiple vehicles are considered as well as a third party service provider. Also, Daganzo’s formula is used not only to estimate the tour length but also the number of vehicles required to perform the service. In our work instead we consider a single vehicle and we use the Daganzo’s formula for inferring how many requests can be fulfilled in future states at each decision epoch.

Finally, another close work is paper Schrottenboer et al. (2021), where the authors study the problem of balancing the consolidation potential and delivery urgency of orders. The problem is a pickup-and-delivery problem with a fleet of vehicles and the objective is to maximize the expected customer satisfaction. The authors propose a cost function approximation approach.

We summarize the major characteristics of the contributions on routing problems with release dates in Table 7 in Appendix 8.2.

### 3. Problem Description

The DOP-rd is defined as follows. A company distributes parcels to its customers from a CDC. Parcels are delivered to the CDC by suppliers throughout the day, i.e., eventually after the moment in which the distribution to customers starts. The moment in which parcels are delivered to the CDC is unknown. In case suppliers’ vehicles are equipped with GPS, the information about their arrivals to the CDC (also called depot in the following) is constantly updated. The company has a single uncapacitated vehicle to perform the service. The goal is to serve as many parcels as possible within a given deadline.

More formally, let  $G = (V, A)$  be a complete graph. The set of vertices  $V$  includes vertex 0, representing the depot, and the set  $N$  of customers. Each customer is associated with an arrival

time of its parcel to the depot, which is called release date, and a delivery location. Packages arrive throughout the day until a predefined deadline  $T_E$ , which also corresponds to the time at which the vehicle has to be back to the depot and to finish the service. Release dates are stochastic and dynamically updated throughout the day. Specifically, the stochastic aspect simulates the situation where suppliers' vehicles carrying the parcels to the CDC might encounter unpredicted events. The dynamic aspect is related to the fact that the information about the arrival of the suppliers' vehicles to the CDC evolves over time, as in the case of GPS equipped vehicles. Each arc  $(i, j) \in A$  is associated with a traveling time  $d_{ij} > 0$  and we assume that the triangle inequality holds. A single uncapacitated vehicle is available to perform the deliveries. We define a route as a tour starting and ending at the depot and visiting customers in between. The set of routes is denoted as  $K = \{1, \dots, |K|\}$ . The goal is to maximize the number of requests served within  $T_E$ .

We now formalize the problem as a Markov Decision Process (MDP). Note that the detailed notation used along the paper is summarized in Table 5 in Appendix 8.1.

### 3.1. The problem as a Markov Decision Process

At each decision epoch  $e$ , we distinguish between customers that have already been served (i.e., their parcels have been delivered),  $N_e^{served}$ , and those that are still unserved,  $N_e^{unserved}$ , where  $N_e^{served} \cup N_e^{unserved} = N$ . There are two categories of unserved customers based on the information available for the release date at decision epoch  $e$ . If the parcel has arrived at the depot, the release date is known and the customer can be served, thus we call this set of customers  $N_e^{known}$ . On the contrary,  $N_e^{unknown}$  denotes the set of customers whose parcels are still to be delivered to the depot. In turn, there are two subsets of  $N_e^{unknown}$ : customers whose information about the release date is not updated until the parcel arrives at the depot are denoted as  $N_e^{static}$ ; customers whose information about release date is updated thanks to GPS equipped vehicles are denoted as  $N_e^{dynamic}$ . Note that  $N_e^{static} \cup N_e^{dynamic} = N_e^{unknown}$ . Finally, we assume that the suppliers' vehicles deliver the parcels to the depot independently from one another, i.e., the release dates are independent. Let  $\tilde{r}_i^e$  be the random variable associated with the release date of customer  $i \in N_e^{unserved}$  at decision epoch  $e$ . This variable is defined as follows:



- for  $i \in N_e^{known}$ ,  $\tilde{r}_i^e = r_i$ , where  $r_i$  is the actual arrival time of the parcel of customer  $i$ .
- for  $i \in N_e^{static}$ , we have  $\tilde{r}_i^e = \tilde{r}_i^0$ , which means that the distribution of the random variable associated with the release date is estimated at time 0 and never updated till the parcel arrives at the depot.
- for  $i \in N_e^{dynamic}$ ,  $\tilde{r}_i^e$  is a random variable representing release date at decision epoch  $e$ . It is dynamically updated throughout the day.

At each decision epoch  $e$ , the company has to decide either to wait for more parcels to arrive or to dispatch the vehicle delivering a subset of the parcels available at the depot. No preemptive return is allowed, i.e., the vehicle comes back to the depot only once all parcels on board are delivered to the corresponding customers.

The MDP components are the following:

- *Decision epochs*: A decision epoch is denoted as  $e \in \{0, \dots, E\}$  and the time associated with it is  $t_e$ . Note that  $T_E$  represents the deadline, i.e., the time at which the vehicle has to be back to the depot and no further deliveries are allowed. The first decision epoch 0 corresponds to the beginning of the delivery operations. At each decision epoch an action is made based on the current information. A decision epoch corresponds to either the arrival of a package at the depot or to the time when the vehicle is back to the depot at the end of a route. If a parcel arrives but the vehicle is out for delivery, or if the vehicle is at the depot but there is no parcel available, the corresponding decision epoch is skipped to the next. To avoid waiting for too long at the depot without having any new arrival of packages, additional decision epochs are defined at equally distant time intervals  $\phi$ , when the action is evaluated according to the updated stochastic information.

- *States*: The state  $S_e = (t_e, N_e^{known}, \{\tilde{r}_i^e\}_{i \in N_e^{unknown}})$  includes all information for evaluating an action at decision epoch  $e$ , i.e., the time of the decision epoch  $t_e$ , the set  $N_e^{known}$  of customers with known release dates (whose parcels have arrived), and the stochastic information of future release dates  $\{\tilde{r}_i^e\}_{i \in N_e^{unknown}}$ . At decision epoch 0, no customer has been served.

- *Actions*: At each decision epoch  $e$ , the company has to decide to either dispatch a subset of accumulated orders or to wait at the depot for new parcels to arrive. The decision involves action

$\mathcal{X}_e$ . Similarly to what has been done in Archetti et al. (2020), an action consists of a new route serving a subset of the unserved customers if dispatching is decided. However, while in Archetti et al. (2020) customers served in the new route could belong to both sets  $N_e^{known}$  and  $N_e^{unknown}$ , we instead reduce the set of customers to be potentially included in the new route to  $N_e^{known}$ . This implies that the new route, in case at least one customer is included, will leave immediately from the depot, contrary to Archetti et al. (2020) where the vehicle might have to wait at the depot in case the route includes at least one customer in  $N_e^{unknown}$ . Formally, the action space at decision epoch  $e$ , denoted by  $X(S_e)$ , includes all possible subsets of  $N_e^{known}$ , each one associated with the TSP route serving all parcels in the subset. The set of customer locations visited in the route associated with the action  $\mathcal{X}_e$  is denoted as  $l(\mathcal{X}_e)$ . This set is empty if the route is empty, i.e., the vehicle is not dispatched and the action consists in waiting at the depot.

- *Transitions:* After action  $\mathcal{X}_e$ , a transition is made from state  $S_e$  to a new state  $S_{e+1}$ . The state  $S_{e+1}$  is described as  $S_{e+1} = (t_{e+1}, N_{e+1}^{known}, \{\tilde{r}_i^{e+1}\}_{i \in N_{e+1}^{unknown}})$ . We have  $t_{e+1} = t_e + t_{route}(\mathcal{X}_e)$  if route is not empty,  $t_{e+1} = \min(t_p, t_e + \phi)$  otherwise, where  $t_{route}(\mathcal{X}_e)$  is the time associated with the route corresponding to action  $\mathcal{X}_e$ ,  $t_p$  is the earliest time when a new parcel arrives while the vehicle is at the depot, and  $\phi$  is the maximum waiting time between two decision epochs as defined earlier. Let  $N_{e+1}^{new}$  be customers whose parcels arrived at the depot between decision epochs  $e$  and  $e+1$ . Correspondingly, the set of customers are updated as  $N_{e+1}^{unknown} := N_e^{unknown} \setminus N_{e+1}^{new}$  and  $N_{e+1}^{known} := N_e^{known} \setminus l(\mathcal{X}_e) \cup N_{e+1}^{new}$ . We assume parcels arrive sequentially, thus  $N_{e+1}^{new}$  is either singleton or empty.

- *Objective:* The objective is to maximize the number of requests served within  $T_E$ . Each action  $\mathcal{X}_e$  chosen when the system is in state  $S_e$ , is guided by a decision rule and brings an *immediate reward*, expressing the number of requests served by the route dispatched at time  $t_e$ , and a future reward. A sequence of decision rules is defined as a *policy* and the optimal policy is the one that maximizes the expected reward at epoch  $E$ .

In decision epoch  $e$ , the goal is to maximize the number of requests served within the time interval  $[t_e, T_E]$ . The value function is formulated as the following recursive formula:

$$V_e(S_e) = \max_{\mathcal{X}_e \in X(S_e)} \{C(S_e, \mathcal{X}_e) + \gamma \mathbb{E}[V_{e+1}(S_{e+1}) \mid (S_e, \mathcal{X}_e)]\}. \quad (1)$$

Here,  $C(S_e, \mathcal{X}_e)$  denotes the immediate reward and  $\gamma \mathbb{E}\{V_{e+1}(S_{e+1}) \mid (S_e, \mathcal{X}_e)\}$  is the expected value associated with decision epoch  $e + 1$ , while  $\gamma$  is a discount factor.

The system suffers from the curse of dimensionality in the number of states, stages and actions. In the recent literature, reinforcement learning is frequently used to deal with this issue (see e.g, Powell (2009) and Van Heeswijk et al. (2019)). In this work, we propose two reinforcement learning approaches for solving the DOP-rd, one based on a policy function approximation and the second on a value function approximation. Both methods are hybridized with a look-ahead strategy, in which future release dates are sampled in a Monte-Carlo fashion. The two approaches, who make a good use of branch-and-cut-based exact methods to improve the quality of decisions, are described in detail in Section 5.

## 4. Upper Bounds and Partial Characterization of Optimal Policy

In the following, we first discuss valid upper bounds which are useful for competitive analysis. We then provide sufficient conditions under which it is optimal for the vehicle to leave the depot immediately in order to serve a subset of available parcels.

### 4.1. Upper Bounds

A first trivial upper bound for the DOP-rd can be calculated by solving an Orienteering Problem (OP) assuming that all unserved parcels are available at the depot (see Section 8.3 of the Appendix for further details). In the following, we instead describe a stronger upper bound obtained a-posteriori, when all information is known.

*Upper Bound with Perfect Information:* A tighter upper bound for the DOP-rd can be computed in a *wait-and-see* fashion, using the perfect information with respect to realized release dates. Indeed, in this case, the optimal solution is the solution of the OP with Release Dates (OP-rd), a deterministic counterpart of the problem studied in this article.

Let us assume that all release dates,  $r_i \geq 0$ ,  $i \in N$  are known. Then, the goal of OP-rd is to find the maximum number of parcels that can be served in the interval  $[0, T_E]$ . A solution consists of

up to  $|\mathcal{K}|$  trips performed by the vehicle, so that the end time of the last trip does not exceed  $T_E$ . The problem is NP-hard, since the OP can be reduced in polynomial time to it. We can model the problem as an ILP making use of the following decision variables:

- $s_i^k$ : binary variable indicating whether parcel  $i$  is served in trip  $k$ ;
- $\xi_{ij}^k$ : binary variable equal to 1 in case arc  $(i, j)$  is traversed in trip  $k$ ;
- $d_k$ : continuous variable representing the starting time of trip  $k$ .

The ILP formulation for the OP-rd is then:

$$\max \sum_{i \in N} \sum_{k \in \mathcal{K}} s_i^k \quad (2a)$$

subject to:

$$\sum_{(i,j) \in A} \xi_{ij}^k = \sum_{(j,i) \in A} \xi_{ji}^k = s_i^k \quad i \in V, k \in \mathcal{K} \quad (2b)$$

$$\sum_{i,j \in S} \xi_{ij}^k \leq \sum_{i \in S \setminus \ell} s_i^k \quad S \subseteq N, |S| \geq 2, \ell \in S, k \in \mathcal{K} \quad (2c)$$

$$\sum_{k \in \mathcal{K}} s_i^k \leq 1 \quad i \in N \quad (2d)$$

$$d_k \geq r_i s_i^k \quad i \in N, k \in \mathcal{K} \quad (2e)$$

$$d_{k+1} = d_k + \sum_{(i,j) \in A} d_{ij} \xi_{ij}^k \quad k \in \mathcal{K} \quad (2f)$$

$$d_{|\mathcal{K}|} = T_E \quad (2g)$$

$$\xi_{ij}^k, s_i^k \in \{0, 1\} \quad i, j \in V, k \in \mathcal{K} \quad (2h)$$

where (2b) are degree constraints, (2c) are the generalized subtour elimination constraints and (2d) fix the number of visits to a customer to be at most one. Constraints (2e)–(2g) determine the starting time of each trip. Specifically, (2e) establish that a trip cannot depart prior to the release date of each customer to be visited, (2f) fix the starting time of a trip as the ending time of the preceding trip and (2g) set the ending time of the last trip equal to  $T_E$ . Note that constraint (2g), as well as the equality in constraint (2f), is due to the fact that waiting time can be shifted to the beginning of the distribution without affecting the structure and the value of the solution,

thus avoiding any waiting time between consecutive trips and at the end, as shown in Archetti et al. (2018). We also add symmetry breaking constraints (2j) and constraints (2i) to strengthen the formulation:

$$\sum_{i \in N} s_i^k \leq |N|s_0^k \quad k \in \mathcal{K} \quad (2i)$$

$$s_0^k \leq s_0^{k+1} \quad k \in \mathcal{K}, k \neq |\mathcal{K}| - 1. \quad (2j)$$

The optimal solution of (2) provides an upper bound to the DOP-rd. In case the optimal solution is not found within a given computation time limit, the upper bound at the end of computation (rounded down to the nearest integer) provides a valid upper bound for the DOP-rd. The model is solved by means of a branch-and-cut algorithm, in which the generalized subtour elimination constraints are separated on the fly. For more details regarding the separation of these constraints, see, e.g., Lucena (1993), Lucena and Resende (2004), Ljubić (2021).

Note that formulation (2) needs an upper bound on the number of trips  $|\mathcal{K}|$  as input. A trivial upper bound is  $|\mathcal{K}| = |N|$ , which is the one used in Archetti et al. (2018). However, a tighter value can be obtained by solving the following integer linear programming problem. Let us order customers  $i \in N$  in increasing order according to the value of release dates (ties are broken arbitrarily). We define  $\alpha_i$  as a binary variable equal to 1 if a direct trip, from the depot to customer  $i$  and back, is performed,  $i \in N$ . A valid upper bound on the number of trips is given by the optimal solution of:

$$\max \sum_{i \in N} \alpha_i \quad (3a)$$

subject to:

$$\alpha_i r_i + \sum_{j=i}^{j=|N|} (d_{0j} + d_{j0}) \alpha_j \leq T_E \quad i \in N \quad (3b)$$

$$\alpha_i \in \{0, 1\} \quad i \in N \quad (3c)$$

Formulation (3) counts the maximum number of direct trips that can be performed within  $T_E$ . Let us call  $UB_{trip}$  the optimal solution of (3).

PROPOSITION 1. Given a vector of deterministic release dates  $r_i$  and deadline  $T_E$ :

1.  $UB_{trip}$  is a valid upper bound on the maximum number of trips  $|\mathcal{K}|$  for the OP-rd;
2. Any optimal solution of (3) is a feasible solution for (2).

*Proof.* See Section 8.4 in the Appendix.

## 4.2. Partial Characterization of Optimal Policy

The following proposition provides sufficient conditions to partially characterize an optimal policy.

PROPOSITION 2.

Let  $\ell$  be the largest number of parcels from  $N_e^{unserved}$  that can be delivered within  $[t_e, T_E]$ . If there exists a subset  $N_{sol} \subseteq N_e^{known}$  of parcels that can be delivered within  $[t_e, T_E]$  such that  $|N_{sol}| = \ell$ , then the optimal policy is to leave the depot immediately and distribute the parcels from  $N_{sol}$ .

*Proof.* Trivial.

Hence, Proposition 2 provides sufficient conditions under which it is optimal to leave the depot and (partially) distribute the available parcels. Checking these conditions requires solving an NP-hard optimization problem. Indeed, given a decision epoch  $e$ , and the set of parcels from  $N_e^{unserved}$ , one has to find an optimal solution of the OP with respect to  $N_e^{unserved}$  with maximum route duration equal to  $T_E - t_e$  and with all parcel profits set to one. In case the value of the optimal solution is  $\ell > |N_e^{known}|$ , then it immediately follows that the condition of Proposition 2 is not satisfied. Otherwise, one has to solve a new OP on the set  $N_e^{known}$  and the condition of Proposition 2 holds if and only if the value  $\ell$  of the optimal solution remains unchanged. Note that, in preliminary experiments, we also implemented a different approach for verifying Proposition 2, where a single OP was solved on  $N_e^{unserved}$  where profits were set to 1 for parcels in  $N_e^{unknown}$ , and to  $1 - \epsilon$  for parcels in  $N_e^{known}$ , with a sufficiently small value of  $\epsilon$ . However, the former approach turned out to be much more efficient than the latter one.

Finally, we note that, for small values of  $|N_e^{known}|$  and large values of  $[t_e, T_E]$ , it is unlikely that Proposition 2 holds and in this case we skip checking whether the underlying conditions hold (see the Section 6 for more implementation details).

## 5. Reinforcement Learning Solution Approaches

The solution approaches are based on Monte-Carlo simulation, where scenario generation is performed at each decision epoch  $e$  according to the state  $S_e$ , and on an approximation of the value of future states, described in Section 5.1. Thus, an on-line approximation scheme is developed, which learns from updated information.

According to formula (1), our goal is to find a policy  $\mathcal{X}_e$  that maximizes the sum of the immediate award  $C(S_e, \mathcal{X}_e)$ , corresponding to the number of parcels delivered in the “route 0” (route that starts now), and the expected number of parcels delivered in future states. To anticipate the value of future states, at each decision epoch  $e$ , we generate a set of scenarios  $\Omega$  associated with unserved customers predicting their release dates  $\tilde{r}_e$ . Then, we approximate the value function of the future states using a batch approach described in Section 5.1. This approximation is used in both approaches determining the decision making policy, which are:

- *Policy Function Approximation through a Consensus Function (PFA)*: for each scenario, we run a deterministic ILP model to determine the set of known requests that should be served with a route starting immediately, and the set of requests included in future routes. Then, a consensus function based on the solutions obtained over all scenarios defines the set of known requests that are served immediately (if any), i.e., with a route leaving from the depot at time  $t_e$ .
- *Look-Ahead Policy with Value Function Approximation (VFA)*: instead of looking at each scenario independently, we now build a two-stage stochastic ILP model in which the first stage consists in determining a route containing requests to deliver at time  $t_e$ , while the second stage concerns the expected rewards obtained in the future states by integrating all scenarios with equal probability.

The two approaches are described in Sections 5.2 and 5.3, respectively (an overview of additional notation used is provided in Table 6 in Appendix 8.1). Both ILP methods are based on state-of-the-art branch-and-cut techniques. Note that, in both approaches, Proposition 2 is checked at each decision epoch (before solving the corresponding ILP) and the procedure is stopped in case it is satisfied, as the best policy is identified by the OP solution defined by the proposition.

### 5.1. Approximation of the Value Function of Future States

At each decision epoch  $e$ , we are given the sets of known customers  $N_e^{known}$  and unknown customers  $N_e^{unknown}$ . As for  $N_e^{unknown}$ , we generate a set of scenarios  $\Omega$  where a scenario  $\omega \in \Omega$  represents a possible realization of the expected release dates for  $i \in N_e^{unknown}$ , sampled according to the distribution associated with  $\tilde{r}_i^e$ . Assume that we are given a scenario  $\omega \in \Omega$ . We propose to determine the route serving a subset of requests from  $N_e^{known}$  while approximating the rewards obtained from serving the remaining ones from  $N_e^{unserved}$  until the deadline. As for the approximation of future routes serving customers in  $N_e^{unserved}$ , we use a batch approach that works as follows.

First, we assume that each future route will serve at most  $\rho$  requests. This will allow us to easily obtain an approximation of the duration of future routes. Also, it is reasonable to assume that in practical applications the company will not wait for too long (i.e., it will not wait for more than  $\rho$  parcels becoming available at the depot) before dispatching the vehicle. Thus, by a proper tuning of the value of  $\rho$  (as shown in Section 8.7), one might obtain a good approximation of the duration of future routes. We apply the formula of Daganzo (1984) to obtain an estimation of the duration of each future route. This formula is commonly used in the literature thanks to its simplicity and accuracy (Sankaran and Wood (2007), Jabali et al. (2012), Van Heeswijk et al. (2019)). According to the formula, given the Euclidean area  $\mathcal{A}$  containing the locations of unserved customers and  $\rho$ , under the assumption that the locations are scattered uniformly and independently, the expected tour duration  $T_D$  is  $0.75\sqrt{\mathcal{A}\rho}$ .

The input to the batch approach is given by the duration of a batch tour  $T_D$  (calculated through the Daganzo's formula mentioned above), the distribution of release dates, and  $\rho$ , the maximum number of requests served in each future route (represented by a batch). All these parameters, together with  $|\Omega|$  (the number of scenarios sampled to approximate the future release dates), are referred to as  $\theta$  in the following. Then, for each scenario  $\omega \in \Omega$ , the batch algorithm works as described in the following. For the ease of reading, we omit the index  $\omega$  in the notations.

The algorithm starts by ordering the unserved requests according to their release dates associated with the realizations of scenario  $\omega$ . Recall that  $\tilde{r}_i^e$  are random variables representing the release



dates for  $i \in N_e^{unserved}$ . For a given scenario  $\omega$  we will denote by  $r_i^e$  the realization of expected release dates (ERDs) for  $i \in N_e^{unknown}$  and actual release dates for  $i \in N_e^{known}$ . The algorithm then divides requests into batches. To serve as many requests as possible, the last route (or batch) is supposed to serve the requests arriving at the latest that can still be feasibly served before the deadline. Thus, the selection is done backwards, i.e., starting from the request with the latest release date, and checking whether it can be included in the last route, i.e., if its release date is not larger than the starting time of the last route, which is  $T_E - T_D$ . Then, the algorithm proceeds iteratively this way until  $\rho$  requests are included in the last route. After that, a new route is created (the second last) associated with a starting time equal to  $T_E - 2T_D$ , and so on. Note that, as the algorithm works backwards, we refer to the last route as route 1 (and so on) in the following. In order to provide a detailed description of the algorithm, the following notation is introduced:

- $K$ : the set of indices for all the batches created;
- $K_0$ : the set of indices for batches with positive spare capacities ( $K_0 \subseteq K$ ). We say that a batch  $k$  has a positive spare capacity if the number of requests in  $N_e^{unknown}$  assigned to  $k$  is less than  $\rho$ ;
- $k(i)$ : the index of the batch request  $i$  is assigned to,  $i \in N_e^{unknown}$ ;
- $\tau_{start}^k$ : the starting time of the route serving batch  $k \in K$ ;
- $\tau_{end}^k$ : the ending time of the route serving batch  $k \in K$ .
- $\rho_k$ : the number of requests in  $N_e^{unknown}$  assigned to the route serving batch  $k \in K$ .

The scheme of the batch algorithm is presented in Algorithm 1. Note that in Algorithm 1, customers in  $N_e^{known}$  are treated differently than those in  $N_e^{unknown}$ . For each customer  $i \in N_e^{unknown}$  such that  $r_i^e \leq T_E - T_D$  the algorithm determines the unique batch  $k(i) \in K$  to which  $i$  is assigned. However, the customers in  $N_e^{known}$  can be assigned to any potential batch  $k \in K$  which has some spare capacities, i.e., which contains less than  $\rho$  unknown requests. The subsets of the batches with spare capacities are denoted by  $K_0$  and a unique assignment of a request  $i \in N_e^{known}$  to some  $k \in K_0$  is obtained through the optimization models proposed in Sections 5.2 and 5.3.

**Algorithm 1** The batch algorithm

**Require:** Time  $t_e$ ; Unserved requests  $N_e^{unserved} = N_e^{known} \cup N_e^{unknown}$ ; Release dates  $\{r_i^e, i \in N_e^{unserved}\}$ ;

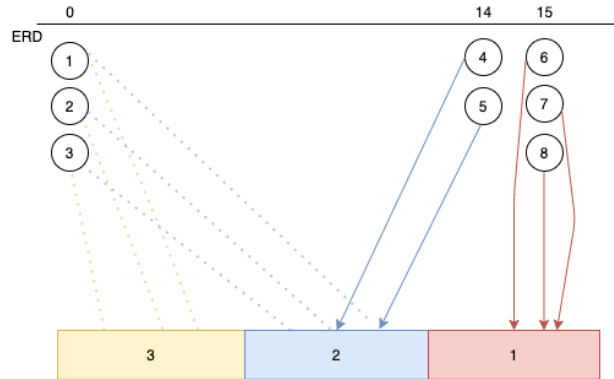
Deadline  $T_E$ ; Maximum number of requests allowed in a batch  $\rho$ ; Duration of a batch route  $T_D$ ;

**Output:** Set  $K_0$ ; Number of batches  $|K|$ ; Batch indices  $\{k(i), i \in N_e^{unknown}\}$ ;  $\{(\tau_{start}^k, \tau_{end}^k, \rho_k), k \in K\}$ ;

- 1:  $n \leftarrow |N_e^{unserved}|$ ;  $K \leftarrow \emptyset$ ;  $K_0 \leftarrow \emptyset$ ;  $k \leftarrow 1$ ;  $t_k \leftarrow T_E - T_D$ ;  $\rho_1 \leftarrow 0$
- 2:  $k(i) \leftarrow 0$ , for all  $i \in N_e^{unknown}$ ;
- 3: Sort the requests from  $N_e^{unserved}$  in increasing order of release dates  $r_i^e$ ;
- 4: Initialize the number of requests assigned to the  $k$ -th batch  $r \leftarrow 0$ ;
- 5: **for**  $i \leftarrow n$  to 1 **do**
- 6:   **if**  $t_k \leq t_e$  **then break**
- 7:   **if** release date  $r_i^e \leq t_k$  **then**
- 8:     **if**  $i \in N_e^{known}$  **then**  $K_0 \leftarrow K_0 \cup \{k\}$
- 9:     **else**  $k(i) \leftarrow k$ ;  $\rho_k \leftarrow \rho_k + 1$
- 10:     $r \leftarrow r + 1$
- 11:    **if**  $r = \rho$  or  $i = 1$  **then**
- 12:      $K \leftarrow K \cup \{k\}$
- 13:      $\tau_{start}^k \leftarrow t_k$ ;  $\tau_{end}^k \leftarrow t_k + T_D$
- 14:      $k \leftarrow k + 1$ ;  $t_k \leftarrow t_k - T_D$ ;  $r \leftarrow 0$ ;  $\rho_k \leftarrow 0$
- 15: **return**  $K_0, K, \{k(i), i \in N_e^{unknown}\}$ , and  $\{(\tau_{start}^k, \tau_{end}^k, \rho_k)\}_{k \in K}$

**Table 1** Input instance for the batch approach (ERD 0 means the parcel  $i$  is at the depot, i.e.,  $i \in N_e^{known}$ )

Total number of requests	8	Max number of requests in a batch $\rho$						3
Number of batches	3	Duration of a batch tour $T_D$						1
Returning time of route 0	TBD	Deadline						16
ID of the requests	1	2	3	4	5	6	7	8
ERD of the requests	0	0	0	14	14	15	15	15



**Figure 1** The assignment of requests to batches (a dashed line connects known requests with batches from  $K_0$ , indicating potential assignments of each  $i \in N_e^{known}$  to some  $k \in K_0$ )

**Table 2** The batches

Batch number	<b>3</b>		<b>2</b>		<b>1</b>		
Unknown requests assigned	-	-	4	5	6	7	8
Max ERD served	0		14		15		
Start time of the tour	13		14		15		
End time of the tour	14		15		16		
$k(i)$	-	-	$k(4) = 2$	$k(5) = 2$	$k(6) = 1$	$k(7) = 1$	$k(8) = 1$
$K_0$	✓		✓		✗		

To explain how the batch approach works, a toy example with 8 requests is displayed in Table 1, and the output is shown in Table 2 and Figure 1. Note that in Table 1 we also consider the ending time of route 0 (whose value is to be defined), that is the route that leaves immediately from the depot. The ending time of route 0 will allow to schedule only those batches that can start after the route 0 is completed. The role of this parameter will become clearer in the following sections where we explain how the output of the batch algorithm is integrated in the two solution approaches. As the deadline is 16, the first batch route should be completed no later than 16. Each route lasts 1 time unit (according to Daganzo's formula) so the three requests with expected release date equal to 15 are assigned to batch 1 (which is mapped as the first future route backward). Correspondingly, the start time of the tour is 15. For batch 2, its ending time is 15 so its starting time is 14. Thus the two requests with expected release dates equal to 14 are assigned to the batch. As batch 2 has a residual capacity of 1, any of the known requests (with ID 1, 2 and 3) can be assigned to it. In addition, requests 1, 2 and 3 can be assigned to batch 3. Table 2 describes the content of batches and displays the values of  $k(i)$ . It also shows that batches 2 and 3 are the ones that can be used to serve known requests (thus, they belong to set  $K_0$ ) with the check mark on the last row. Note that, in case a request  $i \in N_e^{unknown}$  is not assigned to any batch (for example, because its release date is larger than the starting time of route 1 or because there are no sufficient batches to include all requests), then  $k(i) = 0$ . Finally, as shown in the following sections, to evaluate if each future route is executable, we need to check whether the starting time is greater than the ending time of the route leaving immediately from the depot, that is route 0. This is done through the optimization models presented in the following section.

**5.1.1. Optimality Conditions for the Batch Approach** In the following, we show that under some simplifying conditions, the batch approach given by Algorithm 1 provides a polynomial way to calculate an optimal solution. Let us assume that we are at the decision epoch  $e$  and that all the release dates are known with certainty. We denote as  $r$  the vector of release dates. In the following proposition, we still use the notation  $N_e^{unknown}$  referring to parcels whose release date is greater than  $t_e$ .

PROPOSITION 3. If the release dates are deterministic, the duration of each route is  $T_D$ , independently of the location of the requests served, and the vehicle can deliver up to  $\rho$  parcels per route, Algorithm 1 finds (in polynomial time):

1. The maximum number of requests from  $N_e^{unknown}$  that can be served within  $[t_e, T_E]$ , given as  $|N_{\mathcal{O}}|$ , where

$$N_{\mathcal{O}} = \{i \in N_e^{unknown} : k(i) \neq 0\}. \quad (4)$$

2. The maximum number of all requests from  $N_e^{unserved}$  that can be served within the interval  $[t_e, T_E]$ , given as:

$$OPT_{\rho, T_D} = \begin{cases} |N_e^{known}| + |N_{\mathcal{O}}|, & \text{if } |N_e^{known}| < \sum_{k \in K_0} (\rho - \rho_k) \\ |K| * \rho, & \text{otherwise} \end{cases}$$

*Proof.* We first observe that in case  $\rho_k = \rho$  for all  $k \in K$ , then this is trivially the optimal solution. Thus, in the following we discard this case.

1. By construction of the solution  $N_{\mathcal{O}}$ , there exists  $\tilde{k}$ ,  $1 \leq \tilde{k} \leq |K|$ , such that  $\rho_{\tilde{k}} < \rho$ ,  $\rho_k = \rho$  for  $k < \tilde{k}$  and  $\rho_k = 0$  for  $k > \tilde{k}$ . The batch  $\tilde{k}$  is called *critical batch* and corresponds to the only possible batch whose spare capacity is positive but strictly smaller than  $\rho$ . We call this property *the critical batch property*. Suppose that there exists a solution  $N_{\mathcal{O}'} \subseteq N_e^{unknown}$  such that  $|N_{\mathcal{O}'}| > |N_{\mathcal{O}}|$  parcels from  $N_e^{unknown}$  can be delivered within the interval  $[t_e, T_E]$ . Without loss of generality, we can shift the starting time of all batches to the latest possible and then reassigning parcels to the latest possible batch in such a way that the solution  $N_{\mathcal{O}'}$  also

satisfies the critical batch property. We argue that the number of parcels scheduled in batch  $\tilde{k}$  of reordered solution  $N_{\mathcal{O}'}$  cannot be bigger than the one in  $N_{\mathcal{O}}$ . Indeed, by contradiction: let us consider a request  $i$  served in batch  $\tilde{k}$  in  $N_{\mathcal{O}'}$  and not in  $N_{\mathcal{O}}$ . In case  $i$  is not served in any batch  $k < \tilde{k}$  in  $N_{\mathcal{O}}$ , then  $i$  would have been inserted in batch  $\tilde{k}$  in solution  $N_{\mathcal{O}}$  as well. In case, instead, request  $i$  is served in one batch  $k < \tilde{k}$  in  $N_{\mathcal{O}}$ , then it means there exists at least one request  $i'$  served in a batch  $k < \tilde{k}$  in  $N_{\mathcal{O}'}$  but not in  $N_{\mathcal{O}}$ . However, due to the way batches in  $N_{\mathcal{O}}$  are constructed, this means that  $r_i^e \geq r_{i'}^e$ . Given that  $i$  is inserted in batch  $\tilde{k}$  in  $N_{\mathcal{O}'}$ , this means that  $r_{i'}^e \leq \tau_{start}^{\tilde{k}}$ , thus  $i'$  and  $i$  could be swapped in the corresponding batches in  $N_{\mathcal{O}'}$ . By repeating iteratively this procedure on all requests in batch  $\tilde{k}$  in  $N_{\mathcal{O}'}$ , we obtain two identical sets of requests in the critical batch  $\tilde{k}$ .

2. We notice that  $\sum_{k \in K_0} (\rho - \rho_k)$  represents the sum of spare capacities over all routes that can be scheduled within  $[t_e, T_E]$ . Since the parcels from  $N_e^{known}$  are already available at time  $t_e$ , if their number is smaller than the total spare capacity, all of them can be distributed among the routes determined by  $K_0$ , hence guaranteeing that  $|N_e^{known}| + |N_{\mathcal{O}}|$  parcels are delivered in total. If, on the contrary,  $|N_e^{known}|$  is larger than the total spare capacity, only a subset of them will be delivered, so that for each batch, its maximal capacity is used. We emphasize that, due to the assumption that all routes have the same duration, namely  $T_D$ , and the same capacity  $\rho$ , customer locations are irrelevant in this case, and hence, the way how assignment of parcels from  $N_e^{known}$  to batches from  $K_0$  is done, does not affect the optimal solution. ■

Hence, by combining scenario generation and the batch approach, we approximate the value of future routes as a one-step look-ahead policy in which future events are condensed in a single step and future actions are evaluated through batch approximation.

## 5.2. Policy Function Approximation

There are four fundamental ways of deriving policies (Powell (2014)), one of which is based on policy function approximations (PFAs). PFA might consist in a set of simple rules, a lookup table,

or a parametric model that does not involve solving an optimization problem. Our first solution approach falls into this category and it works according to the following rule:

$$\tilde{X}_e^{PFA}(S_e | \theta) = \Phi(\tilde{X}_e^\omega)_{\omega \in \Omega} \quad (5)$$

where

$$\tilde{X}_e^\omega = \arg \max_{\mathcal{X}_e \in X(S_e)} \{C(S_e, \mathcal{X}_e) + \gamma \tilde{V}_{e+1}(\tilde{S}_{e+1} | \omega)\} \quad (6)$$

is the optimal action for the given value function approximation  $\tilde{V}_{e+1}(\tilde{S}_{e+1} | \omega)$  ( $\tilde{S}_{e+1}$  is the approximation of state at  $e + 1$  and  $\tilde{V}$  is an estimate of the value of being in state  $\tilde{S}_{e+1}$ ), under the realization of scenario  $\omega$ . As defined in Section 5.1,  $\theta$  captures all tunable parameters of the model. Function  $\Phi$  is the consensus function for determining the action, considering the best actions over all scenarios (see Section 5.2.1). Thus, the main idea of the proposed PFA is that, for each scenario  $\omega$ , the best action  $\tilde{X}_e^\omega$  is determined, according to the method described in Section 5.2.2. This consists in solving a deterministic ILP where the goal is to determine the route that should leave immediately from the depot (which might be empty) by optimizing the function given by the sum of the immediate reward associated with this route plus the estimated value of future routes (determined through the batch approach described in Section 5.1). Then, a consensus function is applied to determine the best action according to the solution obtained on each of the scenarios. Note that, as we solve an ILP for each scenario independently, there is no uncertainty in the input data to the ILP: indeed, the release dates correspond to the realizations associated with the given scenario, and we refer to this model as *deterministic ILP model* below.

**5.2.1. Consensus Function** We define the consensus function denoted with  $\Phi$  in formula (5) as the policy to find which requests are served in route 0 while considering all solutions associated with the different scenarios. The scheme is reported in Algorithm 2 in Appendix 8.5. The function checks solutions  $\tilde{X}_e^\omega$  over all scenarios  $\omega \in \Omega$ , by focusing on the requests in  $N_e^{known}$  served in route 0 in each solution. The idea is to determine which subset of these requests should be served in route 0. Specifically, a request  $i \in N_e^{known}$  is served if it is included in route 0 in at least  $\frac{|\Omega|}{2}$  solutions. If

at least one request is included in the route, then a Traveling Salesman Problem (TSP) is solved over all requests included to determine the best sequence. Otherwise, the decision is to wait at the depot until the next decision epoch  $e + 1$ .

**5.2.2. Deterministic ILP Model** We now describe the optimization model used to find the optimal policy  $\tilde{X}_e^\omega$  given in (6). The idea is that only the reward of route 0 (which corresponds to  $C(S_e, \mathcal{X}_e)$ ) is calculated with the detailed routing information (since the route is to be executed immediately in case at least one request is assigned to it), while the value and the duration of future routes are estimated through the batch approach (Section 5.1). We define as  $K = \{1, \dots, |K|\}$  the set of future routes. The parameters considered are in the following:

- $\tau_{start}^0$  is a parameter indicating the starting time of the immediate route, thus its value is equal to  $t_e$ .
- $A^{known}$ , it includes all arcs  $(i, j)$  where  $i, j \in N_e^{known} \cup \{0\}$  linking customers with known requests and depot.

In addition, we have the parameters associated with future batches corresponding to the output of the batch approach (see Section 5.1).

The decision variables are the following:

- $x_{ij}^0$ : binary arc variable for route 0, equal to 1 if the route traverses arc  $(i, j) \in A^{known}$ , 0 otherwise.
- $y_i^0$ : binary variable associated with known requests  $i \in N_e^{known}$ , equal to 1 if customer  $i$  is served in route 0, and 0 otherwise.
- $z_k$ : binary variable equal to 1 if future route (batch)  $k \in K$  is executed, 0 otherwise.
- $\tau_{end}^0$ : continuous variable denoting the ending time of the route 0.
- $w_i^k$ : binary variable equal to 1 if known request  $i \in N_e^{known}$  is served in batch  $k \in K_0$ , 0 otherwise.

The deterministic ILP model is given as follows. In line with (6), the first term in the objective function (7a), namely  $\sum_{i \in N_e^{known}} y_i^0$ , maps  $C(S_e, \mathcal{X}_e)$  representing the number of requests served by

action  $\mathcal{X}_e$ . The sum of the two remaining terms in (7a), namely  $\sum_{k \in K} \rho_k z_k$  and  $\sum_{k \in K_0} \sum_{i \in N_e^{known}} w_i^k$  maps  $\tilde{V}_{e+1}(\tilde{S}_{e+1} \mid \omega)$ , representing the approximated number of parcels delivered at future states. Note that in case  $\sum_{i \in N_e^{known}} y_i^0 = 0$ , it means that route 0 does not serve any request and the decision is to wait at the depot until the next decision epoch  $e + 1$ . Then for each scenario  $\omega$ , the corresponding ILP formulation is given as follows:

$$\max \sum_{i \in N_e^{known}} y_i^0 + \gamma \left[ \sum_{k \in K} \rho_k z_k + \sum_{k \in K_0} \sum_{i \in N_e^{known}} w_i^k \right] \quad (7a)$$

subject to:

$$\sum_{(i,j) \in A^{known}} x_{ij}^0 = \sum_{(j,i) \in A^{known}} x_{ji}^0 = y_i^0 \quad i \in N_e^{known} \cup \{0\} \quad (7b)$$

$$\sum_{i,j \in S} x_{ij}^0 \leq \sum_{i \in S \setminus \ell} y_i^0 \quad S \subseteq N_e^{known}, |S| \geq 2, \ell \in S \quad (7c)$$

$$\tau_{end}^0 = \tau_{start}^0 + \sum_{(i,j) \in A^{known}} d_{ij} x_{ij}^0 \quad (7d)$$

$$\tau_{end}^0 \leq T_E \quad (7e)$$

$$\sum_{k \in K_0} w_i^k + y_i^0 \leq 1 \quad i \in N_e^{known} \quad (7f)$$

$$\sum_{i \in N_e^{known}} w_i^k \leq (\rho - \rho_k) z_k \quad k \in K_0 \quad (7g)$$

$$z_{k+1} \leq z_k \quad k = 1, \dots, |K| - 1 \quad (7h)$$

$$\tau_{end}^0 - \tau_{start}^k \leq (T_E - \tau_{start}^k)(1 - z_k) \quad k \in K \quad (7i)$$

$$y_i^0, w_i^k, x_{ij}^0, z_k \in \{0, 1\} \quad i, j \in N_e^{known}, k \in K \quad (7j)$$

Besides, we initialize the model with GSECs of size two and the constraints (7l) to strengthen the formulation:

$$x_{ij}^0 + x_{ji}^0 \leq y_i^0 \quad i, j \in N_e^{known} \quad (7k)$$

$$\sum_{i \in N_e^{known}} y_i^0 \leq |N| y_0^0 \quad (7l)$$

The objective function (7a) maximizes the total number of requests served by route 0 plus the number of requests served by future routes, distinguished in unknown and known requests, multiplied by the discounting factor  $\gamma$ . Constraints (7b), (7c) guarantee that route 0 is a circuit



connected to the depot and prevent generating subtours. Constraint (7d) determines the ending time of route 0, which is computed as its starting time plus the traveling time. Constraints (7e) ensure that route 0 finishes within the deadline  $T_E$ . Constraints (7f) state that the same request cannot be served by both route 0 and a future route. Constraints (7g) guarantee that each future route in  $K_0$  does not serve more than  $\rho$  requests. Note that these constraints are needed for routes in  $K_0$  only, as potentially any request in  $N_e^{known}$  can be assigned to each of these routes. Instead, for the routes in  $K \setminus K_0$ , the constraint is satisfied by construction from the batch approach. Constraints (7h) state that if future route  $k + 1$  is executed, then route  $k$  must be executed too (symmetry breaking constraints). Constraints (7i) ensure that future route  $k$  can start only after finishing route 0. Finally, (7j) define variables domain.

To solve this model, we implemented a branch-and-cut procedure in which the generalized sub-tour elimination constraints (7c) are separated in each node of the branching tree (see Section 6 for further details).

### 5.3. Look-ahead Policy with Value Function Approximation

The second approach we propose is a two-stage stochastic look-ahead model using a value function approximation:

$$\tilde{X}_e^{VFA}(S_e | \theta) = \arg \max_{\mathcal{X}_e \in X(S_e)} \{C(S_e, \mathcal{X}_e) + \gamma \mathbb{E}[\tilde{V}_{e+1}(\tilde{S}_{e+1}) | (S_e, \mathcal{X}_e)]\} \quad (8)$$

where, as before,  $\theta$  captures all tuning parameters,  $\tilde{S}_{e+1}$  is the approximation of state at the epoch  $e + 1$  and  $\tilde{V}_{e+1}(\tilde{S}_{e+1})$  represents the approximation of the value function of being in state  $\tilde{S}_{e+1}$ .

The action  $\tilde{X}_e^{VFA}(S_e | \theta)$  is determined through the solution of a two-stage stochastic integer program as presented in the following.

**5.3.1. Two-stage Stochastic Integer Program** The modeling idea is similar to the one presented in Section 5.2.2, i.e., routing decisions are taken only for route 0 while future routes are approximated with the batch approach. The main difference with respect to the approach presented in Section 5.2 is that a single Two-stage Stochastic Integer Program (2-SIP) is solved using all

scenarios in  $\Omega$ , instead of solving  $|\Omega|$  deterministic ILPs (separately per scenario). Specifically, each scenario  $\omega \in \Omega$  is associated with the realizations of the release dates of the unknown requests  $\{\tilde{r}_i^e\}_{i \in N_e^{unknown}}$ , denoted as  $\tilde{r}^{e1}, \dots, \tilde{r}^{e\omega}$ . Also, we denote as  $p^\omega$  ( $0 < p^\omega < 1$ ) the probability associated with scenario  $\omega \in \Omega$  ( $\sum_{\omega \in \Omega} p^\omega = 1$ ). We first run the batch algorithm on the realization of each scenario to pre-calculate the following parameters:

- $K^\omega$  is the set of indices for all the batches created in scenario  $\omega$ ,  $\omega \in \Omega$
- $k^\omega(i)$  is index of the route serving request  $i \in N_e^{unknown}$  in scenario  $\omega$ . As before  $k^\omega(i) = 0$

means that request  $i \in N_e^{unknown}$  is not served in scenario  $\omega \in \Omega$  in any of the batches;

- $K_0^\omega$  is the subset of batches with spare capacities in scenario  $\omega \in \Omega$ .
- $\tau_{start}^{k\omega}$  is the starting time of route (batch)  $k \in K$  in scenario  $\omega \in \Omega$ .
- $\rho_k^\omega$  is the number of unknown requests assigned to batch  $k \in K$  in scenario  $\omega \in \Omega$ .

As for the decision variables, we keep variables  $x_{ij}^0$ ,  $y_i^0$  and  $\tau_{end}^0$  as defined in the deterministic model (7). In addition, we have the scenario-related variables as follows:

- $w_i^{k\omega}$ :  $i \in N_e^{known}$ ,  $k \in K_0^\omega$ , binary variable equal to 1 if known request  $i$  is served in batch  $k$  in scenario  $\omega \in \Omega$ , and 0 otherwise.

- $z_k^\omega$ : Binary variable equal to 1 if batch  $k \in K$  in scenario  $\omega \in \Omega$  is used, and 0 otherwise. For the ease of reading, we will simply write  $z_{k(i)}^\omega$  instead of  $z_{k^\omega(i)}^\omega$ .

The first stage decision corresponds to determining route 0. In line with the policy function (8), the number of requests served by route 0 is determined by  $\sum_{i \in N_e^{known}} y_i^0$  and matches the immediate reward  $C(S_e, X_e)$ . The second-stage decisions are instead associated with determining the requests served in future routes in each scenario. They correspond to the second term in the objective function (9a) and match the approximation  $\gamma \mathbb{E}[\tilde{V}_{e+1}(\tilde{S}_{e+1}) \mid (S_e, \mathcal{X}_e)]$ . The 2-SIP is formulated as follows:

$$\max \sum_{i \in N_e^{known}} y_i^0 + \gamma \mathbb{E}_\omega[\tilde{V}(y^0, \tau_{end}^0, \tilde{r}^{e\omega})] \quad (9a)$$

$$\text{s.t.} \quad (x^0, y^0, \tau_{end}^0) \text{ satisfies (7b)-(7e), (7k)-(7l)}$$

$$x_{ij}^0, y_i^0 \in \{0, 1\} \quad i, j \in N_e^{known} \quad (9b)$$

where  $\tilde{V}(y^0, \tau_{end}^0, \tilde{r}^e)$  represents the value of the recourse function associated with the second stage. This value depends on the requests served in route 0 ( $y^0$ ), the duration of the route 0 ( $\tau_{end}^0$ ) and the distribution of uncertain release dates for unserved requests ( $\tilde{r}^{e\omega}, \omega \in \Omega$ ). Given a route 0 determined by the vector  $\bar{y}^0$ , with duration  $\bar{\tau}_{end}^0$  and a realization of  $|\Omega|$  scenarios concerning the release dates  $\tilde{r}^e$  (each with probability  $p_\omega > 0$ ), the expected value of the recourse function is calculated as the following ILP:

$$\mathbb{E}_\omega[\tilde{V}(\bar{y}^0, \bar{\tau}_{end}^0, \tilde{r}^{e\omega})] := \max_{w,z} \sum_{\omega \in \Omega} p_\omega \left( \sum_{k \in K^\omega} \rho_k^\omega z_k^\omega + \sum_{k \in K_0^\omega} \sum_{i \in N_e^{known}} w_i^{k\omega} \right) \quad (9c)$$

$$\text{subject to: } \sum_{k \in K_0^\omega} w_i^{k\omega} \leq 1 - \bar{y}_i^0 \quad \omega \in \Omega, i \in N_e^{known} \quad (9d)$$

$$\sum_{i \in N_e^{known}} w_i^{k\omega} \leq (\rho - \rho_k^\omega) z_k^\omega \quad \omega \in \Omega, k \in K_0^\omega \quad (9e)$$

$$z_k^\omega \geq z_{k+1}^\omega \quad \omega \in \Omega, k = 1, \dots, |K^\omega| - 1 \quad (9f)$$

$$\tau_{start}^{k\omega} + (T_E - \tau_{start}^{k\omega})(1 - z_k^\omega) \geq \bar{\tau}_{end}^0 \quad \omega \in \Omega, k \in K^\omega \quad (9g)$$

$$w_i^{k\omega}, z_k^\omega \in \{0, 1\} \quad \omega \in \Omega, i \in N_e^{known}, k \in K^\omega \quad (9h)$$

The objective function (9c) maximizes the expected number of requests served by future routes. The following constraints till (9g) have a similar meaning as in formulation (7) with the difference that they are replicated over all scenarios. Finally, (9h) define variables domain.

In order to solve this 2-SIP model, we implement a branch-and-cut procedure, similar to the one of the deterministic ILP model, with a dynamic separation of the generalized subtour elimination constraints (7c).

## 6. Computational Experiments

In this section, we describe the computational experiments we carry out for validating the performance of the two proposed approaches. The code is developed in Python 3.7 and CPLEX version 20.1.0.0 is used to solve all ILPs, run on a single thread with memory usage limit set to 2GB. We run the simulations on HP Z4 G4 Workstation with Intel(R) Xeon(R) W-2255 CPU @ 3.70GHz.

As for parameter values, we set them as follows. The discount rate  $\gamma$  is set to 0.9, which is a commonly adopted value used for similar approaches in the literature (see, e.g., Völkel (2020)). Also, some preliminary experiments showed that, by varying the value of  $\gamma$ , the performance of the approaches is not significantly affected. The maximum time interval between consecutive decision epochs  $\phi$  is set to 10, which is a sufficiently small value to avoid long waiting times at the depot according to the input instances used in the computational campaign (described in Section 6.1). The number of scenarios  $|\Omega|$  generated in each decision epoch is set to 30 and in our preliminary experiments we noticed no significant performance improvements when considering larger values of  $|\Omega|$ .

As for implementation details, we remark the following. Six branch-and-cut algorithms are implemented: a) to calculate upper bounds with perfect information, b) to check whether the conditions of the partially optimal policy are satisfied, c) to calculate the consensus function of the PFA approach, d) to calculate the value function approximation in the VFA approach, and e) to get the route for the ME approach described in Section 6.2. All these algorithms rely on separation of GSECs which are implemented using lazy cutcallback of the CPLEX solver. To benefit from general-purpose cuts generated by the CPLEX solver, we collect the cuts separated at the root node of the branching tree, and then restart the ILP. We set a time limit for the solution of each branch-and-cut to 5, 10 and 10 minutes for the deterministic, stochastic and myopic ILP model, respectively. Note that the deterministic model is allocated with a shorter computing time as it is solved for each scenario. All other CPLEX parameters are left at their default values. The call to partial characterization of optimal policy is skipped in each decision epoch of PFA and VFA when  $|N_e^{known}| < 0.25|N_e^{unserved}|$  and remaining time  $[t_e, T_E] > 0.75T_E$ . We consider that for such settings there are small chances that conditions of Proposition 2 are satisfied. Concerning the calculation of upper bounds using the model (2), we provide CPLEX with a starting feasible solution corresponding to the solution of model (3) and the time limit is set to one hour.

The remainder of this section is organized as follows. We first describe the input instances in Section 6.1 and the myopic approaches in Section 6.2. Section 6.3 illustrates the benefit of

integrating partial characterization of optimal policy within the PFA and VFA approaches. Finally, Section 6.4 presents the comparison of the two proposed approaches with two myopic approaches.

Additional computational results can be found in the Appendix, where Section 8.7 presents a sensitivity analysis on the value of parameter  $\rho$ , i.e., the maximum number of parcels included in each future route as used in the batch approach.

### 6.1. Input Instances

We use the set of benchmark instances described in Archetti et al. (2020) that are derived from Solomon’s instances (Solomon (1987)). Specifically, instance classes  $C1$ ,  $C2$ ,  $R1$  and  $RC1$  are considered. Each instance is associated with three parameters  $\beta$ ,  $\delta$  and  $T_E$ . The first two parameters,  $\beta$ ,  $\delta$ , define the dispersion of release dates and the percentage of customers with dynamic release dates, respectively. We refer to Section 8.6 in the Appendix and Archetti et al. (2020) for more details. Parameter  $T_E$  is the deadline of the delivery service (that is not defined in the problem studied in Archetti et al. (2020)), which is determined as follows. We first determine the latest actual arrival time of all requests and we denote it as  $T_{standard}$ . Then, we generate four instances with  $T_E$  equal to  $c * T_{standard}$ , where  $c \in \{0.6, 0.8, 1.0, 1.2\}$ . For each customer input data and each choice of  $\beta$ ,  $\delta$  and  $c$ , five different instances (with different seeds) are generated, thus resulting in a benchmark set of 720 instances.

### 6.2. Two myopic approaches

We propose two myopic approaches as a benchmark to compare with. The main idea of both approaches is to dispatch the vehicle immediately in every decision epoch by serving all known requests (if any) that can be feasibly served. The difference among the two lies in how the vehicle route (and eventually the subset of known requests to serve) is determined in each decision epoch. The first approach, denoted as  $ME$  in the following, works as follows: every time the vehicle is back to the depot, in case there is at least one parcel available, the vehicle is immediately dispatched. All parcels available are served if the deadline is not violated. In case the deadline is violated,

there is a need to select a subset of parcels to deliver. This is done through exactly solving an orienteering problem (via a branch-and-cut method) to maximize the number of parcels delivered while satisfying the limited route duration.

The second myopic approach, denoted as *MH*, differs from the first in the way the vehicle route is constructed, which follows the nearest neighbor idea. Specifically, every time the vehicle is at the depot and there is at least one parcel available, the dispatching decision is made by selecting first the closest customer location to the depot then the nearest customer to the one just selected and so on until the remaining time is not enough to serve any new customer or all parcels available are included on the route. The motivation behind the proposal of *MH* as a benchmark is that it mimics practical cases associated with companies that are not equipped with sophisticated optimization tools (like the one needed in *ME*) for decision making.

### 6.3. Benefit of partial characterization of optimal policy

We illustrate the benefit of partial characterization (PC) of optimal policy (see Section 4.2) by comparing the performance of PFA and VFA with and without PC. Results are shown in Table 3 where instances are classified according to the value on  $\delta$  first, then  $\beta$  and finally  $c$ . For each of the two approaches, the table reports the average computing time (in seconds) with and without PC, the average and maximum percentage improvement in solution value of the approach with PC with respect to the corresponding one without PC, and the number of times the approach with PC improved on the solution of the approach without PC.

In Table 3, we observe that when using PC, the running time doubles in both approaches. However, this is compensated by significant solution improvements, especially when all customers are static ( $\delta = 0$ ), release dates are spreaded in a moderate level ( $\beta = 0.5$ ) and time horizons are large ( $c = 1, 1.2$ ). Overall, we have an improvement of 4.98% for VFA and 5.55% for PFA, with maximum improvements being 24% and 29%, respectively.

Given the benefits of PC and the fact that computing time remains reasonable, we show the results of PFA and VFA with PC in the following section.

**Table 3** The comparison of VFA and PFA (with and without partial characterization of optimal policy)

		VFA					PFA				
		t(s)		Improvement			t(s)		Improvement		
$\rho$	#instances	noPC	withPC	AVG(%)	MAX(%)	#impr.	noPC	withPC	AVG(%)	MAX(%)	#impr.
$\delta = 0$	240	30.19	66.89	5.18	15.00	33	42.92	78.57	5.32	15.00	31
$\delta = 0.5$	240	33.28	69.73	4.21	12.00	29	45.15	83.72	5.10	29.00	29
$\delta = 1$	240	37.75	81.89	4.93	24.00	27	82.74	144.24	5.04	24.00	27
$\beta = 0.5$	240	38.76	71.11	8.17	24.00	6	86.69	146.24	12.83	29.00	6
$\beta = 1$	240	32.07	76.60	4.66	15.00	59	45.94	89.22	4.73	15.00	59
$\beta = 1.5$	240	30.40	70.81	4.25	8.00	24	38.18	71.07	4.23	9.00	22
$c = 0.6$	180	31.37	74.45	3.57	4.00	7	43.28	87.28	3.00	3.00	1
$c = 0.8$	180	32.27	72.69	4.80	8.00	15	87.32	144.52	3.73	6.00	22
$c = 1$	180	34.97	69.45	5.86	15.00	29	49.92	87.50	6.59	15.00	29
$c = 1.2$	180	36.35	74.77	4.18	24.00	38	47.23	89.41	4.94	29.00	35
Avg/MAX		<b>33.74</b>	<b>72.84</b>	<b>4.98</b>	<b>24.00</b>	<b>26.7</b>	<b>56.94</b>	<b>102.18</b>	<b>5.55</b>	<b>29.00</b>	<b>26.1</b>

#### 6.4. Performance of PFA and VFA

In this section, we compare the results of PFA and VFA (with PC) against ME and MH. The goal is to show the benefit of incorporating approximation of future information into exact solution methods to improve sequential decision making. Based on the results of sensitivity analysis (see Appendix 8.7), we set the value of  $\rho$  to 15 both in PFA and VFA. The KPI used for the comparison is the best solution found over all approaches. In addition, we also report, for each of the four approaches, the gap with respect to the upper bound with perfect information, obtained by solving formulation (2). This latter measure is related to the competitive analysis where the focus is on measuring the discrepancy in solution quality due to lack of information.

Results are summarized in Table 4, summarized by values of  $\beta$  and  $\delta$ , and in Figure 2 (summarized over  $\beta$ ) and Figure 3 (summarized over  $\delta$ ). Detailed results for all values of  $\beta$ ,  $\delta$  and  $c$  can be found at the link [DetailedResult](#)<sup>3</sup>. In addition, summarized results over the value of  $c$  can be found in the Appendix, Section 8.8.

For each approach, we report the average gap with respect to the best solution found over all approaches (denoted as  $gap\%$ ), the gap with respect to the upper bound with perfect information ( $gap_{ub}\%$ ), the computational time ( $t[s]$ ) and the number of times the best solution was found,

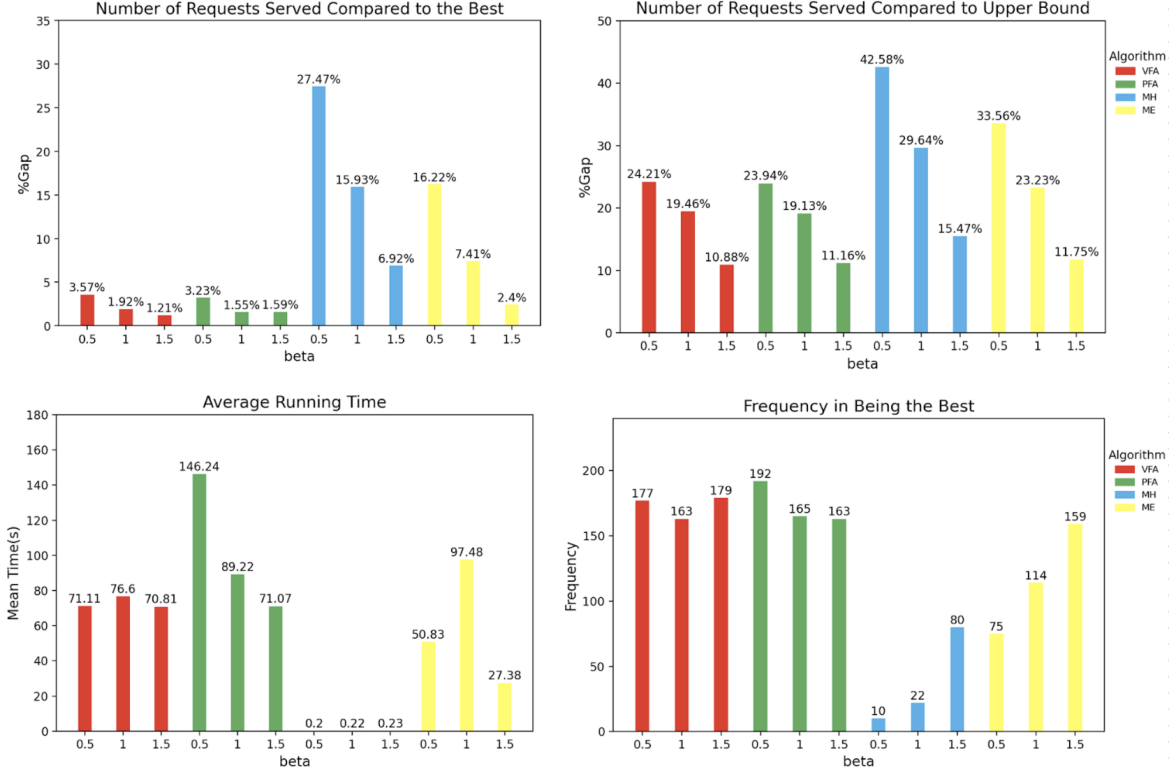
including ties ( $freq$ ). As for  $gap\%$ , we calculated it as follows. Given an input instance, let  $N_p$  denote the value of the solution found by approach  $p$ . The gap with respect to the best solution found over all approaches is calculated as  $1 - \frac{N_p}{\max_{q \in \{VFA, PFA, MH, ME\}} N_q}$ . Instead, the gap with respect to the upper bound is calculated as  $1 - \frac{N_p}{UB}$ , where  $UB$  is the value of the upper bound with perfect information.

**Table 4** Average performance of the four approaches

$\beta$	$\delta$	#instances	VFA				PFA				MH				ME			
			gap[%]	gap <sub>ub</sub> [%]	t[s]	freq	gap[%]	gap <sub>ub</sub> [%]	t[s]	freq	gap[%]	gap <sub>ub</sub> [%]	t[s]	freq	gap[%]	gap <sub>ub</sub> [%]	t[s]	freq
0.5	0	80	1.75	24.36	52.05	64	1.49	24.16	70.90	70	29.59	46.00	0.20	0	15.61	34.93	71.03	30
	0.5	80	5.00	24.70	62.44	55	4.83	24.57	91.23	56	24.28	38.97	0.20	7	14.78	31.29	25.07	27
	1	80	3.97	23.57	98.82	58	3.36	23.08	276.60	66	28.54	42.79	0.20	3	18.28	34.47	56.39	18
	AVG	80	<b>3.57</b>	<b>24.21</b>	<b>71.11</b>	<b>59.0</b>	<b>3.23</b>	<b>23.94</b>	<b>146.24</b>	<b>64.0</b>	<b>27.47</b>	<b>42.58</b>	<b>0.20</b>	<b>3.3</b>	<b>16.22</b>	<b>33.56</b>	<b>50.83</b>	<b>25.0</b>
1	0	80	2.21	19.86	80.63	54	1.49	19.22	97.54	54	16.35	30.19	0.22	0	7.29	23.20	111.99	40
	0.5	80	1.46	22.05	74.51	57	1.59	22.12	86.62	57	17.03	33.03	0.22	12	8.60	26.98	99.95	28
	1	80	2.09	16.47	74.66	52	1.57	16.05	83.49	54	14.42	25.71	0.22	10	6.33	19.51	80.52	46
	AVG	80	<b>1.92</b>	<b>19.46</b>	<b>76.60</b>	<b>54.3</b>	<b>1.55</b>	<b>19.13</b>	<b>89.22</b>	<b>55.0</b>	<b>15.93</b>	<b>29.64</b>	<b>0.22</b>	<b>7.3</b>	<b>7.41</b>	<b>23.23</b>	<b>97.48</b>	<b>38.0</b>
1.5	0	80	0.63	7.44	67.99	70	1.19	7.86	67.25	60	7.44	12.96	0.24	30	2.40	8.79	38.58	47
	0.5	80	1.19	14.03	72.24	56	1.56	14.28	73.32	53	7.82	19.37	0.22	20	3.33	15.73	4.78	49
	1	80	1.81	11.19	72.20	53	2.01	11.35	72.64	50	5.50	14.06	0.23	30	1.46	10.75	38.78	63
	AVG	80	<b>1.21</b>	<b>10.88</b>	<b>70.81</b>	<b>59.7</b>	<b>1.59</b>	<b>11.16</b>	<b>71.07</b>	<b>54.3</b>	<b>6.92</b>	<b>15.47</b>	<b>0.23</b>	<b>26.7</b>	<b>2.40</b>	<b>11.75</b>	<b>27.38</b>	<b>53.0</b>
AVG		80	<b>2.24</b>	<b>18.18</b>	<b>72.84</b>	<b>57.7</b>	<b>2.12</b>	<b>18.08</b>	<b>102.18</b>	<b>57.8</b>	<b>16.77</b>	<b>29.23</b>	<b>0.21</b>	<b>12.4</b>	<b>8.68</b>	<b>22.85</b>	<b>58.57</b>	<b>38.7</b>

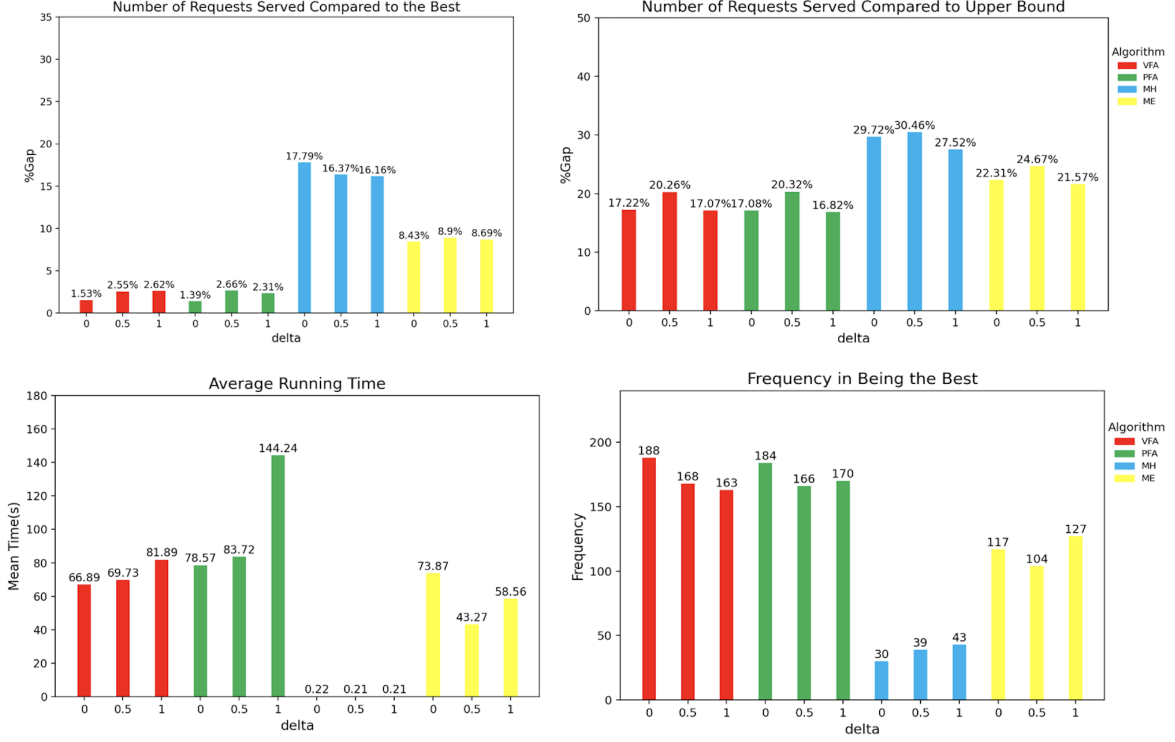
Focusing first on the KPIs related to best known solution ( $gap[\%]$  and  $freq$ ) and on the comparison between VFA and PFA, we notice that, on average, the results obtained with PFA are associated with the smallest gaps, being anyway very similar to the ones obtained with VFA. The same holds when considering the number of best solutions found. The only exception is related to the case of  $\beta = 1.5$  and  $\delta = 0.5$ , however, the differences are always marginal. We instead observe a large difference in terms of computing time, with PFA being much slower than VFA, especially for  $\beta = 0.5$  and  $\delta = 1$ . More specifically, when looking at the third row of Table 4, we can notice that the critical case is the one with  $\beta = 0.5$  and  $\delta = 1$ , where the computing time of PFA is almost three times the one of VFA. We argue that this is due to the fact that, when  $\beta = 0.5$ , due to the lower dispersion of release dates, the number of known requests in each decision epoch increases (as more packages arrive while the vehicle is performing deliveries). Thus, the number of  $x$  and  $y$





**Figure 2** Performance of the approaches based on  $\beta$

variables in (7) increases, slowing down the solution of the formulation. This is valid also for (9). However, given that in PFA formulation (7) is solved for each scenario, the computational burden becomes larger. Moving to the competitive analysis, we notice that both PFA and VFA produce solutions that are between 11% and 25% below the upper bound with perfect information. We also notice that  $\text{gap}_{ub}[\%]$  decreases when increasing the value of  $\beta$ . This might be due to the fact that, when the release dates are more dispersed, the best policy is to let the vehicle leave immediately with a subset of known requests, thus avoiding to wait too long at the depot. Instead, when the dispersion is smaller, it becomes more difficult to judge whether it is more convenient to wait or to start dispatching some parcels. This is also witnessed by the fact that the performance of both myopic approaches, which are based on the idea of dispatching the vehicle immediately with the largest possible subset of known requests, improves for larger value of  $\beta$ . Instead, when focusing on the value of  $\delta$ , the behavior is more flat and there is no clear trend.



**Figure 3** Performance of the approaches based on  $\delta$

Overall, we can conclude that both VFA and PFA largely outperform myopic approaches, the latter ones being considered as proxies of practical policies. The difference in solution quality is marginal and in favor of PFA. However, as VFA is much faster, we believe it is a good compromise between solution quality and computational burden.

Figures 2 and 3 provide graphical representation of obtained results. We observe that when  $\beta$  goes from 0.5 to 1.5, for both  $\text{gap}[\%]$  and  $\text{gap}_{ub}[\%]$ , the performance of all approaches improves progressively, especially for ME and MH. When it comes to the frequency in getting the best solution, we find that the PFA and VFA achieve a significantly better performance when the spread of release dates is the lowest (i.e.,  $\beta = 0.5$ ) but the frequency of getting best results becomes higher in the myopic ones when  $\beta$  grows bigger. This is consistent with the observation above. As for the effect of  $\delta$  (i.e., the percentage of parcels with dynamic release dates), it does not produce a specific trend in general. Overall, we can see that VFA and PFA serve more requests compared to ME and MH, and VFA shows a best tradeoff in terms of running time and the solution quality. Still, we can find that the performance of the myopic approaches varies significantly with the values of  $\beta$ .

Another important observation is that in general the results achieved with PFA and VFA are more stable than the two myopic ones. For the comparison on time scale (see Appendix 8.8), we find similar results.

In summary, we can conclude that VFA achieves the best performance tradeoff in terms of objective value and running time.

## 7. Conclusion

To the best of our knowledge, this article is the first to study the orienteering problem with stochastic and dynamic release dates. For this challenging sequential decision making problem, we propose two reinforcement learning approaches. Both of them rely on MDP with discrete sets of scenarios simulating future realizations of release dates in each decision epoch. They employ the approximation of the value of the future states in two different ways: the PFA approach handles each scenario independently and uses a consensus function to decide on action in each decision epoch. The VFA approach considers all scenarios simultaneously and uses a two-stage stochastic model instead. PFA and VFA are compared with two myopic alternatives, that we consider as proxies of practical policies. Our empirical study confirms benefits of incorporating approximation of future information into exact solution methods to improve sequential decision making. Both PFA and VFA significantly outperform myopic policies, not only in terms of best obtained gaps with respect to the perfect-information upper bounds, but also in terms of frequency with which the best-quality solutions are obtained. The difference in solution quality between PFA and VFA is marginal and in favor of PFA. However, as VFA is much faster, we believe it is a good compromise between solution quality and computational burden.

This paper shows that exact methods provide benefits in solving complex sequential stochastic and dynamic optimization problems, specifically routing problems, which are known to be extremely difficult even in their deterministic counterpart. This opens up opportunities for applying the same methodology to other more complicated problem settings, involving, for example, vehicle capacity, time windows or multiple vehicles.

## Acknowledgements

This work was funded by CY Initiative of Excellence (grant “Investissements d’Avenir” ANR-16-IDEX-0008”). This support is greatly appreciated.

## Endnotes

1. <https://www.statista.com/study/42335/ecommerce-report/>
2. <https://www.statista.com/statistics/1274950/global%2Donline%2Dshoppers%2Dwished%2Dchanges%2Don%2Ddelivery/>
3. [https://github.com/Yuanyuan517/RL-for-OP-with-stochastic-and-dynamic-RD/blob/d2c07ad11caae34d0560c4af5911303caa168565/beta\\_delta\\_ts.csv](https://github.com/Yuanyuan517/RL-for-OP-with-stochastic-and-dynamic-RD/blob/d2c07ad11caae34d0560c4af5911303caa168565/beta_delta_ts.csv)

## References

- Archetti, C. and L. Bertazzi (2021). Recent challenges in routing and inventory routing: E-commerce and last-mile delivery. *Networks* 77(2), 255–268.
- Archetti, C., D. Feillet, A. Mor, and M. G. Speranza (2018). An iterated local search for the traveling salesman problem with release dates and completion time minimization. *Computers & Operations Research* 98, 24–37.
- Archetti, C., D. Feillet, A. Mor, and M. G. Speranza (2020). Dynamic traveling salesman problem with stochastic release dates. *European Journal of Operational Research* 280(3), 832–844.
- Archetti, C., D. Feillet, and M. G. Speranza (2015). Complexity of routing problems with release dates. *European Journal of Operational Research* 247(3), 797–803.
- Behrend, M. and F. Meisel (2018). The integration of item-sharing and crowdshipping: Can collaborative consumption be pushed by delivering through the crowd? *Transportation Research Part B: Methodological* 111, 227–243.
- Bent, R. W. and P. Van Hentenryck (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research* 52(6), 977–987.
- Bertsimas, D. J. and G. Van Ryzin (1991). A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research* 39(4), 601–615.

- Cattaruzza, D., N. Absi, and D. Feillet (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science* 50(2), 676–693.
- Cui, R., M. Li, and Q. Li (2020). Value of high-quality logistics: Evidence from a clash between SF Express and Alibaba. *Management Science* 66(9), 3879–3902.
- Daganzo, C. F. (1984). The distance traveled to visit  $n$  points with a maximum of  $c$  stops per vehicle: An analytic model and an application. *Transportation Science* 18(4), 331–350.
- Fischetti, M., J. J. Salazar González, and P. Toth (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* 10, 133–148.
- Goodson, J. C., J. W. Ohlmann, and B. W. Thomas (2013). Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research* 61(1), 138–154.
- Hong, I., M. Kuby, and A. T. Murray (2018). A range-restricted recharging station coverage model for drone delivery service planning. *Transportation Research Part C: Emerging Technologies* 90, 198–212.
- Jabali, O., M. Gendreau, and G. Laporte (2012). A continuous approximation model for the fleet composition problem. *Transportation Research Part B: Methodological* 46(10), 1591–1606.
- Klapp, M. A., A. L. Erera, and A. Toriello (2018). The one-dimensional dynamic dispatch waves problem. *Transportation Science* 52(2), 402–415.
- Liu, S., L. He, and Z.-J. Max Shen (2021). On-time last-mile delivery: Order assignment with travel-time predictors. *Management Science* 67(7), 4095–4119.
- Ljubić, I. (2021). Solving Steiner trees: Recent advances, challenges, and perspectives. *Networks* 77(2), 177–204.
- Lucena, A. (1993). Tight bounds for the Steiner problem in graphs. *Preprint, IRC for Process Systems Engineering, Imperial College, London*.
- Lucena, A. and M. G. Resende (2004). Strong lower bounds for the prize collecting Steiner problem in graphs. *Discrete Applied Mathematics* 141(1-3), 277–294.
- Oyola, J., H. Arntzen, and D. L. Woodruff (2018). The stochastic vehicle routing problem, a literature review, part I: models. *EURO Journal on Transportation and Logistics* 7(3), 193–221.

- Powell, W. B. (2009). What you should know about approximate dynamic programming. *Naval Research Logistics (NRL)* 56(3), 239–249.
- Powell, W. B. (2014). Clearing the jungle of stochastic optimization. In *Bridging data and decisions*, pp. 109–137. Informa.
- Psaraftis, H. N. (1988). Dynamic vehicle routing problems. *Vehicle routing: Methods and studies* 16, 223–248.
- Reyes, D., A. L. Erera, and M. W. Savelsbergh (2018). Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research* 266(1), 29–34.
- Ritzinger, U., J. Puchinger, and R. F. Hartl (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54(1), 215–231.
- Sankaran, J. K. and L. Wood (2007). The relative impact of consignee behaviour and road traffic congestion on distribution costs. *Transportation Research Part B: Methodological* 41(9), 1033–1049.
- Savelsbergh, M. and M. Sol (1998). Drive: Dynamic routing of independent vehicles. *Operations Research* 46(4), 474–490.
- Schrotenboer, A. H., M. A. Broek, P. Buijs, and M. W. Ulmer (2021). Fighting the e-commerce giants: Efficient routing and effective consolidation for local delivery platforms. *arXiv preprint arXiv:2108.12608*.
- Secomandi, N. (2001). A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research* 49(5), 796–802.
- Secomandi, N. and F. Margot (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research* 57(1), 214–230.
- Shelbourne, B. C., M. Battarra, and C. N. Potts (2017). The vehicle routing problem with release and due dates. *INFORMS Journal on Computing* 29(4), 705–723.
- Soeffker, N., M. W. Ulmer, and D. C. Mattfeld (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research* 298(3), 801–820.
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2), 254–265.
- Souffriau, P. V. W. and D. Oudheusden (2011). The orienteering problem: A survey. *European Journal of Operational Research* 209(1), 1–10.

- 
- Subramanyam, A., F. Mufalli, J. M. Láinez-Aguirre, J. M. Pinto, and C. E. Gounaris (2021). Robust multiperiod vehicle routing under customer order uncertainty. *Operations Research* 69(1), 30–60.
- Ulmer, M. W. and B. W. Thomas (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks* 72(4), 475–505.
- Van Heeswijk, W. J., M. R. Mes, and J. M. Schutten (2019). The delivery dispatching problem with time windows for urban consolidation centers. *Transportation Science* 53(1), 203–221.
- Voccia, S. A., A. M. Campbell, and B. W. Thomas (2019). The same-day delivery problem for online purchases. *Transportation Science* 53(1), 167–184.
- Völkel, M. (2020). *Stochastic Optimization Methods for Supply Chains with Perishable Products*. Logos Verlag Berlin.

## 8. Appendix

### 8.1. Overview of notation used

See Table 5 and Table 6.

**Table 5** Overview of notation used in problem introduction

Problem Description	
$N$	Set of customers
$V$	Set of vertices, $V = \{0\} \cup N$
$A$	Set of arcs connecting nodes $i, j$ , where $i, j \in V$
$G$	A complete graph, $G = (V, A)$
$T_E$	Deadline
$d_{ij}$	Traveling time from $i$ to $j$ , where $i, j \in V$
$K$	Set of routes, $K = \{1, \dots,  K \}$
$e$	Decision epoch, $e \in \{0, \dots, E\}$
$t_e$	Time at decision epoch $e$
$N_e^{\text{served}}$	Set of customers already served before decision epoch $e$
$N_e^{\text{unserved}}$	Set of customers unserved at decision epoch $e$
$N_e^{\text{known}}$	Set of unserved customers available at the depot at decision epoch $e$
$N_e^{\text{unknown}}$	Set of unserved customers whose parcels are still to be delivered to the depot
$N_e^{\text{static}}$	Set of customers whose information about release date is not updated until their parcels arrive at the depot
$N_e^{\text{dynamic}}$	Set of customers whose information about release date is dynamically updated
$\tilde{r}_i^e$	Random variable associated with the release date of customer $i \in N_e^{\text{unserved}}$ at decision epoch $e$
$r_i$	Actual arrival time of the parcel of customer $i \in N_e^{\text{known}}$
Components of MDP	
$\phi$	Time interval between two additional decision epochs
$S_e$	State at decision epoch $e$ , $S_e = (t_e, N_e^{\text{known}}, \{\tilde{r}_i^e\}_{i \in N_e^{\text{unknown}}})$
$X(S_e)$	Action space at decision epoch $e$
$\mathcal{X}_e$	Decision/action at decision epoch $e$ , $\mathcal{X}_e \in X(S_e)$
$l(\mathcal{X}_e)$	Set of customer locations visited in the route associated with the action $\mathcal{X}_e$
$t_{\text{route}}(\mathcal{X}_e)$	Time required to perform the route associated with action $\mathcal{X}_e$
$t_p$	Earliest time when a new parcel arrives while the vehicle is at the depot
$N_{e+1}^{\text{new}}$	Customers whose parcels arrived at the depot between decision epochs $e$ and $e+1$
$V_e(S_e)$	The value at decision epoch $e$ with state $S_e$
$C(S_e, \mathcal{X}_e)$	Number of requests served by action $\mathcal{X}_e$
$\gamma$	Discount factor
Upper Bounds	
$\mathcal{K}$	Number of trips performed by the vehicle
$s_i^k$	Binary variable for trip $k \in \mathcal{K}$ , equal to 1, if customer or depot $i \in V$ visited, 0 otherwise
$\xi_{ij}^k$	Binary arc variable for trip $k \in \mathcal{K}$ , equal to 1, if customer $i \in N$ visited, 0 otherwise
$d_k$	Starting time of trip $k \in \mathcal{K}$
$\alpha_i$	Binary variable equal to 1, if a direct trip from the depot to customer $i \in N$ and back, is performed, 0 otherwise



## 8.2. Summary of literature review

See Table 7.

## 8.3. Upper Bound: the Orienteering Problem

A simple upper bound for the DOP-rd can be obtained by solving a special instance of the Orienteering Problem (OP). The OP is defined on a complete graph  $\hat{G} = (\hat{V}, \hat{A})$  in which we are given arc weights  $\hat{w}_{ij} > 0$ ,  $(i, j) \in \hat{A}$ , and node prizes  $\hat{p}_i > 0$ ,  $i \in \hat{V} \setminus \{0\}$ , where 0 is the depot. The goal is to find a route whose total arc weight does not exceed a budget  $\hat{B} > 0$ , and the total collected prize from visited nodes is maximized. We refer to Souffriau and Oudheusden (2011) for a detailed introduction to the OP.

In each decision epoch  $e$ , we can calculate a valid upper bound on the maximum number of parcels that can be delivered within the interval  $[t_e, T_E]$  in the DOP-rd. This bound is given as a solution of the OP over the set of parcels in  $N_e^{unserved}$ , with all parcel prizes being set to one, and the maximum budget set to  $T_E - t_e$ .

Such obtained upper bound can be used in a competitive analysis. Moreover, it gives rise to the partial characterization of optimal policy described in Section 4.2. The methodology used for solving the OP is based on a branch-and-cut (see, e.g., Fischetti et al. (1998)) which is one of the most effective exact approaches for the OP.

## 8.4. Proof of Theorem 1

*Proof.* A solution of (2) consists of a sequence of non-empty feasible and compatible trips. A trip is feasible if it does not start before any of the release dates of the customers to be visited and finishes before  $T_E$ . Trips are compatible if they visit each customer at most once. Considering a customer  $i$ , the shortest feasible trip serving  $i$  is the one that leaves the depot at time  $r_i$ , goes to  $i$  and comes back to the depot. Constraints (3b) guarantee the feasibility of the direct trips selected: indeed, for each customer  $i$ , constraints (3b) ensure that the trip to  $i$  as well as the following direct trips can be performed within deadline  $T_E$ . Thus, formulation (3) determines the maximum

**Table 6** Overview of notation used in solution approaches

Solution Approaches	
$\Omega$	Set of scenarios predicting their release dates $\tilde{r}_i^e$ at decision epoch $e$ , associated with each customer $i \in N_e^{unserved}$
$\omega$	A realization of possible values of release dates for each customer $i \in N_e^{unserved}$ , $\omega \in \Omega$
$\rho$	Maximum number of requests to be served in each future route
$\mathcal{A}$	Euclidean area containing the locations of unserved customers
$T_D$	Expected tour duration
$\theta$	Tuning parameters including $T_D$ , the distribution of release dates, $\rho$ and $ \Omega $ .
$\tilde{S}_{e+1}$	Approximation of state at $e+1$
$\tilde{V}$	The value of the value function estimated
$\tau_{start}^0$	A parameter indicating the starting time of the immediate route thus its value actually equals to time of the epoch $e$ .
$A^{known}$	Set of arcs $(i, j)$ where $i, j \in N_e^{known} \cup \{0\}$ linking customers with known requests and depot
$p^\omega$	Probability associated with each scenario $\omega \in \Omega$
Batch Approach	
$K$	Set of indices for all the batches created
$K_0$	Set of indices for batches with spare capacities ( $K_0 \subseteq K$ )
$k(i)$	Index of the batch in which the request $i \in N_e^{unknown}$ should be served
$\tau_{start}^k$	Starting time of the route serving batch $k \in K$
$\tau_{end}^k$	Ending time of the route serving batch $k \in K$
$\rho_k$	Number of unknown requests assigned to the route serving batch $k \in K$
Policy Function Approximation	
$\tilde{X}_e^\omega$	Optimal action for the given value function approximation $\tilde{V}$ under scenario $\omega \in \Omega$ in decision epoch $e$
$\Phi$	Consensus function for deciding the solution under the consideration of all the scenarios
$\tilde{X}_e^{PFA}$	Action decided by the consensus function $\Phi$
$x_{ij}^0$	Binary arc variable for route 0, equal to 1 if the route traverses arc $(i, j) \in A^{known}$ , 0 otherwise
$y_i^0$	Binary variable associated with known requests $i \in N_e^{known}$ , equal to 1, if customer $i$ is visited in route 0, and 0 otherwise
$z_k$	Binary variable equal to 1 if future route $k \in K$ is executed, 0 otherwise
$\tau_{end}^0$	Continuous variable denoting the ending time of the route 0
$w_i^k$	Binary variable equal to 1 if known request $i \in N_e^{known}$ is served in batch $k \in K_0$ , 0 otherwise
Value Function Approximation	
$K^\omega$	Set of indices for all the batches created in scenario $\omega$ , $\omega \in \Omega$
$k^\omega(i)$	Index of the route that can serve request $i \in N_e^{unknown}$ in scenario $\omega$ , $\omega \in \Omega$
$K_0^\omega$	Subset of batches with spare capacities in scenario $\omega \in \Omega$
$\tau_{start}^{k\omega}$	Starting time of route $k \in K$ in scenario $\omega \in \Omega$
$\rho_k^\omega$	Number of unknown requests assigned to the route serving batch $k \in K$ in scenario $\omega \in \Omega$
$w_i^{k\omega}$	Binary decision variable equal to 1 if known request $i \in N_e^{known}$ is served in route $k$ in scenario $\omega \in \Omega$ , $k \in K_0^\omega$ , and 0 otherwise
$z_k^\omega$	Binary decision variable equal to 1 if route $k \in K$ in scenario $\omega \in \Omega$ is executed, and 0 otherwise
$\tilde{r}^e$	The distribution of uncertain release dates for unserved requests
$\tilde{V}(y^0, \tau_{end}^0, \tilde{r}^e)$	The value of the recourse function associated with the second stage

Table 7: Literature Classification

Literature	Vehicle	Release dates		Deadline	Objective	MDP	Approach
		Stochastic	Dynamic				
Cattaruzza et al. (2016)	Single	✗	✗	✓	Min(total travel distance)	✗	Hybrid genetic algorithms
Archetti et al. (2015)(1)	Single	✗	✗	✓	Min(total traveling distance within a deadline)	✗	-
Archetti et al. (2015)(2)	Unlimited	✗	✗	✗	Min(the completion time)	✗	-
Reyes et al. (2018)(1)	Single	✗	✗	✓	Min(the completion time)	✗	-
Reyes et al. (2018)(2)	Unlimited	✗	✗	✓	Min(the completion time)	✗	-
Shelbourne et al. (2017)	Multi	✗	✗	✓	Min(a convex combination of the total distance traveled and the total weighted tardiness of delivery)	✗	A path-relinking algorithm based on a hybrid evolutionary framework.
Archetti et al. (2018)	Single	✗	✗	✗	Min(the completion time)	✗	A heuristic approach based on an iterated local search
Archetti et al. (2020)	Single	✓	✓	✗	Min(the completion time)	✓	Two models proposed (stochastic and deterministic), the solution is obtained through an iterated local search
Voccia et al. (2019)	Multi (plus a third party)	✓	✓	✓	Max(#requests served)	✓	Sample-scenario planning and a variable neighborhood search implemented for each scenario
Van Heeswijk et al. (2019)	Multi (plus a third party)	✓	✓	✓	Min(the total dispatching costs)	✓	Approximate dynamic programming using a linear value function approximation to estimate the downstream costs
Klapp et al. (2018)	Single	✓	✓	✓	Min(sum of total travel costs plus penalties for unattended and realized orders)	✓	Three heuristic policies(a priori, direct rollout, roll out a prize-collecting TSP guided by an initial a priori solution)
Schrotenboer et al. (2021)	Multi	✓	✓	✓	Max(the expected customer satisfaction)	✓	A cost function approximation approach that modifies a set-packing formulation

number of direct trips that can be performed within  $T_E$ . Let us assume that there exists a solution  $\tilde{s}$  with a higher number of trips than  $UB_{trips}$ . Given the argument above, at least one of the trip in  $\tilde{s}$  is not a direct trip. Thus, we can arbitrarily remove customers from this trip up to when a single customer remains, making it a direct trip. This procedure can be repeated on all trips containing more than one customer in  $\tilde{s}$ . In this way, we obtain a solution with the same number of trips as in  $\tilde{s}$ , and where all trips are direct trips. However, due to the argument above, the number of trips cannot be greater than  $UB_{trips}$ , and this proves point 1 of the theorem.

As for point 2, feasibility is ensured by constraints (3b): a trip starts not earlier than the delivery date of the customer to be served, and its duration plus the duration of following trips does not exceed  $T_E$ . ■

### 8.5. Algorithm of consensus function

It is described in Algorithm 2.

---

#### Algorithm 2 Consensus Function

---

**Require:**  $N_e^{known}$ ; a set of scenarios  $\Omega$ ; solution  $\tilde{X}_e^\omega$  of scenario  $\omega$ ;

**Output:** a route  $R$ ;

- 1: Initialize the frequency of all requests:  $F_e(i) \leftarrow 0 \ i \in N_e^{known}$ ; Initialize the set of requests to be served as  $I_{sol} \leftarrow \emptyset$ ;
  - 2: **for**  $\omega \in \Omega$  **do**
  - 3:    $I_\omega \leftarrow$  the set of requests served in route 0 of the solution  $\tilde{X}_e^\omega$  of scenario  $\omega$ .
  - 4:   **for**  $i \in I_\omega$  **do**  $F_e(i) \leftarrow F_e(i) + 1$
  - 5: **for**  $i \in N_e^{known}$  **do**
  - 6:   **if**  $F_e(i) \geq \frac{|\Omega|}{2}$  **then**  $I_{sol} \leftarrow I_{sol} \cup \{i\}$
  - 7: **return**  $R \leftarrow$  TSP over  $I_{sol}$
- 

### 8.6. Input Instances

We summarize the procedure used in Archetti et al. (2020) for generating the instances used as benchmark in the current work.

The release dates are generated using a Gaussian distribution with the expectation and the variance obtained by simulating the arrival of the vehicles transporting the packages to the depot. For customers in  $N_e^{dynamic}$  (i.e., the set of customers whose information about release date is dynamically updated along with the parcels' travel to the depot), the release dates are updated by simulating the traveling of the vehicle delivering the parcels to the depot. For each Solomon's instance, a parameter  $\beta \in \{0.5, 1, 1.5\}$  is generated defining the dispersion of release dates. The greater the value of  $\beta$  is, the broader is the interval the release dates are sampled from.

A second parameter  $\delta \in \{0, 0.5, 1\}$  indicates the rate of customers with a dynamically updated release date. When  $\delta$  is 0, it means the release dates of all customers are static.

For each of instance and value of parameters  $\beta$ ,  $\delta$  and  $c$ , five instances have been created by varying the seed for randomly generating the release dates. In our paper, we report the results considering all the five seeds. As for the pairwise Euclidean distances between customer coordinates, they are rounded up to lowest integer values.

### 8.7. Sensitivity Analysis

In this section we present the results of the experiments we made to set a proper value of  $\rho$ , i.e., the maximum number of packages transported in future routes used in the batch approach. We tested four values:  $\rho \in \{5, 10, 15, 20\}$ .

For each of the two approaches  $p \in \{\text{PFA}, \text{VFA}\}$ , and a given input instance, let  $N_{p,\rho}$  denote the number of total requests served, assuming approach  $p$  is applied with a given value of  $\rho$ . Then

$$\Gamma_{p,\rho} = 1 - \frac{N_{p,\rho}}{\max_{\rho \in \{5, 10, 15, 20\}} N_{p,\rho}} \quad (10)$$

provides the gap between the solution obtained for a given value of  $\rho$ , and the best solution among the four different values of  $\rho$ .

In Table 8, we show the percentage gaps for the PFA and VFA approaches, respectively. Each row corresponds to an average calculated over a subset of instances with the fixed value of  $\delta$ ,  $\beta$  and  $c$  parameters on all instances, respectively.

**Table 8** Average percentage gap  $\Gamma_{p,\rho}$  from the best solution found for  $\rho \in \{5, 10, 15, 20\}$  and  $p \in \{\text{PFA}, \text{VFA}\}$ .

	PFA				VFA			
$\rho$	5	10	15	20	5	10	15	20
$\delta = 0$	7.15	5.21	3.36	4.92	6.22	5.20	3.81	5.20
$\delta = 0.5$	6.43	4.54	4.83	5.70	6.19	4.78	4.92	6.38
$\delta = 1$	6.81	4.39	4.40	4.97	6.82	4.27	4.72	4.82
$\beta = 0.5$	12.26	6.69	7.40	9.20	11.99	6.55	8.11	9.60
$\beta = 1$	6.03	5.01	3.31	4.11	5.12	5.00	3.59	4.20
$\beta = 1.5$	2.10	2.44	1.89	2.27	2.13	2.70	1.75	2.61
$c = 0.6$	8.07	6.76	7.58	8.67	7.87	7.12	7.25	9.72
$c = 0.8$	9.66	6.16	4.17	6.32	8.57	5.95	4.78	6.43
$c = 1$	6.24	3.65	3.44	3.30	5.60	3.78	3.49	3.31
$c = 1.2$	3.23	2.28	1.60	2.49	3.60	2.15	2.40	2.42
<b>Avg.</b>	<b>6.80</b>	<b>4.71</b>	<b>4.20</b>	<b>5.20</b>	<b>6.41</b>	<b>4.75</b>	<b>4.48</b>	<b>5.47</b>

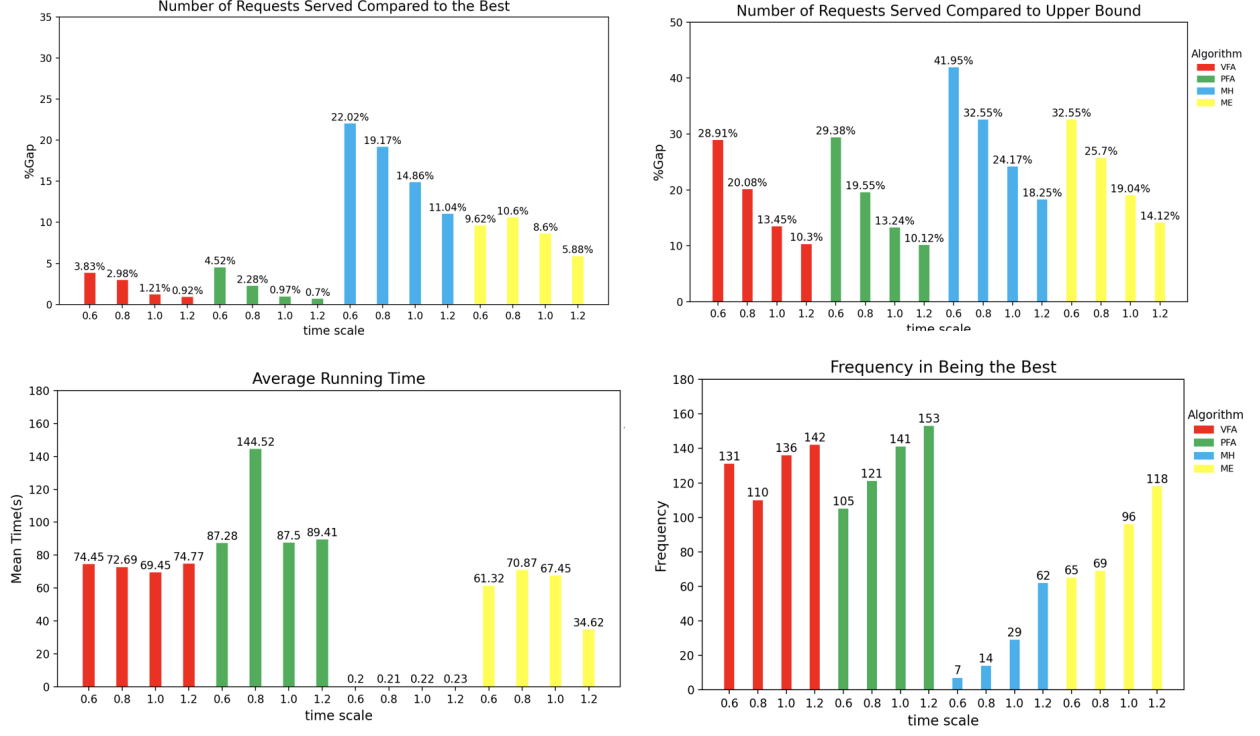
For both PFA and VFA, we see that on average the performance improves when increasing  $\rho$  from 5 to 15, but deteriorates when it increases to 20. The best results are obtained when  $\rho$  is equal to 15. Specifically, when  $\rho = 15$  the gaps are the smallest in most cases while  $\rho = 10$  and 20 show some exceptionally good results in few cases, for example, when  $\delta = 0.5$  in PFA and when  $c = 1.2$  in VFA. Based on the overall results, we opted for a value of  $\rho = 15$  as it provides the best results in general.

## 8.8. Performance of the approaches on time scale

See Table 9 and Figure 4.

**Table 9** The average performance of the four approaches on the basis of  $c$ 

		VFA				PFA				MH				ME			
$c$	#instances	gap[%]	gap_ub[%]	t[s]	freq	gap[%]	gap_ub[%]	t[s]	freq	gap[%]	gap_ub[%]	t[s]	freq	gap[%]	gap_ub[%]	t[s]	freq
0.6	180	3.83	28.91	74.45	131	4.52	29.38	87.28	105	22.02	41.95	0.20	7	9.62	32.55	61.32	65
0.8	180	2.98	20.08	72.69	110	2.28	19.55	144.52	121	19.17	32.55	0.21	14	10.60	25.70	70.87	69
1	180	1.21	13.45	69.45	136	0.97	13.24	87.50	141	14.86	24.17	0.22	29	8.60	19.04	67.45	96
1.2	180	0.92	10.30	74.77	142	0.70	10.12	89.41	153	11.04	18.25	0.23	62	5.88	14.12	34.62	118
<b>AVG</b>	<b>180</b>	<b>2.24</b>	<b>18.18</b>	<b>72.84</b>	<b>129.8</b>	<b>2.12</b>	<b>18.08</b>	<b>102.18</b>	<b>130.0</b>	<b>16.77</b>	<b>29.23</b>	<b>0.21</b>	<b>28.0</b>	<b>8.68</b>	<b>22.85</b>	<b>58.57</b>	<b>87.0</b>



**Figure 4** Performance of the approaches (both PFA and VFA are with partial characterization of optimal policy) on the basis of  $c$

Finally, we compare the effect of the different values of the deadline  $T_E$ . Remember that  $T_E$  is determined as the latest release date in the instance ( $T_{standard}$ ) multiplied by parameter  $c$ .

In Table 9, we see that on average the performance of the four approaches shows an improving trend as the value of  $T_E$  increases. For each  $T_E$ , we observe that PFA gets the highest number of best solutions in almost all cases except when  $c$  is 0.6. On the other hand, we see that when  $T_E = 1.2 \cdot T_{standard}$ , the values of  $freq$  are biggest for all approaches, reflecting the fact that these instances are easier to solve. Finally, we observe that though  $MH$  and  $ME$  provide results in short time, the solution quality is sacrificed.