Contents lists available at ScienceDirect

European Journal of Operational Research

Production, Manufacturing, Transportation and Logistics

Solving large-scale dynamic vehicle routing problems with stochastic requests

Jian Zhang [a,b], Kelin Luo [c,d], Alexandre M. Florio [b,*], Tom Van Woensel [b]

[a] *Department of Electronic Business, South China University of Technology, Guangzhou 510006, PR China*
[b] *Department of Industrial Engineering and Innovation Sciences, Eindhoven University of Technology, Eindhoven 5600MB, Netherlands*
[c] *Institute of Computer Science, University of Bonn, D-53115 Bonn, Germany*
[d] *Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven 5600MB, Netherlands*

A R T I C L E   I N F O

A B S T R A C T

Dynamic vehicle routing problems (DVRPs) arise in several applications such as technician routing, meal delivery, and parcel shipping. We consider the DVRP with stochastic customer requests (DVRPSR), in which vehicles must be routed dynamically with the goal of maximizing the number of served requests. We model the DVRPSR as a multi-stage optimization problem, where the first-stage decision defines route plans for serving scheduled requests. Our main contributions are knapsack-based linear models to approximate accurately the expected reward-to-go, measured as the number of accepted requests, at any state of the stochastic system. These approximations are based on representing each vehicle as a knapsack with a capacity given by the remaining service time available along the vehicle's route. We combine these approximations with optimal acceptance and assignment decision rules and derive efficient and high-performing online scheduling policies. We further leverage good predictions of the expected reward-to-go to design initial route plans that facilitate serving dynamic requests. Computational experiments on very large instances based on a real street network demonstrate the effectiveness of the proposed methods in prescribing high-quality offline route plans and online scheduling decisions.

1. Introduction

We consider the dynamic vehicle routing problem with stochastic requests (DVRPSR), a fundamental problem in freight transport. In the multi-vehicle DVRPSR, a set of *static* requests is known in advance, while other requests (hereafter, *dynamic* requests) arrive randomly during the service period. Vehicles depart from a given location, and must return to that location before the end of the service period. The goal is to serve as many customer requests as possible by assigning an initial route plan to each vehicle, and by routing vehicles dynamically in response to new requests.

Previous research on the DVRPSR is mostly restricted to small- and medium-scale instances on synthetic graphs with Euclidean or Manhattan distances Azi, Gendreau, & Potvin (e.g., 2012); Klapp, Erera, & Toriello (e.g., 2018); Ulmer, Goodson, Mattfeld, & Hennig (e.g., 2019); Ulmer, Mattfeld, & Köster (e.g., 2018a); Voccia, Campbell, & Thomas (e.g., 2019). This paper scales the DVRPSR up to practical sizes: we address the dynamic and real-time routing of

multiple vehicles on a real street network with more than 16,000 nodes (intersections), each corresponding to a potential customer location. Real-time decision-making's tight time frames pose considerable challenges when solving the DVRPSR in our large-scale setting.

A solution to the DVRPSR consists of an initial route plan and an online scheduling policy. The former is computed *offline*, i.e., before the beginning of the service period, and defines the initial vehicle routes covering all static requests. In many recently developed DVRPSR solution methods, initial route plans are computed by myopic heuristics such as cheapest insertion (CI) and savings heuristic van Heeswijk, Mes, & Schutten (e.g., 2019); Ulmer et al. (e.g., 2019); Ulmer et al. (e.g., 2018a), which attempt to minimize the travel time required for serving static requests, but completely ignore the probable realization of dynamic requests. Ideally, however, the initial route plan should also facilitate the acceptance of dynamic requests during the service period. As shown by Bent & Van Hentenryck (2004) and Ferrucci & Bock (2016), exploiting the stochastic knowledge of dynamic requests can improve the quality of the initial routes.

In addition to the initial route plan, the DVRPSR requires an online policy to quickly determine acceptance/rejection decisions,

* Corresponding author.
  E-mail addresses: jianzhang@scut.edu.cn (J. Zhang), kluo@uni-bonn.de (K. Luo), a.de.macedo.florio@tue.nl (A.M. Florio), t.v.woensel@tue.nl (T. Van Woensel).

request-vehicle assignments, and updated routes. Value function approximation (VFA) and rollout algorithms are the most widely used approximate dynamic programming (ADP) methods for computing online policies van Heeswijk et al. (e.g., 2019); Klapp et al. (e.g., 2018); Ulmer et al. (e.g., 2019). Both methods estimate the expected reward-to-go by simulation and can effectively solve small DVRPSR instances. However, when the problem scale becomes realistically large, the application of many VFA policies (e.g., those developed in Ulmer et al. (2018a) and Ulmer et al. (2019)) is restrained by the high dimensionality of lookup tables. Rollout algorithms are unsuitable for large-scale problems as well, since they perform all simulations online and their solution quality and computation time depend largely on the number of simulated scenarios.

Our key methodological contributions are knapsack-based linear models to approximate the expected reward-to-go at any state of the dynamic system. We derive these models by representing each vehicle as a knapsack whose capacity is given by the remaining available service time (or *budget*) along the vehicle's route. We assess the cost (or budget consumption) of assigning future requests to a vehicle by predicting the vehicle's future location along its planned route. We combine these approximations with the decision rules of an optimal policy in order to derive high-performing and efficient scheduling policies. Moreover, we demonstrate how these approximations can be used to select initial route plans that facilitate the acceptance of requests in the dynamic phase.

To summarize, the main contributions of this paper are the following:

1. We propose a sequential optimization model for the DVRPSR, which jointly optimizes the offline decision concerning the initial route plan and the online decisions for scheduling dynamic requests. This is the first DVRPSR model that incorporates, at the same time, multiple vehicles, a real street network, and real-time decision-making. We further characterize the decision rules of optimal scheduling policies.
2. We develop interpretable knapsack-based approximations of the expected reward-to-go, which accurately and efficiently predict the expected number of future accepted requests from any state in our DVRPSR model. These approximations improve the service area coverage achieved by the initial route plan and, in combination with the optimal decision rules, lead to high-performing and efficient online scheduling policies.
3. We demonstrate by an extensive computational study on large-scale instances the merits of initial route plans that consider the spatiotemporal distribution of dynamic requests. Further, we show that the scheduling policies based on the knapsack approximations are suitable for real-time decision-making and, at the same time, outperform conventional ADP methods widely applied to solve dynamic vehicle routing problems (DVRPs).
4. We make freely available the implementation of a generic DVRPSR simulator (see Fig. 1 for a snapshot and https://youtu.be/D57xNfU73as for animated simulations) and a set of standard benchmark instances to allow reproducibility and comparison of results.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 introduces a sequential optimization model for the DVRPSR. Section 4 and Section 5 elaborate on online scheduling policies and offline planning algorithms, respectively. The results of the computational study are presented and discussed in Section 6. Finally, Section 7 draws concluding remarks and proposes directions for further research.

2. Literature review

A detailed review of contributions to the DVRPSR is presented in Section 2.1. The applications of ADP to DVRPs are discussed in Section 2.2.

2.1. Dynamic vehicle routing with stochastic requests

We first focus on the previous research on the DVRPSR. Table 1 provides an overview of the relevant contributions. In terms of problem settings, we indicate, for each paper, whether decisions are made in real-time (i.e., whether decision epochs are triggered by request arrivals), the maximum expected number of requests, the maximum number of vehicles, and the graph type (synthetic graph or street network). A "n/a" (not applicable) in the column "Veh." means that fleet size is not constrained. Table 1 also summarizes the methodologies employed for computing initial route plans and online scheduling policies. A "n/a" in the column "Initial plan" indicates that static requests are not considered.

Early research on the DVRPSR proposes and evaluates several policies which follow certain decision rules (e.g., first-come, first-served) and ignore probabilistic knowledge about future requests when prescribing decisions (e.g., Bertsimas & Van Ryzin, 1991; Gendreau, Guertin, Potvin, & Taillard, 1999; Larsen, Madsen, & Solomon, 2002; Tassiulas, 1996). Chen & Xu (2006) develop a dynamic column generation framework to solve a DVRPSR. Assuming that no probabilistic information of future requests is available, they generate single-vehicle routes (i.e., columns) continuously and solve a set partitioning model periodically with existing columns. Bent & Van Hentenryck (2004), Hvattum, Løkketangen, & Laporte (2006), and Ichoua, Gendreau, & Potvin (2006) are the earliest works to show that exploiting stochastic information of dynamic requests leads to significantly better initial route plans and online scheduling policies. The multiple scenario approaches (MSAs) presented in Bent & Van Hentenryck (2004) and Voccia et al. (2019) compute and reoptimize route plans for multiple scenarios, each corresponding to a static VRP involving known requests and randomly sampled future requests. From multiple route plans, the optimal one is selected by consensus functions. Ichoua et al. (2006) and Ferrucci & Bock (2015, 2016) integrate randomly sampled future requests into tabu search metaheuristics for static VRPs, by which anticipatory DVRPSR policies are derived.

MSAs and anticipatory (meta-)heuristics are applied to DVRPSRs by decomposing the problem into sequential static VRPs Ferrucci & Bock (e.g., 2015, 2016); Hvattum et al. (e.g., 2006); Klapp et al. (e.g., 2018). In these methods, decision epochs occur at the end of fixed time intervals. Thus, there is usually sufficient computation time for reoptimizing static VRPs, but new requests must wait until the end of the current time interval before being scheduled or rejected. To improve customer responsiveness, the decision epochs in many DVRPSRs are triggered by events, e.g., when a vehicle arrives at a node or when a new request arrives. In these cases, sequential stochastic optimization models are more suitable than sequential static VRP formulations. For example, Thomas (2007) and Ulmer et al. (2018a) propose Markov decision processes for single-vehicle DVRPSRs, where a decision epoch occurs every time the vehicle stops at a customer location or returns to the depot. At each epoch, a subset of the newly arrived requests is accepted, and the vehicle's route is updated accordingly.

In most DVRPSRs, some dynamic requests may be rejected due to the limited number of vehicles and hard deadlines of service periods Klapp et al. (e.g., 2018); Ulmer et al. (e.g., 2018a); Voccia et al. (e.g., 2019). It is hence necessary to consider customers' expectations for quick responses from the service provider. If a decision epoch occurs only when a vehicle stops or at predetermined time instants, most responses to dynamic requests will be delayed.
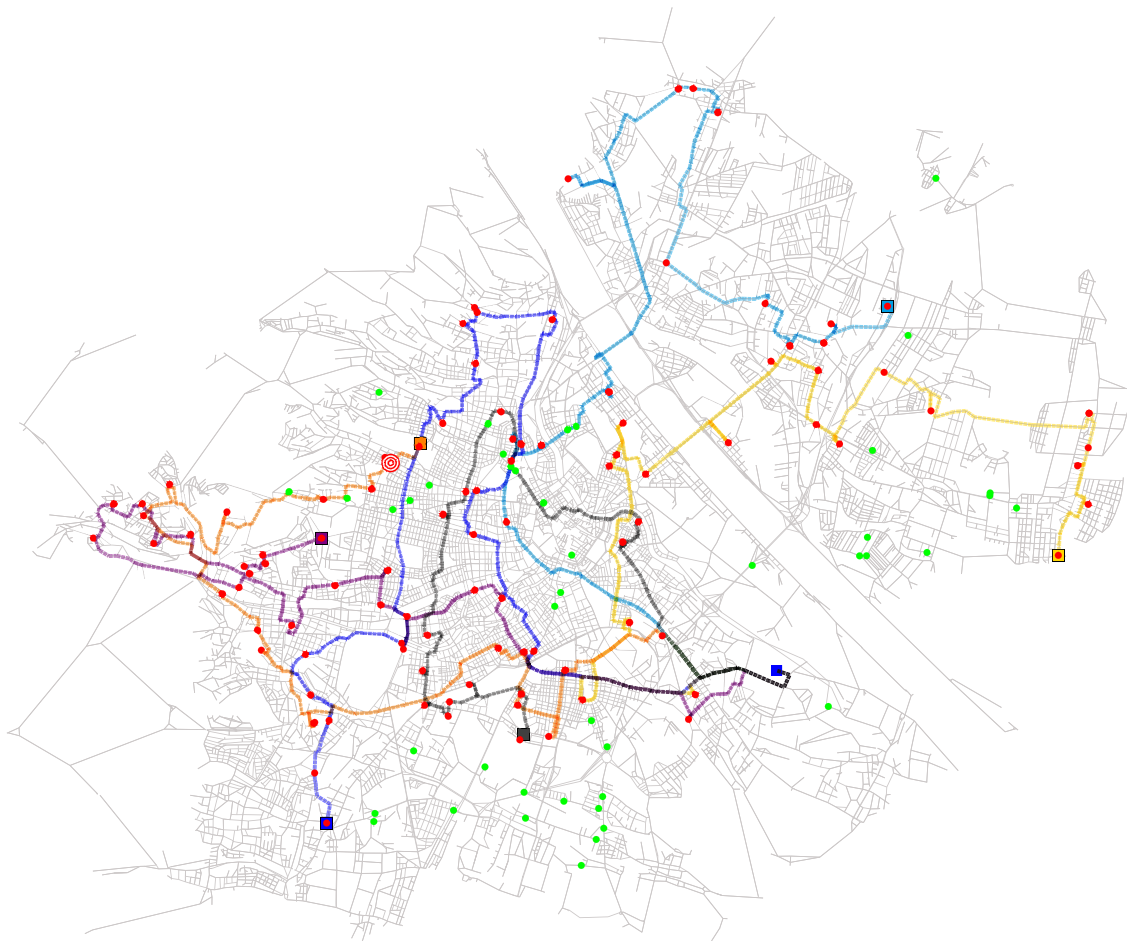
Fig. 1. A Large-scale DVRPSR Instance. *Notes.* The green dots represent the served requests; the red dots represent the accepted requests that have not yet been served; the red circle represents a dynamic request that has just arrived; the blue square in the southeast represents the depot; the other squares represent vehicles' current locations and the lines represent the remaining planned routes (each vehicle is associated with a specific color). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
DVRPSR: Classification of the Relevant Literature.

Literature	Problem setting				Initial plan		Online policy	
	RT	Req.	Veh.	Graph	Method	Ant.	Method	Ant.
Bent & Van Hentenryck (2004)	×	100	n/a	Synth.	MSA	✓	MSA	✓
Chen & Xu (2006)	×	100	n/a	Synth.	CGBH	×	CGBH	×
Hvattum et al. (2006)	×	130	n/a	Synth.	DSHH	✓	DSHH	✓
Ichoua et al. (2006)	✓	240	6	Synth.	TS	✓	TS	✓
Thomas (2007)	×	50	1	Synth.	GRASP	×	Heuristics	✓
Azi et al. (2012)	✓	144	5	Synth.		n/a	ALNS	✓
Ferrucci & Bock (2015, 2016)	×	150	12	SSSN	TS	✓	TS	✓
Klapp et al. (2018)	×	40	1	Synth.	B&C	✓	Rollout	✓
Ulmer et al. (2018a)	×	100	1	Synth.	CI	×	VFA	✓
Ulmer, Soeffker, & Mattfeld (2018b)	×	100	1	Synth.	CI	×	VFA	✓
Ulmer et al. (2019)	×	100	1	Synth.	CI	×	VFA+rollout	✓
van Heeswijk et al. (2019)	×	400	n/a	Synth., SSSN	CW	×	VFA, rollout	✓
Voccia et al. (2019)	×	192	13	Synth.		n/a	MSA	✓
Ulmer (2020)	✓	180	3	Synth.		n/a	VFA	✓
Ulmer & Thomas (2020)	✓	50	1	Synth.		n/a	VFA	✓
This paper	**✓**	**947**	**20**	**LSSN**	**PbCGBH**	**✓**	**PbPs**	**✓**

Notes. RT, real-time; 'Req.', requests; 'Veh.', vehicles; 'Ant.', anticipation; 'Synth.', synthetic; SSSN, small-scale street network; LSSN, large-scale street network; MSA, multiple scenario approach; CGBH, column-generation-based heuristic; DSHH, dynamic stochastic hedging heuristic; TS, tabu search metaheuristic; GRASP, greedy randomized adaptive search procedure; ALNS, adaptive large neighborhood search; B&C, branch-and-cut; CI, cheapest insertion; CW, Clarke-Wright savings heuristic; PbCGBH, potential-based CGBH; PbPs, potential-based policies.

Column "RT" of Table 1 shows that only 4 models define decision epochs as the moments when new requests arrive. Among these, Azi et al. (2012) spend more than 200 seconds to compute a decision for small instances with 3 vehicles and 144 expected requests, whereas in the remaining works, the computation times are not specified.

Concerning problem scales, most DVRPSR instances in the previous research are defined on synthetic graphs with modest numbers of requests and vehicles. In van Heeswijk et al. (2019), several instances based on the real street network of Copenhagen (Denmark) are solved, but the authors consider only 10 customer locations and hence the problem scale is still limited. Ferrucci & Bock (2015) formulate test instances on the real street network of Dortmund (Germany), but the number of nodes (potential customer locations) or the geographical distribution of dynamic requests is not specified. Therefore, no research has addressed solving large-scale DVRPSR instances in real-time.

2.2. Approximate dynamic programming in DVRPs

Sequential stochastic optimization (also referred to as Markov decision process) is a commonly used mathematical model for DVRPs (including DVRPSRs) (Pillac, Gendreau, Guéret, & Medaglia, 2013; Psaraftis, Wen, & Kontovas, 2016; Ritzinger, Puchinger, & Hartl, 2016; Soeffker, Ulmer, & Mattfeld, 2022). However, in practice, solving these models exactly by classical dynamic programming is computationally prohibitive because of the "curses of dimensionality". Consequently, a variety of ADP methods have been developed for DVRPs. The basic idea of ADP is to make decisions according to the estimated reward-to-go, which is generally obtained by simulation. We refer the interested reader to Powell (2011) and Powell, Simao, & Bouzaiene-Ayari (2012) for tutorials on the different ADP methods and their applications to various transportation and logistics problems.

Powell (2011) classifies ADP into four broad categories: myopic policies, look-ahead policies, value function approximation (VFA), and policy function approximation (PFA). Myopic policies guide the selection of decisions in such a way that immediate rewards are maximized. For DVRPs, these policies tend to perform poorly because they do not take into account any forecast information about dynamic requests. Look-ahead policies, on the contrary, approximate the reward-to-go by explicitly sampling and simulating the future. Rollout policies are a typical look-ahead policy class initially proposed by Bertsekas & Tsitsiklis (1996). For applications to DVRPSRs, Ulmer et al. (2019) demonstrate that any base policy with deterministic decision rules can be improved by rollout algorithms. However, look-ahead policies have the critical shortcoming that they perform all simulations online, and hence their performance is highly restricted by the tight time frames of DVRPSRs. For example, the rollout policies employed in Ulmer et al. (2019) simulate 16 scenarios at each decision epoch. For instances with only 100 expected requests on a Euclidean plane, the maximum computation time for selecting a decision exceeds 6 minutes, which is a far too long response time when considering real-time applications.

In contrast to look-ahead policies, VFA policies conduct all simulations offline (i.e., a priori) and hence are more suitable for the applications with limited online computation time. The main component of a VFA policy is an approximation of the value function which captures the expected reward-to-go. Powell et al. (2012) establish a unified VFA framework to address the very large state spaces of DVRPs. Based on this framework, Ulmer et al. (2018a), van Heeswijk et al. (2019), and Ulmer & Thomas (2020) propose non-parametric VFA (N-VFA), parametric VFA (P-VFA), and meso-parametric VFA (M-VFA) policies, respectively, to approximate value functions for DVRPSRs. The main drawback of these VFA policies is the high computational complexity. Both N-VFA and M-VFA rely on lookup tables (completely or partially) which suffer from the curses of dimensionality (Powell & Meisel, 2015). P-VFA requires less computational effort, but its linear basis function ignores non-linearity and cannot approximate the value function in every detail (Ulmer & Thomas, 2020).

The last category of ADP policies, PFA, is similar to VFA in the sense that all simulations are conducted offline, but is easier to implement because it usually only requires tuning the value of a parameter rather than approximating a complex value function. PFA policies are suitable for the cases where the structure of a good decision policy can be easily observed and captured by an analytical function, which returns a decision given a state (Powell et al., 2012). A typical PFA policy is developed by Ulmer & Streng (2019) for a dynamic dispatching problem, in which every vehicle can only perform direct trips between a depot and one of several parcel lockers. At each fixed time point, the policy identifies the locker with the largest amount of parcels to be delivered. If this amount is larger than a threshold value, a vehicle is dispatched to serve that locker; otherwise, all available vehicles wait at the depot. The threshold balances the trade-off between early dispatch for fast deliveries and late dispatch for consolidation, and extensive offline simulations determine its value. For a more complex dynamic dispatching problem with both vehicles and drones, Ulmer & Thomas (2018) propose a PFA policy based on the intuition that it is beneficial to serve remote customers by drones and serve nearby customers by vehicles. With offline simulations, they tune a threshold to measure whether each customer is far enough from the depot so that it should be served by a drone.

The three categories of anticipatory ADP policies (i.e., look-ahead, VFA, and PFA) require a sufficiently large number of online or offline simulations. As a result, it is hard to balance their computational efficiency and solution quality. To the best of our knowledge, there is no proven method to solve accurately large-scale DVRPSR instances while respecting the tight time frames imposed by real-time decision-making. This paper fills this gap by proposing accurate and efficient approximations of the reward-to-go, which lead to practical algorithms to solve large-scale DVRPSRs in real-time. Moreover, the proposed approximations are based on sample scenarios and hence can better cope with the heterogeneity of stochastic information (e.g., time-varying request rates) than VFA (Soeffker et al., 2022). With a set of VFA and rollout policies as benchmarks, we validate through an extensive computational study that the proposed algorithms outperform the conventional ADP policies in terms of both efficiency and quality.

3. Problem definition

In this section, we formally introduce the DVRPSR. Section 3.1 presents general definitions, and Section 3.2 models the DVRPSR as a sequential stochastic optimization problem. The notations used in this section are summarized in Table A.1 in Appendix A.

3.1. General definitions

The DVRPSR is defined on a strongly connected digraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, \ldots, V\}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. Graph $\mathcal{G}$ represents the street network of the service area, and sets $\mathcal{V}$ and $\mathcal{A}$ represent, respectively, road intersections and road segments. We assume that all potential customers are located at intersections. The length of each segment $(i, j) \in \mathcal{A}$ is given by d_{ij}. The service period is represented by the interval $[0, U]$. We use interchangeably the terms *time* and *instant* to refer to a value $u \in [0, U]$. Before the service period begins, a fleet of K vehicles is

stationed at node 0, which corresponds to the location of the depot. Vehicles travel at a constant speed $\bar{s}$, so the travel time along a segment (i, j) is given by $t_{ij} = d_{ij}/\bar{s}$. Further, the duration (under speed $\bar{s}$) of the fastest path from node i to node j is given by $t(i, j)$. If nodes i and j are directly connected by a road segment $(i, j) \in \mathcal{A}$, then $t(i, j) = t_{ij}$; otherwise, $t(i, j) = t_{ik_1} + t_{k_1 k_2} + \ldots + t_{k_n j}$, where $k_1, \ldots, k_n \in \mathcal{V}$ are the nodes along the shortest path from node i to node j.

A customer request consists of a triple (u, i, d), where $u \in [0, U]$ is the request arrival time (alternatively, the time when the request is *generated* by the customer), $i \in \mathcal{V} \setminus \{0\}$ is the customer location, and $d \in \mathbb{R}_{>0}$ is the *duration* of the request, i.e., the service time required by the customer to fulfill the request. A set $\mathcal{S}$ of static requests is known before the beginning of the service period. Static requests must be served by the end of the service period. In addition, dynamic requests arrive randomly during the service period according to a non-homogeneous Poisson process with a request rate $\Lambda(u) = \sum_{i \in \mathcal{V}/\{0\}} \lambda_i(u)$, where $\lambda_i(u)$ are node specific request rates. We denote by $\mathcal{D}(u)$ the ordered (by arrival time) set of dynamic requests generated up to instant u. Hence, $\{|\mathcal{D}(u)|\}_{u \in [0, U]}$ is the counting process associated with the request arrival process $\{\Lambda(u)\}_{u \in [0, U]}$, so $|\mathcal{D}(u)|$ is a random variable indicating the number of dynamic requests generated up to instant u. Finally, the duration of a dynamic request is known at the moment it arrives, and so are the durations of all static requests. The durations of future (not yet generated) requests follow a known probability distribution.

Once a dynamic request arrives, the scheduling policy in effect decides whether to accept or to reject the request. Accepted requests must be served before the end of the service period, and rejected requests are lost (it is assumed that rejected requests are served by competitors). The scheduling policy must take a decision in an online fashion, that is, almost immediately after a dynamic request arrives. A static or accepted dynamic request (u, i, d) is assigned to a vehicle $k \in \{1, \ldots, K\}$. The request is served by routing vehicle k to location i and keeping the vehicle there for an amount of time d. Once a request is served, the vehicle proceeds to the location of another request, or back to the depot if there are no further requests assigned to that vehicle. In particular, waiting is not allowed, except when a vehicle is idle at the depot. The goal in the DVRPSR is to route the K vehicles, so that all requests in $\mathcal{S}$ are served, all vehicles return to the depot no later than instant U, and the number of dynamic requests accepted and served is maximized.

3.2. A Sequential Stochastic optimization model for the DVRPSR

We now propose a model for the DVRPSR inspired by the unified ADP framework of Powell et al. (2012) and by the route-based Markov decision processes proposed by Ulmer, Goodson, Mattfeld, & Thomas (2020) for modeling DVRPs.

3.2.1. Decision epochs and states

Decision epochs correspond to the moments when decisions must be taken. In the DVRPSR, in addition to a first decision concerning the initial route plan to serve requests in $\mathcal{S}$, a decision is also made whenever a dynamic request arrives. Hence, there are in total $1 + |\mathcal{D}(U)|$ decision epochs, where $|\mathcal{D}(U)|$ is uncertain.

The state of the system at the moment a decision must be made is represented by state variables S_t, $t \in \{0, \ldots, T\}$, where $T = |\mathcal{D}(U)|$ is the last decision epoch. State S_0 represents all parameters of the problem, since no uncertainty has been realized before the beginning of the service period. Next, we formally define routes, which are instrumental for characterizing all other states S_t, $t \neq 0$:

Definition 1 (Route). A route is a sequence of pairs $\theta = ((v_0, \delta_0), \ldots, (v_f, \delta_f))$, where $v_0 = v_f = 0$, $(v_i, v_{i+1}) \in \mathcal{A}$,

$i \in \{0, \ldots, f - 1\}$, $\delta_0 = \delta_f = \emptyset$, and δ_i, $i \in \{1, \ldots, f - 1\}$, is the (possibly empty) set of requests to serve when arriving at the i-th node in θ. For notational convenience, $\Delta(\theta) = \cup_{i=1}^{f-1} \delta_i$ denotes the set of all requests served by θ, and $\Delta_u(\theta) = \{(u', i, d) \in \Delta(\theta) : u' \geq u\}$ denotes the ordered (by u') set of all requests served by θ from instant u onwards.

Given a route $\theta = ((v_0, \delta_0), \ldots, (v_f, \delta_f))$ and a starting time $\tau \in [0, U]$, the *budget* $b(\tau, \theta)$ of route θ when it starts at instant τ is defined as the slack time relative to the end of the service period:

$$b(\tau, \theta) = U - \tau - \sum_{i=0}^{f-1} t_{v_i, v_{i+1}} - \sum_{(u,i,d) \in \Delta(\theta)} d.$$

A state S_t, $t \in \{1, \ldots, T\}$, is a $(K + 1)$-tuple $S_t = (V_1, \ldots, V_K, r_t)$, where V_k are vehicle states and $r_t = (u_t, i_t, d_t)$ is the t-th element of $\mathcal{D}(U)$. Each vehicle k is either in an idle state at the depot or in a non-idle state executing a route θ_k. We define the state of an idle vehicle as $V_k = \emptyset$, and define the state of a non-idle vehicle as $V_k = (\tau_k, \theta_k)$, where τ_k is the instant when vehicle k started route θ_k. At any instant, a non-idle vehicle must be either in transit (traveling along a road segment) or in service (fulfilling a request at a customer location). As u_t, d_t, and τ_k are continuous variables, the number of possible states is infinite. The idea of incorporating route information into the state space was originally proposed by Ulmer et al. (2020), and is particularly useful as our solution methods rely heavily on the information of planned routes to guide online decisions.

Since travel times and service times of accepted requests are deterministic, a state collects sufficient information to define precisely the locations of all vehicles. In particular, given a state it is possible to identify the status of each vehicle (idle, in transit or in service), to compute the remaining time before a vehicle reaches the endpoint of the road segment along which it currently travels (if the vehicle is in transit), or to compute the remaining time that a vehicle will stay at a particular node serving requests (if the vehicle is in service).

3.2.2. Offline and online decisions

It is convenient to discuss separately the decision at S_0 and the decisions at other states S_t, $t \neq 0$, since they are structurally different. We denote by $\mathcal{F}$ the feasible decision set at state S_0. Set $\mathcal{F}$ contains all sets of routes $\{\theta_1, \ldots, \theta_{k'}\}$, $k' \leq K$, such that $b(0, \theta_1), \ldots, b(0, \theta_{k'}) \geq 0$ and $\Delta(\theta_1), \ldots, \Delta(\theta_{k'})$ is a partition of $\mathcal{S}$. We represent generically the decision made at S_0 by a decision vector $\mathbf{y} \in \mathcal{F}$. This decision corresponds to an initial route plan and is computed offline (i.e., before the start of the delivery period) by, e.g., one of the algorithms described in Section 5.

We now consider a state $S_t = (V_1, \ldots, V_K, r_t)$, $t \neq 0$. That is, a state where the dynamic request $r_t = (u_t, i_t, d_t)$ has just arrived and an online decision must be taken, in real-time. Given S_t, a *scheduling policy* is the set of rules that determines:

(i) An *acceptance* decision: this is an 'accept' or 'reject' decision relative to request r_t.
(ii) An *assignment* decision: if the request is accepted, this decision allocates the request to a vehicle $k \in \{1, \ldots, K\}$.
(iii) A *routing* decision: when the request is allocated to vehicle k, this decision defines how route θ_k will be adjusted (in case $V_k = (\tau_k, \theta_k)$) or initialized (in case $V_k = \emptyset$) to accommodate the accepted request.

There are $1 + K$ combinations of acceptance and assignment decisions, but in general there are many more possibilities concerning routing decisions, especially when considering the road network graph. The set of possible decisions when the system is at state S_t is denoted by $\mathcal{X}_t$. Each decision $x_t \in \mathcal{X}_t$, $t \neq 0$, specifies

acceptance and, when required, also assignment and routing decisions. Formally, a (deterministic) scheduling policy maps every possible state S_t, $t \neq 0$, to a decision $x_t \in \mathcal{X}_t$. We further denote by $X^\pi(S_t)$ the decision prescribed by policy π when at state S_t. Once a decision is made at state S_t, $t \neq T$, the system evolves and the next decision epoch occurs when the $(t+1)$-th request arrives. State S_{t+1} is specified by the preceding state S_t, the decision $x_t \in \mathcal{X}_t$, and the $(t+1)$-th request $r_{t+1} = (u_{t+1}, i_{t+1}, d_{t+1})$. This functional relationship is represented by the transition function $S^M(\cdot)$:

$$S_{t+1} = S^M(S_t, x_t, r_{t+1}).$$

3.2.3. Rewards and objective function

We denote by $R(S_t, x_t)$ the reward associated with decision $x_t \in \mathcal{X}_t$ when at state S_t. At state S_0, any decision $\mathbf{y} \in \mathcal{F}$ has a reward of 0. At any other state S_t, $t \neq 0$, the reward of a decision is 1 if the dynamic request (u_t, i_t, d_t) is accepted, and 0 otherwise.

The previous definitions allow us to propose the following sequential optimization model (SOM) for the DVRPSR:

$$\max_{\pi \in \Pi, \mathbf{y} \in \mathcal{F}} \mathbb{E}\left[\mathbb{E}\left[\sum_{t=1}^{T} R(S_t, X^\pi(S_t)) \,\Big|\, S_1 \right] \right], \tag{DVRP-SOM}$$

where Π is the set of all scheduling policies. The double expectation is required because state $S_1 = S^M(S_0, \mathbf{y}, r_1)$ is unknown, since it depends on the (also unknown) first dynamic request $r_1 = (u_1, i_1, d_1)$.

Model (DVRP-SOM) maximizes the expected total reward (or the expected number of accepted requests) conditional on the state resulting from the initial decision $\mathbf{y}$. In this work, we are concerned with finding a good policy π as well as a good initial plan $\mathbf{y}$ to be used in combination with π. The online and offline algorithms that we propose in Sections 4 and 5, respectively, are based on the idea of the *potential* $\Phi_\pi(S_t)$ of a state S_t, $t \neq 0$, which is defined as the expected reward-to-go once the system occupies state S_t and is controlled by policy π:

$$\Phi_\pi(S_t) = \mathbb{E}\left[\sum_{t'=t}^{T} R(S_{t'}, X^\pi(S_{t'})) | S_t \right], \qquad t \neq 0. \tag{1}$$

When policy π is relatively simple, (1) can be estimated in a reasonable time by simulation and sample average approximation. An example of such a simple policy is a greedy policy that accepts a request whenever it is feasible to serve it, and assigns requests to vehicles according to cheapest insertion. We further discuss this policy (and its rollout version) in Section 4.6. In the following section, we describe computationally efficient approximations of $\Phi_\pi(S_t)$ when policy π is anticipatory, that is, when it takes into account the spatiotemporal distribution of future requests and prescribes acceptance and assignment decisions accordingly.

4. Online scheduling policies

In this section, we develop several scheduling policies. Across all policies, routing decisions are prescribed according to one of the two routing policies defined in Section 4.1. Section 4.2 characterizes acceptance and assignment decisions in the optimal scheduling policy. Next, in Section 4.3 we derive our potential approximation model following a sequence of interpretable steps. Sections 4.4 and 4.5 introduce two potential-based scheduling policies, and Section 4.6 presents greedy and rollout benchmark policies. All notations used in this section are summarized in Table A.2 in Appendix A.

4.1. Routing policies

When a dynamic request is accepted and assigned to a vehicle in a routing state, it is reasonable to schedule the new request to be served in a position such that the budget of the adjusted route is maximized. Depending on whether it is allowed to reorder unserved requests, this leads to two main routing policies, which are defined next:

Definition 2. (Cheapest Insertion (CI) Routing Policy ρ_{CI}). Consider a request $r = (u, i, d)$ and a vehicle state $V = (\tau, \theta)$. Under the CI routing policy ρ_{CI}, route θ is adjusted into a new route $\rho_{\mathrm{CI}}(\theta, r)$ in such a way that $\Delta_u(\rho_{\mathrm{CI}}(\theta, r)) \setminus \Delta_u(\theta) = \{r\}$, $\Delta_u(\theta)$ is a subsequence of $\Delta_u(\rho_{\mathrm{CI}}(\theta, r))$ and the budget $b(\tau, \rho_{\mathrm{CI}}(\theta, r))$ is maximized.

Definition 3. (Reoptimization Routing Policy ρ_{R}). Consider a request $r = (u, i, d)$ and a vehicle state $V = (\tau, \theta)$. Under the reoptimization routing policy ρ_{R}, route θ is adjusted into a new route $\rho_{\mathrm{R}}(\theta, r)$ in such a way that $\Delta_u(\rho_{\mathrm{R}}(\theta, r)) \setminus \Delta_u(\theta) = \{r\}$ and the budget $b(\tau, \rho_{\mathrm{R}}(\theta, r))$ is maximized.

Given a route θ and a request r, $\rho_{\mathrm{CI}}(\theta, r)$ can be computed in polynomial time on $|\theta|$, while computing $\rho_{\mathrm{R}}(\theta, r)$ requires solving a traveling salesman problem with $|\Delta_u(\theta)| + 1$ nodes.

4.2. Characterizing the optimal scheduling policy

Given a first stage decision $\mathbf{y}$, the optimal scheduling policy π^* is such that:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left[\Phi_\pi(S^M(S_0, \mathbf{y}, r_1)) \right].$$

We now define a class of potential-based policies $\Pi_P \subset \Pi$ such that acceptance and assignment decisions are prescribed according to the following decision rules (DRs):

Decision Rule 1 (Acceptance Rule). Consider that the system is at state S_t, and let x_t^- be the 'reject' decision with respect to request r_t. Then, a policy $\pi \in \Pi_P$ accepts request r_t if and only if there exists an 'accept' decision $x_t^+ \in \mathcal{X}_t \setminus \{x_t^-\}$ such that:

$$1 + \mathbb{E}\left[\Phi_\pi(S^M(S_t, x_t^+, r_{t+1})) \right] \geq \mathbb{E}\left[\Phi_\pi(S^M(S_t, x_t^-, r_{t+1})) \right]. \tag{DR1}$$

Decision Rule 2 (Assignment Rule). Consider that the system is at state S_t, and let $\{\mathcal{X}_t^k\}$ be a partition of $\mathcal{X}_t \setminus \{x_t^-\}$, in which $\mathcal{X}_t^k$ is the set of decisions where r_t is accepted and assigned to vehicle k. Further, for all $\mathcal{X}_t^k \neq \emptyset$, let

$$x_t^k = \arg\max_{x_t \in \mathcal{X}_t^k} \mathbb{E}\left[\Phi_\pi(S^M(S_t, x_t, r_{t+1})) \right].$$

Then, a policy $\pi \in \Pi_P$ assigns an accepted request r_t to the vehicle k^* such that:

$$k^* = \arg\max_k \mathbb{E}\left[\Phi_\pi(S^M(S_t, x_t^k, r_{t+1})) \right]. \tag{DR2}$$

Essentially, (DR1) determines that request r_t is accepted only if the immediate reward offsets the expected decrease in potential (due to budget consumption) at the (random) state S_{t+1}, and (DR2) determines that, if r_t is accepted, it is assigned to the vehicle such that the expected potential at S_{t+1} is maximized. Note that $\pi^* \in \Pi_P$, otherwise π^* is not optimal. Intuitively, good approximations of $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$, $x_t \in \mathcal{X}_t$, can be used to prescribe sensible decisions according to (DR1) and (DR2). Next, we discuss how to efficiently compute such approximations.

4.3. Multiple-knapsack approximation of $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$

Given a request $r_t = (u_t, i_t, d_t)$, our goal is to approximate $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$ for any decision $x_t \in \mathcal{X}_t$, where $S_{t+1} = (V_1, \ldots, V_K, r_{t+1})$ is random since request r_{t+1} is unknown at the current instant u_t. To this end, we propose an approximation based on a multiple-knapsack model where each vehicle state $V_k = (\tau_k, \theta_k)$ is interpreted as a knapsack with capacity $b(\tau_k, \theta_k)$. The objective is to maximize the number of future requests accepted. The cost (in terms of budget consumption) of serving a future request depends on each vehicle state (or knapsack). In particular, it depends on the vehicle location and the remaining route at the moment the future request arrives. The procedure to predict the location of a vehicle at a future instant is based on the following two observations:

Remark 1 (Late Depot Arrival). Since the objective in the DVRPSR is to maximize the number of accepted requests, vehicles will usually return to the depot at an instant near the end of the service period.

Remark 2 (Request Order Preservation). When following the CI routing policy, the relative order of scheduled requests along a route does not change when a new request is assigned to the route. When following the reoptimization policy, the relative order of already scheduled requests usually changes only slightly, if at all, when a new request is assigned.

Our goal of approximating the expected number of requests that can be served from a given state is related, in an inverse or symmetric fashion, to the problem of approximating the length of the minimum-length tour to visit a given set of nodes. The latter is studied in the stream of literature on routing approximations (Daganzo, 1984a; 1984b). These problems are however static, as they refer to approximating the length of distance-optimal routes to visit nodes whose locations are known at the decision time (i.e., approximating the cost of optimal TSP or VRP solutions). In our case, we approximate instead the maximum number of requests that can be served along planned routes (or a *state*) that *change* over time, given the remaining horizon duration. More importantly, our setting is not only stochastic but also dynamic, as node locations are only revealed over time and not all nodes must be served.

Remarks 1 and 2 suggest the concept of *effective speed*, which is useful to predict the location of a vehicle at future instants. The effective speed is defined as the traveling speed such that a route finishes exactly at instant U, assuming no further requests are added to the route. We denote by $\check{s}_{u_t}(V_k)$ the effective speed of a vehicle with state $V_k = (\tau_k, \theta_k)$ at instant u_t (the formula for computing $\check{s}_{u_t}(V_k)$ is given in B.1). Then, it is reasonable to predict the location of vehicle k at a future instant $u' > u_t$ by, from the current location, following route θ_k at speed $\check{s}_{u_t}(V_k)$ for an amount of time $u' - u_t$. By predicting at the current instant u_t the location of vehicle k at a future instant u', we also estimate the set of nodes along θ_k to be traversed after instant u', which we denote by $\mathcal{V}_{u_t}(V_k, u')$. The procedure to predict the vehicle location and determine $\mathcal{V}_{u_t}(V_k, u')$ requires a simple simulation of θ_k and is detailed in B.1. Set $\mathcal{V}_{u_t}(V_k, u')$ is then used to estimate the cost (or budget consumption) $c_{u_t}(V_k, r')$ when assigning a future request $r' = (u', i', d')$ to a vehicle with state V_k:

$$c_{u_t}(V_k, r') = \min_{j \in \mathcal{V}_{u_t}(V_k, u')} t(j, i') + t(i', j) + d'. \tag{2}$$

In (2), we estimate the budget consumption $c_{u_t}(V_k, r')$ by taking the cheapest cost of inserting the future request r' into the yet untraveled route of vehicle k at future instant u'. According to Remark 2, this cost estimation is sensible under both routing policies ρ_{CI} and ρ_R.

We are interested, at the current instant u_t, in approximating $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$. Note that $S_{t+1} = S^M(S_t, x_t, r_{t+1}) = (V_1, \ldots, V_K, r_{t+1})$, so vehicle states $V_1, \ldots, V_K$ are known and already take into account decision x_t. Let $\omega = \{r_1^\omega, \ldots, r_{T_\omega}^\omega\}$ be a sample path of the request arrival process $\{\Lambda(u)\}_{u \in (u_t, U]}$, and $\overline{K} = \{k \in \{1, \ldots, K\} : V_k \neq \emptyset\}$. Then, at time u_t, after decision x_t is taken, the number of requests in ω accepted by the optimal scheduling policy is approximated by the following model with variables z_{kr}, $k \in \overline{K}$ and $r \in \omega$:

$$\text{(MKA)} \quad \phi_{\pi^*}^\omega(S_{t+1}|x_t) = \max \sum_{k \in \overline{K}} \sum_{r \in \omega} z_{kr},$$

$$\text{s.t.} \quad \sum_{r \in \omega} c_{u_t}(V_k, r) z_{kr} \leq b(\tau_k, \theta_k), \quad k \in \overline{K}, \tag{3}$$

$$\sum_{k \in \overline{K}} z_{kr} \leq 1, \quad r \in \omega, \tag{4}$$

$$0 \leq z_{kr} \leq 1, \quad k \in \overline{K}, r \in \omega. \tag{5}$$

Model (MKA) captures the multiple-knapsack structure of the online phase of the DVRPSR. The objective function measures the total number of requests in ω accepted and assigned to a vehicle. For each vehicle, constraints (3) limit the total budget consumption due to assigned requests. These constraints consider the predicted vehicle locations, as determined by effective speeds and planned routes, and charge a vehicle-dependent cost for each accepted request, based on scheduling the request in a position such that budget consumption is minimum. Constraints (4) forbid assigning a request to more than one vehicle. Idle vehicles are ignored in model (MKA), since it is not possible to predict their future locations with the concept of effective speed. Unless the fleet size is over-dimensioned, all vehicles remain busy during the service period and are considered in the model.

We consider continuous instead of binary variables in (5), so we are able to solve model (MKA) fast in a real-time context where computational time is scarce. This is also justified since, first, the coefficients of the formulation have already been approximated, so solving (MKA) with integer variables would not necessarily produce a better approximation; second, $|\omega|$ is usually much larger than $|\overline{K}|$, so the difference between the optimal solution values of (MKA) and its integer counterpart is relatively small. These observations lead us to hypothesize that the approximation attained by (MKA) is sensible, which is confirmed empirically. To achieve stability, when approximating $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$ we take the average of solution values obtained over a set $\Omega = \{\omega_1, \ldots, \omega_H\}$ of H sample paths:

$$\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t] \approx \hat{\Phi}_{\pi^*}(S_{t+1}|x_t) = \frac{1}{H} \sum_{\omega \in \Omega} \phi_{\pi^*}^\omega(S_{t+1}|x_t). \tag{6}$$

We finally note that our approximation scheme requires no parameters other than H.

4.4. Potential-based policy (PbP)

The PbP scheduling policy consists in approximating state potentials according to (6) and applying decision rules (DR1) and (DR2). Given a routing policy ρ and a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$, the policy works as follows:

(i) If there is an idle vehicle k and it can serve request r_t, that is, $u_t + t(0, i_t) + d_t + t(i_t, 0) \leq U$, then r_t is accepted, assigned to vehicle k, and the state of vehicle k changes to $V_k = (u_t, \theta_k)$, where θ_k corresponds to the fastest route from the depot to node i_t, and back.

(ii) Otherwise, a set $\tilde{\mathcal{X}}_t = \{\tilde{x}_t^-\}$ of candidate decisions is initialized, where $\tilde{x}_t^-$ is the 'reject' decision related to request r_t.

(iii) For each non-idle vehicle k, $V_k = (\tau_k, \theta_k)$, we compute the adjusted route $\rho(\theta_k, r_t)$. If $b(\tau_k, \rho(\theta_k, r_t)) \geq 0$, then the decision $\tilde{x}_t^k$ of accepting and assigning r_t to k under routing policy ρ is feasible and is added to $\tilde{\mathcal{X}}_t$.

(iv) A decision is selected from $\tilde{\mathcal{X}}_t$ by evaluating decision rules (DR1) and (DR2) based on approximations $\hat{\Phi}_{\pi^*}(S_{t+1}|\tilde{x}_t)$ of $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|\tilde{x}_t]$, $\tilde{x}_t \in \tilde{\mathcal{X}}_t$, computed by (6).

Different variants of the PbP arise, depending on the adopted routing policy. We denote by $\mathrm{PbP}_\mathrm{R}(H)$ the PbP under routing policy ρ_R, when H sample paths are used to approximate state potentials.

4.5. Simplified potential-based policy (S-PbP)

Step (iv) of policy PbP requires solving model (MKA) under H sample paths for each non-idle vehicle that can serve the new request. Additionally, (MKA) is solved H times to evaluate the potential of the state resulting from the reject decision. Hence, in policy PbP, model (MKA) is created and solved up to $H(K+1)$ times before a decision is taken. In very large instances, this may require more computational time than what is available in a real-time setting. An alternative approximation can be computed faster by considering single-knapsack models. Consider a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$. Given a non-idle vehicle k with $V_k = (\tau_k, \theta_k)$ and a sample path $\omega = \{r_1^\omega, \ldots, r_{T_\omega}^\omega\}$ of process $\{\Lambda(u)\}_{u \in (u_t, U]}$, we define the following model with variables z_r, $r \in \omega$:

$$\text{(KA)} \quad p_{u_t}^\omega(V_k) = \max \sum_{r \in \omega} z_r,$$
$$\text{s.t.} \quad \sum_{r \in \omega} c_{u_t}(V_k, r) z_r \leq b(\tau_k, \theta_k),$$
$$0 \leq z_r \leq 1, \quad r \in \omega.$$

Model (KA) overestimates the number of requests from ω that will be assigned to vehicle k, since it does not take into account other vehicles that may compete for the same requests. To compensate for this bias, we define a coefficient α^ω as the ratio between the potential estimated by model (MKA) and model (KA) before accepting or rejecting request r_t:

$$\alpha^\omega = \frac{\phi_{\pi^*}^\omega(S_{t+1}|\tilde{x}_t^-)}{\sum_{V_k \neq \emptyset} p_{u_t}^\omega(V_k)}. \tag{7}$$

Ratio α^ω falls within the unit interval, and is inversely proportional to the amount of competition among vehicles to serve requests. Then, another approximation of $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|\tilde{x}_t^k]$, where $\tilde{x}_t^k$ is an 'accept' decision that assigns request r_t to vehicle k and changes its state to V_k^+, is given by:

$$\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|\tilde{x}_t^k] \approx \frac{1}{H} \sum_{\omega \in \Omega} \alpha^\omega \left(p_{u_t}^\omega(V_k^+) + \sum_{V_l \neq \emptyset, l \neq k} p_{u_t}^\omega(V_l) \right). \tag{8}$$

Policy S-PbP follows by using (8) to approximate the expected potential of states in step (iv) of the procedure described in Section 4.4. Policy S-PbP requires solving model (MKA) H times for the computation of the numerators of (7). Additionally, for each non-idle vehicle that can serve r_t, model (KA) is solved $2H$ times: H times for the 'accept' and H times for the 'reject' decision. Since variables are continuous, the single-knapsack model (KA) can be efficiently solved in $\mathcal{O}(|\omega| \log |\omega|)$ time without invoking a linear programming solver.

Analogously to $\mathrm{PbP}_\mathrm{R}(H)$, we denote by $\text{S-PbP}_\mathrm{R}(H)$ the S-PbP when routing policy ρ_R is adopted and H sample paths are used to approximate potentials.

4.6. Benchmark policies

We discuss a few benchmark policies in this section.

4.6.1. Greedy policy

Given a routing policy ρ and a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$, the greedy policy is specified as follows:

(i) *Same as step (i) from Section 4.4.*

(ii) Otherwise, request r_t is accepted if and only if there is a non-idle vehicle k, $V_k = (\tau_k, \theta_k)$, such that $b(\tau_k, \rho(\theta_k, r_t)) \geq 0$.

(iii) If r_t is accepted, it is assigned to the non-idle vehicle k, $V_k = (\tau_k, \theta_k)$, such that $b(\tau_k, \theta_k) - b(\tau_k, \rho(\theta_k, r_t))$ is minimum.

Again, different variants of the greedy policy arise, depending on the adopted routing policy. We denote by GP_CI and GP_R the greedy policies obtained by considering routing policies ρ_CI and ρ_R, respectively.

4.6.2. VFA Policy

VFA is a class of ADP methods that mostly rely on lookup tables or linear functional structures to approximate the reward-to-go (Powell et al., 2012). We design a benchmark parametric VFA policy in which the expected reward-to-go $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$ is estimated by a linear function $\hat{\Phi}_\mathrm{VFA}(S_{t+1}|x_t)$ parameterized by a single coefficient $\gamma \in \mathbb{R}_{>0}$. The precise mathematical definition of $\hat{\Phi}_\mathrm{VFA}(S_{t+1}|x_t)$ and the offline procedure to tune parameter γ are detailed in Appendix C. Given a routing policy ρ and a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$, the VFA policy is specified as follows:

(i)-(iii) *Same as steps (i)-(iii) from Section 4.4.*

(iv) A decision is selected from $\tilde{\mathcal{X}}_t$ by evaluating decision rules (DR1) and (DR2) based on approximations $\hat{\Phi}_\mathrm{VFA}(S_{t+1}|\tilde{x}_t)$ of $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|\tilde{x}_t]$, $\tilde{x}_t \in \tilde{\mathcal{X}}_t$, computed by (C.1).

We denote by VFA_CI and VFA_R the VFA policies with routing policies ρ_CI and ρ_R adopted in the online phase, respectively. As the offline procedure to tune parameter γ requires simulating the policy under a large number of sample paths, to guarantee computational efficiency the routing policy ρ_CI is employed in the offline simulations.

4.6.3. Rollout policies

A rollout algorithm is a general procedure to improve the performance of a base policy. It works by simulating candidate decisions, online and under the base policy, and then prescribing the decision with the maximum simulated reward-to-go. We refer to Bertsekas & Tsitsiklis (1996) for more details and a formal introduction to the method. As shown in Table 1, rollout algorithms are frequently employed for solving DVRPs, and for this reason we also define rollout-based benchmark policies.

Given a base policy, a routing policy ρ and a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$, the rollout policy is specified as follows:

(i)-(iii) *Same as steps (i)-(iii) from Section 4.4.*

(iv) For each candidate decision $\tilde{x}_t \in \tilde{\mathcal{X}}_t$, the system is simulated over H sample paths from the current instant u_t until the last instant U, given that decision $\tilde{x}_t$ is taken at state S_t and the base policy is in effect. As a result of the simulations, we obtain $\hat{r}(\tilde{x}_t)$, $\tilde{x}_t \in \tilde{\mathcal{X}}_t$, the approximated reward-to-go associated with decision $\tilde{x}_t$.

(v) The policy prescribes decision $\tilde{x}_t \in \tilde{\mathcal{X}}_t$ such that $\hat{r}(\tilde{x}_t)$ is maximum.

Because of the reduced computational time available in a real-time context and the large number of online simulations required in step (iv) of the rollout procedure, only very efficient scheduling policies are suitable as base policies. Among the policies previously presented, only GP_{CI} and VFA_{CI} are appropriate as base policies, since all other policies require solving multiple integer programs or multiple linear programs before a decision is taken. In fact, as we discuss in Section 6.3, even under the computationally efficient base policies GP_{CI} and VFA_{CI} the decision times of rollout policies in large instances are severely compromised due to the costly online simulations.

Finally, different routing policies can be adopted when defining the set of candidate decisions in step (iii) of the procedure. So, we denote by R_{CI}-$GP_{CI}(H)$ and R_{CI}-$VFA_{CI}(H)$ the rollout policies with base policies GP_{CI} and VFA_{CI}, respectively, where routing policy ρ_{CI} is adopted to generate candidate decisions and H samples paths are simulated in step (iv) to approximate rewards-to-go. Likewise, we define rollout policies R_R-$GP_{CI}(H)$ and R_R-$VFA_{CI}(H)$ when routing policy ρ_R is used to generate candidate decisions.

5. Offline planning algorithms

We now focus on the decision at the initial state S_0. In Section 5.1, we briefly discuss previous methods for route planning in DVRPs, and formalize the problem. Section 5.2 describes a benchmark method based on travel time minimization, and Section 5.3 presents our potential-based approach for route planning.

5.1. Anticipative route planning

A good route plan should cover static requests and leave the system in a favorable state for fitting dynamic requests that arrive during the service period. In this context, previous work proposed designing route plans that include static and sampled dynamic requests (Bent & Van Hentenryck, 2004), waiting strategies in anticipation of the location of dynamic requests (Branke, Middendorf, Noeth, & Dessouky, 2005; Thomas, 2007), combinations between waiting and request sampling (Ichoua et al., 2006), and managing slack times to facilitate acceptance of future requests when time windows are enforced (Mitrović-Minić, Krishnamurti, & Laporte, 2004). For a detailed discussion, we refer to Ichoua, Gendreau, & Potvin (2007). Our potential-based planner is closest to the multiple scenario approach from Bent & Van Hentenryck (2004), as we rely on sample paths from the spatiotemporal request distribution to evaluate the potential of candidate planned routes.

Formally, the offline planning algorithm defines an initial set of routes $\mathbf{y} = \{\theta_1, \ldots, \theta_{k'}\}$, $k' \leq K$, such that all static requests are covered. For the purpose of computing an offline plan, it is useful to reformulate (DVRP-SOM) in a way more suitable for static VRP optimization:

$$(\text{DVRP-SPM}) \quad \max_{\{\theta_1, \ldots, \theta_{k'}\} \subset \Theta, k' \leq K} g(S_1) \equiv \mathbb{E}[\Phi_{\pi^*}(S_1)],$$

$$\text{s.t.} \sum_{k \in \{1, \ldots, k'\}} \left[(u, i, d) \in \Delta(\theta_k) \right] = 1, \quad (u, i, d) \in \mathcal{S},$$

$$V_k = (0, \theta_k), \qquad k \in \{1, \ldots, k'\},$$

$$V_k = \emptyset, \qquad k \in \{k'+1, \ldots, K\}, \quad (9)$$

where $S_1 = (V_1, \ldots, V_K, r_1)$, $r_1 = (u_1, i_1, d_1)$, and $\Theta = \{\theta : b(0, \theta) \geq 0\}$ is the set of all feasible routes.

The feasible solution space of (DVRP-SPM) is defined by the partitioning constraints (9). The states of the vehicles to which a planned route is assigned are initialized accordingly, and it is assumed that these vehicles depart from the depot at instant 0 (i.e., waiting time at the depot is not allowed). The cost function is non-linear and also depends on the optimal (but unknown) policy π^*.

A reasonable step towards a tractable offline planning algorithm is to replace the cost function g by an additively separable (on $V_1, \ldots, V_K$), mostly order-preserving mapping $\tilde{g}$, and solve (DVRP-SPM) with a standard (static) VRP algorithm or heuristic. In the remainder of this section, we discuss two possible ways of implementing this approach.

5.2. Budget-based (myopic) planner

The budget of a route is a suitable proxy for assessing the capacity of the route to accommodate future requests. Therefore, a sensible mapping is given by:

$$\tilde{g}_{\text{myo}}(S_1) = \sum_{V_k \neq \emptyset} b(0, \theta_k) + \sum_{V_k = \emptyset} U. \tag{10}$$

We call $\tilde{g}_{\text{myo}}$ the *myopic* mapping, since it does not anticipate the realization of future dynamic requests. Under cost function (10), model (DVRP-SPM) reduces to a duration-constrained (or distance-constrained) VRP (DCVRP; Toth & Vigo, 2002) defined on an auxiliary, complete digraph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$, which is created in the following way: first, the set of nodes is defined as $\mathcal{V}' = \{0\} \cup \{i : (u, i, d) \in \mathcal{S}\}$; second, with each arc $(i, j) \in \mathcal{A}'$, where $(u_i, i, d_i), (u_j, j, d_j) \in \mathcal{S}$, a cost $t(i, j) + d_j$ is associated. Then, a myopic plan is obtained by solving a DCVRP on $\mathcal{G}'$, with node 0 as the depot, up to K vehicles available, and a duration limit of U on individual routes.

Depending on the number of static requests $|\mathcal{S}|$, finding the optimal solution to the DCVRP may be computationally prohibitive. Moreover, a near-optimal solution is enough to allow us to assess the quality of budget-based first stage DVRPSR decisions, compared to the potential-based plans described in the next section. Therefore, we opt for a heuristic method instead of an exact algorithm to solve the DCVRP. The two-step heuristic works as follows:

(i) A large set of feasible routes $\Theta' \subset \Theta$ is generated by applying column generation (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998) on the linear relaxation of model (DVRP-SPM) under cost function (10). The details of this procedure, which employs standard techniques commonly applied in (static) VRP algorithms, are described in B.2.

(ii) A myopic plan is determined by finding the set of routes $\{\theta_1, \ldots, \theta_{k'}\} \subset \Theta'$, $k' \leq K$, that serve all static requests with minimum total duration. To this end, we employ an off-the-shelf integer programming solver to solve exactly model (DVRP-SPM) under cost function (10) with the restricted set of variables Θ'.

5.3. Potential-based planner

Instead of maximizing budget, the potential-based planner maximizes the potential of the state resulting from the first stage decision. Given sample paths $\Omega = \{\omega_1, \ldots, \omega_H\}$ of the request arrival process $\{\Lambda(u)\}_{u \in [0, U]}$, we define the potential-based mapping $\tilde{g}_{\text{pb}}$:

$$\tilde{g}_{\text{pb}}(S_1) = \frac{1}{H} \sum_{\omega \in \Omega} \sum_{V_k \neq \emptyset} p_0^\omega(V_k), \tag{11}$$

where $V_k = (0, \theta_k)$, $k \in \{1, \ldots, k'\}$, are the initial states of the vehicles to which a route is assigned.

Mapping $\tilde{g}_{\text{pb}}$ takes into account the distribution of future requests and is also additively separable on $V_1, \ldots, V_K$. An algorithm for generating potential-based plans follows:

(i) *Same as step (i) from Section 5.2.*

(ii) For each route $\theta \in \Theta'$, compute the cost coefficient $\sum_{\omega \in \Omega} p_0^\omega((0, \theta))/H$, which correlates with the number of dynamic requests that can be accommodated along θ.

(iii) A potential-based plan is determined by finding the set of routes $\{\theta_1, \ldots, \theta_{k'}\} \subset \Theta'$, $k' \leq K$, that serve all static requests with maximum (estimated) potential. As in step (ii) of the myopic planner, we invoke an integer programming solver to solve exactly model (DVRP-SPM) under cost function (11) with the restricted set of variables Θ'.

Compared to the myopic planner, the potential-based planner favors routes with a total budget that can be more efficiently used to serve dynamic requests. The potential-based mapping promotes flexibility to serve dynamic requests as planned routes with more slack time are preferred over low-budget routes. In addition, whereas the DCVRP-based myopic plan usually consists of few routes, each serving several static requests, the potential-based first stage decision exhibits a more balanced distribution of static requests to vehicles and promotes a better coverage of the service area. By considering a large number of sample paths when evaluating (11), we ensure that the initial plan is robust, in the sense that it performs well, on average, for a variety of possible dynamic request trajectories. Also note that, similar to the potential-based policies, the potential-based planner requires no parameter other than the number of sample paths.

6. Computational study

In this section, we present an extensive computational study on large-scale test instances based on a real street network. Section 6.1 introduces the detailed configurations of the test instances. Section 6.2 compares the myopic and potential-based offline planning algorithms, and Section 6.3 compares the online scheduling policies. All algorithms and simulations are programmed in C++ and executed on a single core of an Intel® Xeon® Gold 6130 (2.1GHz) processor with 8GB of available RAM. IBM® CPLEX® version 12.10 is employed for solving linear and integer programs. When adopting the reoptimization policy ρ_R, the resulting TSPs are solved with a standard branch-and-cut procedure. Finally, all software and test instances developed in this project, with which all results presented in this section can be replicated, are available open-source at https://github.com/amflorio/dvrp-stochastic-requests.

6.1. Test instances

Using geographical data from the OpenStreetMap (OSM) project, we construct our test instances based on the street network of Vienna, Austria. The street network graph is generated from OSM data in the following way: first, we fetch all available data (nodes and ways in OSM terminology) for the city of Vienna; next, we remove the ways that correspond to walkways or service roads, and remove the nodes that are not intersections of two or more streets. The final graph consists of 16,080 nodes (intersections) and 36,424 arcs (street segments) and is illustrated in Fig. 2. The depot, indicated by the square, is located in the southeast of the city, and two request clusters, depicted as circular shaded areas, are defined in the northeast and southwest areas.

The fleet size K varies from 2 to 20, and each vehicle travels at a constant speed $\bar{s} = 20$ km/h. The duration of the service period is set to $U = 600$ minutes. The ratio between the expected number of dynamic requests and the expected total number of requests (static and dynamic) is given by the degree of dynamism $\eta = \mathbb{E}[T/(|\mathcal{S}| + T)]$. We assume $\sum_{i \in \mathcal{V} \setminus \{0\}} \lambda_i(u) = \Lambda$, $u \in [0, U]$, where $\Lambda \in \{0.2, 0.4, 0.8, 1.5\}$ is the (constant) overall request rate per minute. Note that our smallest instances with $\Lambda = 0.2$ and $\eta = $

0.75, such that $\mathbb{E}[T] + |\mathcal{S}| = 160$, are larger than most synthetic instances considered in the previous works mentioned in Table 1.

Given Λ and η, the expected numbers of static and dynamic requests are fixed. Static requests are uniformly distributed on the network nodes. To assess the performance of the proposed methods under distinct dynamic request patterns, we generate dynamic requests according to three spatiotemporal distributions: uniform and time-independent (UTI), clustered and time-independent (CTI), and clustered and time-dependent (CTD). In the UTI distribution, dynamic requests are also uniformly distributed on the network nodes ($\lambda_i(u) = \lambda_j(u)$, for all $i, j \in \mathcal{V} \setminus \{0\}$ and $u \in [0, U]$), while in the CTI and CTD distributions, half of the dynamic requests are expected to originate from a node within the two circular areas with a radius of 3 km (see Fig. 2). The request rate at each node does not change over time in the UTI and CTI distributions ($\lambda_i(u) = \lambda_i(u')$, for all $i \in \mathcal{V} \setminus \{0\}$ and $u, u' \in [0, U]$). In the CTD distribution, the request rates in the northeast and southwest clusters decrease and increase over time, respectively: 80% (20%) of the clustered requests are expected to appear in the northeast (southwest) cluster at the beginning of the service period ($u = 0$), and this percentage decreases (increases) linearly to 20% (80%) at the end of the service period ($u = U$). The UTI distribution models more closely a practical large-scale instance, because the high density of nodes in the inner city induces a natural demand cluster. Distributions CTI and CTD are somewhat arbitrary but useful nevertheless, as they allow testing the policies on request trajectories largely distinct from those given by the UTI distribution. Regardless of which request distribution is adopted, each static or dynamic request's service duration is normally distributed with a mean of 10 minutes and a standard deviation of 150 seconds.

Each test instance is characterized by a certain combination of request rate Λ, degree of dynamism η, fleet size K, and request distribution. For each instance, we independently generate one set of static requests and multiple sets of dynamic requests. Combining the former with one of the latter yields a *request scenario*, which corresponds to a specific problem and is solved individually. Table 2 summarizes all test instances, the number of request scenarios per instance, as well as the offline and online algorithms employed to solve each scenario. We parameterize both the potential-based planner and policies with $H = 50$, as our preliminary experiments indicated that additional sample paths do not lead to improvements in solution quality. Hence, for simplicity, S-PbP$_R$(50) and PbP$_R$(50) are from now on referred to as S-PbP and PbP, respectively. When solving a request scenario under a non-deterministic online policy (all policies except GP$_{CI}$ and GP$_R$), we repeat the solution procedure 15 times to evaluate the policy's performance in expectation. All sample paths required by the non-deterministic policies are independently generated. Considering all instances, request scenarios, and algorithms tested, we simulate a total of 27,964 executions of the DVRPSR. As the total computational time required in this simulation study was quite high (approximately, 248 CPU days), we also make available in the online repository the complete raw simulation results.

Throughout each simulation, the implemented framework records the arrival of requests, the decisions taken, and the states' evolution, and it also outputs each state in a visual format. The visualization of a simulation enables a better interpretation of the decisions taken by the scheduling policy. As an example, at https://youtu.be/D57xNfU73as we can visualize a simulation of policies R$_R$-GP$_{CI}$(50) and PbP on an instance with $\Lambda = 0.4$, $K = 5$ and UTI distributed requests. The main indicators used to evaluate the performance of the algorithms are the *acceptance rate*, defined as the number of dynamic requests accepted divided by the total number of dynamic requests generated, and the *maximum decision time*, which is the maximum computation time required for taking a single decision. The maximum decision time is an impor-

Fig. 2. Graph Representing the Street Network of Vienna, Austria. *Notes.* The square represents the depot, and the two shaded areas represent request clusters..

Table 2
Test Instances and Offline and Online Algorithms Simulated.

Instance parameters					Algorithms		Simulations
Λ	η	K	Dist.	Scen.	Offline	Online	
0.2	0.75	2, 3	UTI CTI CTD	5	MY PB	GP_{CI}, GP_R, R_{CI}-GP_{CI}(10, 25, 50, 100), R_R-GP_{CI}(10, 25, 50, 100), VFA_{CI}, VFA_R, R_R-VFA_{CI}(25, 50, 100)*, S-PbP, PbP	12,270
0.4	0.85	3, 5	UTI CTI CTD	5	MY PB	GP_{CI}, GP_R, R_{CI}-GP_{CI}(10, 25, 50, 100), R_R-GP_{CI}(10, 25, 50, 100), VFA_{CI}, VFA_R, R_R-VFA_{CI}(25, 50, 100)*, S-PbP, PbP	12,270
0.8	0.90	6, 12	UTI CTI CTD	5	PB	GP_{CI}, GP_R, VFA_{CI}, VFA_R, R_{CI}-GP_{CI}(10), R_R-GP_{CI}(10), R_R-VFA_{CI}(10), S-PbP, PbP	3,210
1.5	0.95	10, 20	UTI	1**	PB	GP_{CI}, GP_R, VFA_{CI}, VFA_R, R_{CI}-GP_{CI}(10), R_R-GP_{CI}(10), R_R-VFA_{CI}(10), S-PbP, PbP	214

Notes. Dist., distribution; Scen., scenario; MY, myopic planner; PB, potential-based planner;
The online algorithms marked by * are tested with the potential-based planner only;
** Only one scenario is tested for each large instance with $\Lambda = 1.5$ due to the long simulation time.

tant metric to assess a scheduling policy's suitability for real-time applications.

6.2. Evaluation of offline planning algorithms

To evaluate the performance of the myopic and potential-based planners, we employ both algorithms to compute initial route plans for all instances with $\Lambda \in \{0.2, 0.4\}$. Table 3 compares the average acceptance rates across eight online policies, where columns "Impr." indicate the percentage improvement achieved by the potential-based plan in relation to the myopic plan when a specific online algorithm is used. A more detailed comparison, which shows the results of different request distributions, is provided in Table D.1 in Appendix D. Overall, when combined with the best-performing policies S-PbP and PbP, the potential-based plans improve acceptance rates by 3.6% and 5.0%, respectively. Although myopic and potential-based plans perform equally well in the smallest instance with $\Lambda = 0.2$ and $K = 2$, in all other instances the potential-based plans significantly improve the average accep-

tance rates, regardless of the request distribution and the adopted online policy.

Fig. 3 compares the initial plans generated by the two offline planning algorithms for two instances to develop intuition behind these improvements. In the upper instance, we see that the myopic planner defines two initial routes and leaves one vehicle idle at the depot (which is activated upon arrival of the first dynamic request). This myopic plan maximizes the total budget but ignores the spatiotemporal distribution of dynamic requests. In comparison, the potential-based planner defines one additional route and, by doing so, promotes a better coverage of the dense area in the city center, which constitutes a natural cluster as requests are uniformly distributed over intersections. Similarly, in the lower instance, we observe that the potential-based planner uses the third vehicle to improve the northeast cluster's coverage, where the request rate is high, especially during the early stage of the service period.

Potential-based plans' success is also attributed to a more balanced set of initial routes, which cover similar distances and serve a similar number of static requests. This is achieved simply by

Table 3
Comparison of Offline Planning Algorithms: Acceptance Rate (%) and Relative Improvement (%).

Instance		GP_{CI}			GP_R			R_{CI}-GP_{CI}(50)			R_R-GP_{CI}(50)		
Λ	K	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.
0.2	2	19.5	19.5	0.0	19.8	19.8	0.0	24.2	24.3	+0.4*	24.4	24.3	− 0.1*
	3	39.1	43.6	+11.4	39.6	45.4	+14.8	49.3	50.6	+2.5	49.6	52.0	+4.8
0.4	3	20.0	23.0	+15.0	20.3	23.6	+16.1	26.0	28.0	+7.9	26.1	28.6	+9.5
	5	42.6	49.2	+15.6	43.1	50.1	+16.4	51.4	55.2	+7.2	51.6	55.7	+7.9
Overall		30.3	33.8	+11.7	30.7	34.8	+13.2	37.7	39.5	+4.7	37.9	40.1	+5.9

Instance		VFA_{CI}			VFA_R			S-PbP			PbP		
Λ	K	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.
0.2	2	26.6	26.6	− 0.2*	26.8	26.8	0.0	27.7	27.7	0.0	27.7	27.7	+0.1*
	3	41.6	48.5	+16.6	42.8	50.2	+17.5	52.0	52.7	+1.3	52.5	54.3	+3.5
0.4	3	24.0	29.8	+24.4	24.2	30.0	+24.0	28.7	31.4	+9.2	28.9	31.8	+10.0
	5	45.9	54.3	+18.5	46.2	54.4	+17.7	53.3	55.8	+4.5	54.5	58.0	+6.4
Overall		34.5	39.8	+15.3	35.0	40.4	+15.3	40.5	41.9	+3.6	40.9	43.0	+5.0

Notes. Impr., relative improvement of PB over MY: (PB − MY)/MY×%.
* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).
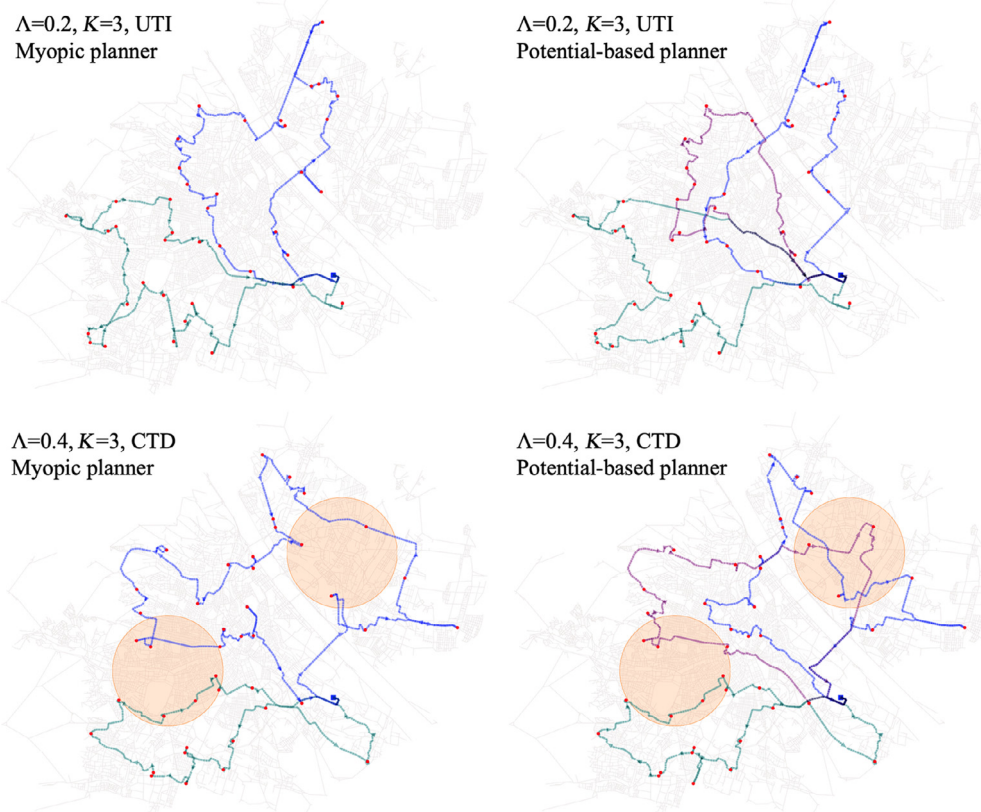
Fig. 3. Comparison of Initial Route Plans. *Notes.* Each red dot represents a static request. Each colored line represents the initial route of a vehicle.. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

evaluating routes under the knapsack-based potential approximation, without requiring additional parameters. The capacity (in terms of budget) of those routes is consumed in a more balanced way along the service period, which gives more flexibility to the scheduling policy when deciding the best vehicle to serve a dynamic request. On the other hand, myopic plans are characterized by very long routes and idle vehicles. While it is possible to enforce the myopic planner to use all vehicles, this alone does not lead to a balanced set of routes as the DCVRP model minimizes the total duration, so additional constraints (and parameters) would be required to achieve the same effect.

Table 4 presents the runtime requirements of both planners. The fleet size and the request distribution have little effect on the offline runtime, so the results correspond to the averages over all

Table 4
Average Runtime of Offline Planning Algorithms.

Λ	$\mathbb{E}[T]$	$|\mathcal{S}|$	$|\Theta'|$	Time (min)	
				MY	PB
0.2	120	40	5986	22	26
0.4	240	42	4656	26	31
0.8	480	53	6279	55	66
1.5	900	47	5748	37	57

Notes. MY, myopic planner; PB, potential-based planner.

fleet sizes and request distributions for each request rate Λ. For each instance, the two planners generate the same route set Θ' because they have the same column generation procedure as their first step. The runtime of the potential-based planner increases
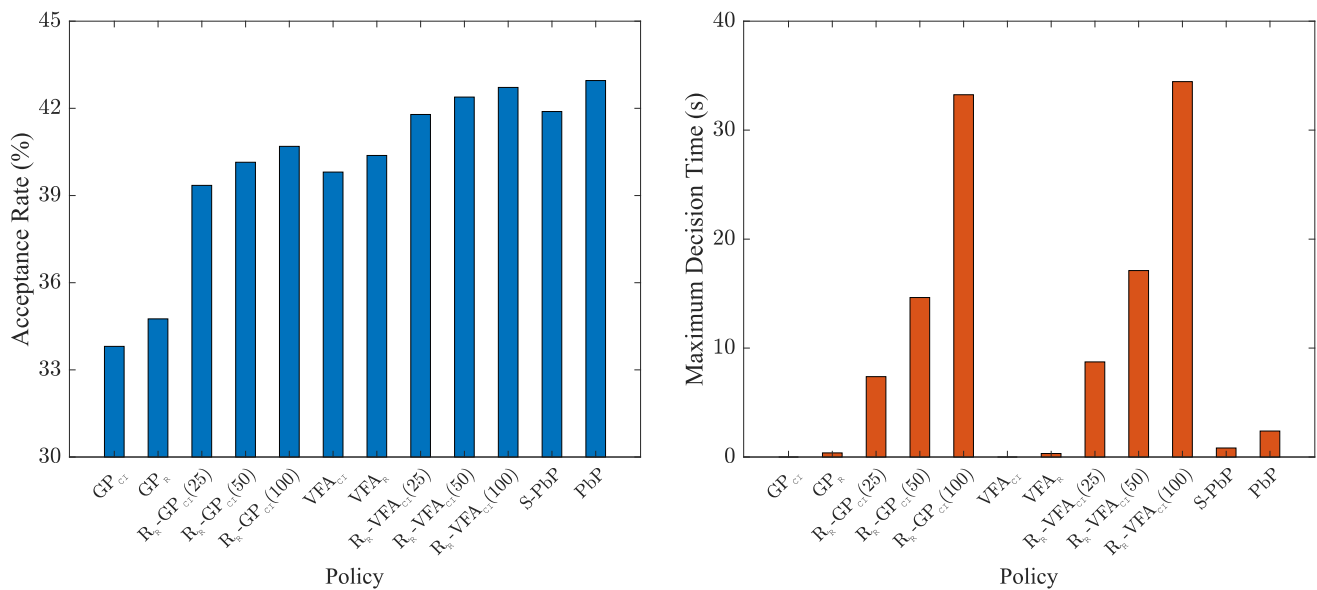
Fig. 4. Comparison of Online Scheduling Policies ($\Lambda \in \{0.2, 0.4\}$).

Table 5
Comparison of Online Scheduling Policies ($\Lambda \in \{0.2, 0.4\}$): Percentage Improvement in the Acceptance Rate Compared to GP_{CI}.

Instance		Online scheduling policy										
			$R_R\text{-GP}_{\text{CI}}(H)$						$R_R\text{-VFA}_{\text{CI}}(H)$			S-PbP
Λ	K	GP_R	$H=25$	$H=50$	$H=100$	VFA_{CI}	VFA_R	$H=25$	$H=50$	$H=100$		PbP
0.2	2	+1.9*	+23.5	+24.9	+25.7	+36.4	+37.7	+39.6	+40.8	+40.9	+42.4	+42.4
	3	+4.3	+17.3	+19.3	+20.8	+11.4	+15.4	+21.1	+22.5	+23.6	+20.9	+24.7
0.4	3	+2.8*	+21.7	+24.5	+26.1	+29.8	+30.7	+34.7	+36.6	+37.4	+36.8	+38.5
	5	+1.8*	+10.3	+13.1	+15.1	+10.4	+10.5	+14.3	+16.6	+17.9	+13.2	+17.7
Overall		+2.8	+16.4	+18.7	+20.4	+17.7	+19.4	+23.6	+25.4	+26.4	+23.9	+27.1

* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).

with Λ, as it requires each route to be simulated $H = 50$ times under sample paths that grow linearly with Λ.

6.3. Evaluation of online scheduling policies

We first compare the online scheduling policies applied to the instances with $\Lambda \in \{0.2, 0.4\}$. As Section 6.2 demonstrates that potential-based plans outperform myopic plans, all results presented in this section are based on the former. Moreover, since the spatiotemporal distribution of dynamic requests has little impact on the performance of online policies, all results presented in this section are averaged over the three distributions. The interested reader is referred to Tables D.2 to D.5 in Appendix D for detailed results. Taking the results of policy GP_{CI} as a reference, Table 5 presents the relative improvements in acceptance rate obtained by the other policies (the acceptance rates of GP_{CI} can be found in Table 3). In addition, the maximum decision times are presented in Table 6.

The best-performing policy is PbP, which displays an average acceptance rate higher than those of the VFA and rollout algorithms, with or without reoptimization, and for any number of sample paths. Moreover, PbP can take decisions much faster than rollout policies. For the same number of sample paths $H = 50$, PbP promotes a sixfold decrease in decision times, on average. Policy S-PbP is even more computationally efficient, at the cost of somewhat lower acceptance rates compared to PbP (but still better than the other policies except $R_R\text{-VFA}_{\text{CI}}(H \geq 50)$, on average). In these instances, VFA-based policies require no more than 60 seconds of

offline computation time for parameter tuning. We also notice that, compared to the CI routing policy ρ_{CI}, the reoptimization routing policy ρ_R improves the acceptance rates slightly in both the greedy and VFA policies, at the cost of short additional computation time per decision.

Fig. 4 compares the overall performance and decision times of all scheduling algorithms (except the rollout algorithms with the CI routing policy ρ_{CI}) graphically. As expected, the performance and decision times of rollout policies increase with the number of sample paths. As also expected, policies GP_{CI} and VFA_{CI} are the least computationally expensive, but the former has the worst solution quality. Although the VFA-based policies lead to drastically better results than the GP-based policies, their solution quality does not surpass that of policy S-PbP until the number of sample paths increases to $H = 50$, at which point their maximum decision times are 21 and 7 times longer than those of S-PbP and PbP, respectively, which limits their application in a real-time context.

Tables 7 and 8 present the acceptance rates and decision times, respectively, obtained in the instances with $\Lambda \in \{0.8, 1.5\}$. In Table 7, the relative improvements over GP_{CI} are provided in parentheses. The number of sample paths in rollout algorithms is limited to $H = 10$ as rollout decision times increase considerably in these large instances. Furthermore, the offline runtime required by the VFA-based policies grows to 15 minutes in the largest instances with $\Lambda = 1.5$ and $K = 20$. Policies S-PbP and PbP perform significantly better than the other algorithms. In addition, we see that S-PbP requires at most 11.3 seconds to compute a decision in the largest instances, which reaffirms S-PbP as a
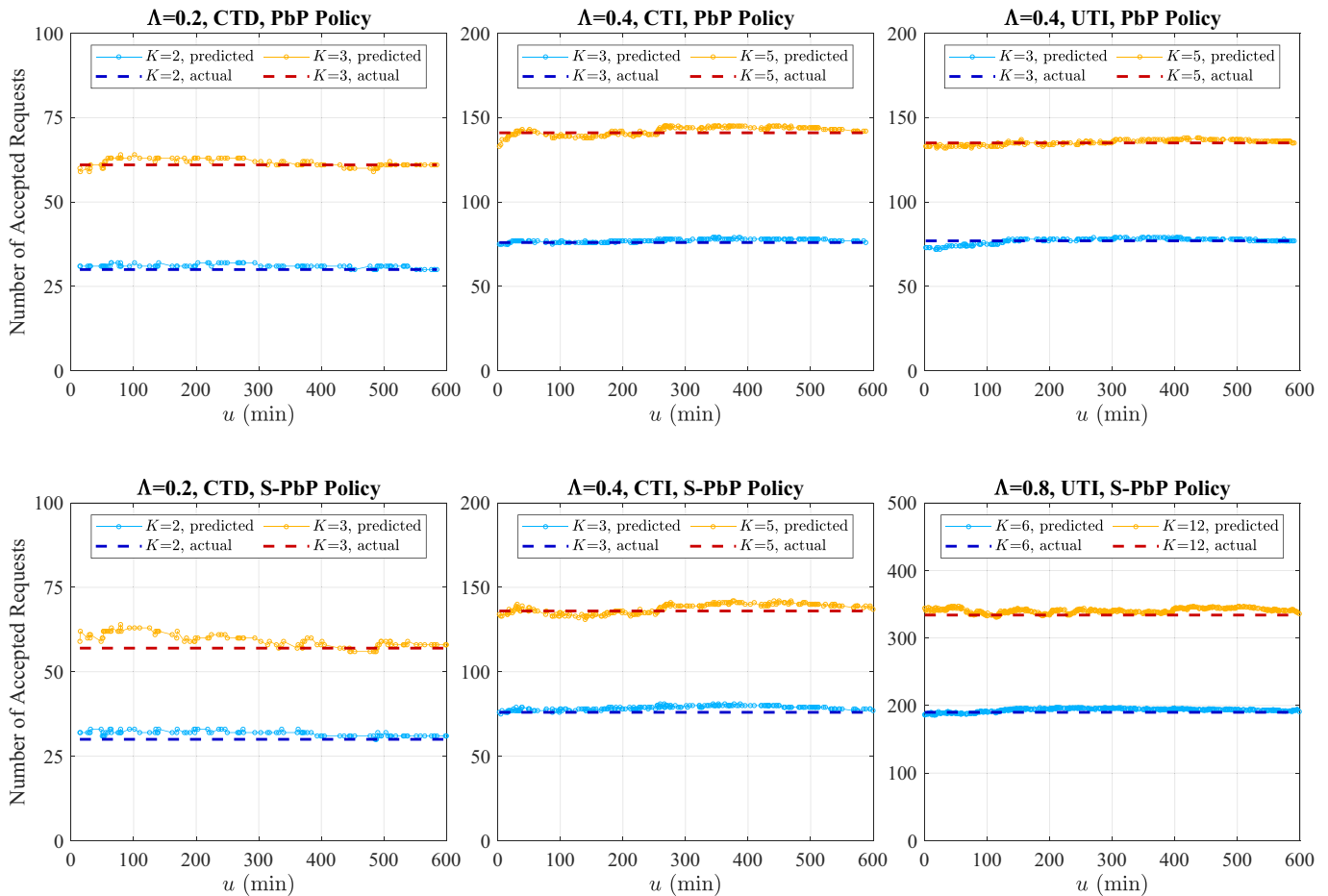
Table 6

Comparison of Online Scheduling Policies ($\Lambda \in \{0.2, 0.4\}$): Maximum Decision Time (seconds).

Instance		Online scheduling policy											
		GP_{CI}	GP_R	R_R-$GP_{CI}(H)$			VFA_{CI}	VFA_R	R_R-$VFA_{CI}(H)$			S-PbP	PbP
Λ	K			$H{=}25$	$H{=}50$	$H{=}100$			$H{=}25$	$H{=}50$	$H{=}100$		
0.2	2	<0.1	0.3	1.0	2.0	3.1	<0.1	0.3	1.0	2.1	3.9	0.4	0.6
	3	<0.1	0.2	1.9	3.7	6.3	<0.1	0.2	1.9	3.5	7.2	0.4	0.6
0.4	3	<0.1	0.4	3.1	6.1	14.1	<0.1	0.3	3.2	7.3	14.5	0.6	1.1
	5	<0.1	0.3	7.4	14.6	33.2	<0.1	0.3	8.7	17.1	34.5	0.8	2.4
Overall		<0.1	0.4	7.4	14.6	33.2	<0.1	0.3	8.7	17.1	34.5	0.8	2.4

Table 7

Comparison of Online Scheduling Policies ($\Lambda \in \{0.8, 1.5\}$): Acceptance Rate (%) and Percentage Improvement Compared to GP_{CI}.

Instance		Online scheduling policy							
Λ	K	GP_{CI}	GP_R	R_R-$GP_{CI}(10)$	VFA_{CI}	VFA_R	R_R-$VFA_{CI}(10)$	S-PbP	PbP
0.8	6	30.7	30.8(+0.6*)	32.9(+7.2)	37.0(+20.5)	36.5(+19.1)	36.9(+20.4)	39.7(+29.3)	40.5(+31.9)
	12	63.9	63.0(- 1.4*)	62.4(- 2.3)	65.9 (+3.1)	66.1 (+3.5)	64.7 (+1.3)	69.4 (+8.7)	73.8(+15.6)
1.5	10	33.4	35.7(+7.0)	34.6(+3.8)	39.7(+19.1)	39.9(+19.7)	39.9(+19.5)	43.1(+29.0)	44.2(+32.5)
	20	62.4	64.5(+3.4)	61.6(- 1.4)	67.1 (+7.4)	66.7 (+6.8)	65.0 (+4.1)	70.8(+13.4)	76.9(+23.2)
Overall		47.3	47.1(- 0.4*)	47.7(+0.8)	51.5 (+8.9)	51.5 (+8.8)	50.9 (+7.6)	54.7(+15.6)	57.4(+21.2)

* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).

Fig. 5. Accuracy of the Multiple-/Single-Knapsack Potential Approximation.

high-performing scheduling policy suitable for real-time decision-making in the DVRPSR. Rollout algorithms with $H = 10$ sample paths, on the other hand, require much longer runtimes (e.g., more than 150 seconds in the largest instances) and are frequently not effective enough to outperform their base policies. Increasing the number of sample paths would lead to unacceptable online decision times (for example, up to six minutes when $H = 25$).

As observed in Table 8, the average decision times of policy S-PbP grow linearly with K. This dependence is explained by the fact that, at each decision epoch, each additional vehicle requires the solution of an extra single-knapsack model. On the other hand, the average response time of policy PbP grows more than linearly with K. In the PbP, each additional vehicle requires solving a new multiple-knapsack model at each epoch, but also increases

Table 8
Comparison of Online Scheduling Policies ($\Lambda \in \{0.8, 1.5\}$): Decision Times (seconds).

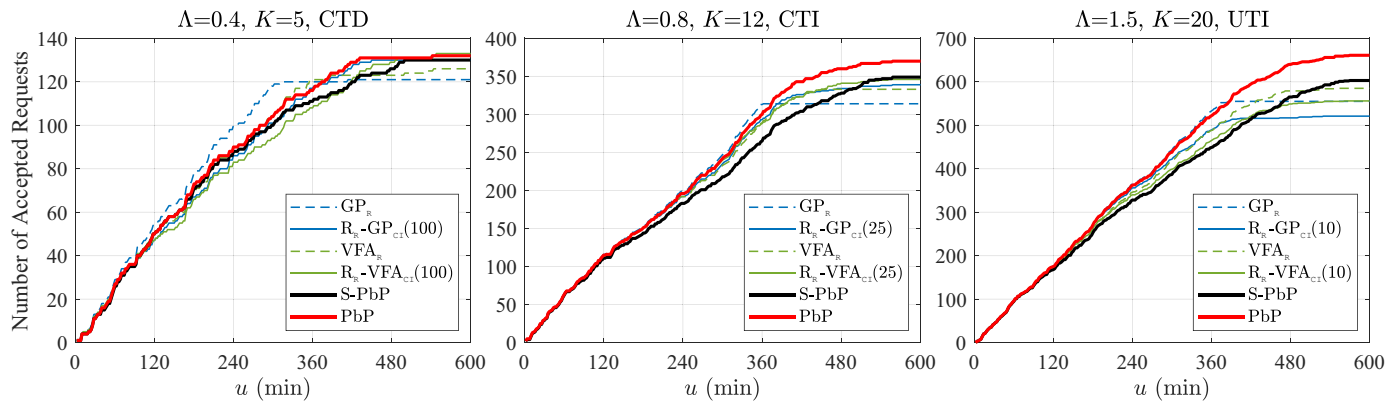| Instance		Online scheduling policy															
		GP_{CI}		GP_R		R_R-$GP_{CI}(10)$		VFA_{CI}		VFA_R		R_R-$VFA_{CI}(10)$		S-PbP		PbP	
Λ	K	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
0.8	6	<0.1	<0.1	0.1	0.6	2.7	8.0	<0.1	<0.1	0.1	0.7	4.0	9.4	0.8	1.9	2.3	6.1
	12	<0.1	<0.1	0.1	0.3	14.4	32.5	<0.1	<0.1	0.1	0.3	14.6	30.4	1.6	3.7	8.9	23.8
1.5	10	<0.1	<0.1	0.1	0.5	13.5	40.0	<0.1	<0.1	0.1	0.4	19.3	39.7	2.6	5.7	11.2	27.0
	20	<0.1	<0.1	0.1	0.2	80.6	158.4	<0.1	<0.1	0.1	0.3	81.8	154.1	5.1	11.3	51.8	124.5
Overall		<0.1	<0.1	0.1	0.6	11.0	158.4	<0.1	<0.1	0.1	0.7	11.9	154.1	1.3	11.3	7.2	124.5

Fig. 6. Acceptance Profiles of Six Policies in Three Request Scenarios.

the complexity of solving models (MKA) for the remaining vehicles (see constraints (3)). The computational tractability of each policy also depends on the response time requirements posed by the underlying application. However, it is clear that policy S-PbP would scale better in instances with request rate $\Lambda \geq 1.5$ and many more than $K = 20$ vehicles.

The excellent performance of PbP and S-PbP is attributed to accurate predictions on the fleet's future service capability. Figs. 5 shows the expected total number of accepted requests as approximated by models (MKA) and (KA) throughout the service period for various request scenarios. Quite remarkably, both models are able to predict with significant accuracy, already at the beginning of the service period (i.e., before any stochasticity is disclosed) and regardless of the request distribution and instance size, the total number of accepted requests during the entire period. In combination with the decision rules applied in the optimal scheduling policy, these predictions lead to high-quality dynamic decisions that can be computed in real-time.

Fig. 6 exhibits the cumulative number of accepted dynamic requests over time (acceptance profiles) of several policies across three request scenarios. Even though data from only 18 simulations are plotted, these profiles' general shape represents well what is generally observed. While the greedy policy GP_R exhausts the fleet's service capability relatively early, the VFA and rollout policies with $H = 25$ sample paths or more prescribe better decisions and achieve higher acceptance rates. Capacity is most efficiently consumed by potential-based policies, which accept requests at a more steady rate until near the end of the service period.

7. Conclusions

We studied a vehicle routing problem where a service provider must determine, in real-time, whether to accept dynamic pickup (or service) requests and how to adjust service routes to accommodate accepted requests. The real-time requirement limits the choice of methods effectively applied to solve large instances of the DVRPSR. In particular, traditional ADP methods such as rollout algorithms resort to expensive online simulations that compromise request response times. This paper introduced knapsack-based approximations of the reward-to-go, upon which efficient online scheduling policies and offline planning algorithms are developed. The approximation models follow simple and interpretable steps justified by observed characteristics of dynamic routes, namely, late depot arrival and request order preservation. The excellent performance of the proposed methods was demonstrated by computational experiments on very large DVRPSR instances.

Several interesting research perspectives are resulting from this work. First, the studied DVRPSR is a canonical problem that can be extended in many directions. Given that a simple linear model approximates the reward-to-go, we are optimistic that the method can be adapted and remain effective to handle side-constraints (e.g., time windows and vehicle capacity), as well as other sources of dynamism such as stochastic and time-dependent travel times. The concept of effective speed is applicable to other problem variants, for example, the DVRP with stochastic pick-up and delivery requests. An interesting research direction is to adapt the knapsack approximations to pick-up and delivery variants, which yield very large DVRP instances as observed in applications such as meal delivery. Moreover, the action space of the DVRPSR can be extended to allow inter-route reoptimization (i.e., reassignment of accepted requests among vehicles), in addition to only intra-route request reordering. Lastly, given the dramatic improvement in acceptance rates enabled by anticipative offline plans, we believe that it is also promising to investigate offline planning algorithms based on the multiple-knapsack approximation model.

Acknowledgement

enna are copyrighted to OpenStreetMap contributors and available from OSM (2020).

Appendix A. Summary of Notations

Tables A.1 and A.2 summarize the problem notations and algo-

Table A1
Problem Notations.

Notation	Description
Global parameters	
$\mathcal{G}$	Graph (street network)
$\mathcal{V}$	Set of nodes (road intersections)
$\mathcal{A}$	Set of arcs (road segments)
d_{ij}	Length of arc $(i,j) \in \mathcal{A}$
$\bar{s}$	Vehicle speed
$t_{ij} = d_{ij}/\bar{s}$	Travel time on arc $(i,j) \in \mathcal{A}$
$t(i,j)$	Fastest travel time between nodes $i, j \in \mathcal{V}$
$[0,U]$	Service period
K	Number of vehicles
$r = (u,i,d)$	Customer request
$u \in [0,U]$	Request arrival time
$i \in \mathcal{V} \setminus \{0\}$	Customer node
$d \in \mathbb{R}_{>0}$	Service time of request
$\mathcal{S}$	Set of static requests
$\mathcal{D}(u)$	Ordered set of dynamic requests generated up to instant u
$\lambda_i(u)$	Request arrival rate at node i and instant u
$\Lambda(u) = \sum_{i \in \mathcal{V} \setminus \{0\}} \lambda_i(u)$	Overall request arrival rate at instant u
State variables	
$T = \|\mathcal{D}(U)\|$	Number of dynamic requests
$\theta = ((v_0, \delta_0), \ldots, (v_f, \delta_f))$	Route
$v_i \in \mathcal{V}, i \in \{1, \ldots, f-1\}$	The i-th node along route θ
$\delta_i, i \in \{1, \ldots, f-1\}$	Set of requests to serve when arriving at node v_i
$\Delta(\theta) = \cup_{i=1}^{f-1} \delta_i$	Set of all requests served by route θ
$\Delta_u(\theta) = \{(u',i,d) \in \Delta(\theta): u' \geq u\}$	Ordered set of requests served by route θ during $[u,U]$
$b(\tau, \theta)$	Budget of route θ when it starts at instant τ
$V_k = (\tau_k, \theta_k)$	State of vehicle k when it starts route θ_k at instant τ_k
$S_t = \{V_1, \ldots, V_K, r_t\}$	State variable at decision epoch $t = \{1, \ldots, T\}$
Decision variables	
$\mathcal{F}$	Feasible decision set at initial state S_0
$\mathbf{y} \in \mathcal{F}$	Offline decision (set of planned routes)
$\mathcal{X}_t$	Feasible decision set at state S_t, $t \neq 0$
$x_t \in \mathcal{X}_t$	Online decision at state S_t, $t \neq 0$
$S_{t+1} = S^M(S_t, x_t, r_{t+1})$	State transition function
Reward & objective function	
$R(S_t, x_t)$	Reward of executing decision x_t at state S_t
Π	Set of feasible policies
$X^\pi(S_t) \in \mathcal{X}_t$	Decision prescribed by policy $\pi \in \Pi$ at state S_t
$\Phi_\pi(S_t)$	Potential of state S_t when controlled by policy $\pi \in \Pi$

rithm notations defined in Section 3 and Section 4, respectively.

Appendix B. Auxiliary Algorithms

B1. Predicting vehicle locations

Consider a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$. The following procedure is used to predict, at the current instant u_t, the location of a non-idle vehicle k, $V_k = (\tau_k, \theta_k)$, at a future instant $u' > u_t$, and to determine, also at the current instant u_t, the set $\mathcal{V}_{u_t}(V_k, u')$ of nodes along θ_k to be traversed after u':

Step 1: Predicting the location of vehicle k at a future instant u':

(i) Let $d_{u_t}(V_k)$ be the remaining distance to be traveled along route θ_k at instant u_t. Then, the effective speed $\check{s}_{u_t}(V_k)$ is

Table A2
Algorithm Notations .

Notation	Description
ρ	Routing policy
$\rho(\theta, r)$	New route resulting from adding request r to route θ by policy ρ
$\pi^* \in \Pi$	Optimal scheduling policy
$x_t^- \in \mathcal{X}_t$	"Reject" decision with respect to request r_t
$x_t^+ \in \mathcal{X}_t \setminus \{x_t^-\}$	"Accept" decision with respect to request r_t
$\mathcal{X}_t^k \subset \mathcal{X}_t$	Set of decisions where r_t is accepted and assigned to vehicle k
$x_t^k \in \mathcal{X}_t^k$	Decision of accepting r_t and assigning it to vehicle k
$\check{s}_{u_t}(V_k)$	Effective speed of a vehicle with state V_k estimated at instant u_t
$d_{u_t}(V_k)$	Remaining distance of route θ_k at instant u_t
$\mathcal{V}_{u_t}(V_k, u') \subseteq \mathcal{V}$	Set of nodes visited by θ_k during $[u', U]$, estimated at instant u_t
$c_{u_t}(V_k, r')$	Cost of assigning future request r' to vehicle k, estimated at instant u_t
$\omega = \{r_1^\omega, \ldots, r_{T_\omega}^\omega\}$	Sample path of the request arrival process $\{\Lambda(u)\}_{u \in [0,U]}$
H	Number of sample paths
$\Omega = \{\omega_1, \ldots, \omega_H\}$	Set of sample paths
$\bar{K} = \{k \in \{1, \ldots, K\}: V_k \neq \emptyset\}$	Set of non-idle vehicles
$\phi_{\pi^*}^\omega(S_{t+1}\|x_t)$	Multiple-knapsack approximation of potential in sample path ω
$p_{u_t}^\omega(V_k)$	Single-knapsack approximation of vehicle k's potential in sample path ω
α^ω	Compensation ratio of single-knapsack approximation
$\hat{\Phi}_{\pi^*}(S_{t+1}\|x_t) \approx \mathbb{E}[\Phi_{\pi^*}(S_{t+1})\|x_t]$	Multiple-/Single-knapsack approximation of the expected potential
$\hat{\Phi}_{\text{VFA}}(S_{t+1}\|x_t) \approx \mathbb{E}[\Phi_{\pi^*}(S_{t+1})\|x_t]$	VFA approximation of the expected potential

given by:

$$\check{s}_{u_t}(V_k) = \frac{d_{u_t}(V_k)}{(d_{u_t}(V_k)/\bar{s}) + b(\tau_k, \theta_k)}.$$

(ii) Let $\theta_k(t) = ((v_t, \delta_t), \ldots, (v_f, \delta_f))$ be the substring of θ_k such that v_t is the current node at which vehicle k is located (if the vehicle is currently in service) or the next node to be reached by vehicle k (if the vehicle is currently in transit).

(iii) Let $\tilde{u}_s$ be the elapsed (service) time since vehicle k arrived at node v_t, if the vehicle is in service, $\tilde{u}_s = 0$ otherwise. Further, let $\tilde{u}_t$ be the remaining (transit) time under the effective speed $\check{s}_{u_t}(V_k)$ before vehicle k reaches node v_t, if the vehicle is in transit, $\tilde{u}_t = 0$ otherwise.

(iv) Taking into account the initial conditions set by $\tilde{u}_s$ and $\tilde{u}_t$, the remaining route $\theta_k(t)$ is simulated under effective speed $\check{s}_{u_t}(V_k)$ for an amount of time $u' - u_t$.

Step 2: Determining $\mathcal{V}_{u_t}(V_k, u')$:

(v) Let $\theta_k'(t) = ((v_t', \delta_t'), \ldots, (v_f, \delta_f))$ be the substring of $\theta_k(t)$ such that v_t' is the (predicted) first unvisited node along route θ_k at instant u'.

(vi) The procedure returns $\mathcal{V}_{u_t}(V_k, u') = \{v: (v, \delta) \in \theta_k'(t)\}$.

B2. Column generation for the DCVRP

Model (DVRP-SPM) under cost function (10) corresponds to a DCVRP defined on graph $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$. Let $\tilde{\Theta}$ be the set of all feasible DCVRP routes, where each route $\tilde{\theta} \in \tilde{\Theta}$ is a non-empty elementary sequence of nodes $\tilde{\theta} = (\tilde{v}_1, \ldots, \tilde{v}_f)$, $\tilde{v}_1, \ldots, \tilde{v}_f \in \mathcal{V}' \setminus \{0\}$. The cost of a route $\tilde{\theta} = (\tilde{v}_1, \ldots, \tilde{v}_f)$ is given by $C_{\tilde{\theta}} = \sum_{i=2}^f c_{v_{i-1}v_i}$, where c_{ij} is the cost of arc $(i,j) \in \mathcal{A}'$ as specified in Section 5.2. Introducing binary variables $z_{\tilde{\theta}}, \tilde{\theta} \in \tilde{\Theta}$, to indicate the routes in the

solution, the DCVRP is rewritten as follows:

(DCVRP-SPM) $\min \quad \sum_{\tilde{\theta} \in \tilde{\Theta}} C_{\tilde{\theta}} z_{\tilde{\theta}},$

$$\text{s.t.} \quad \sum_{\tilde{\theta} \in \tilde{\Theta}} [i \in \tilde{\theta}] z_{\tilde{\theta}} = 1, \qquad i \in \mathcal{V}' \setminus \{0\},$$

$$\sum_{\tilde{\theta} \in \tilde{\Theta}} z_{\tilde{\theta}} \leq K.$$

The linear relaxation of (DCVRP-SPM) cannot be solved directly because the number of variables $|\tilde{\Theta}|$ grows exponentially with $|\mathcal{V}'|$. So, we apply column generation and, in the process, generate a large pool of routes to subsequently use in the offline planning heuristic. Within column generation, the restricted master problem (RMP) refers to the linear relaxation of (DCVRP-SPM) when only a subset $\tilde{\Theta}_R \subset \tilde{\Theta}$ of variables is considered. A profitable variable to add to $\tilde{\Theta}_R$ is identified by solving the pricing problem, which calls for a column $\tilde{\theta}$ with negative reduced cost:

$$C_{\tilde{\theta}} - \sum_{i \in \mathcal{V}' \setminus \{0\}} [i \in \tilde{\theta}] \beta_i - \beta_0 < 0, \tag{B.1}$$

where β_i ($i \in \mathcal{V}' \setminus \{0\}$) and β_0 are, respectively, the dual values associated with the partitioning and the maximum number of vehicles constraints in (DCVRP-SPM), obtained by solving the RMP. Once profitable variables are identified, they are added to $\tilde{\Theta}_R$ and the procedure iterates. Convergence is attained when no further variable satisfies (B.1).

The pricing problem is solved with a labeling procedure, as common in algorithms for several VRP variants (Costa, Contardo, & Desaulniers, 2019). Each label corresponds to a partial path from node 0 to a node $j \in \mathcal{V}' \setminus \{0\}$, and stores the reduced cost accumulated so far along the partial path. To ensure tractability of the labeling procedure, we allow non-elementary routes according to *ng*-route relaxation (Baldacci, Mingozzi, & Roberti, 2011), and control the combinatorial growth of labels with dominance rules.

Appendix C. VFA Policy

Given a state $S_t = (V_1, \ldots, V_K, r_t)$, $r_t = (u_t, i_t, d_t)$, it is clear that r_t should be accepted only if this acceptance leads to an increase in the expected total reward, as stated by decision rule (DR1). Hence, we design a parametric VFA policy with a linear function $\hat{\Phi}_{\text{VFA}}(S_{t+1}|x_t)$, which approximates the expected reward-to-go $\mathbb{E}[\Phi_{\pi^*}(S_{t+1})|x_t]$ based on a single feature, i.e., the total time budget:

$$\hat{\Phi}_{\text{VFA}}(S_{t+1}|x_t) = \begin{cases} \gamma \sum_{V_k \neq \emptyset} b(\tau_k, \theta_k), & \text{if } x_t = x_t^- \in \mathcal{X}_t, \\ \gamma\left(b(\tau_k, \rho(\theta_k, r_t)) + \sum_{i \neq k, V_i \neq \emptyset} b(\tau_i, \theta_i)\right), & \text{if } x_t \in \mathcal{X}_t^k, \end{cases} \tag{C.1}$$

where x_t^- is the 'reject' decision with respect to r_t, $\mathcal{X}_t^k$ is the set of 'accept' decisions where r_t is assigned to vehicle k, and coefficient $\gamma > 0$ is a relative weight that trades off the immediate reward and the reward-to-go. The larger γ is, the more likely request r_t is rejected so as to preserve the fleet's future service capability.

Parameter γ is tuned in advance via offline simulations. As the optimal value of γ depends not only on instance settings (graph type, fleet size, request rate and distribution, etc.) but also on the initial route plan, the tuning process of γ takes place after the initial routes have been fixed by an offline planner and before the beginning of the service period. During the tuning process, we simulate VFA policies with different values of γ over H complete sample paths. The values of γ simulated are selected based on a variable step size search procedure, which assumes that the total reward is concave on γ, as observed empirically. Finally, we select the value of γ which leads to the highest simulated total reward.

Appendix D. Detailed Computational Results

This section presents the detailed computational results with respect to the three different request distributions: UTI, CTI, and

Table D1
Comparison of Offline Planning Algorithms: Acceptance Rate (%) and Relative Improvement (%).

Instance			GP_{CI}			GP_{R}			$\text{R}_{\text{CI}}\text{-GP}_{\text{CI}}(50)$			$\text{R}_{\text{R}}\text{-GP}_{\text{CI}}(50)$		
Λ	K	Dist.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.
0.2	2	UTI	19.9	19.9	0.0	20.3	20.3	0.0	24.2	24.4	+0.6*	24.4	24.4	- 0.1*
		CTI	20.0	20.0	0.0	19.8	19.8	0.0	24.2	24.4	+0.7*	24.5	24.5	0.0
		CTD	18.6	18.6	0.0	19.5	19.5	0.0	24.1	24.1	- 0.1*	24.1	24.1	- 0.3*
	3	UTI	36.6	42.1	+15.1	37.6	44.9	+19.2	47.4	49.4	+4.3	47.6	51.0	+7.2
		CTI	38.4	43.4	+12.8	39.1	44.7	+14.2	49.8	50.6	+1.7	49.8	52.4	+5.3
		CTD	42.2	45.2	+7.0	42.0	46.8	+11.4	50.8	51.6	+1.6	51.4	52.5	+2.2
0.4	3	UTI	20.5	24.2	+17.7	20.8	25.2	+20.9	26.5	28.6	+8.0	26.6	29.5	+11.1
		CTI	21.2	21.9	+3.1*	21.1	23.4	+11.0	26.2	27.8	+6.3	26.6	28.5	+7.0
		CTD	18.2	22.9	+25.8	19.1	22.2	+16.6	25.2	27.6	+9.4	25.2	27.8	+10.4
	5	UTI	42.8	50.0	+16.7	43.1	52.5	+21.6	51.4	56.0	+8.9	51.8	56.3	+8.8
		CTI	43.7	49.9	+14.4	43.4	50.1	+15.5	52.4	56.0	+6.7	52.5	56.5	+7.6
		CTD	41.3	47.8	+15.8	42.8	47.9	+11.9	50.5	53.5	+6.0	50.6	54.2	+7.2
	Overall		30.3	33.8	+11.7	30.7	34.8	+13.2	37.7	39.5	+4.7	37.9	40.1	+5.9

Instance			VFA_{CI}			VFA_{R}			S-PbP			PbP		
Λ	K	Dist.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.	MY	PB	Impr.
0.2	2	UTI	26.9	26.8	- 0.3*	26.9	26.9	+0.1*	27.8	27.7	- 0.2*	27.8	27.7	- 0.4
		CTI	26.0	26.0	- 0.2*	26.4	26.5	+0.3*	27.6	27.7	+0.3*	27.5	27.7	+0.6
		CTD	27.0	27.0	- 0.1*	27.2	27.1	- 0.3*	27.8	27.8	0.0	27.8	27.8	0.0
	3	UTI	40.3	48.6	+20.6	43.1	51.0	+18.4	51.0	53.3	+4.5	51.4	53.7	+4.5
		CTI	41.5	49.5	+19.3	42.5	50.9	+19.8	52.6	52.3	- 0.5*	52.7	54.2	+2.8
		CTD	43.0	47.4	+10.3	42.7	48.8	+14.3	52.4	52.4	0.0	53.2	55.0	+3.2
0.4	3	UTI	24.1	30.3	+25.9	24.6	31.6	+28.2	29.4	32.0	+8.5	29.9	32.5	+8.8
		CTI	24.0	30.0	+24.7	24.1	29.7	+23.3	29.4	32.1	+9.4	29.4	32.4	+10.1
		CTD	23.7	29.1	+22.7	23.9	28.7	+20.2	27.4	30.1	+9.9	27.4	30.5	+11.3
	5	UTI	46.3	54.9	+18.7	45.8	54.2	+18.2	53.1	55.8	+5.2	54.8	59.0	+7.6
		CTI	47.0	55.5	+18.1	47.5	55.8	+17.6	53.9	55.8	+3.5	55.3	58.5	+5.8
		CTD	44.4	52.7	+18.6	45.5	53.3	+17.3	53.1	55.6	+4.9	53.5	56.5	+5.7
	Overall		34.5	39.8	+15.3	35.0	40.4	+15.3	40.5	41.9	+3.6	40.9	43.0	+5.0

Notes. Dist., request distribution; Impr., relative improvement of PB over MY: (PB - MY)/MY×%.
* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).

Table D2
Comparison of Online Scheduling Policies ($\Lambda \in \{0.2, 0.4\}$): Percentage Improvement in the Acceptance Rate Compared to GP_{CI}.

Instance			Online scheduling policy										
				R_R-GP_{CI} (H)					R_R-VFA_{CI} (H)			S-PbP	PbP
Λ	K	Dist.	GP_R	$H{=}25$	$H{=}50$	$H{=}100$	VFA_{CI}	VFA_R	$H{=}25$	$H{=}50$	$H{=}100$		
0.2	2	UTI	+2.0*	+21.2	+22.7	+23.4	+34.6	+35.1	+36.5	+36.9	+37.3	+39.2	+39.3
		CTI	- 0.9*	+21.2	+22.8	+23.4	+30.0	+32.7	+36.1	+37.6	+37.6	+38.9	+38.5
		CTD	+4.7*	+28.3	+29.6	+30.6	+45.1	+45.9	+46.7	+48.3	+48.3	+49.6	+49.9
	3	UTI	+6.5	+19.7	+21.0	+22.3	+15.3	+21.2	+23.7	+24.9	+26.6	+26.5	+27.6
		CTI	+3.0	+18.7	+20.8	+22.5	+14.2	+17.3	+23.0	+24.4	+25.4	+20.7	+25.0
		CTD	+3.5*	+13.8	+16.3	+17.8	+5.0	+8.1	+16.8	+18.3	+19.1	+15.9	+21.7
0.4	3	UTI	+4.1*	+18.8	+22.1	+23.8	+25.4	+30.7	+30.1	+32.3	+32.5	+32.2	+34.4
		CTI	+7.2	+28.3	+30.2	+31.2	+37.2	+35.9	+45.5	+47.3	+48.7	+46.9	+48.3
		CTD	- 2.8*	+18.6	+21.7	+23.6	+27.5	+25.8	+29.1	+31.0	+32.0	+31.9	+33.5
	5	UTI	+5.0	+10.3	+12.7	+14.9	+10.0	+8.4	+13.9	+16.3	+17.6	+11.8	+18.0
		CTI	+0.3*	+10.2	+13.2	+15.5	+11.0	+11.8	+14.1	+16.6	+17.6	+11.7	+17.1
		CTD	+0.1*	+10.3	+13.3	+15.1	+10.1	+11.5	+14.9	+17.0	+18.4	+16.3	+18.1
	Overall		+2.8	+16.4	+18.7	+20.4	+17.7	+19.4	+23.6	+25.4	+26.4	+23.9	+27.1

* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).

Table D3
Comparison of Online Scheduling Policies ($\Lambda \in \{0.2, 0.4\}$): Maximum Decision Time (seconds).

Instance			Online scheduling policy											
					R_R-GP_{CI} (H)					R_R-VFA_{CI} (H)			S-PbP	PbP
Λ	K	Dist.	GP_{CI}	GP_R	$H{=}25$	$H{=}50$	$H{=}100$	VFA_{CI}	VFA_R	$H{=}25$	$H{=}50$	$H{=}100$		
0.2	2	UTI	<0.1	0.3	0.9	2.0	3.1	<0.1	0.2	1.0	2.1	3.8	0.3	0.4
		CTI	<0.1	0.3	1.0	1.7	3.1	<0.1	0.3	0.9	2.0	3.9	0.4	0.4
		CTD	<0.1	0.2	0.9	1.6	3.1	<0.1	0.2	0.9	1.7	3.6	0.4	0.6
	3	UTI	<0.1	0.1	1.9	3.1	6.3	<0.1	0.1	1.8	3.5	6.3	0.4	0.6
		CTI	<0.1	0.2	1.6	3.0	6.0	<0.1	0.2	1.6	3.1	7.0	0.3	0.5
		CTD	<0.1	0.1	1.6	3.7	6.1	<0.1	0.2	1.9	3.2	7.2	0.3	0.5
0.4	3	UTI	<0.1	0.4	3.1	6.1	14.1	<0.1	0.2	3.2	7.3	14.5	0.6	1.0
		CTI	<0.1	0.4	3.1	6.1	14.1	<0.1	0.3	3.1	7.2	14.3	0.6	1.0
		CTD	<0.1	0.3	3.1	6.1	12.0	<0.1	0.2	3.1	6.3	14.2	0.6	1.1
	5	UTI	<0.1	0.1	7.4	14.6	29.2	<0.1	0.1	7.5	14.9	34.3	0.7	2.0
		CTI	<0.1	0.2	7.3	14.6	29.2	<0.1	0.2	8.7	17.1	34.5	0.8	2.2
		CTD	<0.1	0.3	7.2	14.4	33.2	<0.1	0.3	8.6	16.8	33.7	0.8	2.4
	Overall		<0.1	0.4	7.4	14.6	33.2	<0.1	0.3	8.7	17.1	34.5	0.8	2.4

Table D4
Comparison of Online Scheduling Policies ($\Lambda \in \{0.8, 1.5\}$): Acceptance Rate (%) and Percentage Improvement Compared to GP_{CI}.

Instance			Online scheduling policy							
Λ	K	Dist.	GP_{CI}	GP_R	R_R-GP_{CI} (10)	VFA_{CI}	VFA_R	R_R-VFA_{CI} (10)	S-PbP	PbP
0.8	6	UTI	30.9	31.8(+2.9*)	32.3(+4.7)	36.8(+19.4)	37.2(+20.5)	35.9(+16.5)	38.9(+26.2)	39.7(+28.6)
		CTI	30.6	30.0(- 2.0*)	33.3(+8.8)	37.3(+21.6)	36.4(+18.8)	37.8(+23.4)	40.2(+31.3)	41.3(+34.9)
		CTD	30.5	30.7(+0.7*)	33.0(+8.0)	36.7(+20.4)	36.0(+18.1)	37.0(+21.3)	39.8(+30.5)	40.3(+32.2)
	12	UTI	62.9	61.8(- 1.8*)	60.9(- 3.3)	64.6 (+2.7)	64.6 (+2.7)	63.5 (+0.9)	68.3 (+8.5)	72.8(+15.7)
		CTI	64.8	63.5(- 2.0*)	63.8(- 1.5)	66.7 (+2.9)	66.8 (+3.1)	65.7 (+1.4)	69.0 (+6.6)	74.6(+15.2)
		CTD	63.9	63.8(- 0.2*)	62.6(- 2.1)	66.3 (+3.8)	67.0 (+4.8)	65.0 (+1.7)	71.0(+11.1)	74.1(+16.0)
1.5	10	UTI	33.4	35.7(+7.0)	34.6(+3.8)	39.7(+19.1)	39.9(+19.7)	39.9(+19.5)	43.1(+29.0)	44.2(+32.5)
	20	UTI	62.4	64.5(+3.4)	61.6(- 1.4)	67.1 (+7.4)	66.7 (+6.8)	65.0 (+4.1)	70.8(+13.4)	76.9(+23.2)
	Overall		47.3	47.1(- 0.4*)	47.7(+0.8)	51.5 (+8.9)	51.5 (+8.8)	50.9 (+7.6)	54.7(+15.6)	57.4(+21.2)

* indicates the improvements that are NOT statistically significant (i.e., $p > .05$ in paired two-sample t-tests).

Table D5
Comparison of Online Scheduling Policies ($\Lambda \in \{0.8, 1.5\}$): Decision Times (seconds).

Instance			Online scheduling policy														
			GP_{CI}		GP_R		R_R-GP_{CI} (10)		VFA_{CI}		VFA_R		R_R-VFA_{CI} (10)		S-PbP		PbP	
Λ	K	Dist.	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
0.8	6	UTI	<0.1	<0.1	0.1	0.4	2.7	8.0	<0.1	<0.1	0.1	0.4	3.9	9.4	0.8	1.8	2.3	6.1
		CTI	<0.1	<0.1	0.1	0.6	2.7	8.0	<0.1	<0.1	0.1	0.7	4.1	9.4	0.8	1.8	2.2	5.5
		CTD	<0.1	<0.1	0.1	0.6	2.7	7.8	<0.1	<0.1	0.1	0.7	4.1	9.2	0.9	1.9	2.4	6.0
	12	UTI	<0.1	<0.1	0.1	0.2	13.9	29.3	<0.1	<0.1	0.1	0.2	14.6	29.1	1.5	3.4	9.2	23.8
		CTI	<0.1	<0.1	0.1	0.3	13.9	28.1	<0.1	<0.1	0.1	0.3	14.8	30.4	1.6	3.7	9.1	23.3
		CTD	<0.1	<0.1	0.1	0.2	15.5	32.5	<0.1	<0.1	0.1	0.2	14.4	28.0	1.6	3.6	8.5	20.7
1.5	10	UTI	<0.1	<0.1	0.1	0.5	13.5	40.0	<0.1	<0.1	0.1	0.4	19.3	39.7	2.6	5.7	11.2	27.0
	20	UTI	<0.1	<0.1	0.1	0.2	80.6	158.4	<0.1	<0.1	0.1	0.3	81.8	154.1	5.1	11.3	51.8	124.5
	Overall		<0.1	<0.1	0.1	0.6	11.0	158.4	<0.1	<0.1	0.1	0.7	11.9	154.1	1.3	11.3	7.2	124.5

CTD. The results shown in Tables D.1, D.2, D.3, D.4 and D.5 correspond to those shown in Tables 3, 5, 6, 7 and 8 in Section 6, respectively.

References

Azi, N., Gendreau, M., & Potvin, J.-Y. (2012). A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research, 199*(1), 103–112.

Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research, 59*(5), 1269–1283.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329.

Bent, R. W., & Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research, 52*(6), 977–987.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Bertsimas, D. J., & Van Ryzin, G. (1991). A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research, 39*(4), 601–615.

Branke, J., Middendorf, M., Noeth, G., & Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science, 39*(3), 298–312.

Chen, Z.-L., & Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science, 40*(1), 74–88.

Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science, 53*(4), 946–985.

Daganzo, C. F. (1984a). 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.

Daganzo, C. F. (1984b). The length of tours in zones of different shapes. *Transportation Research Part B: Methodological, 18*(2), 135–145.

Ferrucci, F., & Bock, S. (2015). A general approach for controlling vehicle en-route diversions in dynamic vehicle routing problems. *Transportation Research Part B: Methodological, 77*, 76–87.

Ferrucci, F., & Bock, S. (2016). Pro-active real-time routing in applications with multiple request patterns. *European Journal of Operational Research, 253*(2), 356–371.

Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science, 33*(4), 381–390.

van Heeswijk, W. J., Mes, M. R., & Schutten, J. M. (2019). The delivery dispatching problem with time windows for urban consolidation centers. *Transportation Science, 53*(1), 203–221.

Hvattum, L. M., Løkketangen, A., & Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science, 40*(4), 421–438.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science, 40*(2), 211–225.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2007). *Planned route optimization for real–time vehicle routing*. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, & I. Minis (Eds.) (pp. 1–18)). Boston, MA: Springer US.

Klapp, M. A., Erera, A. L., & Toriello, A. (2018). The dynamic dispatch waves problem for same-day delivery. *European Journal of Operational Research, 271*(2), 519–534.

Larsen, A., Madsen, O., & Solomon, M. (2002). Partially dynamic vehicle routing—models and algorithms. *Journal of the Operational Research Society, 53*(6), 637–646.

Mitrović-Minić, S., Krishnamurti, R., & Laporte, G. (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological, 38*(8), 669–685.

OSM (2020). OpenStreetMap project database, accessed August 7, 2020, http://planet.openstreetmap.org.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research, 225*(1), 1–11.

Powell, W. B. (2011). Approximate dynamic programming: Solving the curses of dimensionality. *Wiley Series in Probability and Statistics* (2nd). Hoboken, New Jersey: John Wiley & Sons, Inc..

Powell, W. B., & Meisel, S. (2015). Tutorial on stochastic optimization in energy—-part ii: An energy storage illustration. *IEEE Transactions on Power Systems, 31*(2), 1468–1475.

Powell, W. B., Simao, H. P., & Bouzaiene-Ayari, B. (2012). Approximate dynamic programming in transportation and logistics: A unified framework. *EURO Journal on Transportation and Logistics, 1*(3), 237–284.

Psaraftis, H. N., Wen, M., & Kontovas, C. A. (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks, 67*(1), 3–31.

Ritzinger, U., Puchinger, J., & Hartl, R. F. (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research, 54*(1), 215–231.

Soeffker, N., Ulmer, M. W., & Mattfeld, D. C. (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research, 298*(3), 801–820.

Tassiulas, L. (1996). Adaptive routing on the plane. *Operations Research, 44*(5), 823–832.

Thomas, B. W. (2007). Waiting strategies for anticipating service requests from known customer locations. *Transportation Science, 41*(3), 319–331.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Ulmer, M. W. (2020). Dynamic pricing and routing for same-day delivery. *Transportation Science, 54*(4), 1016–1033.

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Hennig, M. (2019). Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science, 53*(1), 185–202.

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., & Thomas, B. W. (2020). On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics, 9*(2), 100008.

Ulmer, M. W., Mattfeld, D. C., & Köster, F. (2018a). Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transportation Science, 52*(1), 20–37.

Ulmer, M. W., Soeffker, N., & Mattfeld, D. C. (2018b). Value function approximation for dynamic multi-period vehicle routing. *European Journal of Operational Research, 269*(3), 883–899.

Ulmer, M. W., & Streng, S. (2019). Same-day delivery with pickup stations and autonomous vehicles. *Computers & Operations Research, 108*, 1–19.

Ulmer, M. W., & Thomas, B. W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks, 72*(4), 475–505.

Ulmer, M. W., & Thomas, B. W. (2020). Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. *European Journal of Operational Research, 285*(1), 183–195.

Voccia, S. A., Campbell, A. M., & Thomas, B. W. (2019). The same-day delivery problem for online purchases. *Transportation Science, 53*(1), 167–184.