Offline-Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests

Marlin W. Ulmer Justin C. Goodson Dirk C. Mattfeld

Marco Hennig

April 4, 2017

Abstract

Although increasing amounts of transaction data make it possible to characterize uncertainties surrounding customer service requests, few methods integrate predictive tools with prescriptive optimization procedures to meet growing demand for small-volume urban transport services. We incorporate temporal and spatial anticipation of service requests into approximate dynamic programming (ADP) procedures to yield dynamic routing policies for the *single-vehicle routing problem with stochastic service requests*, an important problem in city-based logistics. We contribute to the routing literature as well as to the field of ADP. We combine offline value function approximation (VFA) with online rollout algorithms resulting in a high-quality, computationally tractable policy. Our offline-online policy enhances the anticipation of the VFA policy, yielding spatial and temporal anticipation of requests and routing developments. Our combination of VFA with rollout algorithms demonstrates the potential benefit of using offline and online methods in tandem as a hybrid ADP procedure, making possible higher-quality policies with reduced computational requirements for real-time decision-making. Finally, we identify a policy improvement guarantee applicable to VFA-based rollout algorithms, showing that base policies composed of deterministic decision rules lead to rollout policies with performance at least as strong as that of their base policy.

1 Introduction

By the year 2050, two-thirds of the world's population is expected to reside in urban areas (United Nations, 2015). With many businesses' operations already centralized in cities (Jaana et al., 2013), urbanization coupled with growth in residential e-commerce transactions (Capgemini, 2012) will significantly increase the demand for small-volume urban transport services. Concurrent with rising demand for city-based logistics is a significant increase in the availability of transaction data, enabling firms to better characterize uncertainties surrounding the quantities, locations, and timings of future orders. Although the data required to anticipate customer requests are readily available, few methods integrate predictive tools with prescriptive optimization methods to anticipate and dynamically respond to requests. In this paper, we combine online and offline approximate dynamic programming (ADP) procedures to yield dynamic vehicle routing policies that temporally and spatially anticipate service requests. Our work addresses in part the growing complexities of urban transportation and makes general contributions to the field of ADP.

Vehicle routing problems (VRPs) with stochastic service requests underlie many operational challenges in logistics and supply chain management (Psaraftis et al., 2015). These challenges are characterized by the need to design routes for vehicles to meet customer service requests arriving randomly over a given geographical area and time horizon. For example, package express firms (e.g., local couriers and United Parcel Service) often begin a working day with a set of known service requests and may dynamically adjust vehicle routes to accommodate additional calls arriving throughout the day (Hvattum et al., 2006). Similarly, less-than-truckload logistics or service technicians (e.g., roadside assistance and utilities employees) may periodically adjust preliminary schedules to accommodate requests placed after the start of daily business (Jaillet, 1985; Thomas, 2007). In each of these examples, past customer transaction data can be used to derive probability distributions on the timing and location of potential customers requests, thus opening the door to the possibility of dynamically adjusting vehicle routes in anticipation of future requests.

Although a stream of routing literature focuses on VRPs with stochastic service requests, only a small portion of this research explicitly employs spatial and temporal anticipation of requests and routing developments to dynamically move vehicles in response to customer calls. Figure 1 illustrates the potential value of anticipatory routing when the task is to dynamically direct a vehicle

to serve all requests made prior to the beginning of a work day and as many requests arriving during the work day as possible. Though a dispatcher may manage multiple vehicles, the focus of this example is the assignment of requests to a single vehicle. The upper left portion of Figure 1 shows the vehicle's current position 02:00 hours after the start of work, a tentative route through three assigned customers that must conclude at the depot by 06:00 hours, two new service requests, and three yet-to-be-made and currently unknown service requests along with the times the requests are made. In this example, the vehicle traverses a Manhattan-style grid where each unit requires 00:15 hours of travel time. Because assigning both current requests is infeasible, at least one must be rejected (denoted by a cross through the request), a term we use to indicate the request will not be served by the vehicle. We do not reconsider rejected requests. A rejected request may be served by a third party or on the following work day. The bottom half of Figure 1 depicts the potential consequences of assigning each request, showing more customers can be serviced by assigning the current bottom-right request than by assigning the current top-left request, and thus demonstrating the potential benefit of combining anticipation with routing decisions.

A natural model to join anticipation of customer requests with dynamic routing is a Markov decision process (MDP), a decision-tree model for dynamic and stochastic optimization problems. Although VRPs with stochastic service requests can be formulated as MDPs, for many problems of practical interest, it is computationally intractable to solve the Bellman value functions and obtain an optimal policy (Powell, 2011). Consequently, much of the research in dynamic routing has focused on decision-making via suboptimal heuristic policies. For VRPs with stochastic service requests, while the literature identifies heuristic methods to make real-time route adjustments, many of the resulting policies do not leverage anticipation of customer requests to make better decisions.

One approach to incorporate anticipation into dynamic routing is offline value function approximation (VFA), often consisting of an iterative simulation-optimization procedure to approximate rewards-to-go via aggregate state representations. However, both temporal and spatial anticipation is challenging to achieve in an offline setting. As evidenced by the work of Ulmer et al. (2016), for most problems of practical interest, computational limitations restrict VFAs to low-dimensional state representations. Further, though state-of-the-art, Ulmer et al. (2016) rely solely on temporal aspects of the state variable to estimate rewards-to-go, ignoring the potential benefits of spatial

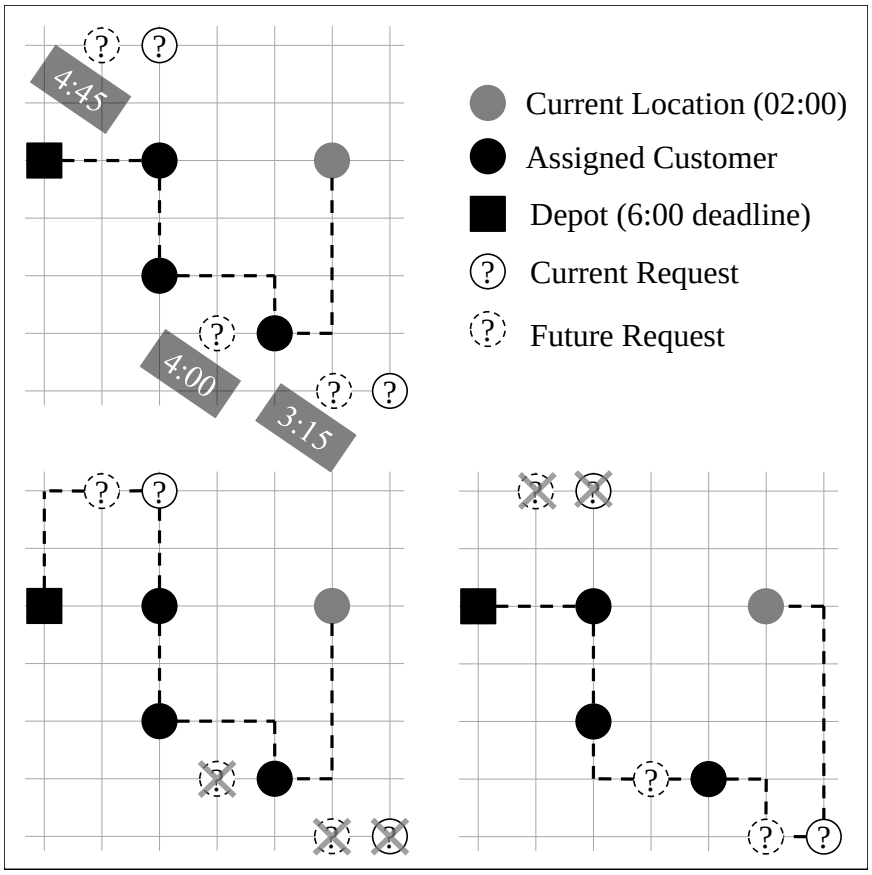


Figure 1: Anticipating Times and Locations of Future Requests

anticipation.

In this paper, we propose a dynamic routing method that augments offline VFA with online rollout algorithms and which is suitable for larger problem instances. Introduced by Bertsekas et al. (1997), a rollout algorithm builds a portion of the current-state decision tree and then uses a given base policy – in this paper the temporal VFA policy of Ulmer et al. (2016) – to approximate the remainder of the tree via their base policy's rewards-to-go. We find rollout algorithms compensate for anticipation of details absent in the base policy, thus a rollout algorithm adds spatial anticipation to the temporal base policy. Indeed, we observe the performance of our offline-online ADP heuristic significantly improves on the performance of the temporal VFA policy in isolation and scales well to large problem instances.

We make contributions to the literature on VRPs with stochastic service requests as well as general methodological contributions to the field of ADP:

Contributions to Vehicle Routing

We make two contributions to the routing literature. First, our offline-online approach yields computationally-tractable, high-quality dynamic routing policies. Further, we achieve temporal-spatial anticipation by pairing with a rollout algorithm the temporal VFA policy of Ulmer et al. (2016). Because a rollout algorithm explicitly builds a portion of the decision tree and looks ahead to potential future states, the resulting rollout policy is anticipatory by definition, even when built on a non-anticipatory base policy. Looking to the broader routing literature and toward general dynamic and stochastic optimization problems, we believe rollout algorithms may serve as a means to enhance anticipatory decision-making and connect data-driven predictive tools with optimization.

Second, we explore the merits of temporal anticipation versus those of spatial anticipation when dynamically routing a vehicle to meet stochastic service requests. Comparing a simulation-based spatial VFA with the temporal VFA of Ulmer et al. (2016), we identify the geographic spread of customer locations as a predictor of the success of temporal versus spatial anticipation. As the distribution of customer locations moves from uniform toward clustered across a service area, anticipation based on service area coverage tends to outperform temporal anticipation and vice versa.

Contributions to Approximate Dynamic Programming

We make three methodological contributions to the broader field of ADP. First, we combine offline VFA with with online rollout algorithms. Our combination of VFAs and rollout algorithms demonstrates the potential benefit of using offline and online methods in tandem as a hybrid ADP procedure. Via offline simulations, VFAs potentially capture the overarching structure of an MDP (Powell, 2011) via low-dimensional state representations. In contrast, online rollout algorithms typically examine small portions of the state space in full detail, but due to computational considerations are limited to local observations of MDP structure. As our work demonstrates, combining VFAs with rollout algorithms merges global structure with local detail, bringing together the advantages of offline learning with the online, policy-improving machinery of rollout. In particular, our computational experiments demonstrate a combination of offline and online effort significantly reduces online computation time while yielding policy performance comparable to that of online or offline methods in isolation.

Second, we identify a policy improvement guarantee applicable to VFA-based rollout algorithms. Specifically, we demonstrate any base policy composed of *deterministic* decision rules – functions that always select the same decisions when applied in the same states – leads to rollout policies with performance at least as good as that of the base policy. Such decision rules might take the form of a VFA, a deterministic mathematical program where stochastic quantities are replaced with their mean values, a local search on a priori policies, or a threshold-style rule based on key parameters of the state variable. This general result explains why improvement over the underlying VFA policy can be expected when used in conjunction with rollout algorithms and points toward hybrid ADP methods as a promising area of research.

Our contributions to ADP extend the work of Li and Womer (2015), which combines rollout algorithms with VFA to dynamically schedule resource-constrained projects. We go beyond Li and Womer (2015) by identifying conditions necessary to achieve a performance improvement guarantee, thus making our treatment of VFA-based rollout applicable to general MDPs. Further, our computational work explicitly examines the tradeoffs between online and offline computation, thereby adding insight to the work of Li and Womer (2015).

Finally, as a minor contribution, our work is the first to combine with rollout algorithms the *indifference zone selection* (IZS) procedure of Kim and Nelson (2001, 2006). As our computational results demonstrate, using IZS to systematically limit the number of simulations required to estimate rewards-to-go in a rollout algorithm can significantly reduce computation time without degrading policy quality.

The remainder of the paper is structured as follows. In §2, we formally state and model the problem. Related literature is reviewed in §3. We describe our offline-online ADP approach in §4 and benchmark heuristics in §5 followed by a presentation of computational experience in §6. We conclude the paper in §7.

2 Problem Statement and Formulation

The VRPSSR is characterized by the need to dynamically design a route for one vehicle to meet service calls arriving randomly over a working day of duration T and within a service region $\mathcal{C}$. The duration limit may account for both work rules limiting an operator's day (U.S. Department

of Transportation Federal Motor Carrier Safety Administration, 2005) as well as a cut-off time required by pickup and delivery companies so deadlines for overnight linehaul operations can be met. The objective of the VRPSSR is to identify a dynamic routing policy, beginning and ending at a depot, that serves a set of early-request customers $\mathcal{C}_{\text{early}} \subseteq \mathcal{C}$ known prior to the start of the working day and that maximizes the expected number of serviced late-request customers who submit requests throughout the working day. The objective reflects the fact that operator costs are largely fixed (ATA Economics & Statistical Analysis Department, 1999; The Tioga Group, 2003), thus companies wish to maximize the use of operators' time by serving as many customers as the day they request possible.

We model the VRPSSR as an MDP. The state of the system at decision epoch k is the tuple $s_k = (c_k, t_k, \bar{\mathcal{C}}_k, \tilde{\mathcal{C}}_k)$, where c_k is the vehicle's position in service region $\mathcal{C}$ representing a customer location or the depot, $t_k \in [0, T]$ is the time at which the vehicle arrives to location c_k and marks the beginning of decision epoch k, $\bar{\mathcal{C}}_k \subseteq \mathcal{C}$ is the set of assigned customers not yet serviced, and $\tilde{\mathcal{C}}_k \subseteq \mathcal{C}$ is a (possibly empty) set of service requests made at or prior to time t_k but after time t_{k-1}, the time associated with decision epoch $k-1$. In initial state $s_0 = (\text{depot}, 0, \mathcal{C}_{\text{early}}, \emptyset)$, the vehicle is positioned at the depot at time zero, has yet to serve the early-request customers composing $\mathcal{C}_{\text{early}}$, and the set of late-request customers is empty. To guarantee feasibility, we assume there exists a route from the depot, through each customer in $\mathcal{C}_{\text{early}}$, and back to the depot with duration less than or equal to T. At final decision epoch K, which may be a random variable, the process occupies a terminal state s_K in the set $\{(\text{depot}, t_K, \emptyset, \emptyset) : t_K \in [0, T]\}$, where the vehicle has returned to the depot by time T, has serviced all early-request customers, and we assume the final set of requests $\tilde{\mathcal{C}}_K$ is empty.

A decision permanently assigns or rejects each service request in $\tilde{\mathcal{C}}_k$ and directs the vehicle to a new location c in service region $\mathcal{C}$. We denote a decision as the pair $x = (a, c)$, where a is a $|\tilde{\mathcal{C}}_k|$-dimensional binary vector indicating assignment (equal to one) or rejection (equal to zero) of each request in $\tilde{\mathcal{C}}_k$. When the process occupies state s_k at decision epoch k, the set of feasible decisions is

$$\mathcal{X}(s_k) = \Big\{(a, c) :$$

$$a \in \{0,1\}^{|\tilde{\mathcal{C}}_k|}, \tag{1}$$

$$c \in \bar{\mathcal{C}}_k \cup \tilde{\mathcal{C}}'_k \cup \{c_k\} \cup \{\text{depot}\}, \tag{2}$$

$$c \neq \text{ depot if } \bar{\mathcal{C}}_k \cup \tilde{\mathcal{C}}'_k \setminus \{c_k\} \neq \emptyset, \tag{3}$$

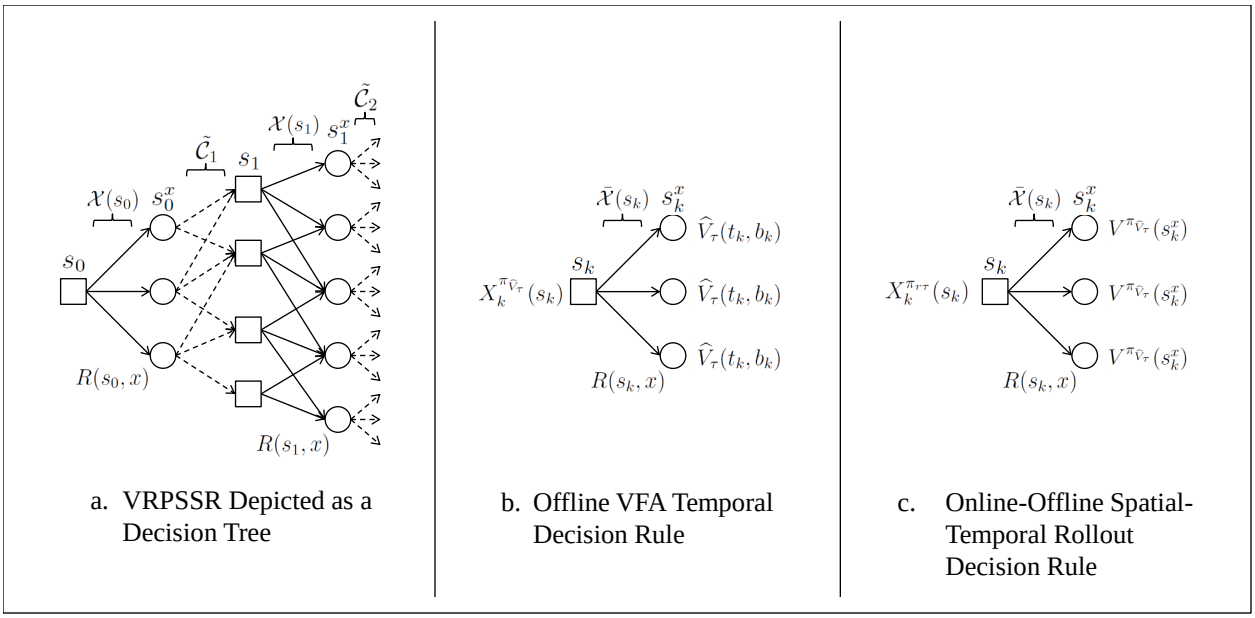
$$\left.\text{feasible routing}\right\}. \tag{4}$$

Condition (1) requires each service request in $\tilde{\mathcal{C}}_k$ to be assigned to this vehicle or rejected. Condition (2) constrains the vehicle's next location to belong to the set $\bar{\mathcal{C}}_k \cup \tilde{\mathcal{C}}'_k \cup \{c_k\} \cup \{\text{depot}\}$, where $\tilde{\mathcal{C}}'_k = \{c \in \tilde{\mathcal{C}}_k : a_{\tilde{\mathcal{C}}_k^{-1}(c)} = 1\}$ is the set of customers assigned by a and $\tilde{\mathcal{C}}_k^{-1}(c)$ returns the index of element c in $\tilde{\mathcal{C}}_k$. Setting $c = c_k$ is the decision to wait at the current location for a base unit of time $\bar{t}$. Per condition (3), travel to the depot is disallowed when assigned customers in $\bar{\mathcal{C}}_k$ and $\tilde{\mathcal{C}}'_k$ have yet to be serviced. Condition (4) requires a route exists from the current location, through all assigned customers, and back to the depot with duration less than or equal to the remaining time $T - t_k$ less any time spent waiting at the current location. Because determining whether or not given values of a and c satisfy condition (4) may require the optimal solution value of an open traveling salesman problem, identifying the full set of feasible decisions may be computationally prohibitive. In the Appendix, we describe a cheapest insertion method to quickly check if condition (4) is satisfied.

When the process occupies state s_k and decision x is taken, a reward is accrued equal to the number of assigned late-request customers: $R(s_k, x) = |\tilde{\mathcal{C}}'_k(s_k, x)|$, where $\tilde{\mathcal{C}}'_k(s_k, x)$ is the set $\tilde{\mathcal{C}}'_k$ specified by state s_k and a, the assignment component of decision x.

Choosing decision x when in state s_k transitions the process to post-decision state $s_k^x = (c_k, t_k, \bar{\mathcal{C}}_k^x)$ where the set of assigned customers $\bar{\mathcal{C}}_k^x = \bar{\mathcal{C}}_k \cup \tilde{\mathcal{C}}'_k$ is updated to include the newly assigned requests. How the process transitions to pre-decision state $s_{k+1} = (c_{k+1}, t_{k+1}, \bar{\mathcal{C}}_{k+1}, \tilde{\mathcal{C}}_{k+1})$ depends on whether or not decision x directs the vehicle to wait at its current location. If $c \neq c_k$, then decision epoch $k+1$ begins upon arrival to position c. Denoting known travel times between two locations in $\mathcal{C}$ via the function $d(\cdot, \cdot)$, the vehicle's current location is updated to $c_{k+1} = c$, the time of arrival to c_{k+1} is $t_{k+1} = t_k + d(c_k, c_{k+1})$, and $\bar{\mathcal{C}}_{k+1} = \bar{\mathcal{C}}_k^x \setminus \{c_k\}$ is updated to reflect service at the vehicle's previous location c_k. If $c = c_k$, then decision epoch $k+1$ begins after the wait time of $\bar{t}$. The arrival time $t_{k+1} = t_k + \bar{t}$ is incremented by the waiting time and $\bar{\mathcal{C}}_k = \bar{\mathcal{C}}_k^x$ is unchanged. At the next decision epoch $k+1$, a new set $\tilde{\mathcal{C}}_{k+1}$ of late-request customers may be observed.

Denote a policy π by a sequence of decision rules $(X_0^\pi, X_1^\pi, \ldots, X_K^\pi)$, where each decision rule



a. VRPSSR Depicted as a Decision Tree

b. Offline VFA Temporal Decision Rule

c. Online-Offline Spatial-Temporal Rollout Decision Rule

Figure 2: MDP Model and Heuristic Decision Rules

$X_k^\pi(s_k) : s_k \mapsto \mathcal{X}(s_k)$ is a function mapping the current state to a feasible decision. Letting Π be the set of all Markovian deterministic policies, we a seek a policy π in Π that maximizes the expected total reward conditional on initial state s_0: $\mathbb{E}[\sum_{k=0}^{K} R(s_k, X_k^\pi(s_k))|s_0]$.

Figure 2a depicts the MDP model as a decision tree, where square nodes represent pre-decision states, solid arcs depict decisions, round nodes are post-decision states, and dashed arcs denote realizations of random service requests. The remainder of Figure 2 is discussed in subsequent sections.

3 Related Literature

In this section we discuss vehicle routing literature where the time and/or location of service requests is uncertain. Following a narrative of the extant literature, we classify each study according to its solution approach and mechanism of anticipation.

For problems where both early- and late-request customers must be serviced, Bertsimas and Van Ryzin (1991), Tassiulas (1996), Swihart and Papastavrou (1999), and Larsen et al. (2002) explore simple rules to dynamically route the vehicle with the objective of minimizing measures of route cost and/or customer wait time. For example, a first-come-first-serve rule moves the

vehicle to requests in the order they are made and a nearest-neighbor rule routes the vehicle to the closest customer. Although our methods direct vehicle movement via explicit anticipation of future customer requests, the online decision-making of rule-based schemes is at a basic level akin to our use of rollout algorithms, which execute on-the-fly all computation necessary to select a feasible decision.

In contrast to the rule-based methods of early literature, Psaraftis (1980) re-optimizes a route through known customers whenever a new request is made and uses the route to direct vehicle movement. Building on the classical insertion method of Psaraftis (1980), Gendreau et al. (1999, 2006) use a tabu search heuristic to re-plan routing when new requests are realized. Similarly, Chen et al. (2006) and Lin et al. (2014) apply route construction and improvement heuristics to dynamically route new requests. Ichoua et al. (2000) augment Gendreau et al. (1999, 2006) by allowing mid-route adjustments to vehicle movement and Mitrović-Minić and Laporte (2004) extend Gendreau et al. (1999, 2006) by dynamically halting vehicle movement via waiting strategies. Ichoua et al. (2006) also explore waiting strategies to augment the method of Gendreau et al. (1999, 2006), but explicitly consider the likelihood of requests across time and space in their wait-or-not decision. Similarly, Branchini et al. (2009) heuristically solve deterministic routing problems at each decision epoch with consideration of various waiting strategies. Additionally, within a genetic algorithm, van Hemert and La Poutré (2004) give preference to routes more capable of accommodating future requests. Likewise, Ferrucci et al. (2013) consider the locations of potential future requests in a tabu search framework. With the exception of Ichoua et al. (2006), van Hemert and La Poutré (2004), and Ferrucci et al. (2013), these heuristic methods only work on currently available information and do not account for uncertainty in future requests. In our work, we seek to explicitly anticipate customer requests across time and space.

Building on the idea of Psaraftis (1980), Bent and Van Hentenryck (2004) and Hvattum et al. (2006) iteratively re-optimize a collection of routes whenever a new request is made and use the routes to direct vehicle movement. Each route in the collection sequences known service requests as well as a different random sample of future service requests. Using a "consensus" function, Bent and Van Hentenryck (2004) and Hvattum et al. (2006) identify the route most similar to other routes in the collection and use this sequence to direct vehicle movement. Ghiani et al. (2009) proceed similarly, sampling potential requests in the short-term future, but use the collection of routes to

estimate expected costs-to-go instead of to directly manage location decisions. Motivated by this literature, the spatial VFA we consider in §5.1 approximates service area coverage via simulation and routing of requests.

Branke et al. (2005) explore a priori strategies to distribute waiting time along a fixed sequence of early-request customers with the objective of maximizing the probability of feasible insertion of late-request customers. Thomas (2007) also examines waiting strategies, but allow the vehicle to dynamically adjust movement with the objective of maximizing the expected number of serviced late-request customers. Using center-of-gravity-style heuristics, the anticipatory policies of Thomas (2007) outperform the waiting strategies of Mitrović-Minić and Laporte (2004). Further, Ghiani et al. (2012) demonstrate the basic insertion methods of Thomas (2007) perform comparably to the more computationally intensive scenario-based policies of Bent and Van Hentenryck (2004), an insight we employ in the spatial approximation of §5.1 where we sequence customers via cheapest insertion. Similar to our work, these methods explicitly anticipate customer requests. However, unlike Thomas (2007), we do not know in advance the locations of potential service requests, thereby increasing the difficulty of the problem and making our methods more general.

In contrast to much of the literature in our review, the methods of Meisel (2011) give explicit consideration to the timing of service requests and to customer locations. Using approximate value iteration (AVI), Meisel (2011) develops spatial-temporal VFAs and obtains high-quality policies for three-customer problems. Acknowledging Meisel (2011) as a proof-of-concept, Ulmer et al. (2016) extend the ideas of Meisel (2011) to practical-sized routing problems by developing computationally tractable methods to identify temporal-only VFAs leading to high-quality dynamic routing policies. We further describe the work of Ulmer et al. (2016) in §4.1 and discuss in §4.2 how the combination of a temporal-only VFA with a rollout algorithm leads to a spatial-temporal policy suitable for large-scale problems.

To conclude our review, we present Table 1 as an additional perspective on the vehicle routing literature where the time and/or location of service requests is unknown. Table 1 classifies the extant literature across several dimensions with respect to the solution method employed and the anticipation of future customer requests. The bottom row of Table 1 represents the work in this paper. A check mark in the "Subset Selection" column indicates the method explicitly addresses customer assignments and rejections. Papers focusing on fewer decisions typically give

Table 1: Literature Classification

	Solution Approach			Anticipation			
Literature	**Subset Selection**	**Online**	**Offline**	**Future Value**	**Stochastic**	**Temporal**	**Spatial**
Psaraftis (1980)	n/a	✓					
Bertsimas and Van Ryzin (1991)	n/a	✓					
Tassiulas (1996)	n/a	✓					
Gendreau et al. (1999)	n/a	✓					
Swihart and Papastavrou (1999)	n/a	✓					
Ichoua et al. (2000)	n/a	✓					✓
Larsen et al. (2002)	n/a	✓					
Mitrović-Minić and Laporte (2004)	n/a	✓					
Bent and Van Hentenryck (2004)	n/a	✓			✓		✓
van Hemert and La Poutré (2004)		✓		✓	✓		✓
Branke et al. (2005)			✓	✓	✓	✓	✓
Ichoua et al. (2006)	n/a	✓			✓	✓	✓
Chen et al. (2006)	n/a	✓					
Gendreau et al. (2006)	n/a	✓					
Hvattum et al. (2006)	n/a	✓			✓		✓
Thomas (2007)		✓			✓	✓	✓
Branchini et al. (2009)	n/a	✓					✓
Ghiani et al. (2009)	n/a	✓		✓	✓	✓	✓
Meisel (2011)	✓		✓	✓	✓	✓	✓
Ferrucci et al. (2013)	n/a	✓			✓		✓
Lin et al. (2014)	n/a	✓					
Ulmer et al. (2016)	✓		✓	✓	✓		✓
Spatial-Temporal Rollout Policy π_{rr}	✓	✓	✓	✓	✓	✓	✓

explicit consideration to feasible vehicle destinations and then employ a greedy procedure to assign or reject service requests, e.g., the insertion method employed by Thomas (2007). An n/a label in the "Subset Selection" column indicates the problem requires all requests receive service by the vehicles. A check mark in the "Online" column indicates some or all of the calculation required to select a decision is conducted on-the-fly when the policy is executed. For example, the sample-scenario planning of Bent and Van Hentenryck (2004) is executed in real time via online simulations. A check mark in the "Offline" column indicates some or all calculation required to select a decision is conducted prior to policy execution. For instance, the VFAs of Meisel (2011) are determined prior to policy implementation by means of offline simulation.

Notably, only our spatial-temporal rollout policy incorporates online and offline methods to both direct vehicle movement and assign a service requests to the vehicle. Further, only our spatial-

temporal rollout policy combines offline simulations' ability to detect overarching MDP structure with online simulations' capacity to identify detailed MDP structure local to small portions of the state space.

Table 1 classifies the anticipation mechanisms of the extant literature across four dimensions. A check mark in the "Future Value" column indicates for each decision considered by the method, the current-period value and the expected future value (or an estimate of the expected future value) are explicitly calculated. For example, Ghiani et al. (2009) estimates via simulation a measure of customers' current and expected future inconvenience, whereas the simple rules of the early literature (e.g., first-come-first-serve) do not explicitly consider future value when directing vehicle movement. A check mark in the "Stochastic" column indicates the method makes use of stochastic information to select decisions. For instance, Hvattum et al. (2006)'s routing of both known customer requests and potential future requests makes use of stochastic information, while Gendreau et al. (1999)'s consideration of only known requests does not. A check mark in the "Temporal" column indicates the method considers times of potential future customer requests when selecting decisions. For example, Branke et al. (2005)'s a priori distribution of waiting time gives explicit consideration to the likelihoods of future request times, but the waiting strategies of Mitrović-Minić and Laporte (2004) do not. A check mark in the "Spatial" column indicates the method considers locations of potential future customer requests when selecting decisions. For instance, the sample-scenario planning of Bent and Van Hentenryck (2004) estimates service area coverage, whereas Ulmer et al. (2016) focus exclusively on temporal anticipation. Excluding our own work, only three of 18 methods anticipate future service requests across all four dimensions.

4 Offline-Online ADP Heuristic

In this section, we present an offline-online ADP heuristic to dynamically route vehicles. We begin in §4.1 by describing the offline component, the temporal VFA of Ulmer et al. (2016). Then, in §4.2, we embed the offline VFA in an online rollout algorithm. As the computational experiments of §6 suggest, the offline-online combination leads to temporal-spatial anticipation and to better decision-making. Then, in §4.3, we discuss the combination of VFAs and rollout generally, providing a condition sufficient to guarantee a VFA-based rollout policy performs at least as well as

the VFA policy in isolation.

Our motivation for combining online and offline methods is two-fold. First, we aim to add spatial anticipation to the temporal VFA of Ulmer et al. (2016), yielding a method that anticipates customer requests and routing developments over time and across the service region. Second, though in principle both temporal and spatial anticipation might be achieved via pure offline or online approaches, our experience suggests VFA becomes prohibitive when considering more than a few aspects of the state variable and online simulations likewise become prohibitive when significant anticipation is executed on-the-fly. Our offline-online approach aims to deliver temporal-spatial anticipation with reduced on-the-fly computation.

In this section and through the remainder of the paper we operate on a subset $\bar{\mathcal{X}}(s_k) \subseteq \mathcal{X}(s_k)$ of the feasible decisions in a given state s_k. We focus on assignment decisions by disallowing waiting and by making routing decisions via cheapest insertion, thereby increasing the computational tractability of our offline-online ADP heuristic and of the benchmark policies. In the Appendix, we detail the simplification and provide a rationale.

4.1 Offline Temporal VFA

Ulmer et al. (2016) base their offline approach on the well-known value functions, formulated around the post-decision state variable:

$$V(s_k^x) = \mathbb{E}\left[\max_{x \in \mathcal{X}(s_{k+1})} \left\{R(s_{k+1}, x) + V(s_{k+1}^x)\right\} \middle| s_k^x\right]. \tag{5}$$

Although solving equation (5) for all post-decision states s_k^x in each decision epoch $k = 0, \ldots, K-1$ yields the value of an optimal policy, doing so is computationally intractable for most problems of practical interest (Powell, 2011). Thus, Ulmer et al. (2016) develop a VFA by focusing on temporal elements of the post-decision state variable. Specifically, Ulmer et al. (2016) map a post-decision state variable s_k^x to two parameters, the time of arrival to the vehicle's current location t_k and time budget b_k, the duration limit T less the time required to service all assigned customers in $\bar{\mathcal{C}}_k^x$ and return to the depot.

Representing their approximate value function as a two-dimensional lookup table, Ulmer et al. (2016) use AVI (Powell, 2011) to estimate the value of being at time t_k with budget b_k. Ulmer

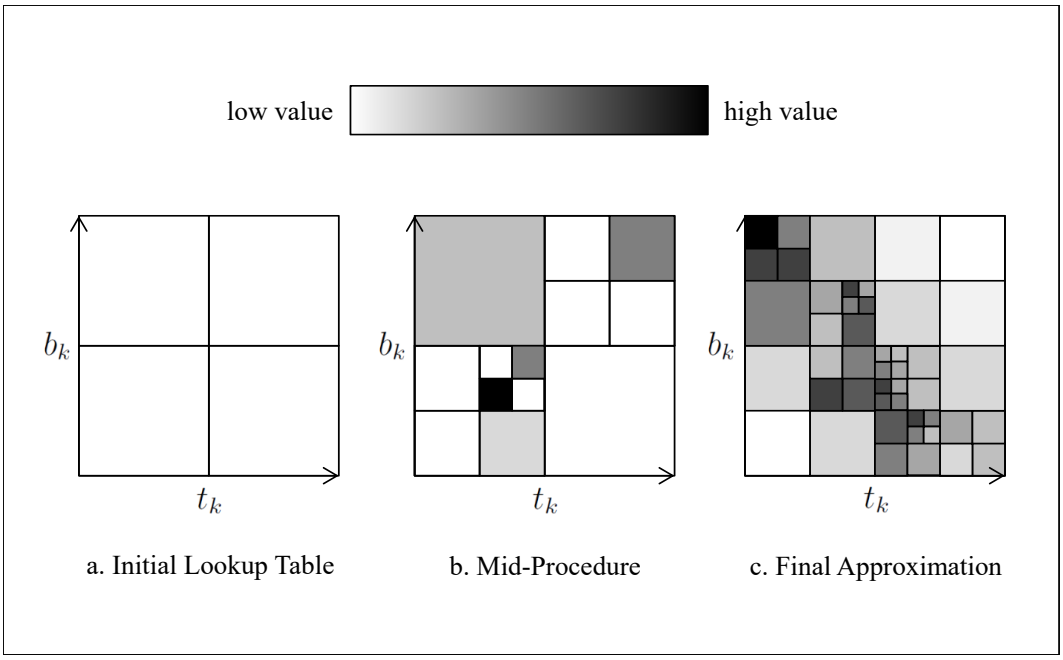


Figure 3: Temporal Value Function Approximation

et al. (2016) build on the classical procedure of iterative simulation, optimization, and smoothing by dynamically adjusting the granularity of the look-up table. At each iteration of the AVI procedure, along a given sample path of customer requests, the look-up table is employed to make Bellman-style decisions, selecting assignment and movement actions that maximize the sum of the immediate reward plus the reward-to-go, as given by the look-up table. Following each simulation, look-up table entries are updated and portions of the look-up table may be subdivided for further exploration in subsequent iterations. The procedure terminates after a given number of iterations.

Figure 3 illustrates the potential evolution of a look-up table across an application of AVI. In Figure 3a, dimensions t_k and b_k are each subdivided into two regions and the value of each time-budget combination is initialized. Figure 3b illustrates the lookup table mid-procedure, where the granularity is less coarse and the estimates of the expected rewards-to-go have been updated. Figure 3c depicts the final VFA, which we denote by $\widehat{V}_\tau(t_k, b_k)$, where we use the Greek letter τ to indicate "temporal." Dynamically identifying important time-budget combinations in this fashion allows the value iteration to focus limited computing resources on key areas of the lookup table, thereby yielding a better VFA.

Following the offline learning phase of the VFA, $\widehat{V}_\tau(t_k, b_k)$ can be used to execute a dynamic

routing scheme. When the process occupies state s_k, the temporal VFA decision rule is

$$X_k^{\pi_{\widehat{V}_\tau}}(s_k) = \underset{x \in \bar{\mathcal{X}}(s_k)}{\arg\max} \left\{ R(s_k, x) + \widehat{V}_\tau(t_k, b_k) \right\}. \tag{6}$$

Figure 2b depicts equation (6), illustrating the rule's consideration of each decision's period-k reward $R(s_k, x)$ plus $\widehat{V}_\tau(t_k, b_k)$, the estimate of the expected reward-to-go from the post-decision state. The VFA policy $\pi_{\widehat{V}_\tau}$ is the sequence of decision rules $(X_0^{\pi_{\widehat{V}_\tau}}, X_1^{\pi_{\widehat{V}_\tau}}, \ldots, X_K^{\pi_{\widehat{V}_\tau}})$. Thus, using only temporal aspects of the state variable, VFA $\widehat{V}_\tau(\cdot, \cdot)$ can be used to dynamically route a vehicle and assign customers via the policy $\pi_{\widehat{V}_\tau}$. For further details, we refer the reader to Ulmer et al. (2016).

4.2 Online Rollout Algorithm

Rollout algorithms, introduced by Bertsekas et al. (1997), aim to improve the performance of a base policy by using that policy in a current state to approximate the rewards-to-go from potential future states. Taking $\pi_{\widehat{V}_\tau}$ as the base policy, we use a post-decision rollout algorithm to approximate rewards-to-go from the post-decision state (Goodson et al., 2015). Because a rollout algorithm explicitly builds a portion of the decision tree, the resulting rollout policy is anticipatory by definition. Thus, as the computational experiments of §6 verify, a rollout algorithm built on policy $\pi_{\widehat{V}_\tau}$ may include more spatial information than $\pi_{\widehat{V}_\tau}$ in isolation. Further, building a rollout algorithm on the temporal VFA of Ulmer et al. (2016) combines offline VFAs' ability to detect overarching MDP structure with online rollout algorithms' capacity to identify detailed MDP structure local to small portions of the state space.

From a given post-decision state s_k^x, the rollout algorithm takes as the expected reward-to-go the value of policy $\pi_{\widehat{V}_\tau}$ from epoch k onward, $\mathbb{E}[\sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\widehat{V}_\tau}}(s_i))|s_k^x]$, a value we estimate via simulation. Let $\tilde{\mathcal{C}}^h = (\tilde{\mathcal{C}}_1^h, \tilde{\mathcal{C}}_2^h, \ldots, \tilde{\mathcal{C}}_K^h)$ be the sequence of service request realizations associated with the h^{th} simulation trajectory and let $V^{\pi_{\widehat{V}_\tau}}(s_k^x, h) = \sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\widehat{V}_\tau}}(s_i), \tilde{\mathcal{C}}_i^h)$ be the reward accrued by policy $\pi_{\widehat{V}_\tau}$ in periods $k+1$ through K when the process occupies post-decision state s_k^x and service requests are $\tilde{\mathcal{C}}^h$. Then, the expected reward of policy $\pi_{\widehat{V}_\tau}$ from state s_k^x onward, is estimated as the average value across H simulations: $V^{\pi_{\widehat{V}_\tau}}(s_k^x) = H^{-1} \sum_{h=1}^{H} V^{\pi_{\widehat{V}_\tau}}(s_k^x, h)$.

Figure 4 illustrates the online and offline aspects of the post-decision state estimate of the

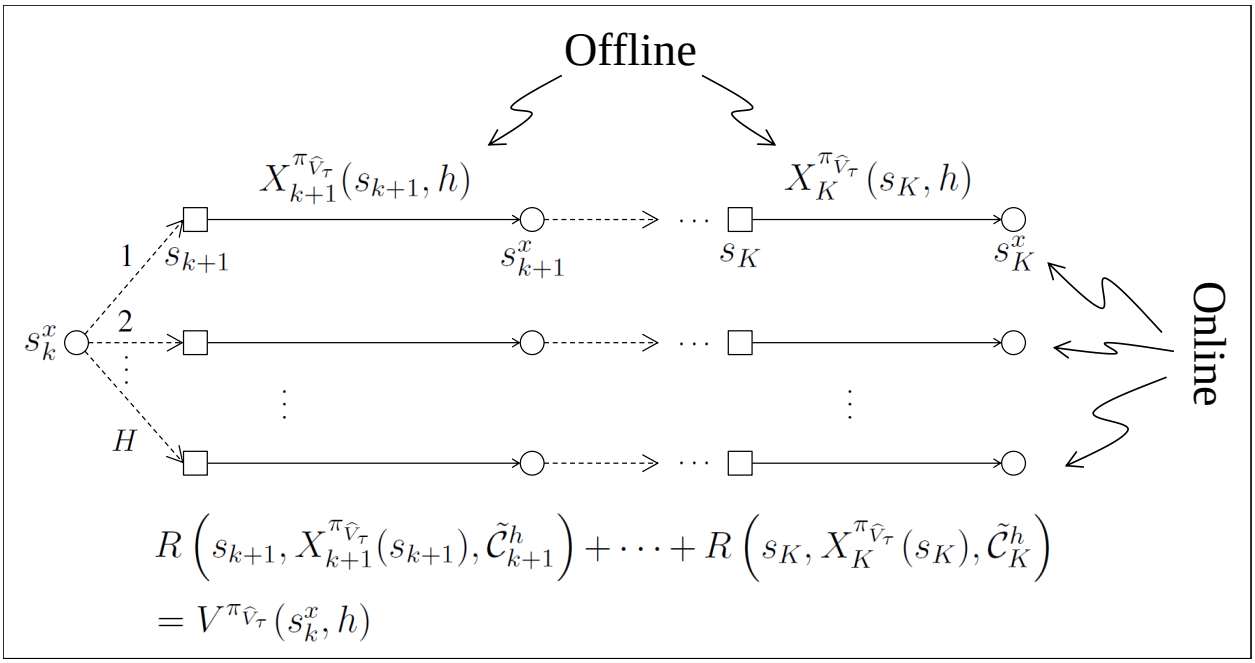


Figure 4: Offline-Online Post-Decision State Evaluation

expected reward-to-go calculation. The h^{th} sample path begins from state s_k^x, evolves via decisions from the offline VFA policy $\pi_{\widehat{V}_\tau}$ and the randomly generated service request realizations $\tilde{\mathcal{C}}^h$, and concludes at a terminal post-decision state. The rewards collected along the h^{th} sample path are summed to calculate $V^{\pi_{\widehat{V}_\tau}}(s_k^x,h)$ and the average across all simulations yields $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$.

Given post-decision estimates of the expected rewards-to-go, the rollout decision rule is

$$X_k^{\pi_{r\tau}}(s_k)=\underset{x\in\bar{\mathcal{X}}(s_k)}{\arg\max}\left\{R(s_k,x)+V^{\pi_{\widehat{V}_\tau}}(s_k^x)\right\}. \tag{7}$$

Figure 2c depicts equation (7), which is implemented in online fashion for realized states. Specifically, when the process occupies a current state s_k, a decision is selected by enumerating the feasible decision set $\bar{\mathcal{X}}(s_k)$. Then, for each decision x, $R(s_k,x)$ is calculated and a transition is made to post-decision state s_k^x where $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$ is computed via the method illustrated in Figure 4. A decision is selected that maximizes the sum of the current-period reward and the estimated reward-to-go. The rollout policy $\pi_{r\tau}$ is the sequence of decision rules $(X_0^{\pi_{r\tau}},X_1^{\pi_{r\tau}},\ldots,X_K^{\pi_{r\tau}})$. For further details on post-decision rollout, we refer the reader to Goodson et al. (2015).

4.3 VFA-Based Rollout Improvement

In addition to serving as a computationally tractable mechanism to incorporate spatial-temporal information into heuristic decision-making for the VRPSSR, our combination of VFAs and rollout algorithms points to the potential of using offline and online methods in tandem as a hybrid ADP procedure. With an eye toward offline-online ADP as a general method, we demonstrate that under a mild condition offline-online decision-making can yield rollout policy performance as good as or better than its base policy performance in expectation. For context, we continue to notate the offline base policy as $\pi_{\widehat{V}_\tau}$ and the online post-decision rollout policy as $\pi_{r\tau}$, but emphasize the discussion extends to more general base and rollout policies.

Our result draws on Goodson et al. (2015), who define a rollout policy $\pi_{r\tau}$ to be *rollout improving* with respect to base policy $\pi_{\widehat{V}_\tau}$ if $\mathbb{E}[\sum_{k=0}^{K} R(s_k, X_k^{\pi_{\widehat{V}_\tau}}(s_k))|s_0] \leq \mathbb{E}[\sum_{k=0}^{K} R(s_k, X_k^{\pi_{r\tau}}(s_k))|s_0]$, meaning the expected reward of the rollout policy is greater than or equal to the expected reward of the base policy. As Goodson et al. (2015) discuss, one way to achieve the rollout improvement property is via a *sequentially consistent* base policy, a policy that always makes the same decisions for a given sequence of states induced by the same sequence of stochastic realizations. The sequence of actions and realizations is called a *sample path*.

A sequentially consistent base policy can be characterized by the set of sample paths induced by policy $\pi_{\widehat{V}_\tau}$ from state s onward, i.e., the collection of trajectories through all possible realizations of service requests where actions are selected via policy $\pi_{\widehat{V}_\tau}$. For any state s' on any of these initial sample paths, consider a second set of sample paths induced by policy $\pi_{\widehat{V}_\tau}$ from state s' onward. If the initial set of sample paths, from s' on, is identical to the second set of sample paths, and if the equivalence holds for all s and for all possible s', then the base policy is said to be sequentially consistent. Goodson et al. (2015) demonstrate sequential consistency as a sufficient condition to achieve rollout improvement.

We define a VFA decision rule $X_k^{\pi_{\widehat{V}_\tau}}$ to be *deterministic* if it returns the same decision every time it is applied in the same state. Proposition 1 states that deterministic VFA decision rules lead to sequentially consistent VFA policies, thus by the results of Goodson et al. (2015) yielding rollout policies that weakly improve over the base policy.

Proposition 1 (Sequentially Consistent VFA Policy)**.** *If VFA decision rule $X_k^{\pi_{\widehat{V}_\tau}}$ is deterministic, then VFA policy $\pi_{\widehat{V}_\tau}$ is sequentially consistent.*

Proof. Let s be a state in the state space and let s' be a state such that it is on a sample path induced by VFA policy $\pi_{\widehat{V}_\tau}$ applied in state s. Let k' index the decision epoch associated with state s'. By the assumption that $X_k^{\pi_{\widehat{V}_\tau}}$ is deterministic for all k, the sequence of decision rules $(X_{k'}^{\pi_{\widehat{V}_\tau}}, X_{k'+1}^{\pi_{\widehat{V}_\tau}}, \ldots, X_K^{\pi_{\widehat{V}_\tau}})$ from epoch k onward is always the same. Because the argument holds for all s and s', the VFA policy $\pi_{\widehat{V}_\tau}$ satisfies Definition 10 of Goodson et al. (2015) and is sequentially consistent. □

With some care, VFA decision rule $X_k^{\pi_{\widehat{V}_\tau}}$ can be made deterministic. Provided VFA $\widehat{V}_\tau(t_k, b_k)$ always returns the same value for a given time t_k and budget b_k, and provided ties in decisions achieving the maximum value in equation (6) are broken the same way each time the decision rule is applied to the same state, then $X_k^{\pi_{\widehat{V}_\tau}}$ is deterministic. Because the guaranteed improvement of the rollout policy over the VFA base policy depends on the exact calculation of the base policy's expected reward-to-go (Goodson et al., 2015), we anticipate the benefits of sequentially consistent VFA policies to become more pronounced as the number of simulations H increases, thus making $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$ a more accurate estimate of $\mathbb{E}[\sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\widehat{V}_\tau}}(s_i))|s_k^x]$. In the computational experiments of §6, $H = 16$ simulations is sufficient to observe improvement of $\pi_{r\tau}$ over $\pi_{\widehat{V}_\tau}$.

Beyond the rollout improvement resulting from a deterministic VFA decision rule, the results of Goodson et al. (2015) imply that post-decision and one-step rollout policies built on a VFA policy with deterministic decision rules yield the same value. In contrast to the decision rule of equation (7), which approximates expected rewards-to-go via policy $\pi_{\widehat{V}_\tau}$ applied from post-decision states, a one-step decision rule applies policy $\pi_{\widehat{V}_\tau}$ in all possible states at the subsequent decision epoch. Because customer locations and service request times may follow a continuous probability distribution, from a given post-decision state, the number of positive-probability states in the subsequent decision epoch may be infinite, thereby rendering a one-step rollout algorithm computationally intractable. Thus, when VFA policy $\pi_{\widehat{V}_\tau}$ is sequentially consistent, the value of post-decision rollout policy $\pi_{r\tau}$ behaves as if its decision rules were able to look a full step ahead in the MDP, rather than looking ahead only to the post-decision state. Similar to the rollout improvement property, equivalence of post-decision and one-step rollout policies depends on the

exact calculation of the base policy's expected reward-to-go, thus the result is more likely to be observed as the number of simulations H is increased.

Finally, although stated in notation specific to the VRPSSR and VFAs, we emphasize Proposition 1 is broadly applicable: any base policy composed of deterministic decision rules is sequentially consistent. Thus, any policy composed of deterministic functions mapping states to the space of feasible decisions is sequentially consistent and achieves the rollout improvement properties of Goodson et al. (2015). In addition to VFAs, such functions might take the form of a math program where stochastic quantities are replaced with their mean values, a local search on a priori policies, or a threshold-style rule based on key parameters of the state variable. We believe viewing policy construction in this way – as a concatenation of decision rules – may simplify the task of identifying feasible MDP policies. Further, although, as Goodson et al. (2015) illustrate, non-sequentially consistent heuristics do not necessarily yield poor performance, making the effort to construct policies via deterministic decision rules provides immediate access to much of the analysis laid out by Goodson et al. (2015).

5 Benchmark ADP Heuristic

To gauge the performance of the offline-online method, we consider the performance of four additional policies: a spatial VFA, a myopic policy, and the incorporation of both into post-decision rollout algorithms. We describe these methods in §5.1 and §5.2, respectively.

5.1 Spatial VFA

Although the temporal VFA of Ulmer et al. (2016) yields computationally-tractable, high-quality policies, its reward-to-go approximation does not utilize spatial information. Explicit consideration of service area coverage may be important, for example, when budget b_k is low. In this scenario, the VFA of Ulmer et al. (2016) may assign a low value to the approximate reward-to-go. However, the true value may depend on the portion of the geographic area covered by the sequence of assigned customers $\bar{\mathcal{C}}_k^x$. Depending on the likelihood of service requests across time and space, a routing of the assigned customers spread out across a large area versus confined to a narrow geographic zone may be more able to accommodate additional customer calls and therefore be more valuable.

The spatial VFA we propose in this section explicitly considers service area coverage via online routing and simulation. Interestingly, preliminary testing of offline ADP methods based on a variety of spatial features did not provide competitive results. Though the literature (Branke et al., 2005; Meisel, 2011) points to offline spatial VFAs, these methods are applied to problem instances with only a few customers and do not scale well to the larger problem instances we consider.

Influenced by the work of Bent and Van Hentenryck (2004), our spatial VFA approximates the post-decision state reward-to-go of equation (5) via simulation of service calls and heuristic routing of those requests. Let $\tilde{\mathcal{C}}^p = (\tilde{\mathcal{C}}^p_1, \tilde{\mathcal{C}}^p_2, \ldots, \tilde{\mathcal{C}}^p_K)$ be the sequence of service request realizations associated with the p^{th} simulation trajectory and let $\tilde{\mathcal{C}}^p(k) = \bigcup_{i=k}^{K} \tilde{\mathcal{C}}^p_i$ be the union of the service requests in periods k through K. From a post-decision state s^x_k, we use cheapest insertion (Rosenkrantz et al., 1974) to construct a route beginning at location c_k at time t_k, through the set of assigned customers $\bar{\mathcal{C}}^x_k$, and through as many service requests as possible in set $\tilde{\mathcal{C}}^p(k+1)$ such that the vehicle returns to the depot no later than time T. The routing procedure assumes requests in $\tilde{\mathcal{C}}^p(k+1)$ may be serviced during any period, thus ignoring the times at which services are requested and constructing a customer sequence based solely on spatial information. Letting Q^p be the number of requests in $\tilde{\mathcal{C}}^p(k+1)$ successfully routed in sample p, the spatial VFA is $\widehat{V}_\sigma(c_k, \bar{\mathcal{C}}^x_k) = P^{-1} \sum_{p=1}^{P} Q^p$, where we use the Greek letter σ to indicate "spatial."

Figure 5 illustrates the process of simulation and routing. The center portion of Figure 5 depicts the set of requests $\tilde{\mathcal{C}}^p(k+1)$ associated with the p^{th} simulation as well as a route from the vehicle's current location c_k at time t_k = 02:00 hours after the start of work, through the ordered set $\bar{\mathcal{C}}^x_k$, and ending at the depot no later than 6:00 hours after the start of work. Similar to Figure 1, in this example the vehicle traverses a Manhattan-style grid where each unit requires 00:15 hours of travel time. The right-most portion of Figure 5 shows the cheapest insertion routing of the requests in $\tilde{\mathcal{C}}^p(k+1)$. In this example, three of the four requests comprising $\tilde{\mathcal{C}}^p(k+1)$ are successfully routed, thus $Q^p = 3$. Repeating this process across all P simulations, then averaging the results, yields $\widehat{V}_\sigma(c_k, \bar{\mathcal{C}}^x_k)$.

In contrast to offline temporal VFA $\widehat{V}_\tau(\cdot,\cdot)$, spatial VFA $\widehat{V}_\sigma(\cdot,\cdot)$ is executed online. To identify a dynamic routing plan, sample requests are generated as needed and reward-to-go estimates are calculated only for post-decision states reachable from a realized current state s_k. When the process occupies state s_k, the spatial VFA decision rule is

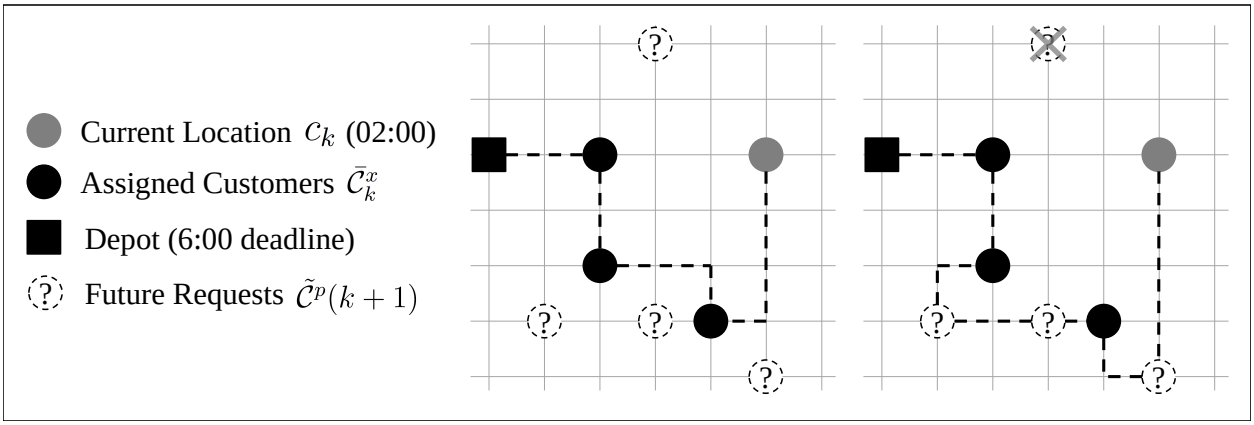


Figure 5: Spatial Value Function Approximation

$$X_k^{\pi_{\widehat{V}_\sigma}}(s_k) = \underset{x \in \bar{\mathcal{X}}(s_k)}{\arg\max} \left\{ R(s_k, x) + \widehat{V}_\sigma(c_k, \bar{\mathcal{C}}_k^x) \right\}. \tag{8}$$

The VFA policy $\pi_{\widehat{V}_\sigma}$ is the sequence of decision rules $(X_0^{\pi_{\widehat{V}_\sigma}}, X_1^{\pi_{\widehat{V}_\sigma}}, \ldots, X_K^{\pi_{\widehat{V}_\sigma}})$.

Similar to equation (7), the rollout decision rule based on the spatial VFA is

$$X_k^{\pi_{r\sigma}}(s_k) = \underset{x \in \bar{\mathcal{X}}(s_k)}{\arg\max} \left\{ R(s_k, x) + V^{\pi_{\widehat{V}_\sigma}}(s_k^x) \right\}, \tag{9}$$

where the post-decision estimate of the reward-to-go is computed as in the method of Figure 4, except decisions on each sample path are made via policy $\pi_{\widehat{V}_\sigma}$ instead of via the temporal VFA policy. The rollout policy $\pi_{r\sigma}$ is the sequence of decision rules $(X_0^{\pi_{r\sigma}}, X_1^{\pi_{r\sigma}}, \ldots, X_K^{\pi_{r\sigma}})$.

5.2 Myopic

We consider a myopic policy to gauge the other policies on their potential for anticipatory decision-making. The decision rule associated with myopic policy π_m selects a decision that maximizes the current-period reward: $X_k^{\pi_m} = \arg\max_{x \in \bar{\mathcal{X}}(s_k)} \{R(s_k, x)\}$. The decision rule associated with myopic-based rollout policy π_{rm} is analogous to equation (7), with the second term estimating the reward-to-go of myopic policy π_m via H simulations. Myopic policies often yield poor performance because they ignore the future impact of current-period decisions. Though, as we show in the computational experiments of §6, π_{rm} improves significantly over π_m, giving a more competitive benchmark for the offline-online ADP heuristic.

6 Computational Experiments

We outline problem instances in §6.1 followed by a discussion of computational results in §6.2. All methods are coded in Java and executed on 2.4GHz AMD Opteron dual-core processors with 8GB of RAM.

6.1 Problem Instances

We develop a collection of problem instances by varying the size of the service region, the ratio of early-request to late-request customers, and the locations of requests. We consider two service regions $\mathcal{C}$, a *large* 20-kilometer-square region and a *medium* 15-kilometer-square region, each with a centrally-located depot.

We treat the number and location of customer requests as independent random variables. Setting the time horizon T to 360 minutes, the number of late-request customers in the range $[1, T]$ follows a Poisson process with parameter λ customers per $T - 1$ minutes. Thus, the number of customers in set $\tilde{\mathcal{C}}_{k+1}$ requesting service in the range $(t_k, t_{k+1}]$ is Poisson-distributed with parameter $\lambda(t_{k+1} - t_k)/(T - 1)$. We consider three values for λ: 25, 50, and 75, which we refer to as *low*, *moderate*, and *high*, respectively. The number of early-request customers in set $\mathcal{C}_{\text{early}}$, observed prior to the start of the time horizon, is Poisson-distributed with rate $100 - \lambda$. Hence, in each problem instance the total number of early- plus late-request customers is Poisson-distributed with parameter 100. Though we consider only a single work period, rejected requests may be included in the early-request customers the following period. The performance of our algorithm on these instances permits such a policy, given the number of rejected requests is typically no larger than the number of early-request customers.

We consider three distributions for customer locations: uniformly distributed across the service area, clustered in two groups, and clustered in three groups. When customers are clustered, service requests are normally distributed around cluster centers with a standard deviation of one kilometer. Taking the lower-left corner of the square service region to be the origin, then for *large* service areas and two clusters, centers are located at coordinates $(5, 5)$ and $(5, 15)$ with units set to kilometers. For *large* service areas and three clusters, centers are located at coordinates $(5, 5)$, $(5, 15)$, and $(15, 10)$. For *medium* service regions, cluster centers and standard deviations are scaled by 0.75.

When requests are grouped in two clusters, customers are equally likely to appear in either cluster. When requests are grouped in three clusters, customers are twice as likely to appear in the second cluster as they are to appear in either the first or third cluster. Although requests outside of the service area are unlikely, such customers are included in realizations of problem instances.

For all problem instances, the travel time $d(\cdot, \cdot)$ between two locations is the Euclidean distance divided by a constant speed of 25 kilometers per hour and rounded up to the nearest whole minute. We set the base time unit $\bar{t}$ to one minute.

All combinations of service area, proportions of early- and late-request customers, and spatial distributions yields a total of 18 problem instances. For each problem instance, we generate 250 realizations of early- and late-request customers and use these realizations to estimate the expected rewards achieved by various policies.

6.2 Discussion

In this section we examine the performance of the offline-online ADP heuristic by comparing policy $\pi_{r\tau}$ to temporal policy $\pi_{\widehat{V}_\tau}$, spatial policy $\pi_{\widehat{V}_\sigma}$, myopic policy π_m, and to rollout policies $\pi_{r\sigma}$, and π_{rm} across the problem instances of §6.1.

Table 2 presents estimates of the expected number of late-request customers serviced by each policy across medium and large service areas; low, moderate, and high values of λ; and uniform, two-cluster, and three-cluster customer locations. Values in parantheses are the standard errors. Each figure is an average of the reward collected across the 250 realizations of the corresponding problem instance. The offline VFA associated with policies $\pi_{\widehat{V}_\tau}$ and $\pi_{r\tau}$ is obtained via 1,000,000 iterations of AVI and a disaggregation threshold of 1.5 (Ulmer et al., 2016). We use $P = 16$ simulations to calculate reward-to-go estimate $\widehat{V}_\sigma(\cdot, \cdot)$ for policy $\pi_{\widehat{V}_\sigma}$. For rollout policies π_{rm} and $\pi_{r\tau}$ we use $H = 16$ simulations to calculate reward-to-go estimate $V^{\pi_{\widehat{V}}}$. Increasing P and H beyond these values leads to higher computation times with relatively little gain in reward. For rollout policy $\pi_{r\sigma}$ we use $P = 4$ simulations to calculate $\widehat{V}_\sigma(\cdot, \cdot)$ and $H = 16$ simulations to calculate $V^{\pi_{\widehat{V}_\sigma}}$. Even at these values of P and H, the computation time required to execute policy $\pi_{r\sigma}$ across all realizations of each problem instance is high, pushing the capacity of our resources. Additionally, each of the P sample trajectories is drawn randomly whenever $\widehat{V}_\sigma(\cdot, \cdot)$ is calculated, thus $\pi_{\widehat{V}_\sigma}$ is not a sequentially consistent base policy.

Table 2: Expected Numbers of Late-Request Customers Served and Standard Errors

	Medium Service Area			Large Service Area		
Policy	**Low λ**	**Moderate λ**	**High λ**	**Low λ**	**Moderate λ**	**High λ**
			Customers Located Uniformly			
π_m	11.8 ($\pm$0.3)	25.3 ($\pm$0.3)	40.5 ($\pm$0.3)	0.2 ($\pm$0.0)	8.1 ($\pm$0.4)	27.7 ($\pm$0.3)
$\pi_{\widehat{V}_\tau}$	12.9 ($\pm$0.3)	29.0 ($\pm$0.3)	45.0 ($\pm$0.3)	0.2 ($\pm$0.0)	11.3 ($\pm$0.5)	34.8 ($\pm$0.3)
$\pi_{\widehat{V}_\sigma}$	12.6 ($\pm$0.3)	28.5 ($\pm$0.3)	45.1 ($\pm$0.3)	0.2 ($\pm$0.1)	10.7 ($\pm$0.5)	33.3 ($\pm$0.3)
π_{rm}	12.7 ($\pm$0.3)	28.2 ($\pm$0.3)	44.2 ($\pm$0.3)	0.2 ($\pm$0.1)	10.6 ($\pm$0.5)	32.9 ($\pm$0.3)
$\pi_{r\tau}$	12.9 ($\pm$0.3)	29.2 ($\pm$0.3)	45.7 ($\pm$0.3)	0.2 ($\pm$0.1)	11.3 ($\pm$0.5)	34.8 ($\pm$0.3)
$\pi_{r\sigma}$	12.8 ($\pm$0.3)	28.6 ($\pm$0.3)	44.9 ($\pm$0.3)	0.2 ($\pm$0.1)	11.1 ($\pm$0.5)	34.0 ($\pm$0.3)
			Customers Located in Two Clusters			
π_m	21.8 ($\pm$0.3)	41.6 ($\pm$0.4)	57.4 ($\pm$0.4)	16.6 ($\pm$0.3)	31.4 ($\pm$0.3)	47.7 ($\pm$0.4)
$\pi_{\widehat{V}_\tau}$	21.8 ($\pm$0.3)	41.3 ($\pm$0.4)	57.1 ($\pm$0.4)	16.7 ($\pm$0.3)	32.5 ($\pm$0.3)	50.1 ($\pm$0.4)
$\pi_{\widehat{V}_\sigma}$	21.7 ($\pm$0.2)	42.0 ($\pm$0.4)	58.4 ($\pm$0.4)	17.0 ($\pm$0.3)	33.4 ($\pm$0.3)	50.8 ($\pm$0.3)
π_{rm}	21.8 ($\pm$0.3)	41.9 ($\pm$0.4)	57.8 ($\pm$0.4)	17.1 ($\pm$0.3)	33.6 ($\pm$0.3)	50.5 ($\pm$0.4)
$\pi_{r\tau}$	21.8 ($\pm$0.3)	42.3 ($\pm$0.4)	57.8 ($\pm$0.4)	17.1 ($\pm$0.3)	33.8 ($\pm$0.3)	51.7 ($\pm$0.4)
$\pi_{r\sigma}$	21.8 ($\pm$0.2)	42.1 ($\pm$0.3)	58.4 ($\pm$0.4)	17.1 ($\pm$0.3)	33.8 ($\pm$0.3)	51.3 ($\pm$0.4)
			Customers Located in Three Clusters			
π_m	20.5 ($\pm$0.2)	37.7 ($\pm$0.3)	54.1 ($\pm$0.4)	12.3 ($\pm$0.3)	27.6 ($\pm$0.4)	42.2 ($\pm$0.3)
$\pi_{\widehat{V}_\tau}$	20.3 ($\pm$0.2)	37.2 ($\pm$0.3)	53.8 ($\pm$0.4)	13.4 ($\pm$0.3)	29.8 ($\pm$0.3)	45.1 ($\pm$0.3)
$\pi_{\widehat{V}_\sigma}$	20.5 ($\pm$0.2)	38.4 ($\pm$0.3)	55.6 ($\pm$0.4)	13.4 ($\pm$0.3)	29.8 ($\pm$0.3)	45.7 ($\pm$0.3)
π_{rm}	20.8 ($\pm$0.2)	38.8 ($\pm$0.3)	55.0 ($\pm$0.4)	13.5 ($\pm$0.3)	29.9 ($\pm$0.3)	46.2 ($\pm$0.3)
$\pi_{r\tau}$	20.8 ($\pm$0.2)	38.9 ($\pm$0.3)	55.3 ($\pm$0.4)	13.6 ($\pm$0.3)	30.7 ($\pm$0.3)	47.0 ($\pm$0.3)
$\pi_{r\sigma}$	20.7 ($\pm$0.2)	38.8 ($\pm$0.3)	55.7 ($\pm$0.4)	13.5 ($\pm$0.3)	30.5 ($\pm$0.3)	46.9 ($\pm$0.3)

Rollout Improvement

Grouped by customer location, Figure 6 aggregates over quantities in Table 2 to display the percent improvement of policies π_{rm}, $\pi_{\widehat{V}_\tau}$, $\pi_{r\tau}$, $\pi_{\widehat{V}_\sigma}$, and $\pi_{r\sigma}$ over myopic policy π_m. Each bar in Figure 6 depicts the improvement of a base policy (solid outline) over π_m and any additional improvement achieved by the corresponding rollout policy (dashed line).

Figure 6 demonstrates each rollout policy performs at least as well as its corresponding base policy, a result predicted by Proposition 1 for policies $\pi_{r\tau}$ and π_{rm}. Further, with only one exception, the disaggregate results of Table 2 indicate policy $\pi_{r\sigma}$ improves upon non-sequentially

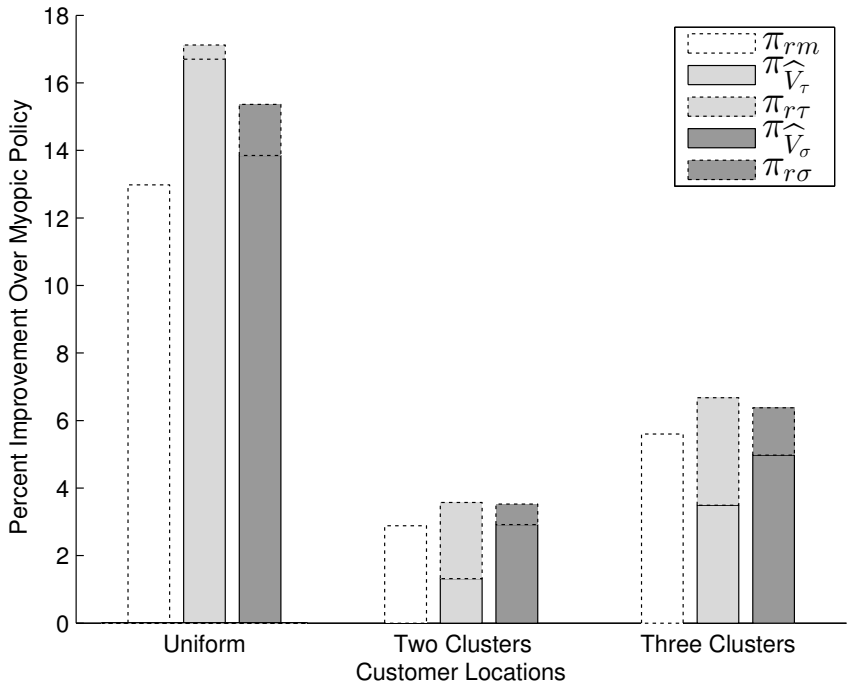


Figure 6: Improvement Over Myopic Policy

consistent base policy $\pi_{\widehat{V}_\sigma}$. When customers are located uniformly across the medium service area and λ is high, spatial policy $\pi_{\widehat{V}_\sigma}$ achieves a reward 0.4 percent higher than that posted by rollout policy $\pi_{r\sigma}$, a discrepancy we expect would be remedied by increasing the number of simulations H. Further, although π_{rm} yields substantial improvement over π_m (7.2 percent on average), we observe higher expected rewards when the rollout algorithm is applied to base policies $\pi_{\widehat{V}_\tau}$ (9.1 percent on average) and $\pi_{\widehat{V}_\sigma}$ (8.4 percent on average), each of which post performance superior to that of the myopic policy. For a VRP with stochastic demand, Novoa and Storer (2008) similarly observe that better base policies yield better rollout policies.

Figure 6 indicates improvement of rollout policy π_{rm} over myopic policy π_m is most pronounced when customer locations are uniform over the service area. When requests are spread randomly across the region, the high variability of customer locations can cause the greedy decision rule of policy π_m to perform poorly, at times assigning requests separated by large distances without considering the future impact of such decisions. As customer locations become more concentrated – from uniform to three clusters to two clusters – the likelihood of such short-sighted decisions decreases, thus lessening the improvement achieved by the rollout algorithm's look-ahead

mechanism.

Figure 6 shows improvement of spatial-temporal rollout policy $\pi_{r\tau}$ over temporal base policy $\pi_{\widehat{V}_\tau}$ is most significant when customer locations are clustered. As additional experiments reveal below, spatial anticipation is more important than temporal anticipation when requests are grouped. Thus, as customer locations become more concentrated, the post-decision look-ahead of policy $\pi_{r\tau}$ has more opportunity to make up for the spatial anticipation absent in policy $\pi_{\widehat{V}_\tau}$. The standard errors in Table 2 support this observation. The difference between the values of policies $\pi_{r\tau}$ and $\pi_{\widehat{V}_\tau}$ is almost always significant when customers are clustered and is insignificant when customers are uniformly distributed.

Similarly, Figure 6 depicts improvement of spatial-temporal rollout policy $\pi_{r\sigma}$ over spatial base policy $\pi_{\widehat{V}_\sigma}$ as being more substantial when customer locations are less concentrated, i.e., in three clusters or uniform versus in two clusters. As we further explore below, we believe policy $\pi_{r\sigma}$ adds temporal anticipation to policy $\pi_{\widehat{V}_\sigma}$, thus enhancing the spatial-only anticipation of the base policy. However, as we discuss below, the high run times required to execute policy $\pi_{r\sigma}$ may limit its practical use.

An important takeaway from Figure 6 is the ability of the post-decision rollout algorithm to compensate for anticipation absent in the base policy. For example, as noted above, rollout policy $\pi_{r\tau}$ adds spatial anticipation to temporal base policy $\pi_{\widehat{V}_\tau}$. In particular, when customers are clustered in two or three groups, the additional anticipation results in comparable performance to rollout policy $\pi_{r\sigma}$, thus suggesting similar levels of spatial-temporal anticipation may be achieved by combining with a rollout algorithm either a temporal or spatial base policy. In contrast, when customers are located uniformly across the service area, rollout policy $\pi_{r\sigma}$ is unable to match the performance of temporal policy $\pi_{\widehat{V}_\tau}$, much less that of rollout policy $\pi_{r\tau}$. These results, taken in conjunction with the computational discussion below, point to rollout policy $\pi_{r\tau}$ as the frontrunner among the six policies we consider.

The high performance of policy $\pi_{r\tau}$ may also be attributed to the combination of offline and online ADP methods. The low-dimensional temporal VFA captures the overarching structure of the MDP and the rollout algorithm observes MDP structure in full detail across medium portions of the state space. Taken together, offline plus online methods allow policy $\pi_{r\tau}$ to merge global structure with local detail. In contrast, the spatial VFA underlying policy $\pi_{r\sigma}$ is an online ADP

technique relying on a relatively small number of real-time simulations to approximate rewards-to-go. Consequently, spatial VFA $\widehat{V}_\sigma(\cdot,\cdot)$ may be unable to detect the overall patterns observed by temporal VFA $\widehat{V}_\tau(\cdot,\cdot)$, potentially leading to lower expected rewards for policy $\pi_{r\sigma}$.

Decreasing Computation via Offline-Online Tradeoffs

Organized similar to Table 2, Table 3 displays the average of the maximum CPU seconds required to select a decision across all 250 realizations of the corresponding problem instance. We report the average maximum CPU seconds (versus the overall average) to highlight the worst-case time required to implement each policy in real time. For policies $\pi_{\widehat{V}_\tau}$ and $\pi_{r\tau}$, figures exclude offline VFA computation.

Across all policies, for a given service region and customer location distribution, CPU requirements tend to increase by an order of magnitude as λ moves from low to moderate and then again as λ moves from moderate to high. These increases in computing time are driven by an increase in the number of feasible decisions in the set $\bar{\mathcal{X}}(\cdot)$, which tends to grow with larger numbers of late-request customers. The highest CPU times belong to rollout policy $\pi_{r\sigma}$. At a given decision epoch, similar to rollout policies π_{rm} and $\pi_{r\tau}$, policy $\pi_{r\sigma}$ uses H simulations to estimate the expected reward-to-go from a given post-decision state. Additionally, along each of the H trajectories, base policy $\pi_{\widehat{V}_\sigma}$ employs P simulations to select a decision at each epoch. Thus, despite its high expected reward, policy $\pi_{r\sigma}$ may be impractical for real-time decision making. Even rollout policy $\pi_{r\tau}$, which performs comparably to policy $\pi_{r\sigma}$ in the vast majority of Table 2 entries, may be of limited practical use when λ is high. Below, we demonstrate how IZS can lower the computational requirements of online decision-making, thereby making policy $\pi_{r\tau}$ viable for real-time routing and assignment decisions.

Seeking a reduction in the CPU requirements for rollout policy $\pi_{r\tau}$, we consider the combined impact of offline and online computation on expected reward. In Table 4, we vary the number of offline AVI iterations from zero (representing the myopic policy) up to 5,000,000 and the number of online simulations H from two up to 128, including as a benchmark the performance of base policy $\pi_{\widehat{V}_\tau}$. Each entry in Table 4 is the average reward achieved across 250 realizations of the problem instance characterized by a large service area, customers grouped in two clusters, and high λ. Darker shades indicate higher expected rewards.

Table 3: Maximum CPU Seconds to Select a Decision

	Medium Service Area			Large Service Area		
Policy	**Low λ**	**Moderate λ**	**High λ**	**Low λ**	**Moderate λ**	**High λ**
			Customers Located Uniformly			
π_m	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\pi_{\widehat{V}_\tau}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\pi_{\widehat{V}_\sigma}$	1.8	16.4	129.3	< 0.01	9.6	125.7
π_{rm}	0.2	1.9	39.8	< 0.01	2.0	85.3
$\pi_{r\tau}$	0.6	3.7	46.7	< 0.01	5.3	113.5
$\pi_{r\sigma}$	633.3	2862.1	22598.2	13.9	1592.6	39013.2
			Customers Located in Two Clusters			
π_m	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.1
$\pi_{\widehat{V}_\tau}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.1
$\pi_{\widehat{V}_\sigma}$	1.8	13.8	88.7	2.0	17.6	175.6
π_{rm}	0.2	2.4	50.9	0.3	5.2	410.5
$\pi_{r\tau}$	0.4	3.7	69.6	0.6	6.0	247.1
$\pi_{r\sigma}$	786.7	3368.2	23761.6	774.5	3644.2	74884.9
			Customers Located in Three Clusters			
π_m	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\pi_{\widehat{V}_\tau}$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\pi_{\widehat{V}_\sigma}$	1.9	12.3	83.2	1.9	19.0	174.2
π_{rm}	0.2	1.4	14.5	0.2	2.7	84.8
$\pi_{r\tau}$	0.5	2.3	29.1	0.6	4.4	96.4
$\pi_{r\sigma}$	759.5	2942.8	21243.9	663.4	3917.4	47749.1

Table 4 illustrates the potential benefit of using offline VFA and online rollout algorithms in tandem as a hybrid ADP procedure. The lower-left and upper-right entries in the body of Table 4 represent pure offline and pure online policies, respectively, the rollout policy with $H = 128$ simulations yielding a 3.8 percent improvement over the temporal VFA policy with 5,000,000 AVI iterations. Complementing offline computation with online computation and vice versa eventually leads to improved rewards, the highest of which is achieved in the lower-right entry of Table 4 with an expected reward of 52.7. This improved reward comes with a cost, however: $H = 128$ online simulations combined with 5,000,000 offline AVI iterations may require as many as 1864 CPU seconds at a given epoch, an impractical figure for real-time decision-making.

Table 4: Impact of Offline Computation on Online Performance

		Online Simulations (H)						
Offline AVI Iterations	$\pi_{\widehat{V}_\tau}$	2	4	8	16	32	64	128
0	47.7	46.0	47.2	49.0	50.5	51.3	51.7	52.0
1,000	43.0	44.2	45.5	47.3	49.0	50.4	51.6	52.0
10,000	46.2	45.3	46.9	48.6	50.2	51.5	51.7	52.3
100,000	49.4	47.4	48.6	50.0	51.0	51.6	52.2	52.3
1,000,000	50.1	48.2	49.4	50.5	51.7	52.3	52.5	52.5
5,000,000	50.1	48.5	49.5	50.7	51.7	52.2	52.3	52.7

Moving away from the extreme entries of Table 4 reveals how offline computation can compensate for reduced online computation. For instance, when $H = 64$ simulations are used in conjunction with 0 offline AVI iterations, rollout policy $\pi_{r\tau}$ yields an expected reward of 51.7. A comparable reward is achieved with $H = 16$ online simulations and 1,000,000 offline AVI iterations. Further, shifting computational effort offline reduces the maximum per epoch online CPU seconds from 1521 to 295, likely a manageable figure for real-time decision-making. Thus, when time to make decisions is limited, increasing offline computation can make up for necessary decreases to online computation.

Decreasing Computation via Indifference Zone Selection

In addition to shifting computation from online to offline, online CPU time may be further reduced via IZS. Developed by Kim and Nelson (2001, 2006), IZS may be employed to reduce the computation required to identify, from a given state s_k, the decision in $\bar{\mathcal{X}}(s_k)$ leading to the largest expected reward-to-go. In particular, in equation (7), IZS may require fewer than H simulations to calculate $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$.

IZS is executed in three phases. In the first phase, for all decisions x in $\bar{\mathcal{X}}(s_k)$, $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$ is initialized via n_{initial} simulations. The second phase identifies, with confidence level $1-\alpha$, the reward-to-go estimates falling within δ (the indifference zone) of the maximum. The third phase discards all $V^{\pi_{\widehat{V}_\tau}}(s_k^x)$ not meeting the phase-two threshold and refines the remaining reward-to-go

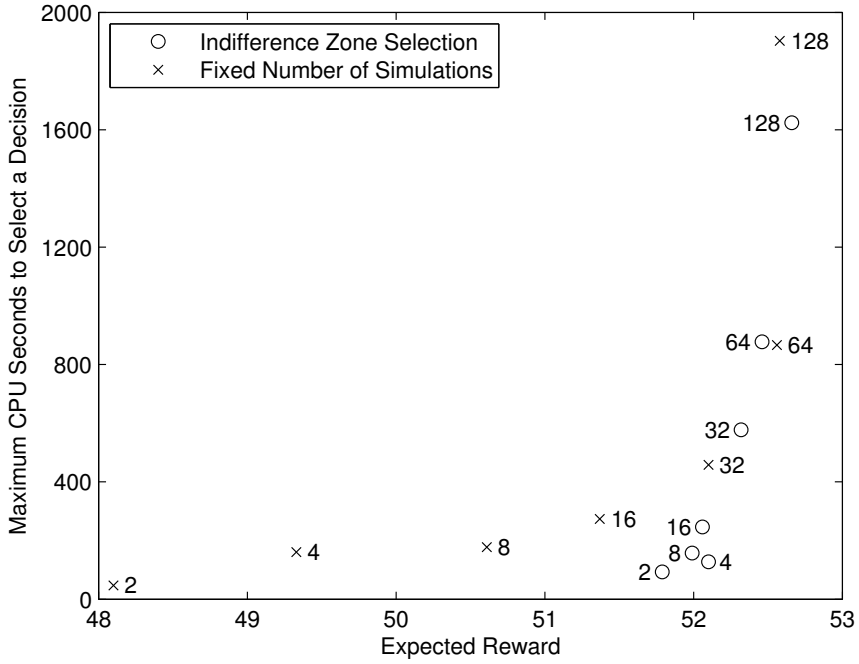


Figure 7: Impact of Indifference Zone Selection on Rewards and CPU Times

estimates via an additional simulation. IZS iterates between phases two and three until only one reward-to-go estimate remains – in which case the procedure returns the corresponding decision – or until the total number of simulations reaches $n_{\max}$ – in which case the procedure returns the decision with the highest reward-to-go estimate. Setting parameter $n_{\max}$ to H ensures at most H simulations (and potentially many fewer) are employed to estimate the reward-to-go from each post-decision state.

To illustrate the potential benefits of IZS, we apply the procedure via rollout policy $\pi_{r\tau}$ to the problem instance with the highest computation times, the instance characterized by a large service region, customer locations grouped in two clusters, and high λ. We set the indifference zone to $\delta = 1$, the confidence parameter to $\alpha = 0.01$, and the maximum number of simulations to $n_{\max} = 128$. Figure 7 displays the impact of IZS on CPU times and on rewards as the number of initial simulations n_{initial} takes on values 2, 4, 8, 16, 32, 64, and 128. As a benchmark to the IZS procedure, we include in Figure 7 the results of fixing to H the number of simulations employed to calculate $V^{\pi_{\hat{V}_\tau}}(\cdot)$. The value of n_{initial} or H is displayed adjacent each point in Figure 7.

Figure 7 suggests IZS can achieve rewards better than or comparable to a fixed-simulation implementation, but with potentially lower CPU times. Notably, setting the number of fixed simu-

lations to $H = 128$ yields an expected reward of 52.58 and a maximum CPU time of 1903 seconds. In contrast, IZS with the number of initial simulations set to $n_{\text{initial}} = 4$ achieves an expected reward of 52.10 and a maximum CPU time of 127 seconds. Thus, a 93.3 percent reduction in CPU time can be achieved with only a 0.9 percent decrease in reward. Further, as n_{initial} increases to 32 and beyond, IZS tends to terminate with n_{initial} total iterations, thus yielding rewards similar to those of the fixed-simulation implementation. Consequently, if per-epoch CPU time is prohibitive when the number of simulations is fixed, the results of Figure 7 suggest IZS with $n_{\text{initial}} < H$ may significantly reduce computation with only marginal detriment to policy quality.

We note the randomly sampled trajectories vary from one simulation to the next, thus decisions taken by the same policy may vary from one realization to another. Consequently, even when n_{initial} and H are set to equivalent large numbers, results may differ slightly.

Temporal vs. Spatial Anticipation

Finally, we examine the impact of spatial and temporal information on anticipation. In particular, we compare the performance of temporal policy $\pi_{\widehat{V}_\tau}$ to that of spatial policy $\pi_{\widehat{V}_\sigma}$. Per Table 2, when customer locations are uniform over the service area, the expected number of late-request customers serviced by policy $\pi_{\widehat{V}_\tau}$ is almost always greater than or equal to the expected reward accrued by policy $\pi_{\widehat{V}_\sigma}$, thus suggesting current time t_k and time budget b_k are better predictors of the reward-to-go than explicit information about customer locations and service area coverage. In contrast, when customer locations are grouped in two or three clusters, policy $\pi_{\widehat{V}_\sigma}$ almost always outperforms policy $\pi_{\widehat{V}_\tau}$, indicating current location c_k and the tour through assigned customers $\bar{\mathcal{C}}_k$ trump temporal considerations when approximating the value function.

To further investigate the impact of customer locations on the performance of temporal and spatial policies, we construct a set of problem instances varying the proportion of customers located in clusters and the proportion of customers uniformly distributed across the service area. Specifically, given a large service area and high λ, γ percent of customers are drawn from the two-cluster location distribution and $100 - \gamma$ percent of the customers are drawn from the uniform location distribution. Varying γ from zero to 100 by increments of 10, Figure 8 depicts for each problem instance the ratio of the expected reward achieved by policy $\pi_{\widehat{V}_\sigma}$ to that accrued by policy $\pi_{\widehat{V}_\tau}$.

The upward trend in Figure 8 confirms the relationship suggested by the results of Table 2

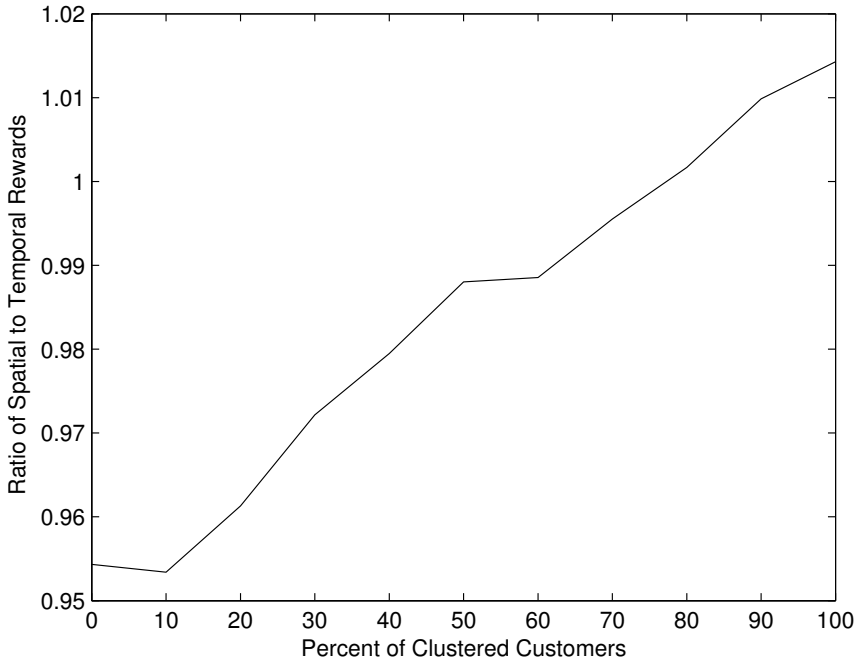


Figure 8: Impact of Customer Locations on Temporal and Spatial Policy Performance

and further suggests the performance of the spatial policy surpasses that of the temporal policy when at least 80 percent of customer locations are clustered in two groups. Intuition suggests the relationship of Figure 8 results from decreased variability in customer locations as the distribution moves from uniform to clustered. Specifically, the sequences of service calls $\tilde{\mathcal{C}}^p$ simulated to calculate spatial VFA $\widehat{V}_\sigma(\cdot,\cdot)$ are better approximations of actual request locations when customers are grouped versus randomly dispersed over the service area. Thus, spatial information more accurately anticipates rewards-to-go than temporal considerations when customer locations are more predictable, but temporal information becomes key as location variability rises.

To conclude our discussion, we identify rollout policy $\pi_{r\tau}$ as the all-around best among the six policies considered in our experiments. Not only does policy $\pi_{r\tau}$ achieve rewards at least as high as the other policies, the computation can be shifted online or offline depending on available computing resources and the time available to select decisions. Additional computing concessions may be realized via IZS.

7 Conclusion

Recognizing the VRPSSR as an important problem in urban transportation, we study heuristic solution methods to obtain policies that dynamically direct vehicle movement and manage service requests via temporal and spatial anticipation. Our work integrates predictive tools with prescriptive optimization methods, making contributions to the vehicle routing literature as well as general methodological contributions to the field of ADP:

First, our offline-online approach yields computationally-tractable, high-quality dynamic routing policies which achieve temporal-spatial anticipation by pairing with a rollout algorithm the state-of-the-art temporal VFA. We observe the resulting rollout policies compensate for anticipation absent in the base policies. Second, we identify the geographic spread of customer locations as a predictor of the success of temporal versus spatial anticipation, showing the latter performs better as customer locations move from uniform to clustered across the service area. Third, our combination of VFAs and rollout algorithms demonstrates the potential benefit of using offline and online methods in tandem as a hybrid ADP procedure, making possible higher quality policies with reduced computing requirements for real-time decision-making. Fourth, we identify a policy improvement guarantee applicable to VFA-based rollout algorithms, thus explaining why improvement over the underlying VFA policy can be expected. Our result is broadly applicable: any base policy composed of deterministic decision rules is a sequentially consistent heuristic. Fifth, our work is the first to combine rollout algorithms with IZS, significantly reducing the computation required to evaluate rewards-to-go without degrading policy quality.

Future research might extend our work to dynamic VRP variants. For example, vehicle travel times tend to vary not only spatially but across time, thus a combination of spatial and temporal anticipation may yield high-quality dynamic routing policies when travel times are uncertain. Additionally, the demand for bicycles across a bike-sharing network fluctuates based on location and time-of-day, thereby suggesting spatial-temporal anticipation may be helpful when devising inventory-routing policies to coordinate bike movement. An alternative direction for future research might be to explore enhancements to offline-online ADP methods. For example, in our work we identify VFAs offline and then embed the VFAs in an online rollout algorithm. It may be possible to iteratively improve the VFAs based on the performance of the rollout algorithm, per-

haps by updating VFA parameters if the rollout policy yields significantly different rewards than the VFA policy.

Acknowledgements

The authors gratefully acknowledge the suggestions of Barrett Thomas as well as the comments of anonymous referees. Justin Goodson wishes to express appreciation for the support of Saint Louis University's Center for Supply Chain Management.

References

ATA Economics & Statistical Analysis Department (1999). Standard trucking and transportation statistics. Technical report, American Trucking Association, Alexandria, VA.

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

Bertsekas, D., J. Tsitsiklis, and C. Wu (1997). Rollout algorithms for combinatorial optimization. *Journal of Heuristics 3*(3), 245–262.

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

Branchini, R., A. Armentano, and A. Løkketangen (2009). Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers and Operations Research 36*, 2955–2968.

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

Capgemini (2012, September). Number of global e-commerce transactions from 2011 to 2015 (in billions). `http://www.statista.com/statistics/369333/number-ecommerce-transactions-worldwide/`, Accessed on September 21, 2015.

Chen, H.-K., C.-F. Hsueh, and M.-S. Chang (2006). The real-time time-dependent vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review 42*(5), 383–408.

Ferrucci, F., S. Bock, and M. Gendreau (2013). A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods. *European Journal of Operational Research 225*(1), 130–141.

Gendreau, M., F. Guertin, J.-Y. Potvin, and R. Séguin (2006). Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies 14*(3), 157–174.

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

Ghiani, G., E. Manni, A. Quaranta, and C. Triki (2009). Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review 45*(1), 96–106.

Ghiani, G., E. Manni, and B. W. Thomas (2012). A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. *Transportation Science 46*(3), 374–387.

Goodson, J., B. Thomas, and J. Ohlmann (2015). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. Working paper, Saint Louis University, `http://www.slu.edu/~goodson/papers/GoodsonRolloutFramework.pdf`, Accessed on June 18, 2015.

Goodson, J., B. Thomas, and J. Ohlmann (2016). Restocking-based rollout policies for the vehicle routing problem with stochastic demand and duration limits. *Transportation Science 50*(2), 591–607.

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

Ichoua, S., M. Gendreau, and J.-Y. Potvin (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science 34*(4), 426–438.

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

Jaana, R., S. Sven, M. James, W. Jonathan, and A.-B. Yaw (2013). Urban world: The shifting global business landscape.

Jaillet, P. (1985). *Probabilistic traveling salesman problem*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Kim, S.-H. and B. L. Nelson (2001). A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS) 11*(3), 251–273.

Kim, S.-H. and B. L. Nelson (2006). On the asymptotic validity of fully sequential selection procedures for steady-state simulation. *Operations Research 54*(3), 475–488.

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

Li, H. and N. Womer (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research 246*, 20–33.

Lin, C., K. L. Choy, G. T. Ho, H. Lam, G. K. Pang, and K. Chin (2014). A decision support system for optimizing dynamic courier routing operations. *Expert Systems with Applications 41*(15), 6917–6933.

Meisel, S. (2011). *Anticipatory Optimization for Dynamic Decision Making*, Volume 51 of *Operations Research/Computer Science Interfaces Series*. Springer.

Mitrović-Minić, S. and G. Laporte (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological 38*(7), 635–655.

Novoa, C. and R. Storer (2008). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research 196*(2), 509–515.

Powell, W. (2011). *Approximate Dynamic Programming, 2nd Edition*. New York, NY: John Wiley and Sons.

Powell, W. B., M. T. Towns, and A. Marar (2000). On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science 34*(1), 67–85.

Psaraftis, H., M. Wen, and C. Kontovas (2015). Dynamic vehicle routing problems: three decades and counting. To appear in *Networks*.

Psaraftis, H. N. (1980). A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science 14*(2), 130–154.

Rosenkrantz, D. J., R. E. Stearns, and P. M. Lewis (1974). Approximate algorithms for the traveling salesperson problem. In *Switching and Automata Theory, 1974., IEEE Conference Record of 15th Annual Symposium on*, pp. 33–42. IEEE.

Swihart, M. R. and J. D. Papastavrou (1999). A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *European Journal of Operational Research 114*(3), 447–464.

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

The Tioga Group (2003, January). Chapter 3: Truck transportation. In *Project 99-130: Goods Movement Truck and Rail Study*.

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

Ulmer, M. W., D. C. Mattfeld, and F. Köster (2016). Budgeting time for dynamic vehicle routing with stochastic customer requests. *To appear in Transportation Science*. `http://www.winfo.tu-bs.de/hp/upload/paper/ulmer/WP_Ulmer_BudgetingTime.pdf`, Accessed on August 26, 2015.

United Nations (2015, September). Proportion of population in cities worldwide from 1985 to 2050. [Online; accessed 21-September-2015].

U.S. Department of Transportation Federal Motor Carrier Safety Administration (2005). Hours of service final rule for truck drivers. https://www.fmcsa.dot.gov/regulations/hours-of-service, Accessed on November 24, 2015.

van Hemert, J. I. and J. A. La Poutré (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. In *Parallel Problem Solving from Nature-PPSN VIII*, pp. 692–701. Springer.

Appendix

Decision Space Reduction

Because the size of feasible action set $\mathcal{X}(s_k)$ may increase exponentially with $|\bar{\mathcal{C}}_k|$ and $|\tilde{\mathcal{C}}_k|$ and because condition (4) may be computationally prohibitive to check, our offline-online heuristic operates on a subset $\bar{\mathcal{X}}(s_k) \subseteq \mathcal{X}(s_k)$ obtained by making two adjustments. First, we disallow waiting. Our experience suggests explicit consideration of the waiting decision significantly increases computation without leading to substantially better policies (Ulmer et al., 2016). Second, we take $\bar{\mathcal{C}}_k$ to be an ordered set, fix the sequence of assigned customers composing $\bar{\mathcal{C}}_k$, and use cheapest insertion of the customer requests in $\tilde{\mathcal{C}}_k$ as a proxy for condition (4). Specifically, for a given value of binary vector a in condition (1), a route is constructed by inserting the customers in $\tilde{\mathcal{C}}'_k$ into $\bar{\mathcal{C}}_k$ via standard cheapest insertion (Rosenkrantz et al., 1974) with the constraint that current location c_k begin the sequence. The vehicle's next location c associated with this value of a is the second element of the cheapest insertion route, the element immediately following current location c_k. If the travel time of the resulting route is less than or equal to the remaining time $T - t_k$, then the decision belongs to $\bar{\mathcal{X}}(s_k)$, otherwise it is excluded. Finally, the decision selected from $\bar{\mathcal{X}}(s_k)$ determines the sequence of assigned customer requests for the next decision epoch. The initial sequencing of the early-request customers $\mathcal{C}_{\text{early}}$ is also performed via cheapest insertion.

Although reducing the space of decisions in this fashion improves the computational tractability for our offline-online ADP heuristic, it is possible that neglecting alternative routing sequences may remove from consideration decisions leading to higher objective values. However, the literature suggests sophisticated routing methods do not necessarily lead to substantially better outcomes

(Powell et al., 2000; Goodson et al., 2016). In particular, for the VRPSSR we observe only minor improvement when replacing the cheapest insertion route with a route corresponding to an optimal solution of an open traveling salesman problem, a computationally-intensive routing procedure.