Budgeting Time for Dynamic Vehicle Routing with Stochastic Customer Requests

Marlin W. Ulmer, Dirk C. Mattfeld, Felix Köster
Technische Universität Braunschweig, Germany
m.ulmer@tu-braunschweig.de

Abstract

Parcel services route vehicles to pick up parcels in the service area. Pickup requests occur dynamically during the day and are unknown before their actual request. Due to working hour restrictions, service vehicles only have limited time to serve dynamic requests. As a result, not all requests can be confirmed. To achieve an overall high number of confirmed requests, dispatchers have to budget their time effectively by anticipating future requests. To determine the value of a decision, i.e., the expected number of future confirmations given a point of time and remaining free time budget, we present an anticipatory time budgeting heuristic (ATB) drawing on methods of approximate dynamic programming. ATB frequently simulates problem’s realization to subsequently approximate the values for every vector of point of time and free time budget to achieve an approximation of an optimal decision policy. Since the number of vectors is vast, we introduce the dynamic lookup table (DLT), a general approach adaptively partitioning the vector space to the approximation process. Compared with state-of-the-art benchmark heuristics, ATB allows an effective use of the time budget resulting in anticipatory decision making and high solution quality. Additionally, the DLT significantly strengthens and accelerates the approximation process.

Keywords: dynamic vehicle routing, stochastic customer requests, time budget, subset selection, approximate dynamic programming, dynamic lookup table
1 Introduction

Last mile delivery presents an increasingly difficult challenge for logistic service providers. Due to a significant increase in e-commerce sales, the number of shipped customer to customer (C2C) and business to customer (B2C) parcels has grown significantly. In Germany, the B2C and C2C market increased about 50% in the last five years (Esser and Kurte 2015). Many small vendors use online market places to sell their products directly to the purchaser. For successful e-commerce, delivery times and delivery costs are two of the main influence factors (Lowe et al. 2014). Customers expect reasonably priced, fast, and reliable service (Ehmke 2012). For fast delivery, about 20% of all parcels are shipped via courier or express delivery. Many of the parcels are picked up directly at the seller (in the following called customer) and are processed the same day (Hvattum et al. 2006). Some early requests are known at the start but most of these pickups are requested throughout the shift (Lin et al. 2010). To serve these requests, courier express and parcel services schedule vehicles dynamically. The collected parcels are then transported to the depot to be shipped long haul (Pillac et al. 2013). In urban transportation, drivers’ wages are the main cost factor. Fuel costs are secondary. Dispatchers aim on utilizing the time budget induced by the driver’s shift to serve as many customers as possible (Thomas 2007). Usually, the early request customers (ERCs) have to be served. Then, some free time budget is left for serving late request customers (LRCs) requesting during the shift. To include LRCs, the dispatcher can dynamically adapt the planned tour (Gendreau et al. 2006). Since the overall time budget is limited, not all of the LRCs may be confirmed. For every LRC, the dispatcher decides about a confirmation or a rejection (Gendreau et al. 1999). Usually, the dispatcher accumulates incoming LRCs until the vehicle reaches the next customer’s location. Then, the subset of accumulated LRCs to be confirmed is selected (Meisel 2011). Every other LRC is rejected. Dispatchers aim on avoiding many rejections, since they may be served by an expensive third party, or may be postponed to following days (Angelelli et al. 2009).

In this paper, we consider the dynamic vehicle routing problem with stochastic service requests (VRPSSR). An uncapacitated vehicle serves customers in a service area. It starts its tour at a depot and returns to the depot before a time limit exceeds. A set of ERCs is known in the beginning and must be served. During the shift, new stochastic LRCs request service. Decisions about the subset
of LRCs to confirm are made, when the vehicle arrives at a customer. A decision is feasible, if all remaining ERCs and confirmed LRCs can be served and the vehicle returns to the depot within the time limit. The objective is to find a decision policy maximizing the expected rewards, i.e., number of confirmed LRCs.

Since current decisions impact the expected future rewards, information about future requests have to be anticipated for effective decision making. To this end, the service provider can derive stochastic probabilities of customer requests for certain regions of the service area by making prognoses about customer behavior (Dennis 2011). For anticipation, the stochastic information is integrated by methods of approximate dynamic programming (ADP, Powell 2011). Methods of ADP evaluate decisions with respect to the expected number of future rewards (called values). These values are approximated by forward programming, generally via simulation. Since for the VRPSSR the customers demand fast responses, the calculation time in the online decision state is highly limited. The ADP-simulations have to be conducted a priori, i.e., offline. The approximated value then can be efficiently accessed in the online decision state. Still, the values have to be stored. For the VRPSSR, the locations of potential customers are not known beforehand and requests may occur in the entire service area. A storage of an individual value for every possible decision state is not applicable and an aggregation and/or a partitioning of the state space to a lower dimensional space is mandatory.

In this paper, we present an anticipatory time budgeting approach (ATB). ATB seeks to estimate the future value of current decisions using an offline simulation that operates on an aggregated and partitioned state space. For aggregation, we identify two suitable temporal parameters as indicators for a decision’s value: the point of time and the free time budget. The free time budget is the amount of time left, if the vehicle serves all current customers, i.e., the not yet served ERCs and the confirmed and not yet served LRCs. The point of time and free time budget prove to be suitable indicators since they act as proxy for several spatial and temporal attributes of the states.

The aggregation of states to point of time and time budget leads to a two-dimensional vector space. For storage of the values, ATB then partitions the vector space to a lookup table (LT) and approximates the values for every entry of the LT. The quality of approximation significantly depends on this partitioning (Barto 1998, p. 193). To achieve an accurate approximation both a sufficient number of observations for every entry of the LT and a differentiation between states of
heterogeneous value are required. Since these characteristics are generally dependent on the problem’s and instances’ structures, static a priori partitioning methods are not able to capture these structures and provide insufficient approximation. We introduce the dynamic LT (DLT) adapting the partitioning according to the approximation process. The DLT starts with an initial partitioning and subsequently updates the partitioning with respect to both the number of observations of the entry and the value deviation within these observations. This allows a fast and reliable approximation, since every entry is observed a sufficient number of times. At the same time, it allows a detailed approximation, since entries generally represent only states with similar values. The DLT is a general partitioning algorithm and, in combination with a suitable aggregation, can be applied to many stochastic dynamic decision problems.

To show the benefits of ATB and the DLT, we compare ATB with state-of-the-art benchmark heuristics and the DLT with conventional partitioning algorithms. ATB increases the solution quality for the VRPSSR substantially. Compared to ATB based on conventional partitioning approaches, the DLT provides the highest solution quality. The approximation process is accelerated, the solution quality is generally independent of tuning, and the required memory for storage of the values is reduced significantly. In essence, the DLT enables fast, reliable, efficient, and effective approximation.

This paper offers contributions to the solution of the VRPSSR, understanding of the problem area, and to the general ADP methodology. This work transfers offline methods of ADP to the VRPSSR, a vehicle routing problem with stochastic customer requests and unknown request locations. Until now, ADP has been applied to problems with a small number of possible requests and known customer locations (Thomas 2007, Meisel 2011). Further, the presented ATB outperforms state-of-the-art benchmark heuristics significantly. The extensive analysis of the results allows understanding the functionality of ATB and the impact of point of time and time budget to the expected future confirmations. With the DLT, we introduce a general state space partitioning algorithm allowing effective and efficient approximation and simultaneously deriving insights of problem, instance, and solution structure.

This paper is outlined as follows. In §2, we present and discuss the related literature focusing on dynamic vehicle routing problems with stochastic customer requests. In §3, we formally define the VRPSSR. In §4, we present the anticipatory time budgeting approach ATB. Therefore, we
briefly recall the functionality of ADP and present the required state space aggregation based on the dependencies of point of time, free time budget, and value. We introduce the dynamic lookup table in §5. In §6, we conduct an extensive computational evaluation for a variety of real-world sized instances differing in customer distribution, service area size, and ratio of dynamic customers. In §7, we analyze the behavior of ATB and DLT in detail. The paper concludes with a summary of the results and directions for future research in §8.

2 Literature Review

The VRPSSR is a stochastic and dynamic vehicle routing problem based on the definitions of Kall and Wallace (1994). The problem is stochastic because not all information is provided in the beginning, but is revealed over time. It is dynamic because the problem setting allows adaptions of decision making regarding the revealed new information. In the literature, stochastic impacts include uncertainty in travel times (Thomas and White III 2007, Lecluyse et al. 2009, Ehmke et al. 2015), in service times (Daganzo 1984, Larsen et al. 2002), in the magnitude of customer demands (Secomandi 2000, Novoa and Storer 2009, Goodson et al. 2013), and in the presence of customer requests (Psaraftis 1980, Bent and Van Hentenryck 2004). An extensive review of the different dynamic vehicle routing problems is given by Pillac et al. (2013). In our work, we focus on work considering stochastic customer requests. For these problems, decisions consider both routing and request confirmations.

Table 1 shows an overview of dynamic routing approaches with stochastic customer requests. We classify work regarding the objective, the customer representation, and the solution approach. A part of the work aims at maximizing the number of served customers within the shift. This work is represented by customers (cust.) in the Objective-column. Some problems require to serve all customers minimizing the travel time, waiting time, number of vehicles, or the deviation from time windows. This is represented by time. For these problems, rejections are not considered.

In real-world routing problems, future requests generally follow a stochastic spatial-temporal distribution. Some problem definitions and instances base on a geographical aggregation (Campbell 2006) to simplify the problem structure. In some work, The vast number of possible customer locations within the service area is simplified to a set of nodes in a graph model. This bears the
Table 1: Literature Classification

<table>
<thead>
<tr>
<th>Problem Setting</th>
<th>Solution Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>Representation</td>
</tr>
<tr>
<td>Psaraftis (1980)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Bertsimas and Van Ryzin (1991)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Tassiulas (1996)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Gendreau et al. (1999)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Swihart and Papastavrou (1999)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Ichoua et al. (2000)</td>
<td>time &amp; cust. distribution</td>
</tr>
<tr>
<td>Larsen et al. (2002)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Mitrović-Minić and Laporte (2004)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Thomas and White III (2004)</td>
<td>time &amp; cust. graph</td>
</tr>
<tr>
<td>Bent and Van Hentenryck (2004)</td>
<td>time &amp; cust. distribution</td>
</tr>
<tr>
<td>Van Hemert and La Poutré (2004)</td>
<td>customers graph</td>
</tr>
<tr>
<td>Branke et al. (2005)</td>
<td>customers distribution</td>
</tr>
<tr>
<td>Ichoua et al. (2006)</td>
<td>time &amp; cust. distribution</td>
</tr>
<tr>
<td>Gendreau et al. (2006)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Hvattum et al. (2006)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Thomas and White III (2007)</td>
<td>time &amp; cust. graph</td>
</tr>
<tr>
<td>Thomas (2007)</td>
<td>customers graph</td>
</tr>
<tr>
<td>Ghiani et al. (2009)</td>
<td>time distribution</td>
</tr>
<tr>
<td>Meisel (2011)</td>
<td>customers graph</td>
</tr>
<tr>
<td>Ghiani et al. (2012)</td>
<td>customers graph</td>
</tr>
<tr>
<td>Ulmer et al. (2015)</td>
<td>customers distribution</td>
</tr>
</tbody>
</table>

| Anticipatory Time Budgeting | customers | distribution | e | e | ✓ |

risk of a discrepancy between (aggregated) decision and practical implementation and, therefore, of inefficient solutions (Ulmer and Mattfeld 2013). The Representation-column of Table 1 indicates the customer representation. If potential customers are represented by a graph, the maximal number of nodes in the considered instances is shown in the Nodes-column.

Solution approaches often aim at the provision of sufficient coverage of the service area to efficiently include new requests or on anticipatory subset selection to maintain free time budget for future requests. We classify the applied solution approaches regarding their impacts on area coverage and subset selection. We differentiate between implicit (i) and explicit (e) impacts. If an approach aims at achieving sufficient area coverage to potentially serve many future requests, the coverage is considered explicitly. If the approach actively decides about the subset to confirm allowing the rejection of feasible requests, it is called explicitly regarding the subset selection.
Notably, approaches can consider both features explicitly or both implicitly.

We review the approaches regarding their anticipation of future requests, i.e., whether the approaches incorporate stochastic information about possible future requests in current decision making. Anticipatory approaches are indicated by a checkmark (✓) in the Anticipation-column.

The initial work on stochastic customer requests analyzes routing heuristics to reduce the expected travel times. First come, first serve-policies are applied by Psaraftis (1980), Bertsimas and Van Ryzin (1991), Swihart and Papastavrou (1999), and Larsen et al. (2002). Tassiulas (1996) partitions the service region and subsequently serves the subareas. Gendreau et al. (1999, 2006) combine tabu search and an adaptive memory with a rolling horizon algorithm to dispatch customer requests to a fleet of vehicles. Ichoua et al. (2000) decide about routing given time windows and a time budget. Other approaches are waiting strategies (e.g., wait at start, wait at end), for instance applied by Mitrović-Minić and Laporte (2004). These approaches do not consider stochastic information about future requests in decision making and are therefore not anticipatory.

Anticipatory approaches can be divided in offline and online approaches (Ritzinger et al. 2015). Offline approaches do not require extensive computational effort within the execution of the algorithm, but achieve a decision policy within an offline learning phase. Offline algorithms are for example applied by Meisel (2011) using methods of approximate dynamic programming to evaluate post-decision states. Offline algorithms in vehicle routing generally suffer from the post-decision state space dimensionality because the values of post-decision states are stored for the execution phase (Powell 2011). For this storage, a suitable state space representation is required. This representation needs to be of low dimensionality while it simultaneously needs to incorporate the distinguishing characteristics impacting the value of a post-decisions state. To this end, Meisel (2011) integrates information about every potential customer location in the state space representation. Hence, the representation’s size exponentially increases with every additional customer location. Meisel (2011) is only able to apply the approaches for 49 possible customer locations.

Online anticipation is mainly achieved by sampling approaches. Within the execution phase, sampling approaches simulate a set of future events to evaluate current decisions. Sampling allows a more detailed consideration of future events, but requires significant computational effort within the execution phase. To anticipate stochastic customer requests in vehicle routing, future customer requests are sampled according to the spatial distribution or the graph. These requests are used to
evaluate different routes and decisions. Bent and Van Hentenryck (2003, 2004) introduce a sample-scenario planning approach, where future customer requests are sampled and integrated in plans containing a set of routes. This approach is also used by Flatberg et al. (2007) and Sungur et al. (2010). Ghiani et al. (2009) use short term sampling for a dynamic pick-up and delivery problem. Hvattum et al. (2006) approach a real-world case study with sampling. They use historical data of customer requests to minimize the expected travel time. Online sampling approaches consider subset selection only implicitly. Explicit subset selection would lead to an exponential increase of the already high calculation times and even to computational intractability (Powell and Ryzhov 2012, p. 203ff).

Thomas and White III (2004), Van Hemert and La Poutré (2004), Branke et al. (2005), Ichoua et al. (2006), Thomas and White III (2007), and Thomas (2007) propose waiting heuristics including information about future requests to achieve explicit consideration of the area coverage. These heuristics select the routing, waiting locations, and waiting times according to potential future customers. Ghiani et al. (2012) introduce an anticipatory insertion waiting approach (AI), where waiting considers the center of gravity (COG) of the potential and feasible future requests. Ghiani et al. (2012) compared AI with the sample-scenario planning approach by Bent and Van Hentenryck (2004) and achieved similar results, even though the sampling approach requires high computational effort. Therefore, we draw on AI to benchmark ATB in our computational evaluation.

The majority of the approaches aims at providing a sufficient coverage and only decides about the subset to confirm implicitly. Generally, the largest feasible subset is included in the tour. Meisel (2011) and Ulmer et al. (2015) explicitly decide about confirmations. For the VRPSSR, Ulmer et al. (2015) propose a cost benefit heuristic (CBH) comparing the required time to insert a subset with the relative gain of confirmations. If the ratio exceeds a global threshold, the candidate subset is rejected. The threshold is determined a-priori via simulation. The area coverage is considered only implicitly by Ulmer et al. (2015). To the best of our knowledge, CBH is the only approach allowing explicit subset selection for problem instances of real-world size. Therefore, we use CBH as a benchmark heuristic.

With ATB, we introduce the first offline approach evaluating post-decision states regarding their expected future confirmations, explicitly considering both the subset selection and the area coverage for problem settings of real-world size.
Table 2: Problem Notations and MDP Components

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time horizon</td>
<td>$T = [0, \ldots, t_{\text{max}}]$</td>
</tr>
<tr>
<td>Service area</td>
<td>$A$</td>
</tr>
<tr>
<td>Depot</td>
<td>$D$</td>
</tr>
<tr>
<td>Vehicle speed</td>
<td>$v$</td>
</tr>
<tr>
<td>Overall set of realizations</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>Realization</td>
<td>$\omega \in \Omega$</td>
</tr>
<tr>
<td>Customers of realization $\omega$</td>
<td>$C^\omega = {C_1^\omega, \ldots, C_H^\omega}$</td>
</tr>
<tr>
<td>Early request customers</td>
<td>$C_0^\omega$</td>
</tr>
<tr>
<td>Late request customers of a realization</td>
<td>$C_+^\omega$</td>
</tr>
<tr>
<td>Spatial-temporal probability distribution</td>
<td>$\Xi$</td>
</tr>
<tr>
<td>Request time of customer $C_i^\omega \in C^\omega$</td>
<td>$t_i \in T$</td>
</tr>
<tr>
<td>Location of customer $C_i^\omega \in C^\omega$</td>
<td>$f(C_i^\omega) \in A$</td>
</tr>
<tr>
<td>Travel time between customers $C_i^\omega, C_j^\omega \in C^\omega$</td>
<td>$d(f(C_i^\omega), f(C_j^\omega))$</td>
</tr>
</tbody>
</table>

Decision points: $k = 0, \ldots, K$
Initial state: $S_0$
State in decision point $k$: $S_k$
Decisions available in decision point $k$: $X(S_k)$
Post-decision state in decision point $k$: $S_k^x$
Point of time in decision point $k$: $t(k)$
Vehicle position in decision point $k$: $P_k$
Customers to serve in decision point $k$: $C_k$
Customer requests in decision point $k$: $C_k^c = \omega_{k+1} \subset \omega$
Confirmed requests in decision point $k$: $C_k^c$
Reward in decision point $k$: $\mathcal{R}_k$
Next customer to visit: $C_{k+1}$
Planned tour in decision point $k$, given decision $x$: $\Theta_k^x$
Tour duration $\Theta_k^x$: $d(\Theta_k^x)$
Free time budget assuming $\Theta_k^x$: $b^\prime(k)$

3 The VRPSSR

In the following, we present the formulation for the VRPSSR and model the problem as a Markov decision process (MDP, Puterman 2014).

3.1 Problem Formulation

The notation for the VRPSSR is depicted in the upper part of Table 2. In the lower part of Table 2, the terminology required for the Markov decision process is defined. Let $T = [0, 1, \ldots, t_{\text{max}}]$
be the time horizon with time limit $t_{\text{max}}$. Let $A$ be the service area. The vehicle starts and ends its tour in a depot $D \in A$. The vehicle travels with a monotone speed $v$. In the following, we describe a stochastic problem realization $\omega \in \Omega$ of the overall set of realizations $\Omega$. Notably, in the decision making process, the customers are unknown before their request time. A problem realization $\omega \in \Omega$ consists of a set of customers $C^\omega = \{C_1^\omega, \ldots, C_h^\omega\}$, according request times $t_i \in T$, and locations $f(C_i^\omega) \in A, \forall C_i^\omega \in C^\omega$ as depicted in Equation (1):

$$\omega = \{(C_1^\omega, t_1^\omega, f(C_1^\omega)), \ldots, (C_h^\omega, t_h^\omega, f(C_h^\omega))\}. \quad (1)$$

That is, each customer $C_i^\omega \in C^\omega$ is associated with vector $(t_i^\omega, f(C_i^\omega))$ consisting of a request time $t_i^\omega \in T$ and a location $f(C_i^\omega) \in A$. For each customer $C_i^\omega \in E^\omega$, the request time $t_i^\omega$ and location $f(C_i^\omega)$ is generated from a spatial and temporal probability distribution $(t_i^\omega, f(C_i^\omega)) \sim \Xi : T \times A \rightarrow [0, 1]$. The travel time between two customers is defined by $d(f(C_1), f(C_2))$.

The customers are divided in two temporal classes as shown in Equation (2); Early request customers $C_0^\omega = \{C_i^\omega \in C^\omega : t_i^\omega = 0\}$ requesting service in $t = 0$ and late request customers $C_+^\omega = \{C_i^\omega \in C^\omega : t_i^\omega > 0\}$ requesting service in $t > 0$. The customers $C_0^\omega$ are known in the beginning and must be served:

$$C^\omega = C_0^\omega \cup C_+^\omega = \{C_0^\omega, \ldots, C_m^\omega\} \cup \{C_{m+1}^\omega, \ldots, C_h^\omega\}. \quad (2)$$

For a problem realization $\omega \in \Omega$, the dispatcher aims at maximizing the number of confirmed late request customers. In unlikely cases, where the tour duration to serve all ERCs already exceeds the time limit, all ERCs are served and no confirmation of dynamic requests is allowed.

### 3.2 Markov Decision Process

To formally describe the decision-making process, we model the VRPSSR as a Markov decision process. We further formally define the free time budget $b^\ast(k)$ later utilized in ATB.

The required terminology is depicted in the lower part of Table 2. In an MDP, in each decision point $k$, a state $S_k$ of a finite state space $S = \{S_0, \ldots, S_q\}$ and a subset of possible decisions $X(S_k) \subseteq X = \{x_1, \ldots, x_r\}$ depending on state $S_k$ is given. Each decision $x \in X(S_k)$ in state $S_k \in S$ generates an immediate reward $R : S \times X(S_k) \rightarrow \mathbb{R}$. The outcome of each combination
\[(S_k, x) \in S \times X(S_k)\] is known with certainty and is defined as the post-decision state \[S^* k \in P = S \times X.\] \[P\] denotes the post-decision state space. Given a post-decision state, a transition leads to a subsequent state \[S_{k+1} = (S^* k, \omega_k).\] \[S_{k+1}\] is determined by realization \[\omega_k : P \rightarrow S\] of the set of problem realizations \[\omega_k \in \Omega.\]

For the VRPSSR, the initial state \[S^0 0\] is defined by the set of ERCs \[C^0 0\] of the realization \[\omega.\] A decision point occurs when the vehicle is located at a customer or initially at the depot. A state at decision point \[k\] consists of the point of time \[t(k) \in T,\] the vehicle’s position \[P^k \in A,\] and a set of customers to visit \[C^\omega k = C^0 0(k) \cup C^\omega + (k)\] containing the not-yet-served subset of ERCs \[C^0 0(k) \subseteq C^0\] and the not-yet-served subset of confirmed LRCs \[C^\omega + (k) \subseteq C^\omega.\] Additionally for \[k > 0,\] a set of requests \[\omega_{k-1} = C^\omega k(k) \subseteq C^\omega\] occurred between the last decision point \[k - 1\] and decision point \[k\] is given.

For the VRPSSR, decisions contain both a confirmation and a movement action. Decisions \[X(S_k)\] contain the confirmation action selecting the subset \[C^\omega c(k) \subseteq C^\omega c(k)\] to confirm. As a movement action, the next customer \[C^k_{\text{next}} \in C^0 0(k) \cup C^\omega c(k) \cup C^\omega + (k) \cup \{P^k\}\] to visit is selected. If \[C^k_{\text{next}} = P^k,\] the vehicle idles at its current location for one time step \[\bar{t}.\] The idle time can be extended to \[w\] time steps by repeatedly setting \[C^k_{\text{next}} = P^k + j\] for \[0 \leq j < w.\] The immediate reward \[R(S_k, x) = |C^\omega + (k)|\] of a decision \[x\] given state \[S_k\] is defined by the cardinality of the confirmed subset of requests.

A decision \[x\] is feasible if there exists at least one feasible tour \[\Theta^x k = (\theta_1, \ldots, \theta_o)\] starting at the vehicle’s position \[\theta_1 = P^k,\] traveling to the next customer \[\theta_2 = C^\omega_{\text{next}},\] including all remaining confirmed late request and early request customers, and ending at the depot \[\theta_o = D.\] The tour duration of \[\Theta^x k\] is defined as \[d(\Theta^x k)\] in Equation (3).

\[
\bar{d}(\Theta^x k) = \sum_{i=1}^{o-1} d(f(\theta_i), f(\theta_{i+1})) \tag{3}
\]

A planned tour \[\Theta^x k\] is feasible if \[\bar{d}(\Theta^x k) \leq t_{\text{max}} - t(k),\] i.e., the tour duration of \[\Theta^x k\] allows returning to the depot within the time limit \[t_{\text{max}}.\] The free time budget \[b^x(k)\] of \[\Theta^x k\] is then calculated as shown in Equation (4):

\[
b^x(k) = t_{\text{max}} - t(k) - \bar{d}(\Theta^x k) \tag{4}
\]
A decision $x$ results in a post-decision state $S^x_k$ containing the time $t(k)$, the vehicle’s location $P_k$, the next customer to visit $C^k_{\text{next}}$, and a set of remaining customers $C^c_k \cup C^\omega(k)$. The travel to the next customer and realization $C^r_{k+1}$ lead to a transition to decision point $k+1$. State $S_{k+1}$ consists of position $P_{k+1} = C^k_{\text{next}}$, time $t(k+1) = t(k) + d(f(P_k), f(C^k_{\text{next}}))$, and the set of customers $C^\omega_{k+1} = C^\omega_k \cup C^\omega_c(k) \setminus C^k_{\text{next}}$. Further, $\omega_{k+1}$ provides a set of new requests $\omega_{k+1} = C^\omega_r(k+1) = \{C^\omega_i \in C^\omega_+ : t(k) < t_i \leq t_{k+1}\}$. Since customer requests are stochastic, $C^\omega_r$ is only revealed at decision point $k+1$ and unknown before. The overall realization $\omega \in \Omega$ is split regarding the time of the decision points as depicted in Equation (5):

$$\omega = \{(C_1, t_1, f(C^r_1)), (C_2, t_2, f(C^r_2)), (C_3, t_3, f(C^r_3)), \ldots, (C_{n-1}, t_{n-1}, f(C^\omega_{n-1})), (C_n, t_n, f(C^\omega_n))\}.$$ (5)

The process terminates in state $S_K$, when the vehicle has returned to the depot, i.e., $C^r_K = \emptyset$ and $P_K = D$.

In Figure 1, an exemplary state $S_k$ is shown. In $t = 60$, a set of three customers $C^\omega_k = \{C_1, C_2, C_3\}$ must be served. Two new requests $C^r_{\omega}(k) = \{C^r_1, C^r_2\}$ are given. Decisions consist of four confirmation actions and up to six movement actions. The applied decision $x$ consists of confirmation action $C^c(k) = \{C^\gamma_1\}$, i.e., to accept request $C^\gamma_1$ and to reject request $C^\gamma_2$. This leads to the confirmation of one customer and an immediate reward of $R_k(S_k, x) = 1$. The next customer to visit $C^k_{\text{next}}$ is set to $C^\gamma_1$. The resulting post-decision state is depicted on the right side of Figure 1. $S^x_k$ contains four customers, the next customer to visit is depicted by the bold line. The dashed line...
indicates a feasible tour $\Theta^x_k$. If we assume for this example the tour duration to be $d(\Theta^x_k) = 200$ and a time limit of $t_{\text{max}} = 360$ minutes, the resulting free time budget is $b^x(k) = 360 - 60 - 200 = 100$ minutes.

Since the customer requests are stochastic, the number of confirmations is a random variable. The objective for this stochastic dynamic decision problem is to identify an optimal decision policy $\pi^* \in \Pi$ that maximizes the expected number of confirmations beginning from an initial state $S_0$. A policy $\pi : S \rightarrow X$ is a sequence of decision rules $(X^\pi_0, \ldots, X^\pi_K)$ to assign every state $S_k \in S$ to a decision $X^\pi_k(S_k) \in \mathcal{X}(S_k)$. $X^\pi_k(S_k)$ is the decision rule dependent on state $S_k$ induced by $\pi$ in decision point $k$. An optimal policy $\pi^*$ maximizes the expected rewards over all decision points subsequently applying $\pi^*$ as depicted in Equation (6):

$$\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{k=0}^K R(X^\pi_k(S_k)) | S_0 \right].$$  \hspace{1cm} (6)

For every state $S_k$, $\pi^*$ holds the Bellman equation as depicted in Equation (7). In every decision point $k$, $\pi^*$ selects the decision maximizing the sum of immediate reward and the expected future rewards, as depicted by the Bellman equation in Equation (7). The second term of Equation (7) is called the value $V(S^x_k)$ of a post-decision state $S^x_k$.

$$X^\pi^*_k(S_k) = \arg \max_{x \in \mathcal{X}(S_k)} \left\{ R(S_k, x) + \mathbb{E} \left[ \sum_{j=k+1}^K R(S_j, X^\pi^*_j(S_j)) | S_k \right] \right\}$$ \hspace{1cm} (7)

4 Anticipatory Time Budgeting

In this section, we describe how the values for a state can be approximated via approximate value iteration (AVI, Powell 2011, p. 127ff), a method of ADP. We then motivate and define a suitable state space aggregation ATB required for AVI. ATB aims on anticipatory confirmation decisions. To determine the movement action, ATB draws cheapest insertion routing (Rosenkrantz et al. 1974, CI) as described in §A.1.
4.1 Approximate Value Iteration

In this section, we depict the correlations between values $V$ and policies $\pi$ and show how values are approximated via AVI.

The expected future rewards are represented by the value of a post-decision state and depend on the applied policy. For the VRPSSR, the value reflects the expected number of future confirmations. The value for any given policy $\pi$ can be recursively calculated applying decision rule $X_j^\pi$ in all subsequent decision points $j = k + 1, \ldots, K - 1$ for all possible transitions given a post-decision state $S_k^x$. The achieved value $V^\pi(S_k^x)$ of the regarding post-decision state depending on $\pi$ as depicted in Equation (8):

$$V^\pi(S_k^x) = \mathbb{E} \left[ \sum_{j=k+1}^{K} R(S_j, X_j^\pi(S_j)) | S_k^x \right]. \quad (8)$$

As shown in Equation (7), the decision rule of an optimal policy $\pi^*$ maximizes the sum of the immediate reward and the value of the post-decision state. Equation (7) can be reformulated as depicted in Equation (9):

$$X_k^\pi(S_k) := \arg \max_{x \in X(S_k)} \left\{ R(S_k, x) + V^\pi(S_k^x) \right\}. \quad (9)$$

Conversely, the assignment of specific values to all post-decision states defines a decision policy if Equation (9) is applied. Instead of directly approximating a policy $\pi^\alpha$ to the optimal policy $\pi^*$, AVI aims at approximating the values $V^\pi$ to the expected number of future confirmations of an optimal policy $V^\pi$.

The procedure of AVI is depicted in Figure 2. AVI starts with initial values $V^{\pi_0}$ inducing a policy $\pi_0$. Then, AVI subsequently simulates a subset of the problem realizations $\tilde{\Omega} \subset \Omega$. Within each simulation run $i$, for the given realization $\omega^i \in \tilde{\Omega}$, policy $\pi_{i-1}$ is applied according to the Bellman equation drawing on the current values $V^{\pi_{i-1}}$. If $V^{\pi_{i-1}}$ is applied in a state, this is called exploitation of the approximated values. Still, decisions may be selected randomly in some states to avoid local optima and to force an exploration of the state space. After each simulation run $i$, the values are updated with the realized confirmations $V_{\omega}^\pi(S_k^x)$ according to Equation (10). Parameter $\alpha$ defines the step size of the approximation process.
Figure 2: Approximate Value Iteration (Soeffker et al. 2016)

\[ V^{\pi_i}(S^x_k) = (1 - \alpha)V^{\pi_i-1}(S^x_k) + \alpha V_\omega(S^x_k) \] (10)

Subsequently, the values \( V^{\pi_i} \) and policy \( \pi_i \) approximate the values of an optimal policy \( V^{\pi^*} \) and the optimal policy \( \pi^* \) respectively. The final policy \( \pi_{|\Omega|} \) is then applied in the execution phase. The advantage of AVI is that extensive (offline) simulation can be conducted. This enables explicit online subset selections, since the evaluation of the subsets is already integrated in the approximated values. To allow storage of the values for the large number of post-decision states, we present an aggregated state space representation of reasonable size. We aggregate the state space to a two-dimensional vector space. AVI approximates the values for these vectors instead of values for particular states. In the following, we motivate and describe the procedure of aggregation.

4.2 Aggregation

In this section, we motivate the selection of point of time \( t(k) \) and free time budget \( b^\gamma(k) \) as indicators for the values \( V(S^x_k) \) and present the according aggregation \( \mathfrak{A} \) mapping post-decision states to two-dimensional vectors \( (t(k), b^\gamma(k)) \). In the following motivational consideration, we denote the figures simplified as \( t, b, \) and \( V \). We further utilize the concept of the insertion time \( \gamma \), i.e., the required time in future decision points to insert a new request and the coverage \( \kappa \) of the service area, i.e., the percentage of the service area covered by the planned tour \( \Theta^x_k \) (Branke et al. 2005).
Figure 3 shows the influencing factors on $V$, $\gamma$, $\kappa$, and $b$ and their dependencies for a specific $t$. The influence of a factor is indicated by an arrow. If the arrow is assigned to a plus symbol, an increase in the influencing factor results in an increase in the influenced factor. The minus symbol indicates an opposed effect. The gray arrow assigned with an $x$ reflects the position the dispatcher is able to control the dynamics. The dispatcher decides about the requests to confirm and the according routing. The applied decision results then in an adapted tour $\Theta$ and a tour duration $\bar{d}$. $\bar{d}$ is correlated to the coverage $\kappa$. $\kappa$ influences the insertion time $\gamma$. Hence, $b$ and $\gamma$ are dependent on $\bar{d}$. If $\bar{d}$ is high, $\gamma$ and $b$ are low. Notably, $t$ itself is another substantial influencing factor to $V$. The earlier $t$, the more customers will request in the future and the higher $V$ may get. Point of time $t$ influences the factors and the fortitude of the depicted dependencies in Figure 3 significantly.

In Figure 4, we show in idealized and simplified terms how $V$ is influenced by $b$ for a specific $t$. Time budget $b$ is depicted on the x-axis varying between $0 \leq b \leq 0.4$ of the overall shift duration. Budget $b = 0$ indicates that all of the time budget is already consumed by previous confirmations depicted on the right y-axis. As discussed, the insertion time $\gamma$ for future requests, depicted on the left y-axis, directly depends on $b$. The higher $b$, the higher is $\gamma$. Notably, $\gamma$ differs over the remaining points of time and is dependent on the applied policy. The depiction in Figure 4 is a simplification. The value $V$ depends on $b$ and $\gamma$. Given $b = 0$, $\gamma$ is low, but no free time budget remains to insert a request. Given a high $b$, $\gamma$ increases significantly and $V$ stagnates. The
dispatcher aims on maximizing the overall number of confirmations. If $V$ is provided for every vector $(t, b)$ in every decision situation, an anticipatory decision can be selected maximizing the sum of immediate confirmations and $V$ by means of the Bellman equation.

Based on the aforementioned considerations, ATB applies an aggregation $\mathfrak{A} : \mathcal{P} \rightarrow \mathcal{Q} \subseteq \mathbb{N}^2$ to represent post-decision states $\mathfrak{A}(S^x_k) = p^k$ by 2-dimensional vectors $p^k = (t(k), b^x(k)) \in \mathcal{Q}$. Each vector in $\mathcal{Q}$ only contains the numerical parameters point of time and free time budget. The resulting representation $\mathcal{Q}$ is defined in Equation (11):

$$\mathcal{Q} = \{\mathfrak{A}(S^x_k) : S^x_k \in \mathcal{P}\}. \quad (11)$$

The value of a post-decision state $S^x_k$ can now be represented by the value $\hat{V}^\pi$ of the vector $p^k$: $V^\pi(S^x_k) = \hat{V}^\pi(\mathfrak{A}(S^x_k)) = \hat{V}^\pi(p^k)$. The application of $\mathfrak{A}$ results in a significantly smaller vector space $\mathcal{Q}$. Since for the VRPSSR both parameters are discrete, $\mathcal{Q}$ can be associated with a 2-dimensional lookup table (LT) with dimensions $t, b \in \{0, \ldots, t_{\max}\}$.

If we assume a minute-by-minute representation of $T$, the cardinality of LT $\mathcal{Q}$ is still high. The number of entries is $|\mathcal{Q}| \leq \frac{1}{2}(t_{\max})^2$, since the sum of point of time $t(k)$ and free time budget $b^x(k)$ is smaller than or equal to $t_{\max} \geq t(k) + b^x(k)$. As it is shown by Ulmer et al. (2015), an approximation based on $\mathcal{Q}$ eventually results in sufficient solution quality. Nevertheless, the high number of entries requires extensive approximation with a high number of simulation runs. A

![Figure 4: Expected Number of Future Confirmations per Free Time Budget $b$](image)
minute-by-minute consideration of point of time and free time budget may not be necessary. Since the approximation quality depends on the number of observations, this detailed representation may even impede the approximation process and the solution quality as discussed in Barto (1998, p. 193). For a more efficient and effective approximation, ATB draws on a partitioned LT $\mathcal{E}$. $\mathcal{E}$ is defined by a partitioning $\mathcal{I} : \mathcal{Q} \rightarrow \mathcal{E}$ grouping vectors $p_1, \ldots, p_q \in \mathcal{Q}$ to a set represented by an LT-entry $\eta = \mathcal{I}(p_1) = \cdots = \mathcal{I}(p_q)$. In the following, we introduce the dynamic lookup table, a partitioning adapting to the approximation process.

5 Dynamic Lookup Table

In this section, we introduce the dynamic lookup table (DLT). We motivate the requirement for DLTs, depict the DLT functionality, and finally embed the DLT in the literature.

A priori defined, static LTs (SLTs), e.g., with equidistant interval lengths of the parameters, often impede the approximation process due to the tradeoff between reliability and level of detail. First, a frequent entry observation is mandatory to achieve a reliable and efficient approximation. This is only provided by a partitioning with a limited number of entries (coarse-grained). Second, the partitioning must enable the differentiation between heterogeneous states for an effective approximation. Hence, the partitioning must allow to partition these states to different entries (fine-grained). For an exemplary analysis of this tradeoff, the interested reader is referred to §A.2 of the Appendix. Since for ATB, the vector space is two-dimensional, we describe the functionality of a two-dimensional DLT in this remainder of this section. Still, the DLT is not limited to two dimensions. The general algorithm for AVI combined with DLT and $\zeta \geq 1$ dimensions is presented in §A.3.

5.1 Functionality

The DLT meets the aforementioned tradeoff. To this end, the DLT adapts the partitioning $\mathcal{I}$ according to the approximation process. The DLT starts in a partitioning $\mathcal{I}^0$ and LT $\mathcal{E}^0$ with large intervals and subsequently decreases the interval lengths in some areas during the approximation process. In the beginning, a fast, coarse-grained approximation of the few initial entries is provided. During the approximation, the interval length is decreased for some “interesting” areas; that
is, an entry is separated in a set of new entries resulting in a dynamic partitioning $\mathcal{I}^j$ and LT $\mathcal{E}^j$ in simulation run $j$. The DLT allows a fine-grained approximation within these areas. For areas of no interest, the partitioning stays in the original design. An evolution of the DLT is exemplified in Figure 5. In the left, initial partitioning, the intervals are homogenous and coarse-grained. During the simulation runs, some areas are considered in more detail, as seen in the central partitioning. In the final partitioning, seen on the right, the lower left area is highly separated and considered in detail, while the large interval lengths of the initial partitioning in the upper right corner remain. Beside the advantage of a fast and effective approximation process, DLT may also allow to derive insights of the problem and solution structure. Areas with a high separation may indicate important areas or usual patterns for the problem. In the following, we describe how the DLT is adapted over the approximation process.

### 5.2 Entry Selection

An entry $\eta = \mathcal{I}^j(p)$ of partitioning $\mathcal{I}^j$ and DLT $\mathcal{E}^j$ can only be considered in more detail if a sufficient number of entry observations is given. Otherwise, the new entries may not be observed frequently enough to derive a reliable approximation. Therefore, we consider the number of observations $N(\eta)$ to decide whether we are able to consider $\eta$ in more detail. Frequently observed entries have an essential impact on the entire approximation process. The evaluation of those entries needs to be very accurate and a detailed consideration is desirable. A more detailed consideration of an entry is additionally required if states of heterogeneous value are grouped in a single entry. This is indicated by a high deviation within the entry’s value, represented by the standard
deviation $\sigma(\eta)$. 

To achieve a problem-agnostic method, we draw the relative values $N(\eta)/\bar{N}$ and $\sigma(\eta)/\bar{\sigma}$ to determine suitable entries. $\bar{N}$ and $\bar{\sigma}$ are the averages of all entries $\eta_i \in \mathcal{E}$. A relatively high number of observations is indicated by $N(\eta)/\bar{N} > 1$, a relatively high standard deviation by $\sigma(\eta)/\bar{\sigma} > 1$. A separation of an entry $\eta$ is conducted if Equation (12) is satisfied.

$$\frac{N(\eta)\sigma(\eta)}{\bar{N}\bar{\sigma}} \geq \tau \quad (12)$$

A multiplication of the two components allows deterministic entries without deviation to remain unseparated. Further, entries with only a few observations are not split to allow a reliable approximation. The tuning parameter $\tau$ defines the separation frequency of the DLT. A small $\tau$ results in a fast separation of many entries, a large $\tau$ only selects entries with outstanding characteristics in both number of observations and deviation. For comparison of the impact of $N$ and $\sigma$ to the approximation process, we define two DLT-approaches: DLT($N, \sigma$) considering both attributes and DLT($N$) only considering the number of observations in Equation (12). Preliminary tests have shown that DLT($\sigma$) only considering the standard deviation results in inferior approximation for this problem.

### 5.3 Update Algorithm

In the update process of $\mathcal{I}^i$, entries $\eta = \mathcal{I}^i(p)$ satisfying Equation (12) are separated to a set of four new entries $\eta_1 = \mathcal{I}^{i+1}_1(p), \ldots, \eta_4 = \mathcal{I}^{i+1}_4(p)$ by dividing both intervals in half. Values, observations, and deviations are transferred to the new entries and the former entry is replaced. Because of the lack of further knowledge about the distribution of the values and observations, the number of observations and the deviation are equally divided to the new entries and the values remain. It may be useful to start the update process after a few, e.g. 50, initial simulation runs to avoid premature separation.

### 5.4 Dynamic Partitioning in the Literature

To the best of our knowledge, a method of ADP dynamically partitioning a vector space with respect to the entry observations and value deviation has not been presented in the literature. Nev-
ertheless, the idea of dynamic state space partitioning is not new. A similar concept of dynamically changing the partitioning has been introduced by Bertsekas and Castaño (1989). For an infinite-horizon problem with $|S| = 100$ and a known transition probability matrix $P$, states are partitioned based on their residuals resulting in a faster approximation processes. In contrast to Bertsekas and Castaño (1989), the DLT can be applied to problems of large state spaces and unknown transition probability functions.

Since the DLT for the VRPSSR only consists of two dimensions, the resulting structure of the DLT can be associated with a quadtree (Finkel and Bentley 1974), a tree structure for the efficient storage of data. The concept of quadtrees was already combined with Q-learning by Clausen and Wechsler (2000). They propose to use different levels of detail for fractal image compression. Still, the main difference is that the DLT combines the characteristics of the quadtree with the AVI approximation process. For a quadtree, the focus lays on achieving an efficient structure to store given data. AVI focuses on approximating values given a partitioning. The DLT updates both structure and data (values) in tandem.

Finally, the DLT can be compared to the weighted LT (WLT) by George et al. (2008). Instead of adapting a partitioning, the WLT draws on a set of static partitionings. The value is the weighted sum of the according individual entries. The WLT adapts these weights with respect to the approximation process. This may accelerate and strengthen the approximation process but on the expense of significant memory consumption as we show in our computational evaluation in §6. As tests indicate, this memory consumption impedes the application of WLTs for vector-spaces of higher dimension (Ulmer et al. 2016).

6 Computational Evaluation

In this section, we first describe the set of test instances and tune the algorithm parameters. Then, we present the results of ATB and the benchmark heuristics for the defined instances. We show that ATB is superior regardless of the degree of dynamism and the service area size. Since ATB does not explicitly consider spatial information like vehicle and customer locations, we further show how the success of ATB depends on the customer distribution.
Table 3: Instance Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time limit</td>
<td>$t_{\text{max}} = 360 \text{ min}$</td>
</tr>
<tr>
<td>Expected number of customer</td>
<td>$n = 100$</td>
</tr>
<tr>
<td>Vehicle speed</td>
<td>$v = 25 \text{ km/h}$</td>
</tr>
<tr>
<td>Service area</td>
<td>$\mathcal{A}<em>{15}, \mathcal{A}</em>{20}$</td>
</tr>
<tr>
<td>Depot</td>
<td>Center of $\mathcal{A}$</td>
</tr>
<tr>
<td>Degree of Dynamism</td>
<td>0.50, 0.75</td>
</tr>
<tr>
<td>Spatial distribution</td>
<td>$\mathcal{F}<em>U, \mathcal{F}</em>{2C}, \mathcal{F}_{3C}$</td>
</tr>
</tbody>
</table>

### 6.1 Instances

The set of the instances are derived from Bent and Van Hentenryck (2004), Hvattum et al. (2006), Thomas (2007), and Meisel (2011). The instance parameters are listed in Table 3. For a detailed depiction of the instance generation process, the interested reader is referred to §A.6 in the Appendix.

To analyze the impact of the customer distribution, we generate three different customer distributions varying in their requirement for considering spatial information. The first distribution $\mathcal{F}_U$ provides uniformly distributed customer requests. We assume that spatial information may be secondary for $\mathcal{F}_U$ and ATB allows a generally anticipatory budgeting of the time. For the other two distributions $\mathcal{F}_{2C}$ and $\mathcal{F}_{3C}$, customers are accumulated in clusters. An example with 50 realized customers for these distributions is shown in Figure 6. The distribution $\mathcal{F}_{2C}$ provides customer requests in two clusters. The request distribution is symmetrical to the depot located in the center of the service area. We assume that for $\mathcal{F}_{2C}$ the dependency between point of time, free time budget, and area coverage is strong, since the vehicle’s tour may always follow a similar behavior over time due to the distribution’s symmetry. This behavior especially manifests in the point(s) of time the vehicle changes clusters. We assume that ATB may perform well for $\mathcal{F}_{2C}$. We further examine the policies for $\mathcal{F}_{3C}$, where three clusters are heterogeneously distributed in the service area. For this distribution, the sequence of cluster visits and the times of cluster changes may differ. As a result, the area coverage is not strongly represented in the temporal attributes of ATB and the solution quality of ATB may be impeded. For $\mathcal{F}_{3C}$, the consideration of spatial information, e.g. the
vehicle’s location or the sequence of cluster visits, may be necessary for high quality anticipation. The results of the test runs later confirm our assumptions.

6.2 Benchmark Heuristics and Tuning

We compare ATB to anticipatory insertion (AI) by Ghiani et al. (2012) and cost benefit (CBH) by Ulmer et al. (2015). We select AI and CBH because they both focus on a single aspect in their anticipation, routing and confirmations respectively. AI waits close to the center of gravity defined by the locations of potential future requests. AI focuses on anticipatory coverage of the service area without explicit subset selection. CBH compares the rewards of a decision with the resulting consumption of the free time budget. CBH decides explicitly about subset selection neglecting the consideration of area coverage. For definition and tuning of AI and CBH, the interested reader is referred to §A.4. We compare the DLT to static lookup tables (SLTs) and a weighted lookup table (WLT) by George et al. (2008). The SLT partitioned each dimension of the vector space in equidistant intervals. The WLT draws on a set of SLTs, updates the SLT entries and adapts the weighting of the entries per SLT to the approximation process. For definition of SLT and WLT, the interested reader is referred to §A.5.

In the following, we describe the tuning for DLT, SLT, and WLT as well as AVI. We consider \( L = 5 \) different levels of interval lengths, starting with intervals of length 16 \((l = 5)\) up to the

![Figure 6: Exemplary Realization of \( F_{2C} \) and \( F_{3C} \) for \( A_{20} \)]
discrete representation of \( l = 1 \). The point of time and time budget consideration varies from 16 minutes down to minute by minute. We test lookup tables with static interval length \( \text{SLT}(I) \) for all levels \( I = 2^{l-1}, l = 1, \ldots, 5 \). For the weighted and dynamic LTs, we consider all five partitioning levels \( (l = 5 \text{ up to } l = 1) \). The initial partitioning of the DLT is \( \text{SLT}(16) \). We apply DLTs with \( \tau = 1.0, 1.25, \ldots, 2.0 \). For \( \tau = 1.0 \), an entry is separated if its divergence in observations and standard deviation is above average. For \( \tau = 2.0 \), the divergence needs to be two times as high.

We select high initial values of \( \hat{V}_0 = 1,000 \) for every entry. This results in the selection of not yet observed entries and forces exploration in the beginning of the approximation process. Since the values are equal, the ATB-policy with \( \hat{V}_0 \) can be associated with a myopic policy always selecting the largest feasible subset of requests. We update the values at the end of each simulation run to the moving average, i.e., \( \alpha = \frac{1}{N(\theta)} \). For all LTs, we run \( |\hat{\Omega}| = 1 \) million simulation runs. To avoid premature separation, we prohibit updates of the DLT in the first 50 simulation runs. For every tuning of DLT\( (N, \sigma) \) and SLT, we run 10,000 test runs to determine the best tuning for the particular instance setting. The according policies are then used for the evaluation.

### 6.3 Solution Quality

For every instance setting, we run 10,000 test runs. The results for ATB drawing on DLT\( (N, \sigma) \) with best tuning and the benchmark heuristics are shown in Table 4.

Comparing the benchmark heuristics, CBH outperforms AI for every instance setting. The explicit subset selection of CBH is advantageous compared to the explicit area coverage consideration of AI. Especially for instances with \( \mathcal{F}_U \) and \( \mathcal{A}_{20} \), the difference is high since the explicit subset selection avoids customers far-off the current tour. ATB enables both subset selection and consideration of area coverage due to the aforementioned dependencies of point of time, free time budget, and area coverage, even though ATB does not explicitly include spatial information in the evaluation of decisions. To analyze the ability of ATB to consider the area coverage, we analyze the improvement of ATB compared to CBH.

ATB provides the best solution quality for nearly every instance. ATB performs well for the distributions \( \mathcal{F}_U \) and \( \mathcal{F}_{2C} \). Even though the solution quality in \( \mathcal{F}_{2C} \) is already high for CBH, ATB allows significant improvements up to 4.3% given \( \text{dod} = 0.75 \). The distribution \( \mathcal{F}_{2C} \) is symmetrical. Therefore, the cluster sequence is not relevant for ATB. As a result, the dependency
Table 4: Solution Quality and Comparison to CBH (in %)

<table>
<thead>
<tr>
<th>Service area</th>
<th>( A_{15} )</th>
<th>( A_{20} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>( F_U )</td>
<td>( F_{2C} )</td>
</tr>
<tr>
<td>ATB</td>
<td>58.5</td>
<td>82.2</td>
</tr>
<tr>
<td>AI</td>
<td>51.0</td>
<td>81.9</td>
</tr>
<tr>
<td>CBH</td>
<td>55.4</td>
<td>82.0</td>
</tr>
<tr>
<td>Improvement ATB to CBH (in %)</td>
<td>5.6</td>
<td>0.2</td>
</tr>
</tbody>
</table>

\( dod = 0.50 \)

<table>
<thead>
<tr>
<th>Service area</th>
<th>( A_{15} )</th>
<th>( A_{20} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td>( F_U )</td>
<td>( F_{2C} )</td>
</tr>
<tr>
<td>ATB</td>
<td>59.6</td>
<td>78.6</td>
</tr>
<tr>
<td>AI</td>
<td>53.5</td>
<td>77.7</td>
</tr>
<tr>
<td>CBH</td>
<td>57.1</td>
<td>77.7</td>
</tr>
<tr>
<td>Improvement ATB to CBH (in %)</td>
<td>4.4</td>
<td>1.2</td>
</tr>
</tbody>
</table>

\( dod = 0.75 \)

between the temporal parameters and the area coverage is strong and ATB is able to achieve high solution quality without explicitly considering spatial information for this customer distribution. In the instances with \( A_{15} \) and \( F_{3C} \), the solution quality of ATB is inferior to CBH. This results from the varying sequences of visited clusters given \( F_{3C} \). The future rewards depend on the clusters the vehicle already has visited and the clusters the vehicle will visit in the remaining time of the shift, i.e., the coverage. This dependency varies over the realizations. Since spatial information is not considered by ATB, the solution quality is impeded for \( F_{3C} \). We justify these assumptions in the following analysis.

7 Analysis

In this section, we analyze the DLT and the ATB algorithm. The analysis contains three parts. First, we depict the effectiveness and efficiency of the approximation process for partitioning approach DLT compared to WLT and SLT in §7.1. We show how the DLT is able to adapt to the problem and highlight the advantageous in approximation speed with respect to SLT as well as the significantly
more efficient entry storage compared to WLT. In \S7.2, we analyze the routing behavior of ATB and show how ATB is able to provide high quality solutions for $F_{2C}$ in contrast to $F_{3C}$. In \S7.3, we finally compare the dependencies of tour duration $\hat{d}$, insertion time $\gamma$, point of time $t$, free time budget $b$, and expected values $V$ to the analytical considerations made in \S4.2. We show that for $F_U$ the assumptions hold while for $F_{2C}$ the dependencies significantly differ. In the analysis, we focus on instances with $dod = 0.75$ and $A_{20}$.

### 7.1 Partitioning: Dynamic Lookup Table

The in \S6 depicted results are achieved for ATB with DLT($N, \sigma$). In this section, we compare these results with ATB based on the partitioning approaches DLT($N$), SLT, and WLT. DLT($N$) neglect the value deviation within the entries. An SLT is a single, static partitioning. The solution quality significantly depends on the a priori selection of the SLT. The WLT draws on a set of static partitionings. Hence, WLT may provide a high quality but on the expense of substantial storage requirement. We show how DLTs outperforms SLTs in solution quality and approximation speed and WLT regarding the required number of entries and storage efficiency.

We first compare the achieved solution quality regarding the partitioning approaches depicted
Table 5: Comparison of Partitioning Approaches

<table>
<thead>
<tr>
<th>Service area</th>
<th>Distribution</th>
<th>$\mathcal{A}_{15}$</th>
<th>$\mathcal{A}_{20}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mathcal{F}_U$</td>
<td>$\mathcal{F}_{2C}$</td>
<td>$\mathcal{F}_{SC}$</td>
</tr>
<tr>
<td>Served Dynamic Requests (in %)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$dod = 0.50$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLT</td>
<td>+</td>
<td>58.4</td>
<td>81.9</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>54.2</td>
<td>79.1</td>
</tr>
<tr>
<td>DLT($N, \sigma$)</td>
<td>+</td>
<td>58.5</td>
<td>82.2</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>58.4</td>
<td>81.4</td>
</tr>
<tr>
<td>DLT($N$)</td>
<td>+</td>
<td>58.4</td>
<td>81.2</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>58.3</td>
<td>81.2</td>
</tr>
<tr>
<td>WLT</td>
<td></td>
<td>58.4</td>
<td>82.2</td>
</tr>
<tr>
<td>$dod = 0.75$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLT</td>
<td>+</td>
<td>59.8</td>
<td>77.6</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>57.2</td>
<td>74.6</td>
</tr>
<tr>
<td>DLT($N, \sigma$)</td>
<td>+</td>
<td>59.6</td>
<td>78.6</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>59.5</td>
<td>77.4</td>
</tr>
<tr>
<td>DLT($N$)</td>
<td>+</td>
<td>59.8</td>
<td>77.9</td>
</tr>
<tr>
<td></td>
<td>$\varnothing$</td>
<td>59.7</td>
<td>77.5</td>
</tr>
<tr>
<td>WLT</td>
<td></td>
<td>59.0</td>
<td>77.9</td>
</tr>
<tr>
<td>Number of Entries (in 1,000)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$dod = 0.50$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLT($N, \sigma$)</td>
<td></td>
<td>24.1</td>
<td>22.6</td>
</tr>
<tr>
<td>DLT($N$)</td>
<td></td>
<td>18.5</td>
<td>22.2</td>
</tr>
<tr>
<td>WLT</td>
<td></td>
<td>59.5</td>
<td>57.0</td>
</tr>
<tr>
<td>Reduction DLT($N, \sigma$) to WLT (in %)</td>
<td>59.5</td>
<td>60.4</td>
<td>58.8</td>
</tr>
<tr>
<td>Reduction DLT($N$) to WLT (in %)</td>
<td>69.0</td>
<td>61.1</td>
<td>61.9</td>
</tr>
<tr>
<td>$dod = 0.75$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLT($N, \sigma$)</td>
<td></td>
<td>19.9</td>
<td>17.9</td>
</tr>
<tr>
<td>DLT($N$)</td>
<td></td>
<td>19.0</td>
<td>19.6</td>
</tr>
<tr>
<td>WLT</td>
<td></td>
<td>62.5</td>
<td>52.7</td>
</tr>
<tr>
<td>Reduction DLT($N, \sigma$) to WLT (in %)</td>
<td>68.1</td>
<td>66.0</td>
<td>65.9</td>
</tr>
<tr>
<td>Reduction DLT($N$) to WLT (in %)</td>
<td>69.6</td>
<td>62.9</td>
<td>64.3</td>
</tr>
</tbody>
</table>

in Table 5. The best results of the tuning group are indicated by $+$, the average results by $\varnothing$. The best results for SLT, WLT, and DLT are relatively similar. This results from the high number of 1 million simulation runs allowing for a detailed approximation. Still, there are significant differ-
The x-axis represents the point of time; the y-axis represents the free time budget.

ences in the approximation speed and the impact of tuning for SLT and DLT. By comparing the best and average solution quality of the SLTs, we can observe a gap of up to 4.2% in confirmations. As assumed, the static and a priori definition of the partitioning has a significant impact on the solution quality. Even though SLTs may allow sufficient approximation for one interval size, for others the solution quality is inferior. For the DLTs, the results are relatively independent from tuning. The average results of DLT$(N, \sigma)$ are always higher compared to the average results of SLT and provide up to 4.2% more confirmations. In some cases, they even outperform the best SLT. Except for one case, DLT$(N, \sigma)$ performs as well as or better than WLT and for half of the instance settings, even the average results of DLT$(N, \sigma)$ are at least as good as the WLT. DLT$(N)$ provides similar results as the DLT$(N, \sigma)$ in best and average tuning. For the problem under consideration, a partitioning solely based on observations is sufficient and the standard deviation may be neglected.

This may be explained by the continuous decrease of expected future confirmation, i.e., values and as a result of standard deviation over time $t$. This results in a more detailed partitioning for early points of time $t$ which does not add significant benefit to the approximation.
Even though the WLT achieves similar results compared to the DLTs, the DLTs allows a significantly more efficient storage of the values. The number of entries of the best tuning and the reduction by DLT compared to WLT are depicted in the lower part of Table 5. Over all instances, the reduction of required entries compared to WLT reaches up to 68.1% for DLT\((N, \sigma)\) and even up to 79.9% for DLT\((N)\). The reduction and the number of entries is dependent on the instance setting under consideration. Given a high dod or a small service area \(A_{15}\) and, therefore, a high initial time budget, significantly more states and entries can be observed resulting in partitionings with many separations. Although memory may be relatively inexpensive, due to the exponential entry-increase by adding additional dimensions to the LT, DLTs may be able to handle higher-dimensional LTs compared to WLT.

Another advantage of the DLT is the effective approximation process. Figure 7 shows the ATB-approximation process for SLT, WLT, and DLT\((N, \sigma)\) given \(F_{2C}, dod = 0.75,\) and \(A_{20}\). To allow a detailed depiction, the x-axis only represents the first 200,000 simulation runs. On the y-axis, the according solution quality is depicted. We averaged the results over 1,000 simulation runs for a better presentation. We compare the tuned LTs providing the best solution quality after 1 million simulation runs and the average results of the tuning group. The best approximation in the first 10,000 simulation runs is achieved by WLT quickly converging to similar results as the best DLT. Even the average DLT allows a faster approximation than the best SLT. Within less than 50,000 simulation runs, the average DLT achieves a number of confirmations the best SLT only reaches after 200,000 simulation runs. In essence, the DLT allows a significantly more effective approximation.

This effective approximation is achieved by the adaptive partitioning of the DLT. We now analyze the resulting structure. We select DLT\((N)\) after 10,000 simulation runs, since it illustrates the DLT-behavior most clearly. The corresponding LT-structure is shown in Figure 8. A correlation between point of time \(t\) and free time budget \(b\) over the time horizon can be identified. This structure resembles the general \(t\) and \(b\) dependency, i.e., the decrease of free time budget over time. For these generally observed areas of the vector space, a more detailed consideration is both feasible and necessary.
7.2 Routing Behavior

In the following, we analyze the routing behavior induced by ATB to explain ATB’s strong performance for $\mathcal{F}_{2C}$ compared to $\mathcal{F}_{3C}$. This routing behavior impacts the temporal development of the area coverage. If the instance setting allows for a recurrent general routing behavior, the dependency between the temporal attributes and the area coverage is strong. This is of significant importance especially for clustered customers, since the coverage significantly drops, when the vehicle leaves a cluster. The aggregation needs to enable the consideration of these cluster changes. We show that, in contrast to $\mathcal{F}_{3C}$, for $\mathcal{F}_{2C}$ ATB is able to achieve a general routing behavior and the correlation between point of time, free time budget, and area coverage is strong.

The routing behavior manifests in the point of times, the vehicle changes between the clusters. In Figure 9, we depict these times for the two distributions. Since the behavior depends on the applied policy, it differs over the approximation process. We additionally depict the cluster changes resulting from the initial myopic policy of ATB with $\hat{V}_0$. On the left side, the development for $\mathcal{F}_{2C}$ is shown, on the right side for $\mathcal{F}_{3C}$.

The y-axis represents per point of time the percentage of test runs, the vehicle leaves a cluster. We can identify two cluster changes for $\mathcal{F}_{2C}$. Hence, the vehicle generally serves customers in one clusters, travels to the second clusters, and finally returns to the first cluster. The times of the cluster changes for the initial myopic policy vary significantly. For ATB, the first cluster change mainly occurs between $100 \leq t \leq 125$. We have externally determined the average travel duration $d = 20$ between the two cluster of $\mathcal{F}_{2C}$. Generally, the vehicle arrives in the second cluster not later than around $t = 144$. The second cluster change occurs between $260 \leq t \leq 270$ and is even more distinct. Hence, for $\mathcal{F}_{2C}$, ATB achieves a general routing behavior. ATB is able to maintain this behavior for every realization and to determine when to change clusters. This is remarkable because no explicit spatial information is included in the ATB state aggregation.

On the right side of Figure 9, the development for $\mathcal{F}_{3C}$ is shown. For this instance setting, the vehicle changes clusters several times. For the purpose of presentation, we focus only on the first two cluster changes. A slight difference to the myopic policy can be observed. For ATB, the vehicle usually leaves clusters earlier. Still, the times of cluster changes vary significantly. In some cases, the vehicle only serves a few customers in the first cluster and then directly travels to the next cluster around $t = 15$. In other cases, the first cluster change occurs around $t = 180$. The
time of the second cluster change varies widely from around $t = 60$ up to $t = 300$. This indicates that the cluster sequence varies over the realizations. As assumed, a general routing behavior cannot be identified and the correlation between point of time, time budget, and area coverage is weak. Even though ATB still provides high quality solutions for $F_3C$ compared to the benchmark heuristics, the implicit inclusion of spatial information about the vehicle’s position and the area coverage vanishes. To improve solution quality for these instance settings, either a standard cluster sequence may be determined binding for every realization or the integration of spatial information in the aggregation may be necessary.

### 7.3 Budgeting Time

In this final section, we analyze how ATB exploits the dependencies depicted in Figure 3 of §4.2 and how ATB budgets the time accordingly. Therefore, we again compare ATB with the initial myopic policy induced by $\hat{V}_0$. For the instance settings $dod = 0.75$ and $A_{20}$, we show how for uniformly distributed requests given $F_U$ the dependencies align to the analytical considerations while for clustered customers given $F_2C$, the dependencies significantly differ. Since a determination of the area coverage is challenging, we draw on the insertion time $\gamma$ as proxy. We differentiate between the average insertion time of all requests $\gamma^a$ and the realized insertion time of the confirmed requests $\gamma^c$. The average insertion time $\gamma^a_t$ per new request in time $t$ is calculated as depicted in Equation (13). Decision $x_a$ confirms all requests, regardless the time limit. $\gamma^a_t$ is only calculated if new requests are given in decision point $k$: $|C^r_k| > 0$. 

![Figure 9: Cluster Changes per Point of Time for ATB and the initial myopic policy](image)
\[ \gamma_i^a = \frac{d(\Theta_k^{x_c}) - d(\Theta_k)}{C_k^r} \]  
\[ \gamma_i^c = \frac{\hat{d}(\Theta_k^{x_c}) - d(\Theta_k)}{C_k^c} \]  

The realized insertion time \( \gamma_i^c \) per confirmed customer and \( t \) is defined in Equation (14). Decision \( x_c \) is the decision selected by the regarding approach. \( \gamma_i^c \) is only calculated if requests are confirmed: \( |C_k^c| > 0 \).

Figure 10 depicts the impact of ATB and the initial myopic policy on the tour duration \( \tilde{d} \) and the insertion times \( \gamma^a, \gamma^c \) as well as the value \( \hat{V} \) depending on \( b \) for a specific point of time \( t = 144 \). The left side of Figure 10 shows the results for \( \mathcal{F}_U \), the right side for \( \mathcal{F}_{2C} \). Figure 10a and Figure 10b depict the development of the tour duration \( \tilde{d} \) over time. For myopic, \( \tilde{d} \) continuously increases until the free time budget is expended. Notably, for ATB, \( \tilde{d} \) increases after \( t = 120 \) for \( \mathcal{F}_{2C} \). This is the time after the first change of clusters, when many new requests are confirmed instantly the vehicle arrives in the new cluster. As assumed, \( \tilde{d} \) is strongly correlated to the actual area coverage indicated by the according average insertion time \( \gamma^a \) in Figure 10c and Figure 10d respectively. \( \gamma^a \) is significantly lower for myopic compared to ATB. The correlation of \( \tilde{d} \) and \( \gamma^a \) is therefore strong.

For \( \mathcal{F}_U \), \( \gamma^a \) shows a continuous increase over time. For ATB and \( \mathcal{F}_{2C} \), we can observe a peak around \( t = 135 \). \( \gamma^a \) for requests around this point of time is high and then decreases for \( t > 135 \). At this point of time, the vehicle served all customers in the first cluster and arrived in the second. Since at this point new requests within the first cluster are highly expensive to include, \( \gamma^a \) rises. Afterward, the area coverage in the first cluster is reestablished and \( \gamma^a \) decreases.

Considering the realized insertion time \( \gamma^c \) in Figure 10e and Figure 10f, we can observe that for \( \mathcal{F}_U \), ATB in average allows constant \( \gamma^c \) per customer over time. As expected, myopic allows a high amount in the beginning, but is not able to insert later requests since the time budget is consumed. Given \( \mathcal{F}_{2C} \), a significant peak is shown for ATB between \( 120 \leq t \leq 180 \). The highest \( \gamma^c \) can be observed around \( t = 144 \) meaning that at this point of time the vehicle usually arrived in the second cluster. Around this time, ATB identifies the requirement to spend significantly more of the free time budget to reestablish the area coverage as already depicted in the top right figure of Figure 10.

As shown in Figure 9, the vehicle arrives in the second cluster usually not later than \( t = 144 \).
Figure 10: Dependencies for the Expected Number of Confirmations given $\mathcal{F}_U$ and $\mathcal{F}_{2C}$
We now analyze how this arrival impacts the values $\hat{V}$. Figure 10g and Figure 10h show $\hat{V}$ for $t = 144$ and differing $b$. On the x-axis, $b$ is depicted. For $F_U$, the development of the values follows the idealized assumptions of Figure 4. An increase of $b$ results in a declining increase of $\hat{V}$. Given $F_{2C}$, a jump discontinuity can be observed at $b = 60$. For $b < 60$, the value drops drastically. A time budget lower than $b = 60$ reflects that the vehicle has already arrived in the second cluster and confirmed customers. At arrival, a large subset of requests is confirmed. Hence, the post-decision state value drops. Given $b \geq 60$, the vehicle has not arrived in the cluster yet. A high number of confirmations can be expected and the value is significantly higher than for $b < 60$. ATB is able to identify this characteristic and, therefore, is able to adapt to the spatial and temporal request distribution.

8 Conclusion and Future Research

In this paper, we have presented the VRPSSR, a dynamic vehicle routing problem with stochastic customer requests. The requests follow a temporal and spatial distribution and the request times and customer locations are unknown until the request occurs. We have depicted that anticipation of future confirmations can be achieved by evaluating decisions with respect to the point of time and the free time budget left. With ATB, we have developed an approach exploiting this dependency via approximate value iteration to approximate the expected number of future confirmations for a decision state. For a variety of instances, ATB is able to outperform state-of-the-art benchmark heuristics.

As an offline approach, ATB allows fast decision making within the execution phase. ATB requires an aggregation of states to vectors of point of time and free time budget and a subsequent partitioning of the vector space to a lookup table for storing the expected values. The LT-entries then are determined by intervals of point of time and free time budget. With the dynamic lookup table, we introduced a new approach adapting to the problem’s and instances’ specifics. A comparison to classical partitioning approaches indicates that the DLT is both efficient and effective. DLTs provide high quality solutions and fast approximation processes.

ATB does not draw on spatial information in the state space aggregation. This allows excellent anticipation if the customer distributions allow ATB to achieve a recurrent, general routing be-
behavior. For other instance settings, future work may identify suitable spatial and spatial-temporal attributes to extend ATB allowing an explicit consideration of customers’ and vehicle’s locations. The considered problem may further be extended to multi-vehicle and multi-periodical settings. Rejected customers might be postponed to a following period as presented by Angelelli et al. (2009). For these problems, anticipatory algorithms need to consider free time budgets for several vehicles in the current and the following periods. This requires aggregations of higher dimensionality and may enhance the advantages of the DLT. Due to its generality, DLT may not only be applied to the VRPSSR but might be a valuable tool in the entire field of approximate dynamic programming, especially for problems with complex value function structures. The functionality of the DLT might be extended by partitioning individual parameters, by merging of entries, or by the addition and reduction of parameters depending on the approximation process.

Appendix

A.1 Routing and Insertion Decisions

ATB evaluates current confirmations by anticipating their impact on the number of expected future confirmations, i.e., the value. To determine the realized movement action and the positions in the tour where new requests are inserted, we draw on cheapest insertion (CI) routing (Rosenkrantz et al. 1974) for both the ATB method and the benchmark approaches. CI has two main advantages. First, it reflects the routing methods applied in practice. Usually, the dispatcher plans a sequence of the ERCs in the beginning of the day and subsequently adds new customers to the existing tour. Second, the application of CI is efficient and allows fast feasibility checks and decision making within the execution phase. Given a set of new requests $C \cup k$, all approaches evaluate every potential candidate subset to confirm regarding feasibility and rewards. This results in a high number of candidate tours. Here, an efficient routing algorithm is mandatory to allow dynamic decision making.

The tour is initialized and updated regarding the following procedure: In $t = 0$, CI starts with a pre-decision tour $\Theta_0 = (D, D)$ only consisting of the depot. In every decision point $k$, a pre-decision tour $\Theta_k$ and a set of candidate subsets $C_k(k) \subseteq C_k(k)$, $i = 1, \ldots, 2^{\|C_k(k)\|}$, i.e., the power set of new requests is given. For every candidate subset, CI subsequently selects the cheapest
request of the subset regarding the insertion time and adds it at the cheapest insertion position in
the tour. For \( t = 0 \), the ERCs are already given, i.e., \( C_c^0(0) = C_0^c \). The routing results in a
decision \( x \) and a post-decision tour \( \Theta^x_k = (\theta_0, \ldots, \theta_o) \). If the selected movement action of \( x \) is not
waiting, the first customer in the tour \( C_{\text{next}} = \theta_1 \) is the next customer to visit. After the travel to
the next customer, the visited customer is removed from \( \Theta^x_k \). This results in the pre-decision tour
\( \Theta_{k+1} = (\theta_1, \ldots, \theta_o) \). If waiting is applied, the pre-decision tour \( \Theta_{k+1} \) is identical to \( \Theta^x_k \). If no
customers are left to serve, i.e., \( \Theta_k = (\mathcal{P}_k, \mathcal{D}) \), but free time budget is left, the vehicle idles at the
current location for all approaches. If no time is left, the vehicle returns to the depot.

A candidate subset and the according decision are considered feasible if the resulting candidate
post-decision tour does not exceed the time limit. Because the decision space is reduced, CI may
reject candidate subsets, which are feasible regarding a different routing approach, e.g., an optimal
traveling salesman solution.

A.2 Example: Impact of Partitioning

In this section, we show the impact of partitioning for a simplified, single-stage example. To
show the impact of the number of entries to the approximation process, we especially consider
the required number of simulation runs for a sufficient approximation. We consider a partitioning
in the highest level of separation resulting in four entries \( \mathcal{Q} = \{p_1, \ldots, p_4\} \). Additionally, we
consider a partitioning \( \mathcal{Q} = \{\bar{p}\} \) containing only a single entry. In \( \mathcal{Q}, \bar{p} \) represents \( p_1, \ldots, p_4 \). The
value in every entry in \( \mathcal{Q} \) follows a normal distribution with expected value \( V(p_i) \) and standard
deviation \( \sigma^2(p_i) \). Additionally, let \( \nu_i \) be the probability of observing \( p_i \). The according values are
shown in Table A1. The entry values behave heterogeneously, the expected values and deviations
rise from \( p_1 \) to \( p_4 \). The according values of \( \bar{p} \) are a result of the partitioning.

To show the impact of the number of entries on the approximation process, we calculate the
expected necessary number of observations \( n^*_i \) for every entry and the total number of simulation
runs \( n^* \) for termination, i.e., for sufficient approximation in every entry. As a termination criterion,
we allow a difference of the average values \( \bar{V} \) to the expected values up to 0.05. Further, we
calculate the number of required observations \( \bar{n} \) for entry \( \bar{p} \) and compare the results. For each
entry, we calculate the distribution of the expected realizations average (i.e., \( \alpha = \frac{1}{\bar{p}} \)). Then,
we derive the probability \( P_k \) that the average lies in the allowed deviation range after \( k \) entry

36
Table A1: Expected Entry Values and Deviation

<table>
<thead>
<tr>
<th>Entries</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$\bar{p}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>1.0</td>
<td>2.0</td>
<td>4.0</td>
<td>6.0</td>
<td>3.0</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.0</td>
<td>1.0</td>
<td>3.0</td>
<td>4.0</td>
<td>4.3</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

$n^*_i$ | 0 | 1,537 | 4,610 | 6,146 | 6,590 |

observations. We determine $n^*_i$ as the minimal number of observations with probability higher than $\mathbb{P}_k > 95.0\%$. Let $\hat{V}_l(p_i)$ be the value of the $l$th entry realization of $p_i$. Then, the minimum number of observations for entry $p_i$ can be calculated as shown in Equation (A1):

$$n^*_i = \arg \min_{k \in \mathbb{N}^+} \left\{ \mathbb{P}_k \left( \left| V(p_i) - \frac{1}{k} \sum_{l=1}^{k} \hat{V}_l(p_i) \right| \leq 0.05 \right) \geq 0.95 \right\}. \quad (A1)$$

The total number of required simulation runs $n^*$ is the maximum number of individual observations considering the probability of observing the entry as depicted in Equation (A2):

$$n^* = \max_{i \in \{1, \ldots, 4\}} \left\{ \frac{n^*_i}{\nu_i} \right\}. \quad (A2)$$

Rarely visited entries increase the necessary number of simulation runs of the algorithm. In the example, $p_4$ requires the most visits for termination with $n^*_4 = 6,146$. Due to the probability $\nu_4 = 0.1$ of observing $p_4$, the expected number of runs for termination of the whole process is $n^* = \frac{n^*_4}{\nu_4} = 61,460$. For $Q$, a sufficient approximation is expected after 61,460 simulation runs. We now show that partitioning $Q$ can reduce the number of required simulation runs significantly. The expected value of $\bar{p} \in Q$ is the weighted sum of the single expected values as depicted in Equation (A3):

$$V(\bar{p}) = \sum_{i=1}^{m} \nu_i V(p_i) = 3.0. \quad (A3)$$

The variance $\sigma^2(\bar{p})$ can be calculated as shown in Equation (A4):

$$37$$
\[ \sigma^2(\bar{p}) = \mathbb{E}V(\bar{p})^2 - (\mathbb{E}V(\bar{p}))^2 = \sum_{i=1}^{4} \nu_i V(p_i)^2 - \left(\sum_{i=1}^{4} \nu_i V(p_i)\right)^2 = 4.3. \]  

(A4)

The probability distribution of \( V(\bar{p}) \) is the weighted sum of the single distributions. The number of necessary observations to achieve a maximal deviation of 0.05 with a probability of at least 95% is \( \bar{n} = 6,590 \). The number of necessary simulation runs is reduced by 89.3% compared to \( Q \). For our example, partitioning \( \bar{Q} \) allows a significantly faster approximation. Nevertheless, the partitioning has a large impact on decision making. As we can see from Equation (A4), the variance of \( V(\bar{p}) \) exceeds the variance of all original entries \( p_1, \ldots, p_4 \). Partitioning \( \bar{Q} \) results in a rise of the deviation of the entry value. We additionally experience a bias \( |V(\bar{p}) - V(p_i)| \) up to 3.0 for all former entries. Using \( \bar{p} \) results in a less accurate representation and may lead to ineffective decisions.

Evidently, partitioning \( \bar{Q} \) results in a suboptimal solution quality. We consider a decision point \( S \) and two possible decisions \( x_a, x_b \). Decision \( x_a \) leads to entry \( p_1, x_b \) to \( p_4 \). The immediate rewards are \( R(S, x_a) = 2.0 \) and \( R(x_b) = 1.0 \). Considering the Bellman equation, given \( \bar{Q} \), the overall values are \( R(S, x_a) + V(p_1) = 3.0 \) and \( R(S, x_b) + V(p_4) = 7.0 \). Hence, we choose \( x_b \) and achieve an expected overall outcome of 7.0. With SLT \( \bar{Q} \), the two decisions result in the same entry \( \bar{p} \). Hence, decision \( x_a \) is chosen with overall outcome of 3.0. Due to partitioning \( \bar{Q} \), we experience a loss of 4.0. In essence of the example, \( \bar{Q} \) allows a faster approximation, but simultaneously leads to a loss in solution quality. We experience a tradeoff between accuracy and approximation efficiency.

### A.3 Dynamic Lookup Table: Algorithm

In this section, we present the general procedure of AVI combined with a \( \varsigma \)-dimensional DLT. The procedure is depicted in Algorithm 1.

Inputs are the aggregation function \( \mathfrak{A} \), partitioning \( \mathcal{I}^0 \), the resulting initial LT \( \mathcal{E}^0 \), the initial values \( V^{m_0} \), a set of realizations \( \{\omega^1, \ldots, \omega^m\} \), the step size \( \alpha \), and the threshold \( \tau \). Outputs are the values \( V^m \) after \( m \) simulation runs, the according partitioning \( \mathcal{I}^m \), and DLT \( \mathcal{E}^m \). Notably, partitioning and DLT are equivalent. Hence, only storing \( \mathcal{E} \) instead of both \( \mathcal{E} \) and \( \mathcal{I} \) may be sufficient. In our computational experiments, we implemented the DLT as a Java Hashmap. We experienced, that this implementation allows an efficient storage, fast value access, and fast updates of the DLT.
Algorithm 1: AVI and Dynamic Lookup Table

**Input:** $A, T^0, E^0, V^m, \{\omega^1, \ldots, \omega^m\}, \alpha, \tau$

**Output:** $V^m, T^m, E^m$

2 // Initialization
3 $i \leftarrow 1$, $N \leftarrow 0$, $\sigma \leftarrow 0$
4 for all $\eta_k \in E^0$ do
5   $\sigma(\eta_k) \leftarrow 0$, $N(\eta_k) \leftarrow 0$
6 end
8 // Simulation
9 while ($i \leq m$) do
10   $k \leftarrow -1$
11   $\mathcal{P} = \emptyset$
12   while ($S_k^e \neq S_K$) do
13     $k \leftarrow k + 1$
14     if $k \geq 1$ then $S_k \leftarrow (S_{k-1}^e, \omega_{k-1}^e)$
15     else $S_k = S_0$
16     for all $x \in X(S_k)$ do
17       $\eta_k^e \leftarrow T^{i-1}(\mathfrak{X}(S_k^e))$
18     end
19     $x \leftarrow \arg\max_{x \in X(S_k)} \{ R(S_k, x) + V^{m+1}(\eta_k^e) \}$
20     $S_k^e \leftarrow (S_k, x)$
21     $R_k \leftarrow R(S_k, x)$
22     $\mathcal{P} \leftarrow \mathcal{P} \cup \{T^{i-1}(\mathfrak{X}(S_k^e))\}$
23 end
25 // Update
26 $R_{-1} \leftarrow 0$
27 for all $\eta_k^e \in \mathcal{P}$ do
28   $V^m(\eta_k^e) \leftarrow (1 - \alpha)V^{m+1}(\eta_k^e) + \alpha R_k$
29   $N(\eta_k^e) \leftarrow N(\eta_k^e) + 1$, $\sigma(\eta_k^e) \leftarrow \text{UpdateSigma}(\sigma(\eta_k^e), R_k)$
30   $\bar{N} \leftarrow \bar{N} + 1$, $\bar{\sigma} \leftarrow \text{UpdateSigma}(\bar{\sigma}, R_k)$
31 end
32 for all $\eta_k^e \in \mathcal{P}$ do
33   if $\frac{N(\eta_k^e)}{\bar{N}} \geq \tau$ then
34     UpdatePartition($T^{i-1}, \eta_k^e$)
35     UpdateTable($E^{i-1}, \eta_k^e$)
36   end
37   $i \leftarrow i + 1$
38 end

39 return $V^m, T^m, E^m$
In the initialization, $\bar{\sigma}$, $\bar{N}$ and the individual $\sigma$ and $N$ per entry $\eta \in \mathcal{E}^0$ are set to zero. Then, the algorithm subsequently simulates the realizations. For each realization $\omega^i$, the algorithm simulates states, decisions based on current values, and stochastic transitions to the next states and stores observed entries and rewards. Initially, the set of observed entries $\mathcal{P}$ is empty. For every $k \geq 1$, state $S_k = (S_{k-1}^x, \omega^i_{k-1})$ is the combination of the previous post decision state $S_{k-1}^x$ and the according part of realization $\omega^i$. In a given state $S_k$, the algorithm now creates the set of accessible entries $\{ \eta \in \mathcal{E}^{i-1} : \exists x \in X(S_k) : \eta = T^{i-1}\mathcal{A}(S_k^x) \}$. The decision $x$ is selected, where the according entry $\eta^x_k$ maximizes the Bellman equation:

$$x = \arg \max_{x \in X(S_k)} \left\{ R(S_k, x) + V^{\pi_{i-1}}(\eta^x_k) \right\}.$$  

(A5)

This entry $\eta^x_k$ is then stored in the set of observed entries $\mathcal{P}$. The reward $R(S_k, x)$ is stored in $R_k$. The simulation of $\omega^i$ terminates when $S_k = S_K$. After each simulation run $i$, the values $V^{\pi_m}$, $\bar{\sigma}$, $\bar{N}$, and $\sigma(\eta)$, $N(\eta)$ for the observed entries $\eta \in \mathcal{P}$ are updated. Finally, these entries are analyzed with respect to threshold $\tau$ and the partitioning and DLT are updated accordingly. In the following, we describe the functions $UpdateSigma$, $UpdatePartition$, and $UpdateTable$.

Update of Sigma

The standard deviation $\sigma$ of a set of $n$ observations with realized values $V_1, \ldots, V_n$ and average value $\bar{V}$ is usually calculated as depicted in Equation (A6).

$$\sigma = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (V_i - \bar{V})^2}.$$  

(A6)

The standard deviation $\sigma$ has to be recalculated for every simulation run. A recalculation regarding Equation (A6) would require the storage of every realized value $V_j$. In many cases, the number of simulation runs of AVI is large and an efficient storage not possible. The Steiner Translation Theorem allows a reformulation of Equation A6 as depicted in Equation A7.

$$\sigma = \sqrt{\sum_{j=1}^{n} V_j^2 - \frac{1}{n} (\sum_{j=1}^{n} V_j)^2}.$$  

(A7)

As a result, only the values of $\sum_{j=1}^{n} V_j^2$ and $\sum_{j=1}^{n} V_j$ have to be stored. These values can be efficiently updated given a new observation.
Update of Partitioning and Table

In the update process of $i^i$, entries $\eta = i^i(p)$ are analyzed with respect to the threshold:

$$\frac{N(\eta)\sigma(\eta)}{N\sigma} \geq \tau \quad \text{(A8)}$$

These entries are separated to a set of $2^c$ new entries $\eta_1 = i^{i+1}(p), \ldots, \eta_2^c = i^{i+1}(p)$ by dividing all intervals in half. Values, observations, and deviations are transferred to the new entries and the former entry is replaced. Because of the lack of further knowledge about the distribution of the values and observations, the number of observations and the deviation are equally divided to the new entries and the values remain.

A.4 Benchmark Heuristics for the VRPSSR

In this section, we describe the benchmark heuristics anticipatory insertion (AI) and cost-benefit heuristic (CBH) for the VRPSSR. These heuristics are compared to ATB. We further describe the required tuning.

Anticipatory Insertion

AI was introduced by Ghiani et al. (2012) and draws on waiting strategies by Thomas (2007). The main idea of AI is to idle at certain locations in the service area to maintain a high coverage of the service area and, therefore, to insert new requests efficiently. AI draws on a myopic confirmation action, i.e., does not explicitly decide about subset selection. To determine at which locations to wait, Ghiani et al. (2012) use the center-of-gravity (COG) longest wait strategy calculating the COG of all feasible potential future customers. The vehicle waits at the customer location, which allows the latest departure time serving the remaining customers and a dummy customer located at the COG. The COG is recalculated in every decision point. We call this approach AI. For a dynamic routing problem with known customer locations, AI is able to achieve similar results as the sample-scenario approach by Bent and Van Hentenryck (2004). AI requires significantly less calculation effort in the execution phase and maintains the sequence of confirmed customers throughout the execution of the tour because of CI routing. For the presented problem, potential future customers are not known. To apply AI, we sample a sufficient number of future customers in
every decision point. Then, we calculate the COG of the feasible sampled customers and proceed as described in Ghiani et al. (2012).

**Tuning.** To determine the center of gravity, we sample a set of 100 spatial realizations \( F \subset \mathcal{A} \) of the respective distribution \( \mathcal{F} \). Each sampled customer is checked for feasibility in the current tour. This results in a subset of feasible realizations \( \tilde{F} \subset F \). If \( |\tilde{F}| = 0 \), the COG is set to the depot. Else, the COG is calculated regarding Equation (A9).

\[
\text{COG} = \left( \sum_{(a_x, a_y) \in \tilde{F}} \frac{a_x}{|\tilde{F}|}, \sum_{(a_x, a_y) \in \tilde{F}} \frac{a_y}{|\tilde{F}|} \right).
\]  

(A9)

**Cost Benefit Heuristic**

CBH originates from the idea of Ichoua et al. (2000) weighting the additional travel time resulting from a decision against the according reward. Ulmer et al. (2015) transfer this idea to determinate the acceptance of candidate subsets. Subset candidates requiring a relatively high consumption of the free time budget are rejected. As a result, CBH conserves a large free time budget to include future customers, but at the expense of fewer immediate confirmations and reduced service area coverage. Given a tour \( \Theta_k \) and a candidate subset \( C_k^c \), CBH compares the insertion time (“cost”) of decision \( x \) inserting \( C_k^c \) with the rewards (“benefit”) as depicted in Equation (A10):

\[
\kappa \cdot \frac{|\Theta_k^x|}{|\Theta_k|} \geq \frac{\bar{d}(\Theta_k^x)}{d(\Theta_k) + d^*}.
\]

(A10)

On the left side of Equation (A10), the relative benefit of a decision, i.e., the increase in customers to serve is calculated. On the right, the duration of the new tour compared to the current tour is calculated. The parameters \( \kappa \) and \( d^* \) allow tuning regarding the instances. The average insertion time of adding a customer is dependent on the instances. Parameter \( \kappa \) scales the benefit compared to the costs. Parameter \( d^* \) defines a free amount of time budget relative to the tour length. This enables the insertion of customer requests especially in the beginning, when the tour might be short and insertion times are above-average.

**Tuning.** To determine the parameters \( d^*, \kappa \), for every instance setting, we run 100 test runs offline. For every test run, we apply candidate parameter vectors \((d_k^*, \kappa_c)\) with \( d_k^* \in T \) and \( \kappa_c \in [0, 2] \). We select the parameter combination maximizing the overall sum of confirmations for the
A.5 Benchmark Partitioning Algorithms

In this section, we describe two alternative partitioning approaches to be compared to the DLT. First, we present an a priori equidistant and static partitioning approach, the static LT (SLT). Second, we present the weighted LT (WLT) drawing on a set of static partitionings by George et al. (2008).

**Static Lookup Table**

Usually, partitionings generate static LTs with equidistant interval lengths. An exemplary LT-entry $\eta \in \mathcal{E}$ consists of a set of vectors within an interval of lengths $l$: $\eta = \{(t, b), (t + 1, b), \ldots, (t + l - 1, b), \ldots, (t + l - 1, b + l - 1)\}$. The resulting LT is $\mathcal{E}_l$. The SLT has the shortcoming that an a priori definition of the intervals may impede the solution quality because heterogeneous states are assigned to the same value. A suitable interval length may differ given different instances and even within an instance. This means that the required level of time-consideration varies and every a priori interval selection provides suboptimal approximation.

**Weighted Lookup Table**

To combine the advantages of different interval lengths $l_1, \ldots, l_L$ and to allow differing levels of detail, George et al. (2008) and Powell (2009) propose to combine multiple partitionings $\mathcal{I}_1, \ldots, \mathcal{I}_L$ and the resulting SLTs $\mathcal{E}_1, \ldots, \mathcal{E}_L$. Let $V^{l_i}(\mathcal{I}_i(p))$ be the value of $\eta_{l_i} = \mathcal{I}_i(p)$ in LT $\mathcal{E}_{l_i}$ of a weighted LT (WLT) $\mathcal{E}_{WLT} = \bigcup_{i=1}^{L} \mathcal{E}_{l_i}$. The overall value $V_{WLT}(p)$ is calculated as the weighted sum of the single LT-values with weights $w_{l_1}, \ldots, w_{l_L}$ for every LT as depicted in Equation (A11):

$$V_{WLT}(p) = \sum_{i=1}^{L} w_{l_i}(\mathcal{I}_i(p)) V^{l_i}(\mathcal{I}_i(p)).$$  \hspace{1cm} (A11)$$

The weights are calculated considering the experienced variance within the entries $\sigma^2_{l_i}(\mathcal{I}_i(p))$, the number of observations $N_{l_i}(\mathcal{I}_i(p))$, and the bias $\mu_{l_i}(\mathcal{I}_i(p))$ of entry $\mathcal{I}_i(p)$. The bias $\mu_{l_i}(\mathcal{I}_i(p)) = |V^{l_i}(\mathcal{I}_i(p)) - V^{l_i}(\mathcal{I}_i(p))|$ is defined by the difference between value $V^{l_i}(\mathcal{I}_i(p))$ and the value of the
lowest partitioning level $V^{i_1} (\mathcal{I}_i (p))$. A formula for the weights calculating the total variation of the error and considering all three impacts is provided by Powell (2009) as depicted in Equation (A12):

$$w_{l_i} (\mathcal{I}_i (p)) \propto \left( \frac{\sigma_{l_i}^2 (\mathcal{I}_i (p))}{N_{l_i} (\mathcal{I}_i (p))} + \mu_{l_i} (\mathcal{I}_i (p))^2 \right)^{-1}. \quad (A12)$$

The weighting favors LTs with large intervals in the beginning for a fast first approximation. Later, LTs with small intervals are weighted higher to achieve a more differentiated evaluation. The weight $w_{l_i}$ for an LT $\mathcal{E}_i$ is increased by a relatively small variance and bias and a large number of observations. In the beginning, the frequently observed entries in the LTs with large intervals allow a fast first estimation of the value function. During the subsequent approximation process, the weights of the more detailed LTs with small intervals increase because of the relatively small variance and bias. Further, WLT allows avoiding ineffective LT-areas. For instance, areas in the LT providing relatively low numbers of expected future confirmations are excluded early in the approximation process. Hence, the approximation is focused on the effective areas. WLT may allow a faster approximation in the beginning and high quality solutions in the end without any tuning necessary. Nevertheless, WLT leads to increased memory consumption due to the high number of entries. Instead of a single LT, $L$ LTs of different partitioning levels are required.

### A.6 Instance Generation

In this section, we describe the process of instance generation in detail. The closed service area $\mathcal{A} \subseteq \mathbb{R}^2$ is rectangular defined by the points $(0, 0)$ and $(x_{\text{max}}, y_{\text{max}}) \in \mathbb{R}^2$. Time is represented minute by minute, $T = \{0, 1, \ldots, t_{\text{max}}\}$. The expected number of overall customers is $n = \mathbb{E}_{\omega \in \Omega} |\mathcal{C}_\omega|$. The expected number of early request customers $\mathbb{E} |\mathcal{C}_0| = n_0$ depends on the degree of dynamism $dod \in [0, 1]$ as depicted in Equation (A13) (Larsen et al. 2002):

$$n_0 = (1 - dod) \cdot n. \quad (A13)$$

The number of customers and the request times for a realization are generated by a Poisson process $\Psi$ (Haight 1967). The number of ERCs is generated by $\Psi (n_0)$. The probability distribution $\Xi$ for request times and locations is divided into two independent probability distributions. Request times of late request customers are (discretely) uniformly distributed over time $t \sim U_{\mathbb{Z}} [1, t_{\text{max}} - 1]$. 

44
Customer locations $f(C^\omega) \in \mathcal{A}$ are realizations $f \sim \mathcal{F}$ of the spatial probability distribution $\mathcal{F} : \mathcal{A} \rightarrow [0, 1]$.

A realization of the request time is again conducted by a Poisson process $\mathcal{P}$ for every minute $0 < t < t_{\text{max}}$. Given two points of time $0 < t^i < t^h < t_{\text{max}}$, this results in an expected number of customer requests of $n^h_{\omega i} = \mathbb{E}_{\omega \in \Omega} \left| \{C^\omega_i \in C^\omega : t^i < t_i \leq t^h\} \right|$ accumulated in time $t^i < t_i \leq t^h$ as described in Equation (A14):

$$n^h_{\omega i} = \text{dod} \cdot n \cdot \frac{t^h - t^i}{|T| - 2}. \quad \text{(A14)}$$

The travel time $d : \mathcal{A} \times \mathcal{A} \rightarrow \mathbb{N}$ for two customers $C^\omega_1, C^\omega_2 \in \mathcal{C}^\omega$ with locations $f(C^\omega_1) = (a^1_x, a^1_y), f(C^\omega_2) = (a^2_x, a^2_y) \in \mathcal{A}$ is Euclidean and rounded up to minutes as depicted in Equation (A15). The minimal travel time is set to $\bar{t} = 1$ minute.

$$d(f(C^\omega_1), f(C^\omega_2)) = \max \left( \left| \frac{((a^1_x - a^2_x)^2 + (a^1_y - a^2_y)^2)^{1/2}}{v} \right|, 1 \right) \quad \text{(A15)}$$

The time limit is set to $t_{\text{max}} = 360$ minutes. We test the approaches for a large ($\mathcal{A}_{20} : x_{\text{max}} = 20\, \text{km}, y_{\text{max}} = 20\, \text{km}$) and small service area ($\mathcal{A}_{15} : x_{\text{max}} = 15\, \text{km}, y_{\text{max}} = 15\, \text{km}$). The vehicle travels with a speed of $v = 25\, \text{km/h}$. The depot is located in the center of the area $\mathcal{D}_{20} = (10, 10)$, respectively $\mathcal{D}_{15} = (7.5, 7.5)$. The average number of customer requests per day is $n = 100$. We examine instances with a moderate ($\text{dod} = 0.50$) and high ($\text{dod} = 0.75$) number of dynamic requests. We define three spatial distributions. We consider uniformly distributed customers ($\mathcal{F}_U$) and customer distributions grouped in two ($\mathcal{F}_{2C}$) or three clusters ($\mathcal{F}_{3C}$). Within the clusters, the customer locations are two-dimensional normally distributed.

In the following, we define the spatial distribution functions for $\mathcal{A}_{20}$. Given $\mathcal{F}_U$, a realization $f(C) = (a_x, a_y)$ is defined as $a_x, a_y \sim U[0, 20]$. For $\mathcal{F}_{2C}$, the customers are equally distributed to each cluster. The cluster centers are located at $\mu_1 = (5, 5), \mu_2 = (15, 15)$. The standard deviation within the clusters is $\sigma = 1$. The distribution is therefore point-symmetrical to the depot. For $\mathcal{F}_{3C}$, the cluster centers are located at $\mu_1 = (5, 5), \mu_2 = (5, 15), \mu_3 = (15, 10)$. 50% of the requests are assigned to cluster two, 25% to each other cluster. The standard deviations are set to $\sigma = 1$. For $\mathcal{A}_{15}$, all spatial parameters are reduced by factor 0.75.
References

Angelelli, Enrico, Nicola Bianchessi, Renata Mansini, Maria G Speranza. 2009. Short term strategies for a
dynamic multi-period routing problem. Transportation Research Part C: Emerging Technologies 17(2)
106–119.


1362–1363.

Bent, Russell W, Pascal Van Hentenryck. 2004. Scenario-based planning for partially dynamic vehicle


Bertsimas, Dimitris J, Garrett Van Ryzin. 1991. A stochastic and dynamic vehicle routing problem in the

Branke, Jürgen, Martin Middendorf, Guntram Noeth, Maged Dessouky. 2005. Waiting strategies for dy-
namic vehicle routing. Transportation Science 39(3) 298–312.


Clausen, Clifford, Harry Wechsler. 2000. Quad-q-learning. Neural Networks, IEEE Transactions on 11(2)
279–294.

Daganzo, Carlos F. 1984. Checkpoint dial-a-ride systems. Transportation Research Part B: Methodological
18(4) 315–327.


Ehmke, Jan F. 2012. Integration of Information and Optimization Models for Routing in City Logistics,

Ehmke, Jan F, Ann M Campbell, Timothy L Urban. 2015. Ensuring service levels in routing problems with

Esser, Klaus, Judith Kurte. 2015. KEP 2015. Marktanalyse, Bundesverband Paket und Expresslogistik e. V.

Informatica 4(1) 1–9.


