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

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

December 3, 2015

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 broader field of ADP. We identify the geographic spread of customer locations as a key predictor of the success of temporal versus spatial anticipation. We combine value function approximation (VFA) with rollout algorithms as a means of enhancing the anticipation of the VFA policy, resulting in spatial-temporal rollout policies. 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 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 the 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 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 incorporate temporal and spatial anticipation of service requests into approximate dynamic programming (ADP) procedures to yield dynamic vehicle routing policies. 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 delivery routes for uncapacitated 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, 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). Likewise, less-than-truckload logistics can be approximated via the routing of an uncapacitated vehicle (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 temporal and spatial anticipation to dynamically move vehicles in response to customer requests. Figure 1 illustrates the potential value of anticipa-
Figure 1: Anticipating Times and Locations of Future Requests

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. Our research aims to fill this gap, exploring means to connect temporal and spatial anticipation of service requests with routing optimization.

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, 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 value function approximation (VFA) with the temporal VFA of Ulmer et al. (2015), 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.

Second, we design dynamic routing policies by pairing with a rollout algorithm our spatial VFA policy and the temporal VFA policy of Ulmer et al. (2015). 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 case the temporal or spatial VFA policy – to approximate the remainder of the tree via the base policy’s rewards-to-go. 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. Further, we find rollout algorithms compensate for anticipation absent in the base policy, thus a rollout algorithm adds spatial anticipation to a temporal base policy and adds temporal anticipation to a spatial base policy. Indeed, we observe the performance of our spatial-temporal rollout policies improves on
the performance of temporal and spatial VFA policies in isolation. Looking to the broader routing literature and toward general dynamic and stochastic optimization problems, we believe rollout algorithms may serve as a connection between data-driven predictive tools and optimization.

**Contributions to Approximate Dynamic Programming**

We make three methodological contributions to the broader field of ADP. First, 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). However, as evidenced by the work of Meisel (2011), for most problems of practical interest, computational limitations restrict VFAs to 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 VFAs and rollout policies in §4 followed by a presentation of computational experience in §5. We conclude the paper in §6.

2 Problem Statement and Formulation

The vehicle routing problem with stochastic service requests (VRPSSR) is characterized by the need to dynamically design a route for one uncapacitated vehicle to meet service calls arriving randomly over a working day of duration $T$ and within a service region $C$. 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 $C_{\text{early}} \subseteq C$ 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 possible.

We model the VRPSSR as an MDP. The state of the system at decision epoch $k$ is the tuple $s_k = (c_k, t_k, \tilde{C}_k, \bar{C}_k)$, where $c_k$ is the vehicle’s position in service region $C$ 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 $k$, $\tilde{C}_k \subseteq C$ is the set of confirmed customers not yet serviced, and
\(\tilde{C}_k \subseteq C\) 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 \(s_0 = (\text{depot}, 0, C_{\text{early}}, \emptyset)\), the vehicle is positioned at the depot at time zero, has yet to serve the early-request customers composing \(C_{\text{early}}\), 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 \(C_{\text{early}}\), 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 \(\tilde{C}_K\) is empty.

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

\[
\mathcal{X}(s_k) = \left\{ (a, c) : \right. \\
\left. a \in \{0, 1\}^{\mid\tilde{C}_k\mid}, \right. \\
\left. c \in \tilde{C}_k \cup \tilde{C}_k' \cup \{c_k\} \cup \{\text{depot}\}, \right. \\
\left. c \neq \text{depot} \text{ if } \tilde{C}_k \cup \tilde{C}_k' \setminus \{c_k\} \neq \emptyset, \right. \\
\left. \text{feasible routing} \right\}.
\]

Condition (1) requires each service request in \(\tilde{C}_k\) to be accepted or rejected. Condition (2) constrains the vehicle’s next location to belong to the set \(\tilde{C}_k \cup \tilde{C}_k' \cup \{c_k\} \cup \{\text{depot}\}\), where \(\tilde{C}_k' = \{c \in \tilde{C}_k : a_{\tilde{C}_k^{-1}}(c) = 1\}\) is the set of customers confirmed by \(a\) and \(\tilde{C}_k^{-1}(c)\) returns the index of element \(c\) in \(\tilde{C}_k\). Setting \(c = c_k\) is the decision to wait at the current location for a base unit of time \(\tilde{t}\). Per condition (3), travel to the depot is disallowed when confirmed customers in \(\tilde{C}_k\) and \(\tilde{C}_k'\) have yet to be serviced. Condition (4) requires a route exists from the current location, through all confirmed 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
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 §4, 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 confirmed late-request customers: $R(s_k, x) = |\tilde{C}_k^x(s_k, x)|$, where $\tilde{C}_k^x(s_k, x)$ is the set $\tilde{C}_k$ specified by state $s_k$ and $a$, the confirmation component of decision $x$.

Choosing decision $x$ when in state $s_k$ transitions the process to post-decision state $s_k^x = (c_k, t_k, \tilde{C}_k^x)$ where the set of confirmed customers $\tilde{C}_k^x = \tilde{C}_k \cup \{c\}$ is updated to include the location component $c$ of decision $x$. How the process transitions to pre-decision state $s_{k+1} = (c_{k+1}, t_{k+1}, \tilde{C}_{k+1}, \tilde{C}_{k+1})$ 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 $c$ when a new set $\tilde{C}_{k+1}$ of late-request customers is observed. 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 $\tilde{C}_{k+1} = \tilde{C}_k \setminus \{c_k\}$ 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 $\bar{t}$ when a new set $\tilde{C}_{k+1}$ of late-request customers is observed. The arrival time $t_{k+1} = t_k + \bar{t}$ is incremented by the waiting time and $\tilde{C}_k = \tilde{C}_k^x$ is unchanged.

Denote a policy $\pi$ by a sequence of decision rules $(X_0^\pi, X_1^\pi, \ldots, X_K^\pi)$, where each decision rule $X_k^\pi(s_k) : s_k \mapsto X(s_k)$ is a function mapping the current state to a feasible decision. Letting $\Pi$ be the set of all Markovian deterministic policies, we a seek a policy $\pi$ in $\Pi$ 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
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.

gies 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. Additionally, within a genetic algorithm, van Hemert and La Poutré (2004) give preference to routes more capable of accommodating future requests. With the exception of Ichoua et al. (2006) and van Hemert and La Poutré (2004), these heuristic methods do not account for uncertainty in request locations and times. In our work, we seek to explicitly anticipate customer requests across time and space.

Building on the idea of Psaraftis (1980), Bent and Van Hentenyck (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 Hentenyck (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 §4.2 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 examine 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 Hentenyck (2004), an insight we employ in the spatial approximation of §4.2 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
<table>
<thead>
<tr>
<th>Literature</th>
<th>Solution Approach</th>
<th>Anticipation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Subset Selection</td>
<td>Online</td>
</tr>
<tr>
<td>Psaraftis (1980)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Bertsimas and Van Ryzin (1991)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Tassiulas (1996)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Gendreau et al. (1999)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Swihart and Papastavrou (1999)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Ichoua et al. (2000)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Larsen et al. (2002)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Mitrović-Minić and Laporte (2004)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Bent and Van Hentenryck (2004)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>van Hemert and La Poutré (2004)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Branke et al. (2005)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ichoua et al. (2006)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Gendreau et al. (2006)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Hvattum et al. (2006)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Thomas (2007)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ghiani et al. (2009)</td>
<td>n/a</td>
<td>✓</td>
</tr>
<tr>
<td>Meisel (2011)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ulmer et al. (2015)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Spatial-Temporal Rollout Policy \( \pi_{\tau r} \) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

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 considers at least the decisions in \( \tilde{\mathcal{X}}(\cdot) \), a subset of the feasible decisions \( \mathcal{X}(\cdot) \) we define in §4. Papers
focusing on fewer decisions typically give explicit consideration to feasible vehicle destinations and then employ a greedy procedure to accept 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. 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. 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. Notably, only our spatial-temporal rollout policy incorporates online and offline methods to both direct vehicle movement and accept or reject service requests. Further, only our spatial-temporal rollout policy 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.

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. (2015) focus exclusively on temporal anticipation. Excluding our own work,
only three of 18 methods anticipate future service requests across all four dimensions.

4 Heuristic Solution Methods

In §4.1 and §4.2, we describe temporal and spatial methods to approximate the value function, respectively. Then, in §4.3, we present rollout algorithms based on the temporal and spatial approximation policies. Because the size of feasible action set $\mathcal{X}(s_k)$ may increase exponentially with $|\tilde{C}_k|$ and $|\tilde{\tilde{C}}_k|$ and because condition (4) may be computationally prohibitive to check, our heuristic solution methods operate on a subset $\tilde{\mathcal{X}}(s_k) \subseteq \mathcal{X}(s_k)$ 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. Second, we take $\tilde{C}_k$ to be an ordered set, fix the sequence of confirmed customers composing $\tilde{C}_k$, and use cheapest insertion of the customer requests in $\tilde{\tilde{C}}_k$ 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 $\tilde{\tilde{C}}_k$ into $\tilde{\tilde{C}}_k$ 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 $\tilde{\mathcal{X}}(s_k)$, otherwise it is excluded. Finally, the decision selected from $\tilde{\mathcal{X}}(s_k)$ determines the sequence of confirmed customer requests for the next decision epoch. The initial sequencing of the early-request customers $C_{\text{early}}$ is also performed via cheapest insertion.

Although reducing the space of decisions in this fashion improves the computational tractability for our heuristic solution procedures, it is possible that neglecting alternative routing sequences may remove from consideration decisions leading to higher objective values. However, our experience suggests routing confirmed customers with search methods more sophisticated than cheapest insertion does not identify feasible decisions leading to substantially better outcomes.

4.1 Temporal Value Function Approximation

Our first heuristic solution method is that of Ulmer et al. (2015), which we summarize in this section. Ulmer et al. (2015) base their approach on the well-known value functions, formulated
around the post-decision state variable:

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

Although solving equation (5) for all post-decision states $s^x_k$ 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. (2015) develop a VFA by focusing on temporal elements of the post-decision state variable. Specifically, Ulmer et al. (2015) map a post-decision state variable $s^x_k$ 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 confirmed customers in $\bar{C}_k$ and return to the depot.

Representing their approximate value function as a two-dimensional lookup table, Ulmer et al. (2015) use AVI (Powell, 2011) to estimate the value of being at time $t_k$ with budget $b_k$. Ulmer et al. (2015) build on the classical procedure of iterative simulation, optimization, and smoothing by dynamically adjusting the granularity of the look-up table. Figure 3 illustrates the process. 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 $\hat{V}_\tau(t_k, b_k)$, where we use the Greek letter $\tau$ 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, $\hat{V}_\tau(t_k, b_k)$ can be used to execute a dynamic routing scheme. When the process occupies state $s_k$, the temporal VFA decision rule is

$$X^x_{\pi_{\hat{V}_{\tau}}} (s_k) = \arg \max_{x \in \mathcal{X}(s_k)} \left\{ R(s_k, x) + \hat{V}_\tau(t_k, b_k) \right\}. \quad (6)$$

Figure 2b depicts equation (6), illustrating the rule’s consideration of each decision’s period-$k$ reward $R(s_k, x)$ plus $\hat{V}_\tau(t_k, b_k)$, the estimate of the expected reward-to-go from the post-decision state. The VFA policy $\pi_{\hat{V}_{\tau}}$ is the sequence of decision rules $(X^x_{\pi_{\hat{V}_{\tau}}} , X^x_{\pi_{\hat{V}_{\tau}}} , \ldots , X^x_{\pi_{\hat{V}_{\tau}}} )$. Thus, using only temporal aspects of the state variable, VFA $\hat{V}_\tau(\cdot, \cdot)$ can be used to dynamically route a vehicle via the policy $\pi_{\hat{V}_{\tau}}$. For further details, we refer the reader to Ulmer et al. (2015).
4.2 Spatial Value Function Approximation

Although the temporal VFA of Ulmer et al. (2015) 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. (2015) 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 confirmed customers $\tilde{\mathcal{C}}_k$. Depending on the likelihood of service requests across time and space, a routing of the confirmed 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. Influenced by the work of Bent and Van Hentenryck (2004), the spatial VFA we propose in this section explicitly considers service area coverage.

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 $\tilde{\mathcal{C}}^p = (\tilde{\mathcal{C}}_1^p, \tilde{\mathcal{C}}_2^p, \ldots, \tilde{\mathcal{C}}_K^p)$ be the sequence of service request realizations associated with the $p^{th}$ simulation trajectory and let $\tilde{\mathcal{C}}_k^p = \bigcup_{i=k}^{K} \tilde{\mathcal{C}}_i^p$ be the union of the service requests in periods $k$ through $K$. From a post-decision state $s_k^p$, we use cheapest insertion (Rosenkrantz et al., 1974) to construct a route beginning at loca-
Figure 4: Spatial Value Function Approximation

tation $c_k$ at time $t_k$, through the set of confirmed customers $\bar{C}_k^x$, and through as many service requests as possible in set $\bar{C}_p(k+1)$ such that the vehicle returns to the depot no later than time $T$. The routing procedure assumes requests in $\bar{C}_p(k+1)$ 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 $\bar{C}_p(k+1)$ successfully routed in sample $p$, the spatial VFA is $\hat{V}_\sigma(c_k, \bar{C}_k^x) = P^{-1} \sum_{p=1}^{P} Q_p$, where we use the Greek letter $\sigma$ to indicate “spatial.”

Figure 4 illustrates the process of simulation and routing. The center portion of Figure 4 depicts the set of requests $\bar{C}_p(k+1)$ 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 $\bar{C}_k^x$, 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 4 shows the cheapest insertion routing of the requests in $\bar{C}_p(k+1)$. In this example, three of the four requests comprising $\bar{C}_p(k+1)$ are successfully routed, thus $Q_p = 3$. Repeating this process across all $P$ simulations, then averaging the results, yields $\hat{V}_\sigma(c_k, \bar{C}_k^x)$.

In contrast to offline temporal VFA $\hat{V}_\tau(\cdot, \cdot)$, spatial VFA $\hat{V}_\sigma(\cdot, \cdot)$ 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
\[ X_k^{\pi_{\sigma}}(s_k) = \arg \max_{x \in \bar{X}(s_k)} \left\{ R(s_k, x) + \hat{V}_{\sigma}(c_k, \bar{C}_k) \right\}. \]  

Figure 2c depicts equation (7), illustrating the rule’s consideration of each decision’s period-\(k\) reward \(R(s_k, x)\) plus \(\hat{V}_{\sigma}(c_k, \bar{C}_k)\), the estimate of the expected reward-to-go from the post-decision state. The VFA policy \(\pi_{\hat{V}_{\sigma}}\) is the sequence of decision rules \((X_0^{\pi_{\sigma}}, X_1^{\pi_{\sigma}}, \ldots, X_K^{\pi_{\sigma}})\).

### 4.3 Spatial-Temporal Rollout Algorithms

Although the VFAs of §4.1 and §4.2 explicitly incorporate temporal and spatial information into reward-to-go approximations, respectively, neither method gives simultaneous consideration to both. While integrating coverage area into the offline learning procedure of Ulmer et al. (2015) is a logical way to tap the potential benefits of spatial-temporal anticipation, doing so is computationally prohibitive. In this section, we use rollout algorithms, an online ADP method, to enhance the spatial anticipation of temporal policy \(\pi_{\hat{V}_{\tau}}\) and the temporal anticipation of spatial policy \(\pi_{\hat{V}_{\sigma}}\).

Rollout algorithms, introduced by Bertsekas et al. (1997), aim to improve the performance of a base policy – in this case \(\pi_{\hat{V}_{\tau}}\) or \(\pi_{\hat{V}_{\sigma}}\) – by using that policy in a given current state to approximate the rewards-to-go from potential future states. Because a rollout algorithm explicitly builds a portion of the decision tree, the resulting rollout policy is anticipatory by definition. Thus, a rollout algorithm built on base policy \(\pi_{\hat{V}_{\tau}}\) may include more spatial information than \(\pi_{\hat{V}_{\tau}}\) in isolation. Similarly, a rollout algorithm built on base policy \(\pi_{\hat{V}_{\sigma}}\) may contain additional temporal features than \(\pi_{\hat{V}_{\sigma}}\) by itself.

Taking as base policies \(\pi_{\hat{V}_{\tau}}\) and \(\pi_{\hat{V}_{\sigma}}\), we consider two post-decision rollout algorithms, each of which uses the assigned base policy to approximate rewards-to-go from the post-decision state (Goodson et al., 2015). In what follows, we sometimes refer to the VFA base policy as \(\pi_{\hat{V}}\), recognizing this as a placeholder for either temporal VFA policy \(\pi_{\hat{V}_{\tau}}\) or spatial VFA policy \(\pi_{\hat{V}_{\sigma}}\). From a given post-decision state \(s_k^x\), the rollout algorithm takes as the expected reward-to-go the value of policy \(\pi_{\hat{V}}\) from epoch \(k\) onward, \(\mathbb{E}[\sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\hat{V}}}(s_i))|s_k^x]\), a value we estimate via simulation. Let \(\bar{C}^h = (\bar{C}_1^h, \bar{C}_2^h, \ldots, \bar{C}_K^h)\) be the sequence of service request realizations associated with the \(h\)th simulation trajectory and let \(V^{\pi_{\hat{V}}}(s_k^x, h) = \sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\hat{V}}}(s_i), \bar{C}_i^h)\) be the reward accrued by policy \(\pi_{\hat{V}}\) in periods \(k + 1\) through \(K\) when the process occupies post-decision state \(s_k^x\) and service requests are \(\bar{C}^h\). Then, the expected value of policy \(\pi_{\hat{V}}\) from state \(s_k^x\) onward is estimated as the
average value across \( H \) simulations: 
\[
V^\pi_(s^\pi_k) = H^{-1} \sum_{h=1}^H V^\pi_(s^\pi_k, h).
\]

Given the post-decision state estimate of the expected reward-to-go \( V^\pi_(s^\pi_k) \), the rollout decision rule is

\[
X^\pi_r(s_k) = \arg \max_{x \in \mathcal{X}(s_k)} \{ R(s_k, x) + V^\pi_(s^\pi_k) \},
\]

where the subscripted \( r \) indicates “rollout.” In Figure 2d, we append \( r \) with \( \tau \) and \( \sigma \) as references to the underlying base policies. Figure 2d depicts equation (8) for decision rules \( X^\pi_{r-r} \) and \( X^\pi_{r-s} \), illustrating the rules’ consideration of each decision’s period-\( k \) reward plus \( V^\pi_(s^\pi_k) \) and \( V^\pi_(s^\pi_k) \), respectively. As shown in Algorithm 5 of Goodson et al. (2015), the post-decision rollout algorithm applies the rollout decision rule of equation (8) to select decisions in observed states, yielding the rollout policy \( \pi_r = (X^\pi_{r-0}, X^\pi_{r-1}, \ldots, X^\pi_{r-K}) \).

### 4.4 VFA-Based Rollout Improvement

In addition to serving as a mechanism to incorporate spatial-temporal information into heuristic decision selection 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. Building a rollout algorithm on the temporal VFA of Ulmer et al. (2015) 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. Further, under a mild condition, we show in Proposition 1 that a VFA policy \( \pi_\hat{V} \) is a sequentially consistent heuristic, a condition that guarantees rollout policy \( \pi_r \) performs at least as well as policy \( \pi_\hat{V} \) (Goodson et al., 2015). To guarantee this weak improvement, Proposition 1 requires the VFA decision rule \( X^\pi_k \) to return the same decision every time it is applied in the same state, a condition we call deterministic.

**Proposition 1** (Sequentially Consistent VFA Policy). If VFA decision rule \( X^\pi_k \) is deterministic, then VFA policy \( \pi_\hat{V} \) is a sequentially consistent heuristic.

**Proof.** Let \( s \) be a pre- or post-decision state in the state space and let \( s' \) be a pre- or post-decision state such that it is on a sample path induced by VFA policy \( \pi_\hat{V} \) applied in state \( s \). If \( s' \) is a pre-decision state, let \( k' \) index the associated decision epoch. Otherwise, if \( s' \) is a post-decision
state, let $k'$ index the decision epoch associated with the subsequent period. By the assumption that $X_k^{\pi_V}$ is deterministic, the sequence of decision rules from epoch $k$ onward is the same: $(X_k^{\pi_V}, X_{k+1}^{\pi_V}, \ldots, X_K^{\pi_V})$. Because the same argument holds for all $s$ and $s'$, the VFA policy $\pi_{\hat{V}}$ satisfies Definition 10 of Goodson et al. (2015) and is a sequentially consistent heuristic.

With some care, temporal and spatial VFA decision rules $X_k^{\pi_{\hat{V}_\tau}}$ and $X_k^{\pi_{\hat{V}_\sigma}}$ can be made deterministic. Provided VFA $\hat{V}_\tau(t_k, b_k)$ 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 $X_k^{\pi_{\hat{V}_\tau}}$ is deterministic. Similarly, if the calculation of $\hat{V}_\sigma(t_k, b_k)$ uses a common set of simulated service requests $\{\hat{C}_p\}_{p=1}^P$ across all applications of the approximation, and if ties in decisions achieving the maximum value in equation (7) are broken the same way each time the decision rule is applied to the same state, then $X_k^{\pi_{\hat{V}_\sigma}}$ is deterministic. Finally, 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 $V^{\pi_{\hat{V}}}$ a more accurate estimate of $\mathbb{E}\left[\sum_{i=k+1}^{K} R(s_i, X_i^{\pi_{\hat{V}}}(s_i))|s_k^x\right]$. In the computational experiments of §5, $H = 16$ simulations is sufficient to observe improvement of $\pi_r$ over $\pi_{\hat{V}}$.

Beyond the rollout improvement resulting from a sequentially consistent 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 (8), which approximates expected rewards-to-go via policy $\pi_{\hat{V}}$ applied from post-decision states, a one-step decision rule applies policy $\pi_{\hat{V}}$ 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 $\pi_{\hat{V}}$ is a sequentially consistent heuristic, the value of post-decision rollout policy $\pi_r$ 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 a sequentially consistent heuristic. Thus, any policy composed of deterministic functions mapping states to the space of feasible decisions is a sequentially consistent heuristic 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 Computational Experience

We outline problem instances in §5.1 followed by a discussion of computational results in §5.2. All methods are coded in Java and executed on 2.4GHz AMD Opteron dual-core processors with 8GB of RAM.

5.1 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 $C$, a large 20-kilometer-square region and a small 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 $\lambda$ customers per $T - 1$ minutes. Thus, the number of customers in set $\tilde{C}_{k+1}$ 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 $\lambda$: 25, 50, and 75, which we refer to as low,
moderate, and high, respectively. The number of early-request customers in set \( C_{\text{early}} \), 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.

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 small 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 Euclidian 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.

5.2 Discussion

In this section we compare and contrast the performance of temporal policy \( \pi_{\hat{V}} \), spatial policy \( \pi_{\hat{V}_s} \), and spatial-temporal post-decision rollout policies \( \pi_{rT} \) and \( \pi_{rS} \) across the problem instances of §5.1. As benchmarks, we also consider the performance of a myopic policy \( \pi_m \) with period-\( k \) decision rule \( X_{k}^{\pi_m}(s_k) = \arg \max_{x \in A(s_k)} \{ R(s_k, x) \} \) and of a post-decision rollout policy \( \pi_{rm} \) with base policy \( \pi_m \). The period-\( k \) decision rule associated with policy \( \pi_{rm} \) is analogous to equation (8) with the second term estimating the reward-to-go of the myopic policy via \( H \) simulations.
Table 2 presents estimates of the expected number of late-request customers serviced by each policy across small and large service areas; low, moderate, and high values of \( \lambda \); and uniform, two-cluster, and three-cluster customer locations. Each figure is an average of the reward collected across the 250 realizations of the corresponding problem instance. The offline VFA associated with policies \( \hat{\pi}_\tau \) and \( \pi_{r\tau} \) is obtained via 1,000,000 iterations of AVI and a disaggregation threshold of 1.5 (Ulmer et al., 2015). We use \( P = 16 \) simulations to calculate reward-to-go estimate \( \hat{V}_\sigma(\cdot, \cdot) \) for policy \( \hat{\pi}_\sigma \). For rollout policies \( \pi_{rm} \) and \( \pi_{r\tau} \) we use \( H = 16 \) simulations to calculate reward-to-go estimate \( V_{\pi, \sigma} \). 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 \( \hat{V}_\sigma(\cdot, \cdot) \) and \( H = 16 \) simulations to calculate \( V_{\pi, \sigma} \). 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.

**Temporal vs. Spatial Anticipation**

We first compare the performance of temporal policy \( \pi_{\hat{V}_\tau} \) to that of spatial policy \( \pi_{\hat{V}_\sigma} \). Per Table 2, when customer locations are uniform over the service area, the expected number of late-request customers serviced by policy \( \pi_{\hat{V}_\tau} \) is almost always greater than or equal to the expected reward accrued by policy \( \pi_{\hat{V}_\sigma} \), thus suggesting current time \( t_k \) and time budget \( b_k \) are better predictors of the reward-to-go than service area coverage. In contrast, when customer locations are grouped in two or three clusters, policy \( \pi_{\hat{V}_\sigma} \) almost always outperforms policy \( \pi_{\hat{V}_\tau} \), indicating current location \( c_k \) and the tour through confirmed customers \( C_k \) 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 \( \lambda \), \( \gamma \) 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 \( \gamma \) from zero to 100 by increments of 10, Figure 5 depicts for each problem instance the ratio of the expected reward achieved by policy \( \pi_{\hat{V}_\sigma} \) to that accrued by policy \( \pi_{\hat{V}_\tau} \).

The upward trend in Figure 5 confirms the relationship suggested by the results of Table 2.
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 5 results from decreased variability in customer locations as the distribution moves from uniform to clustered. Specifically, the sequences of service calls $\tilde{C}^p$ simulated to calculate spatial VFA $\tilde{V}_o(\cdot, \cdot)$ 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.
Figure 5: Impact of Customer Locations on Temporal and Spatial Policy Performance

Rollout Improvement

Grouped by customer location, Figure 6 aggregates over quantities in Table 2 to display the percent improvement of policies $\pi_{rm}$, $\pi_\tau$, $\pi_{r\tau}$, $\pi_\sigma$, and $\pi_{r\sigma}$ over myopic policy $\pi_m$. Each bar in Figure 6 depicts the improvement of a base policy (solid outline) over $\pi_m$ 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. With only one exception, the disaggregate results of Table 2 also demonstrate the improvement afforded by a deterministic VFA decision rule. When customers are located uniformly across the small service area and $\lambda$ is high, spatial policy $\pi_\sigma$ 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 $H$, as discussed in §4.4. Further, although $\pi_{rm}$ yields substantial improvement over $\pi_m$ (7.2 percent), we observe higher expected rewards when the rollout algorithm is applied to base policies $\pi_\tau$ (9.1 percent) and $\pi_\sigma$ (8.4 percent), 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
Figure 6: Improvement Over Myopic Policy

Figure 6 indicates improvement of rollout policy $\pi_{rm}$ over myopic policy $\pi_m$ 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 $\pi_m$ to perform poorly, at times confirming 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 $\pi_{rt}$ over temporal base policy $\pi_t$ is most significant when customer locations are clustered. Recalling the intuition surrounding Figure 5, 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 $\pi_{rt}$ has more opportunity to make up for the spatial anticipation absent in policy $\pi_t$. Similarly, Figure 6 depicts improvement of spatial-temporal rollout policy $\pi_{r\sigma}$ over spatial base policy
\( \pi_\sigma \) as being more substantial when customer locations are less concentrated, i.e., in three clusters or uniform versus in two clusters. Drawing again on Figure 5, we believe policy \( \pi_{r}\sigma \) adds temporal anticipation to policy \( \pi_\sigma \), thus enhancing the spatial-only anticipation of the base policy.

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 \( \pi_\tau \). 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 \( \pi_\tau \), 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 small 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 \( \hat{V}_\sigma(\cdot, \cdot) \) may be unable to detect the overall patterns observed by temporal VFA \( \hat{V}_\tau(\cdot, \cdot) \), 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 \( \pi_\tau \) 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 \( \lambda \) moves from low to moderate and then again
Table 3: Maximum CPU Seconds to Select a Decision

<table>
<thead>
<tr>
<th>Policy</th>
<th>Low $\lambda$</th>
<th>Moderate $\lambda$</th>
<th>High $\lambda$</th>
<th>Low $\lambda$</th>
<th>Moderate $\lambda$</th>
<th>High $\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Small Service Area</td>
<td></td>
<td></td>
<td>Large Service Area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\pi_m$</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>$\pi_{\hat{V}_r}$</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>$\pi_{\hat{V}_c}$</td>
<td>1.9</td>
<td>16.4</td>
<td>129.3</td>
<td>&lt; 0.01</td>
<td>9.6</td>
<td>125.7</td>
</tr>
<tr>
<td>$\pi_{rm}$</td>
<td>0.2</td>
<td>1.9</td>
<td>39.8</td>
<td>&lt; 0.01</td>
<td>2.0</td>
<td>85.3</td>
</tr>
<tr>
<td>$\pi_{rr}$</td>
<td>0.6</td>
<td>3.7</td>
<td>46.7</td>
<td>&lt; 0.01</td>
<td>5.3</td>
<td>113.5</td>
</tr>
<tr>
<td>$\pi_{r\sigma}$</td>
<td>633.3</td>
<td>2862.1</td>
<td>22598.2</td>
<td>13.9</td>
<td>1592.6</td>
<td>39013.2</td>
</tr>
</tbody>
</table>

Customers Located Uniformly

Customers Located in Two Clusters

Customers Located in Three Clusters

as $\lambda$ moves from moderate to high. These increases in computing time are driven by an increase in the number of feasible decisions in the set $\bar{X}(\cdot)$, 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 $\pi_{rm}$ and $\pi_{rr}$, 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 $\pi_\sigma$ 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 $\pi_{rr}$, which performs comparably to policy $\pi_{r\sigma}$ in the vast majority of Table 2 entries, may be of limited practical use when $\lambda$ is high.
Table 4: Impact of Offline Computation on Online Performance

<table>
<thead>
<tr>
<th>Offline AVI Iterations</th>
<th>Online Simulations ((H))</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\pi_\tau)</td>
<td>47.7</td>
<td>46.0</td>
<td>47.2</td>
<td>49.0</td>
<td>50.5</td>
<td>51.3</td>
<td>51.7</td>
</tr>
<tr>
<td>1,000</td>
<td></td>
<td>43.0</td>
<td>44.2</td>
<td>45.5</td>
<td>47.3</td>
<td>49.0</td>
<td>50.4</td>
<td>51.6</td>
</tr>
<tr>
<td>10,000</td>
<td></td>
<td>46.2</td>
<td>45.3</td>
<td>46.9</td>
<td>48.6</td>
<td>50.2</td>
<td>51.5</td>
<td>51.7</td>
</tr>
<tr>
<td>100,000</td>
<td></td>
<td>49.4</td>
<td>47.4</td>
<td>48.6</td>
<td>50.0</td>
<td>51.0</td>
<td>51.6</td>
<td>52.2</td>
</tr>
<tr>
<td>1,000,000</td>
<td></td>
<td>50.1</td>
<td>48.2</td>
<td>49.4</td>
<td>50.5</td>
<td>51.7</td>
<td>52.3</td>
<td>52.5</td>
</tr>
<tr>
<td>5,000,000</td>
<td></td>
<td>50.1</td>
<td>48.5</td>
<td>49.5</td>
<td>50.7</td>
<td>51.7</td>
<td>52.2</td>
<td>52.3</td>
</tr>
</tbody>
</table>

Seeking a reduction in the CPU requirements for rollout policy \(\pi_\tau\), 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 \(\pi_\tau\). 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 \(\lambda\). Darker shades indicate higher expected rewards.

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.

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 \(\pi_\tau\) 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 $\bar{X}(s_k)$ leading to the largest expected reward-to-go. In particular, in equation (8), IZS may require fewer than $H$ simulations to calculate each $\hat{V}_{\pi}(s_k^x)$.

IZS is executed in three phases. In the first phase, for all decisions $x$ in $\bar{X}(s_k)$, $\hat{V}_{\pi}(s_k^x)$ is initialized via $n_{\text{initial}}$ simulations. The second phase identifies, with confidence level $1 - \alpha$, the reward-to-go estimates falling within $\delta$ (the indifference zone) of the maximum. The third phase discards all $V_{\pi}(s_k^x)$ not meeting the phase-two threshold and refines the remaining reward-to-go 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 $n_{\text{max}}$ – in which case the procedure returns the decision with the highest reward-to-go estimate. Setting parameter $n_{\text{max}}$ 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 $\pi_{\tau\tau}$ to the problem instance characterized by a large service region, customer locations grouped in two clusters, and high $\lambda$. We set the indifference zone to $\delta = 1$, the confidence parameter to $\alpha = 0.01$, and the maximum number of simulations to $n_{\text{max}} = 128$. Figure 7 displays the impact of IZS on CPU times and on rewards as the number of initial simulations $n_{\text{initial}}$ 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 $V_{\pi}(\cdot)$. The value of $n_{\text{initial}}$ or $H$ is displayed adjacent each point in Figure 7.
Figure 7: Impact of Indifference Zone Selection on Rewards and CPU Times

Figure 7 suggests IZS can achieve rewards comparable to a fixed-simulation implementation, but with potentially lower CPU times. Notably, setting the number of fixed simulations 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 $n_{\text{initial}}$ increases to 32 and beyond, IZS tends to terminate with $n_{\text{initial}}$ 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.

To conclude our discussion, we identify rollout policy $\pi_{r_T}$ as the all-around best among the six policies considered in our experiments. Not only does policy $\pi_{r_T}$ 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.
6 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 both the vehicle routing literature as well as general methodological contributions to the field of ADP. First, 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. Second, we pair with rollout algorithms temporal and spatial VFA policies and observe the resulting rollout policies compensate for anticipation absent in the base policies. 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. Our computational work concludes that combining the temporal VFA of Ulmer et al. (2015) with a rollout algorithm yields a computationally tractable dynamic routing policy that achieves high expected rewards.

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 comments and suggestions of Barrett Thomas. Justin Goodson wishes to express appreciation for the support of Saint Louis University’s Center for Supply Chain Management.

References


