Bus Inf Syst Eng 61(3):261–275 (2019)
https://doi.org/10.1007/s12599-019-00582-7

RESEARCH PAPER

Adaptive State Space Partitioning for Dynamic Decision Processes

Ninja Soeffker · Marlin W. Ulmer · Dirk C. Mattfeld

Received: 28 June 2018 / Accepted: 13 November 2018 / Published online: 28 January 2019

Abstract With the rise of new business processes that require real-time decision making, anticipatory decision making becomes necessary to use the available resources wisely. Dynamic real-time problems occur in many business fields, for example in vehicle routing applications with stochastic customer service requests expecting a fast response. For anticipatory decision making, offline simulation-based optimization methods like value function approximation are promising solution approaches. However, these methods require a suitable approximation architecture to store the value information for the problem states. In this paper, an approach is proposed that finds and adapts this architecture iteratively during the approximation process. A computational proof of concept is presented for a dynamic vehicle routing problem. In comparison to conventional architectures, the proposed method is able to improve the solution quality and reduces the required architecture size significantly.

Keywords Approximate dynamic programming · Dynamic service routing · State space partitioning · Data-driven modeling and simulation · Simulation-based optimization

1 Introduction

Many modern service providers offer services at customers' homes that are requested through modern communication technologies, often even for the same day. Examples are courier, mobility, repair services, and transportation services in general. A major part of the corresponding business model is the definition of the business process modeling the response to the customer request. In many of these business processes, customers call while the service vehicles are already on the road. Customers expect a fast response to their service request and preferably a service fulfillment on the same day. The number of business models containing such business processes constantly increases and thus, more providers are challenged by the task of successful service fulfillment (Speranza 2018; Savelsbergh and Van Woensel 2016).

In contrast to traditional planning problems with all information available and sufficient time for computation, service providers now face dynamic decision processes with a sequence of decision states. In every state, they need to make decisions about assignments of the new customer requests and the corresponding dynamic routing of the vehicles in real-time. On the operational level, the provider has predefined and limited workforce resources to fulfill the service requests. The resources are reflected in the time available within the drivers' shifts. Because providers operate in highly competitive markets, decisions need to utilize these time resources both effectively and efficiently. Decisions are generally determined under incomplete information because new information such as new customer requests is likely to be available in later states. Because current decisions impact the resources available later in the day, an anticipation of potential future developments is necessary in current decision making to balance immediate and possible future revenue. For anticipation, providers often have access to large amounts of historical transactional data. However, the pure availability of this data is not sufficient for a decision support in a decision

Accepted after one revision by the editors of the special issue.

N. Soeffker (✉) · Jun-Prof. Dr. M. W. Ulmer · Prof. Dr. D. C. Mattfeld
Technische Universität Braunschweig, Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
e-mail: n.soeffker@tu-braunschweig.de

state. The combination of real-time responses, the pressure to use resources effectively, and the need to anticipate potential future developments challenges service providers in their operational decision making. Service providers have to apply methods of prescriptive analytics that belong to the group of data-centric mechanisms (Kowalczyk and Buxmann 2014) and enable "automated decision making by offering advanced predictive capabilities" (Jarke 2014). The necessary decision support tools have to derive decisions nearly instantaneously in every state, incorporate the information provided by predictive analytics, and allow for a suitable use of the operational resources.

Because extensive calculations are inhibited by the real-time decision making requirement during the execution of the business process, calculations need to be shifted to a "learning" phase. The idea is to learn information offline and use the information in the online decision state without additional calculations necessary. The literature provides several learning methods associated with keywords like reinforcement learning (Sutton and Barto 1998), neuro-dynamic programming (Bertsekas and Tsitsiklis 1996), or approximate dynamic programming (Powell 2011). These methods show similar structures: A large number of simulations is conducted mimicking the process of decision making under exogenous information (for example, customer requests over the day). Here, the historical data provides possible exogenous information. The selected decisions and their short- and long-term contribution to the objective function are stored in an approximation architecture. To allow storage of the information, this architecture usually condenses the state information by means of aggregation and state space partitioning. A partitioning maps states to a set of representatives and evaluates these representatives. The partitioning is used to guide the simulations. The information for the representatives is frequently updated based on newly observed information. In the end, the "trained" architecture can then be used to select suitable decisions in the online execution.

One common offline learning approach in the literature is value function approximation (VFA), a method of approximate dynamic programming. VFA operates on the idea of maximizing the immediate revenue plus the expected future revenue in every state (Bellman's Equation, Bellman 1957). The expected future revenue is also called the *value* of the state immediately after choosing a decision. The VFA approximates these values by means of simulation. Applying VFA can be successful for small problems, but a transfer to more complex problems, for example from the field of dynamic vehicle routing, is challenging. For these problems, the space of potential states is vast and a partitioning of states is needed. Because of the non-linear combinatorics in the decision process, different areas of the state space are visited more often than others and some are never visited at all. These non-linear combinatorics occur, for example, due to the non-linear behavior of the routing as well as the general dynamism in the problem. Conventional state space partitionings as well as VFA-approaches assuming a parametric dependency between state and value therefore provide inferior approximations (Ulmer et al. 2018).

The reliability of the values' approximation decreases when the number of representatives is large because states are not observed frequently enough to obtain a good estimate. If the number of representatives is small, heterogeneous states are evaluated similarly. In both cases, the result is an impeded approximation and eventually inferior decision making. Further, the level of detail required varies over the state space. Some "important" areas in the state space may require a closer look while other areas are unimportant for the approximation process. The importance of areas may also shift because of changes in the decision making due to the learned values. Furthermore, some areas of the state space may be less represented or observed. Thus, an iterative method is needed that (1) focuses on the important areas in the state space, (2) subsequently adapts these areas over the approximation process, and (3) accounts for the accuracy of approximation of different areas of the state space.

In this paper, we propose such a method, the adaptive state space partitioning (ASSP). Our method iterates between state space partitioning and VFA. In each iteration, ASSP re-designs the state space partitioning based on the states observed in the last iterations of the VFA. It therefore focuses on the "important" areas. Because not all areas of the state space are covered equally, a handling of state outliers is required for decision making. Thus, we integrate a term reflecting approximation accuracy in the VFA to allow a reliable approximation. The overall result is an adaptive state space partitioning tailored to the state space required by the problem.

We provide a computational proof of concept for our method. We therefore draw on a prominent dynamic routing problem from the literature, the dynamic vehicle routing problem with stochastic service requests (VRPSSR) (Thomas 2007). We show how our method is able to achieve anticipation and an increase in solution quality while anticipation is not possible for a conventional state space partitioning of the same size. In a comprehensive analysis, we show how our method leads to adaptive shifts in the partitioning and how the consideration of approximation accuracy impacts both partitioning and solution quality. We further show that our method is able to reduce the architecture size compared to conventional architectures to obtain the same solution quality.

This paper is structured as follows. We present the general business process of service routing in Sect. 2. In

Sect. 3, we present the framework of Markov decision processes to model sequential decision making problems. In these two sections, we further describe and model the VRPSSR that we use for our proof of concept. The solution method value function approximation is described in Sect. 4. We motivate the need for a state space partitioning and present related literature in Sect. 5. In Sect. 6, we propose our approach ASSP. We then present our computational proof of concept in a computational study based on a problem from literature in Sect. 7. The paper ends with a conclusion in Sect. 8.

2 Business Processes with Uncertain Customer Requests

In this section, we discuss the general business process with uncertain customer requests. To this end, we draw on the generalized BPMN (business process model and notation) model in Sect. 2.1. In Sect. 2.2, we then provide the problem description of the VRPSSR as a special case of the general business process.

2.1 Business Process Model and Notation Model

Uncertain customer requests can be observed in a variety of business processes. For example, technician service providers receive calls during the day requesting timely repair. Other application areas are passenger transportation services such as dial-a-ride or taxis. Uncertain customer requests are further very common in the courier and parcel business where customers request the pickup or delivery of goods. All these business processes have in common that not all customers are known in advance but subsequently request over the course of day. Further, the customers expect a fast response to their requests. Finally, in these business processes, the services are conducted while new requests come in.

In the following, we describe the general procedure the providers conduct when a new customer requests service. To this end, we draw on the BPMN-notation. The process for a single customer is depicted in Fig. 1. The process consists of two entities: the customer and the service provider. Each entity is represented by a pool. The pool for the service provider contains two swim lanes: one for the service administration and one for the drivers conducting services. We view the process from a provider's point of view. Thus, we collapse the pool of the customer. The process starts with the customer's service request. This request can be issued via phone or via internet devices. At that point of time, the provider obtains the customer's data, for example the address and the requested service. This request is processed by the service administration of the service provider. Before the service administration can make a decision, he or she checks the statuses of the drivers. In some business processes, this status check is already automatized. In other business processes, the status has to be checked manually, for example by means of a phone call. Once the service administration receives the information about the drivers' statuses, service plans can be calculated. These plans determine whether the customer can be served at all and, if yes, which driver should serve the customer and how the drivers' routes should be updated to incorporate the service for the new customer. Once the plans are determined, both drivers and customer are informed. If the customer cannot be served, the customer is informed about the rejection of the service request and the process ends. If the customer can be served, service details are provided to the customer and a driver is informed about route updates. In an increasing number of modern business processes, the customer expects that the time span between the request for service and the reception of service details is very small. Thus, the time for calculating the plans is very limited. In the case of an acceptance of the service request, the process terminates once the driver informs the service administration about the conduction of service at the customer.

Notably, the described process is started whenever a customer requests service. Thus, it is repeated frequently during the day. Furthermore, the individual processes for the customers are interconnected because the services are all carried out by the same driver workforce. The combination of several subsequently incoming, unknown customer requests leads to a stochastic and dynamic problem.

2.2 The VRPSSR

In this section, we describe the dynamic vehicle routing problem with stochastic service requests (VRPSSR) as a special case of the described problem field. The VRPSSR and its variances have been frequently studied in the literature (Thomas 2007).

In this problem, a vehicle serves subsequently requesting customers in a service area. Due to working hours of the driver, the time horizon for serving customer requests is limited. The vehicle starts and ends its tour at a depot. Some customer requests are known in advance and have to be served during the time horizon. Other customers request service during the day. These requests are unknown for the service provider until their time of request. However, the (stochastic) distribution of requests over time and service area is known. Serving a customer requires a service time. While the vehicle is on the road serving customers, new customers request service. Because of the limited time horizon, usually not every customer can be served. Thus, each dynamic request can be either accepted for same-day

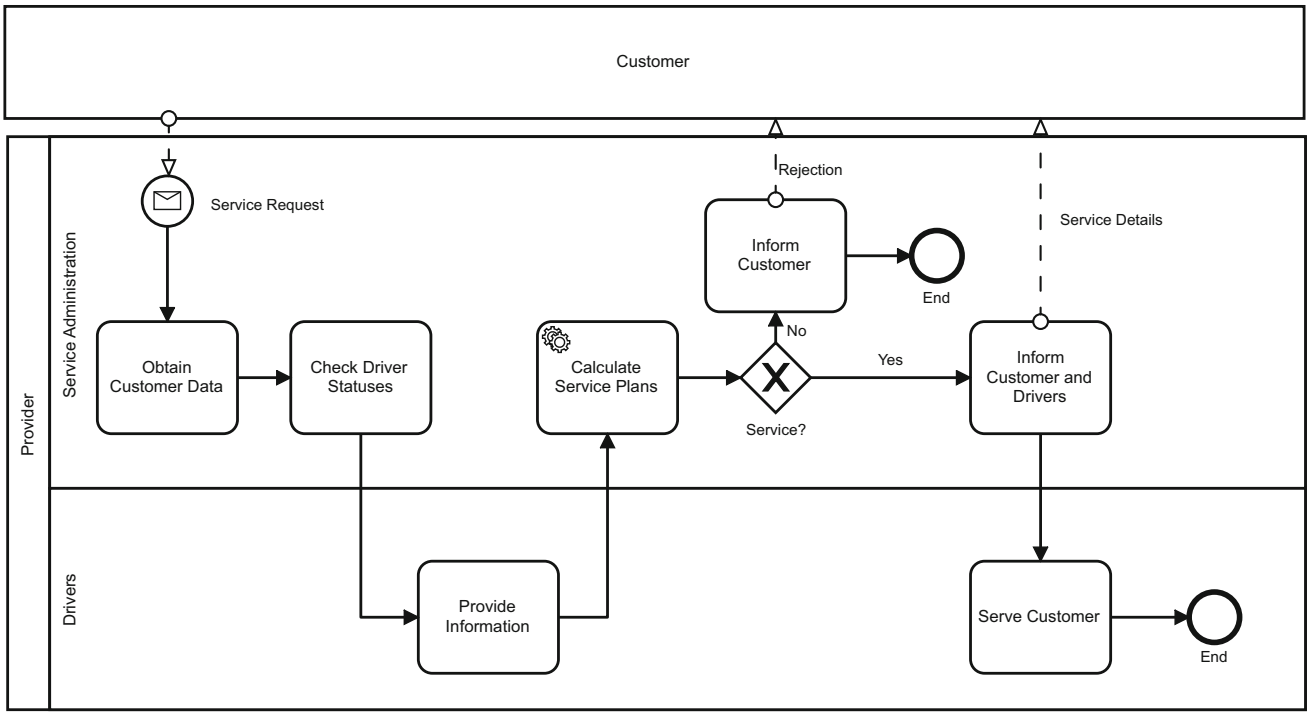

Fig. 1 BPMN of a generalized business process with dynamic customer requests

service or the customer is rejected. For this problem, it is assumed due to safety reasons that communication is prohibited while the driver is traveling between customers. Thus, the provider and driver communicate only when the driver finished service at a customer. Then, the service provider determines which subset of the new requests to accept for same-day service and how to include the accepted customers in the tour. When making a decision, the dispatcher has information about the current time in the horizon, the vehicle location, and the tour still planned for the vehicle. The dispatcher therefore knows how much of the resource time is still available and how much can be used for the future service of customers. The time between the end of serving all accepted customers and the end of the time horizon is also called "slack".[1] For the VRPSSR, the objective for the service provider is to maximize the expected number of accepted (and served) dynamic customer requests.

Notably, while we focus on the VRPSSR in this work, our work could be applied to other dynamic problems with stochasticity as well. Examples for other possible application areas include, for example, financial decisions, where prices or interest rates are subject to stochasticity and decisions have to be made under uncertainty about the future, or decisions about production of goods where demand is highly volatile or production machines are rather unreliable.

3 Markov Decision Process Models

The problem class described in Sect. 2 requires sequential decision making by the service provider due to the updated information about customer service requests. In this section, we recall the concept of Markov decision processes (MDPs, Puterman 2014) as a mathematical modeling framework for these types of problems. We then model the VRPSSR as an MDP.

3.1 Modeling Stochastic Dynamic Problems as Markov Decision Processes

Stochastic and dynamic problems can be modeled as Markov decision processes (Puterman 2014) as those provide the appropriate framework for subsequent decision making. In an MDP, subsequent decision points occur. In a decision point $k = 1, \ldots, K$, the situation can be described by the according state S_k. The state encapsulates all the information currently accessible for the decision maker. This information is problem-specific. In dynamic vehicle routing, this information usually comprises the point of time, the set of customers still to serve, and new requests.

[1] Because both point of time and slack are good indicators for the number of additional customers the vehicle can serve in the future, we will later use the combination of point of time and slack to aggregate our state space.

In every state, a decision x has to be chosen, for example about the assignment of new customers or the update of the current routes. Also, there is a reward $R(S_k, x)$ associated with the combination of state S_k and decision x that represents the immediate contribution of the decision to the objective function.

The combination of state S_k and decision x leads to a deterministic post-decision state (PDS) S_k^x. In the PDS S_k^x, the situation is changed due to the chosen decision, but there is no influence of a stochastic transition yet. After the PDS, stochastic information is revealed and the transition ω leads to the next decision state S_{k+1}. This sequence continues until $k = K$ at the end of the decision horizon.

A solution to an MDP is a policy π from the set of policies Π. A policy π is a sequence of decision rules X_k^π determining a decision $X_k^\pi(S_k)$ in every state S_k. The optimal policy $\pi^* \in \Pi$ maximizes the expected overall reward.

3.2 Markov Decision Process Model for the VRPSSR

For the VRPSSR, the vehicle has to start and end its tour at the depot. The customers known in advance C^{adv} have to be served, thus, an initial tour θ_0 including the locations of all customers in C^{adv} is created. A decision point k occurs when a customer was just served. The according decision state S_k contains information about the time t_k in the shift $T = [0, t_{\max}]$, the current location of the service vehicle l_k^v in the service area, the set of customers C_k that still have to be served, about the route currently planned for the vehicle θ_k, and about the set of new requests C_k^{rq} that occurred between decision point $k-1$ and decision point k. A state can therefore be described as $S_k = (t_k, l_k^v, C_k, \theta_k, C_k^{\text{rq}})$. Decisions $x(S_k)$ have to be made about the acceptance of the customer requests and, due to the routing component, about the update of the route. The customer requests that are accepted for same-day service are denoted $C_k^a \subseteq C_k^{\text{rq}}$. The route θ_k is then updated to θ_k^x to include C_k^a. The reward $R_k(S_k, x)$ is the number of accepted dynamic requests, that is $R_k(S_k, x) = |C_k^a|$. The post-decision state (PDS) S_k^x results from the combination of state S_k and decision x. It therefore contains information about the time (still t_k), the location of the vehicle (still l_k^v), about the updated set of accepted customer requests, that is $C_k^x = C_k \cup C_k^a$, and about the updated routing θ_k^x. A PDS can therefore be described as $S_k^x = (t_k, l_k^v, C_k^x, \theta_k^x)$. Notably, the slack in S_k^x can be directly derived by $t_{\max}$, t_k, and the duration of θ_k^x. While the vehicle travels to the next customer and the customer is being served, the stochastic transition ω takes place and reveals new customer requests C_{k+1}^{rq}.

Travel times between two locations l^1 and l^2 in the service area are denoted by $t(l^1, l^2)$. Service time is denoted by t^s. When the vehicle reaches the next customer and the service of this next customer is completed, the next decision point $k+1$ occurs at $t_{k+1} = t_k + t(l_k^v, l_{k+1}^v) + t^s$ and the next decision state S_{k+1} is provided. The MDP terminates when the vehicle reaches the depot at the end of the time horizon after serving all customers.

In an MDP, decisions are made in every decision point. A solution to an MDP is a policy π that assigns a decision to take $x(S_k)$ to every state S_k. For our problem, the objective is to find a policy that maximizes the expected number of dynamic requests that are accepted for same-day service.

4 Value Function Approximation

For MDPs, the optimal policy π^* maximizes the expected sum of rewards over the decision horizon. Maximizing the expected overall rewards also means that in every decision point, the sum of immediate reward $R_k(S_k, x_k)$ and expected future rewards given the post-decision state is maximized as stated in Bellman's Equation (Bellman 1957):

$$X_k^{\pi^*}(S_k) = \arg\max_{x \in X(S_k)} \left\{ R(S_k, x) + \mathbb{E}\left[\sum_{j=k+1}^{K} R\left(S_j, X_j^{\pi^*}(S_j)\right) | S_k \right] \right\}.$$

The expected future rewards are also called value of the post-decision state ($V(S_k^x)$) as they describe the expected contribution to the objective function after a post-decision state S_k^x and, therefore, what it is worth to be in this post-decision state. For small MDPs, we could compute the value for each PDS recursively by means of dynamic programming. The value of a PDS then captures the expectation of the decision tree following the PDS. When the values of each PDS are available, the optimal policy can then choose the decision maximizing the sum of immediate reward and value of the resulting PDS. However, the space of possible PDSs is usually too large to calculate each possible PDS's value individually. Instead of dynamic programming, we therefore apply value function approximation (VFA), a method of approximate dynamic programming (ADP, Powell 2011). The functionality is depicted in Fig. 2.

Here, the values of the PDSs are not calculated, but approximated by means of simulation. This means that many decision horizons are simulated offline to provide the values for the online decision making. Since no further approximation takes place during the online application of the resulting policy, runtimes in the online decision application are negligible.

A VFA starts with initial values for the PDSs. Then, a set of sampled realizations of the MDP are simulated. The observed sum of rewards at the end of each sample run is

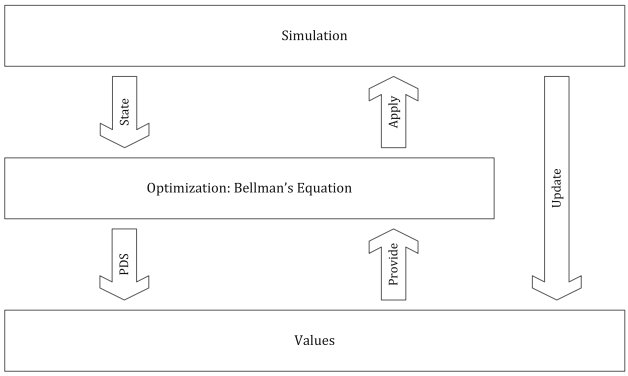

Fig. 2 Functionality of value function approximation (Soeffker et al. 2016)

used to update the approximated values of the PDSs. The updated values are then used for decision making via Bellman's Equation in the next iteration. Thus, the decision making is based on a more reliable information basis in each new iteration and the solution quality increases over the number of simulations. In general, VFA could be conducted assuming parametric dependencies between PDS features and the value (parametric VFA) or without assuming such dependencies (non-parametric VFA). For parametric VFA, the problem structure has to be known in advance to determine possible dependencies. In this article, we focus on non-parametric VFA to maintain a high flexibility regarding the application. Usually, values are approximated not for individual PDSs but for groups of PDSs depending on the chosen approximation architecture. We discuss the choice of the approximation architecture in the next section.

In the MDP, a PDS description contains all relevant information to describe it unambiguously. For the problem described above, the PDS space contains dimensions for the time and for locations of the vehicle and the customers that still have to be served. Because the dimensionality of the PDS space is generally large, an aggregation of the PDS space to a set of state features has to be performed to reduce the size for the VFA. Possible ways to aggregate are to "ignore a dimension, discretize it, or use any of a variety of ways to reduce the number of possible values of a state vector" (Powell 2011). This aggregation of MDP-states to features has to be performed carefully with a certain knowledge of the problem.

As mentioned earlier, for the VRPSSR, we aggregate the PDS to the features time and slack (Ulmer et al. 2015). These two features are relevant for the decision making as they both refer to the resource time. The feature time describes how much of the time horizon already passed (and cannot be spent anymore) and how much is still left. The feature slack describes how much of the remaining time is still free to be used for new customer requests. This leads to a two-dimensional vector space: one dimension for the point of time, the other dimension for the remaining slack.

5 State Space Partitioning

In this section, we first motivate in Sect. 5.1 why we need a state space partitioning for the application of a value function approximation. We then describe relevant approaches from the literature in Sect. 5.2.

5.1 Motivation

When applying value function approximation, the value of being in a PDS has to be approximated for each possible PDS. Approximating each value individually usually requires high memory and high computational efforts as a sufficiently high number of observations is necessary to adequately approximate a value. If the number of PDSs to evaluate is too large and each one is evaluated individually, then the number of observations per PDS is usually not large enough, so no values are learned or the values are not reliable. Furthermore, the number of value observations impacts the decision making during the approximation process. If values are observed sparsely, decision making may be misguided leading to a vicious circle of poor decision making and wrong approximations. Thus, the state space is usually partitioned to a set of representatives. Observed PDSs are then assigned to their closest representative and the value of each representative is updated by the observed assigned PDSs.

Not every possible PDS is actually relevant in practice and some PDSs are similar enough that they could be evaluated in the same way. Therefore, it is beneficial to find partitionings of the state space that are focused on the relevant areas of the state space, grouping similar states to the same representative and different states to different representatives. However, typical approaches are to partition the state space like a lookup table with equidistant intervals such that the representatives are equally distributed in the state space. Furthermore, the importance of areas in the state space may shift due to the approximation process. Because static and equidistant partitionings are not able to meet these requirements, several alternative approaches were proposed in the literature. We will discuss these approaches in the next section.

5.2 Related Literature

In this section, we present different state space partitioning approaches from the literature.

The most typical approach to achieve a state space partitioning is a static lookup table where intervals are chosen for the domains of all features that describe a state. Here, all states that are within the same intervals fall into the same partition and are assumed to be similar. However, larger intervals may lead to partitions that aggregate dissimilar states while smaller intervals lead to many partitions in which values have to be determined. Due to this tradeoff and because details about the problem structure may not be available in such detail, it is challenging to find a suitable state space partitioning in advance. A possible alternative are weighted lookup tables (see Powell 2011) where several static lookup tables are created with different interval lengths and the according values are combined in a weighted manner. This approach may be inefficient if areas of the state space are not visited, and it is still necessary to have problem-specific knowledge in advance to determine the interval sizes. Also, since multiple lookup tables have to be stored, the memory requirements are high and this approach is therefore not applicable to more complex problems (Ulmer et al. 2018).

A recent approach for state space partitioning for dynamic vehicle routing problems was proposed by Ulmer et al. (2017) who suggest to start with a lookup table with large interval sizes and successively split the entries of the lookup table if there are a lot of states aggregated in the entry and/or if the states aggregated in the entry turn out to be rather dissimilar. Since changes to the lookup table are made during the approximation of values, it is an adaptive approach. A similar idea can be found in Whiteson et al. (2007) who extend work of Sherstov and Stone (2005) and propose an adaptive approach and split entries whenever a learning plateau of the values is reached, that is, the values do not change anymore. As we show in our computational study, relevant areas may become irrelevant and vice versa over the approximation process. Even though these methods adapt to the approximation process, they are not able to revert partitioning decisions once made, even when the focus of the partitioning should shift. This requires unnecessary storage effort and may impede the approximation.

Kishima and Kurashige (2013) propose to completely ignore certain state features and Jin et al. (2009) and Sarkar et al. (2000) build on the fact that for some problems, a certain final state has to be reached and that the problem can be either decomposed into smaller problems (Jin et al. 2009) or that some states are very likely to lead to the same next state (Sarkar et al. 2000). These approaches, however, are very specific and require specific prior problem knowledge. Ikonen et al. (2016) reduce the state space for a process control problem by an iterative clustering. While the general idea of that work is very close to the one proposed here, the authors assume that there is a limited number of possible decisions and that the desired output is a policy consisting of state-decision pairs. In the iterations, they therefore use simulation to determine the transition probability matrices between state-decision pairs and the subsequent state. For many dynamic problems like the one we will apply, this is not realistic. Because customers may request at arbitrary times and arbitrary locations, transition probabilities cannot be calculated. While there is a lot of work on state space partitioning in the area of reinforcement learning, many of these ideas are not applicable to the problem we are approaching. In many problems tackled by reinforcement learning, the objective is to reach a certain state that is linked with high rewards (or the only positive reward) and the approach has to determine how to reach this state (Lee and Lau 2004). One example for this could be a maze in which the decision maker has to start at some location and has to find the exit. The state space consists of all locations and the decisions could consist of four different directions. Here, the objective is to reach one certain state which is assigned a high value. Furthermore, in many applications, the combination of state and decision is evaluated which is not applicable to typical problems in dynamic vehicle routing as there are too many possible states and decisions. In the case of our problem, the MDP is substantially more complex, the uncertainty is high, and the sizes of state and decision spaces are large.

The most recent approach for a problem-specific state space partitioning for a dynamic vehicle routing problem was suggested by Soeffker et al. (2016). Similar to the work presented in this section, Soeffker et al. (2016) apply the idea of using the observed PDSs of the simulation to generate to state space partitioning from it. To this end, observed states are clustered. The partitioning is defined in one step and then fixed. Our proposed approach shows similarities in the clustering approach although applying a different distance measure. However, our approach draws on iterative steps of clustering and considers the idea of handling outliers that may not be approximated accurately. The method of Soeffker et al. (2016) can therefore be seen as a special and simplified case of the proposed ASSP.

6 Adaptive Problem-Specific State Space Partitioning (ASSP)

In this section, we present our algorithm that iteratively creates a new state space partitioning based on the outcomes of the VFA. We first give a general overview over our method and then define the individual components.

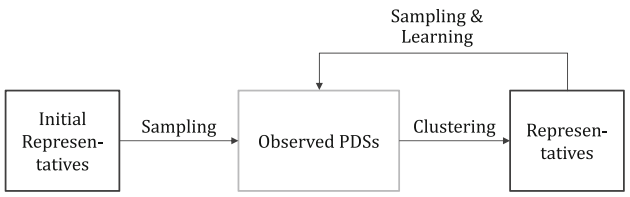

Fig. 3 Functionality of ASSP

6.1 Overview

The general idea of ASSP is to (1) focus on relevant areas of the state space and (2) iteratively shift the partitioning over the approximation process. ASSP therefore uses observed states in the VFA to update the state space partitioning and to use the updated partitioning in the next iteration of VFA. The procedure is visualized in Fig. 3. ASSP starts with an initial partitioning and initial representatives. These initial representatives are generated by sampling states. ASSP then uses this partitioning in a VFA where realizations of the MDP are sampled and the value for the representatives are learned. During the VFA, ASSP stores the observed states and then uses the states to generate a new partitioning and new representatives. This partitioning is then again used for VFA, etc. This continues until a termination criterion is met.

Over the iterations, the partitioning is subsequently moved to important areas of the state space and because these areas are represented in more detail, the VFA's quality increases leading in the best case to a "virtuous circle" of better representation and better approximation. That is, before the first iteration, the decision making is based on the initial representatives. Since the initial representatives are located without knowledge of the problem structure, decision making is likely to be obstructed. The observed states therefore do not provide a very high solution quality, but are the basis for choosing new representatives. In the next round of decision making and learning, however, the observations collected are already based on more suitable representatives and the then following iteration of representatives will be located even better.

The heterogeneous partitioning in the state space and the focus on important areas lead to the problem of handling occasional outliers. We may observe PDSs far away from the next representative. An evaluation of these PDSs with the value of the representative may be misleading. Thus, we introduce an approximation accuracy term (ACT) incorporating the distance between observed PDS and representative in the evaluation of the PDS.

In the next Sects. 6.2 and 6.3, we motivate and explain the ideas of iterative clustering and ACT. In Sect. 6.4, we provide the algorithmic details of ASSP.

6.2 Iterative Clustering

In the following, we describe how we generate the representatives based on the observed PDSs. Before starting the process, a number of representatives n is externally defined. We further define a distance measure between two PDSs $(\cdot,\cdot) \rightarrow \mathbb{R}_+$. This distance measure is needed to map PDSs to the closest representatives and to allow a clustering of the PDSs to representatives.

For the initial clustering, we start with an empty partitioning. We then run a number of m simulation runs. Because of the empty partitioning, all PDSs values are mapped to zero. Thus, for these first m runs, a myopic policy is used for decision making. Over the simulations, we collect all selected PDSs. At the end of the m simulation runs, we apply a clustering algorithm on the observed PDSs to determine the set of representatives $\mathcal{R}$. The clustering algorithm can be chosen arbitrarily. However, in our computational study, we use n-medoid.[2] This procedure determines n PDSs as representatives in a way that the sum of distances between PDSs and closest representatives is minimal. We chose this clustering approach because it allows us to choose an existing PDS as representative. In contrast to clustering-methods operating on averages over the PDSs, this procedure is also suited in case a PDS contains non-numerical features.

Starting from the initial clustering and representatives $\mathcal{R}$, we then iteratively repeat the following VFA-procedure: All representatives are assigned an initial value. In our computational study, we initialize with a high value to enforce exploration of unobserved representatives. Again, m simulation runs are conducted. We use Bellman's Equation in each state. After each approximation run, the representatives with observed associated PDSs are updated with the corresponding values, for example, by means of a running average. Again, all observed PDSs are stored. Because of the different decision making, we observe different states than in our previous iteration. After the m simulation runs, the observed PDSs are again clustered providing new representatives. This procedure continues for I iterations. To obtain values for the final set of representatives, we run a potentially more thorough approximation of M simulation runs. Eventually, we obtain the set of representatives $\mathcal{R}$ and their values. We can then use this set and the values for instant decisions in the online decision state.

[2] In the literature, the method is often referred to as k-medoid. However, because in MDPs, decision points are denoted by variable k, we chose n to avoid a duplicated use of variables.

6.3 Approximation Accuracy

When methods of VFA have to be applied to problems with larger state spaces, the number of representatives is crucial for memory, computational efforts, and approximation success. It therefore may be necessary to restrict the number of representatives to a small number (compared to the possible number of states). Our method suggests to move these representatives to important areas of the state space. However, if the representatives focus on these areas of the state space, others may not be depicted sufficiently anymore.

This leads to the challenge that the accuracy of approximation may vary for different states. Observed states are mapped to the closest representative. States close to a representative may have a more accurate evaluation compared to states further away. Using the same value of the representative for these states may therefore impede the decision making and the approximation process.

To consider this potential "inaccuracy" caused by the distance between the PDS and its representative, we introduce an *approximation accuracy correction term* (ACT). The ACT is inspired by the idea of penalizing "choosing actions that deviate from the external domain expert" (Powell 2011). In our case, the domain expert is replaced by the automated state space partitioning. Figure 4 depicts how the estimation of the value of a PDS changes when ACT is applied. In this example, the state space contains two dimensions. It is therefore directly connected to the proposed aggregation to time and slack for the VRPSSR.

In Fig. 4, two representatives are shown in dark colors as well as a newly observed PDS whose value has to be estimated. The PDS is mapped to the closest representative. Without ACT, the PDS is evaluated with the value of this representative. To account for a potential inaccuracy, the ACT is now subtracted from the value. The size of the ACT is based on the distance between PDS and representative.

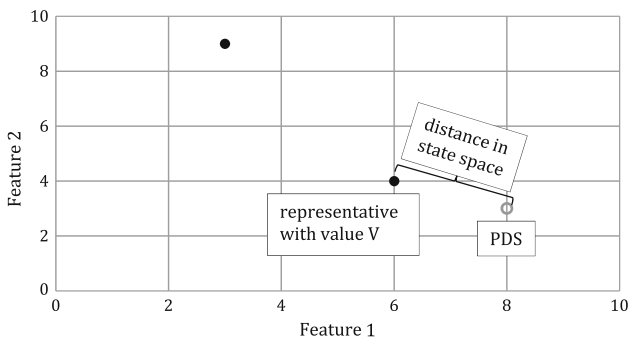

Fig. 4 Impact of ACT on estimated value for potential PDS

The ACT can be chosen arbitrarily. In our computational study, we use a linear term over the distance. To tune the magnitude of the ACT, we introduce a factor p. Given an ACT-factor p, Bellman's Equation is then described as

$$\arg\max_{x\in X(S_k)}\left\{R(S_k,x)+V\left(Rep\left(S_k^x,\mathcal{R}\right)\right)-p\cdot d\left(S_k^x,Rep\left(S_k^x,\mathcal{R}\right)\right)\right\}$$

with $Rep(S_k^x,\mathcal{R})$ being the representative in $\mathcal{R}$ that is associated to PDS S_k^x.

Notably, the magnitude of the ACT impacts the balance between exploration and exploitation of the state space. A small ACT leads to exploration because outliers are evaluated relatively high. A large ACT leads to exploitation because outliers are penalized significantly. We will later show this impact in our analysis.

6.4 Algorithm for ASSP

In this section, we present the algorithm for ASSP in pseudocode. The procedure is shown in Algorithm 1. Inputs are the number of representatives n, the number of iterations I, the number of simulations per iteration m, the number of simulations for the final iteration M, and the ACT-parameter p. The algorithm then traverses the iterations starting in the initial iteration $i = 1$. In this iteration, the set of representatives $\mathcal{R}$ is empty. For each iteration, a set of observations $\mathcal{O}$ is collected.

At the beginning of each iteration, the values V are initialized with function $Initialize()$. Except for the final iteration $(i = I + 1)$, m simulations are conducted per iteration. In each simulation run, the algorithm collects the set of observed PDSs $\mathcal{S}^x$ and the set of obtained rewards R^{set}. It then subsequently generates decision states by sampling exogenous information in function $GenerateExogeneous(S_k,x)$. This function depends on the previous state and the selected decision. To determine a decision in a state, Bellman's Equation is applied combined with the ACT.

After each simulation run, the values are updated with function $Update(V,\mathcal{O},\mathcal{S}^x,R^{\text{set}})$. The update depends on the previous values and observations as well as the observed PDSs and rewards. At the end of each iteration except for the final one, a new set of representatives is generated based on function $Clustering(\mathcal{O},n)$. Input for this function are the observed PDSs and the number of representatives n. After the final iteration, the algorithm terminates and returns the final set of representatives $\mathcal{R}$ and their values V.

Algorithm 1: Adaptive State Space Partitioning

Input : Number of Representatives n, Number of Iterations I, Number of Simulations m, Number of Final Approximation Runs M, ACT-Parameter p.
Output : Set of Representatives $\mathcal{R}$, Value Function V

// Initialization
$i \leftarrow 1$
$\mathcal{R} \leftarrow \emptyset$ // Initialization of the Representatives
while $(i \leq I+1)$ // Run I Iterations (plus Initial and Final)
do
  $\mathcal{O} \leftarrow \emptyset$// Initialization of the Observations
  $V \leftarrow Initialize()$ // Initialization of the Values
  if $(i == I+1)$ **then** $m \leftarrow M$ // M Runs In Final Iteration
  while $(j \leq m)$// Run m MDP-Simulations
  do
    $k \leftarrow 0$ // Decision Points
    $S_0^x \leftarrow \emptyset$
    $\mathcal{S}^x \leftarrow \emptyset$ // Initialization Set of PDSs
    $R^{\text{set}} \leftarrow \emptyset$ // Initialization Set of Rewards
    $R_0 \leftarrow 0$
    while $(S_k^x \neq S_K)$ // Stop when Termination State S_K is Reached
    do
      $k \leftarrow k+1$
      $\omega_k^i \leftarrow GenerateExogenous(S_k, x)$ // Generate Exogenous Information
      $S_k \leftarrow (S_{k-1}^x, \omega_k^i)$ // Generate new Decision State
      $v \leftarrow 0$ // Initialize Value
      for all $x \in \mathcal{X}(S_k)$ // Bellman's Equation plus ACT
      do
        $S_k^x \leftarrow (S_k, x)$
        if $(i == 1)$ **then** $v_{\text{temp}} \leftarrow R(S_k, x)$
        else $v_{\text{temp}} \leftarrow R(S_k, x) + V(Rep(S_k^x, \mathcal{R})) - p \cdot (d(S_k^x, Rep(S_k^x, \mathcal{R}))$
        if $(v_{temp} > v)$ **then**
          $v \leftarrow v_{\text{temp}}$
          $x^* \leftarrow x$
        end
      end
      $S_k^x \leftarrow (S_k, x^*)$ // Observed PDS
      $R_k \leftarrow R_{k-1} + R(S_k, x)$
      $\mathcal{S}^x \leftarrow \mathcal{S}^x \cup \{S_k^x\}$
      $R^{\text{set}} \leftarrow R^{\text{set}} \cup \{R_k\}$
    end
    // Update Observations and Values
    $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{S}^x$
    if $i > 1$ **then** $V \leftarrow Update(V, \mathcal{O}, \mathcal{S}^x, R^{\text{set}})$
    $j \leftarrow j+1$
  end
  // Clustering
  if $i < I+1$ **then** $\mathcal{R} \leftarrow Clustering(\mathcal{O}, n)$// Determining Representatives for Next Iteration
  $i \leftarrow i+1$
end
// Termination
return *Final Representatives* $\mathcal{R}$*, Final Value Function* V

7 Computational Study

In this section, we present our proof of concept by means of a computational study. We draw on a dynamic vehicle routing problem with stochastic customer requests (Thomas 2007) that was described and modeled earlier. In the following, we first describe the design of experiments, that is, instance details, the benchmark policies, and the tuning. We then compare the solution quality to the benchmarks and conduct an analysis of the functionality of our proposed method.

7.1 Design of Experiments

In this section, we present the instances, the parameter tuning, and the benchmarks.

7.1.1 Instances

We assume a service area of 20 km × 20 km with a depot located in the center. The time horizon T available for service is 480 min and the time required for the service at a customer t^s is 5 min. The distance between two customers

in the service area is Euclidean and the vehicle drives at a constant speed of 25 km/h. A total of 60 customer requests is expected per day. We set the Degree of Dynamism (Larsen et al. 2002) to 75%. That means that on average 25% (or 15 customers) of the customers are known in advance and 75% (or 45 customers) request over the course of the day. The temporal distribution of the dynamic requests is uniform over the time horizon and the spatial distribution of all customer requests is uniform over the service area.

7.1.2 Parameter Tuning

We tune our method as follows. As pointed out earlier, an aggregation of PDSs is necessary. As described, we aggregate the features to the current point of time and the slack, that is, the remaining time budget after the vehicle returns to the depot after visiting all customers in the tour. For the routing component, we apply a cheapest insertion heuristic for both the initial tour as well as for future customers that are inserted. This means that in a decision point with r new customer requests, there are 2^r possible options to accept or reject customer requests. For each of these options, a routing update is determined and the decision is made based upon the immediate reward and the expected future rewards of the PDS the decision would lead to.

Based on preliminary tests, we set the number of representatives to $n = 200$ and the number of approximation runs per iteration to $m = 300$. In contrast to many adaptive approaches from the literature, we fix this number of representatives. In the computational study, however, we also provide a small section about the impact of the number of representatives. In every state, we apply pure exploitation, that is, always follow Bellman's Equation. We update values of observed states with the running average over all observed values associated with the representative.

For the approximation accuracy correction term, we choose a term that is deducted from the sum of immediate and expected future rewards. Here, this term consists of the distance between the PDS to be evaluated and the corresponding representative that is multiplied with a constant factor p between 0 and 0.4 in steps of 0.05.

We set the number of iterations to $I = 10$. After these iterations, we use the final clustering for a longer approximation phase of $M = 97{,}000$ runs resulting in a total of 100,000 simulation runs. Because the approximation process depends on the sampled realizations, we conduct each test five times and use averages over the solution quality. We select the tuning leading to the highest average solution quality, a factor of $p = 0.05$. For evaluation, we run 10,000 additional evaluation runs without updating the representatives.

7.1.3 Benchmarks

In this section, we describe the two benchmark policies we use. One policy provides an estimate of our obtained solution quality for the problem at hand. The other policy is chosen to analyze the value of our ASSP-procedure.

For the problem, we test a myopic policy that does not anticipate future events, but only maximizes the immediate rewards available through customer acceptances. This benchmark is applied to set our results in context and to highlight the advantage of anticipation as well as the use of value function approximation. In addition to the myopic benchmark, we also apply test settings with value function approximation to analyze the influence of the iterative clustering combined with the ACT. Therefore, we apply a VFA using equidistant representatives that are not moved during the approximation phase. These representatives are equally distributed in the feasible time-slack-space. The approximation phase also consists of 100,000 simulation runs and the evaluation phase also consists of 10,000 simulation runs. The number of representatives is also 200 representatives. We test this VFA with and without an ACT. The best factor p for the equidistant representatives is 0.1.

7.2 Comparison with the Benchmarks

In this section, we first provide the results in terms of the solution quality for our benchmarks and the best parameter setting for our approach. We then demonstrate the efficiency of the ASSP compared to equidistant representatives.

7.2.1 Solution Quality

In the following, we analyze the objective value of the MDP, the average number of accepted customers per day for the 10,000 evaluation runs. The results are shown in Fig. 5 for the myopic acceptance policy, for the equidistant representatives without an ACT (0.0 p) and with the best

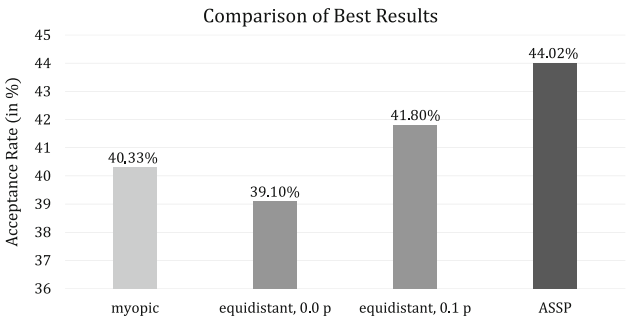

Fig. 5 Acceptance rates for the myopic policy, for equidistant representatives with an ACT of 0.0 and 0.1, and for the ASSP

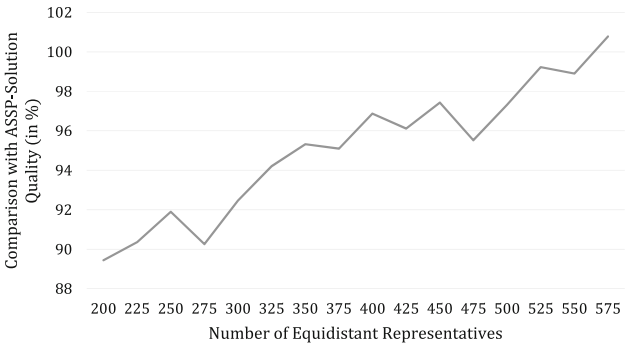

Fig. 6 Comparison of solution quality of ASSP (200 representatives) with solution qualities of equidistant representative sets

ACT (0.10 p) as well as for the ASSP. We further calculate the average improvement of the ASSP over the policies in percent. We calculate the improvement as:

$$\frac{\text{ASSP} - \text{Benchmark Policy}}{\text{Benchmark Policy}}. \quad (1)$$

The non-anticipatory myopic approach achieves an average acceptance rate of 40.3%. The ASSP significantly outperforms the myopic policy and achieves about 44.0% on average. This is an improvement of 9.15%. We observe that the VFA with equidistant representatives and without an ACT is not able to achieve anticipation and only reaches 39.1%. The ASSP achieves an improvement of 12.59% compared to this policy. Thus, the number of representatives is not sufficient for anticipation if they are equally distributed in the service area. If an ACT is applied in combination with the equidistant representatives, the solution quality increases to 41.8% showing the merits of the ACT. However, ASSP still achieves an improvement of 5.31% over this approach.

7.2.2 State Space Efficiency

In order to demonstrate the efficiency of the ASSP, we now analyze how many equidistant representatives are needed to reach the solution quality reached by ASSP. To this end, we subsequently increase the number of representatives of the equidistant approach in steps of 25. We compare the achieved solution quality with the value for ASSP with 200 representatives. We show the development in Fig. 6. On the x-axis, the number of equidistant representatives are shown. On the y-axis, the percentage of the ASSP-value is shown. A percentage of 100% indicates the same value as ASSP.

We observe a general increase in solution quality with respect to the number of representatives. This can be expected because with an increasing number, we obtain a better coverage of the state space. Because the approximation process is impacted by the stochastic realizations, we occasionally observe small decreases, for example for 275 representatives.

However, even with twice the number of representatives than ASSP, the solution quality is still worse. The same solution quality is only achieved with 575 representatives. Thus, the ASSP is more efficient in the requirement of representatives and stored values. For the same solution quality, the ASSP reduces the required number of representatives by about 65%. This is an important observation because an equidistant representation may run into memory issues for larger and more complex state spaces. If ASSP requires fewer representatives, it may alleviate some of these issues.

7.3 Analysis

We now analyze the functionality of the ASSP in detail. We first provide insight into the structure of the resulting state space partitioning. We then depict the impact of the ACT-factor p on the solution quality and show when ASSP is especially beneficial.

7.3.1 Structure of State Space Partitioning

In this section, we provide insights into the resulting state space partitioning. Because we run each setting five times, we focus on the instance where the objective value is closest to the mean over the five instances. First, we demonstrate how the state space partitioning changes with respect to the iterations. To this end, we compare the set of representatives of the first and last iteration. Figure 7 shows both sets in the time-slack space for the ASSP with an ACT-factor of $p = 0.05$. The x-axis depicts the time. The y-axis depicts the slack. The grey dots represent the first set of representatives. The black dots represent the tenth set.

First of all, we observe that regardless the iteration, the representatives show a general tendency. With increasing time, the slack decreases. This can be expected since more

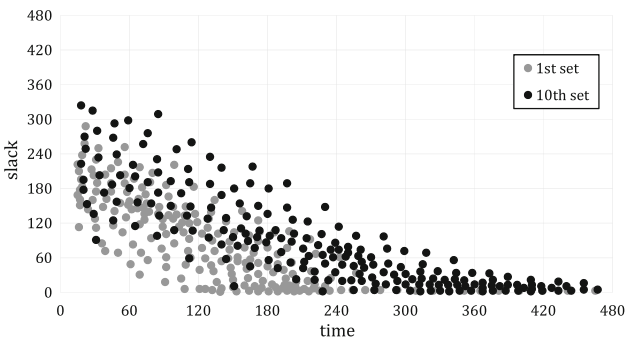

Fig. 7 First and last set of ASSP-representatives, $p = 0.05$

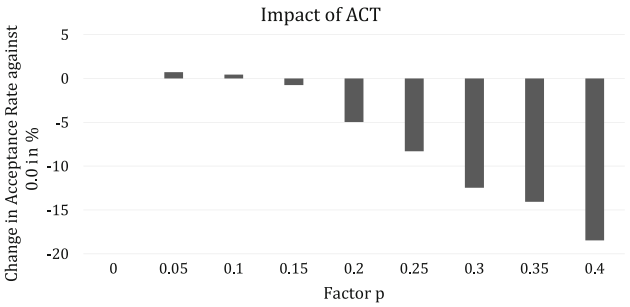

Fig. 8 Solution quality for a varying ACT-factor compared to ASSP without ACT

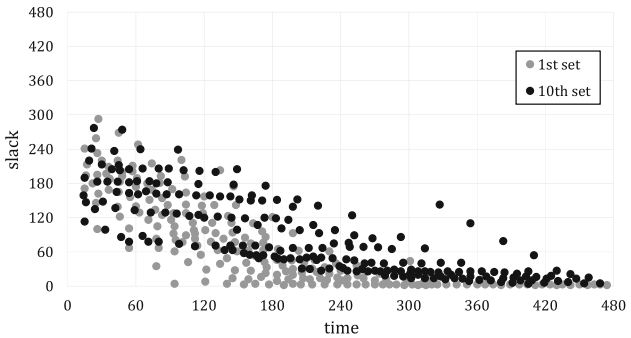

Fig. 9 First and last set of ASSP representatives, factor $p = 0.40$

customers are accepted over the day and the slack decreases. Our method is able to discover this phenomenon and adapts its representation to it. If we compare the sets of representatives for the first and tenth iteration, different parts of the state space are covered. We observe an upward shift. The tenth set of representatives covers areas where at the same time more slack is available compared to the first set. This indicates a change in the policy. The first policy leads to more consumption of the slack in the beginning while the tenth policy saves slack to accept customers later in the day. Thus, different areas of the state space are observed in these two cases.

7.3.2 *Impact of ACT-Factor*

In order to show the impact of the ACT on the resulting solution quality, we vary the factor p from 0.0 to 0.4 in steps of 0.05. We benchmark the results with ASSP without an ACT. We show the results in Fig. 8. The x-axis depicts the ACT-factor p. The y-axis depicts the solution quality compared to ASSP without an ACT. We observe an increase and then a constant decrease. This indicates the tradeoff between exploration and exploitation. Without an ACT, outliers are evaluated the same as the closest "regular" state. They may be selected frequently leading to a poor solution quality. With a larger ACT, outliers are less frequently observed even if they are promising. This impedes the exploration of the state space as we will highlight later in this section. We see that for our instance setting, a moderate ACT balances this tradeoff and leads to the highest solution quality.

We now analyze how the ACT impacts the exploration of the state space. To this end, we compare the representatives using an ACT-factor of $p = 0.05$ as depicted in Fig. 7 with the set of representatives using a factor of $p = 0.40$ which is depicted in Fig. 9. We recall that a factor of 0.40 resulted in a poorer solution quality.

As for ASSP with $p = 0.05$, we observe a shift in the representatives between the first and tenth set. However, the shift is less distinct. Furthermore, the "cloud of

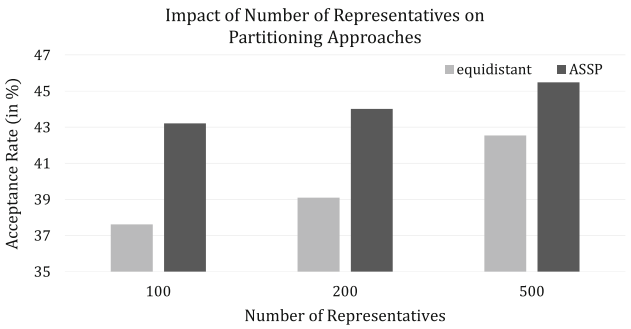

Fig. 10 Acceptance rates for different numbers of representatives

representatives" is more narrow if a factor $p = 0.40$ is applied. For a factor of $p = 0.05$, we have representatives with slack of more than 300 min. Also, parts of the state space with slack of about 0 min are already covered at a time of 200 min. For $p = 0.4$, we observe that the representative with the highest slack has a slack less than 300 min and that the first representatives with a slack of about 0 min occur only around a time of 360. This result indicates that a too high ACT impedes exploration of the state space and therefore the discovery of more valuable states and policies.

7.3.3 *Impact of Number of Representatives*

In our main study, we set the number of representatives to 200 based on preliminary tests. In the following, we analyze how the number of representatives impacts the performance of our method as well as the benchmarks. In Fig. 10, we depict the solution quality for 100, 200, and 500 representatives for equidistant representatives without ACT and for ASSP representatives.

It can be seen that both approaches provide increasing solution qualities with an increasing number of representatives. ASSP outperforms equidistant representatives regardless the number of representatives. We further observe that even ASSP with only 100 representatives is

able to outperform the results of 500 equidistant representatives. Thus, ASSP enables a better solution quality with substantially fewer representatives. Another observation is that the differences decrease with increasing number of representatives. The more representatives are given, the better is the coverage of the relevant state space areas even for the equidistant representatives.

8 Conclusion

With an increasing need to tackle dynamic problems, solution methods for these have to be able to cope with their specific characteristics. To apply value function approximation (VFA), a suitable state space partitioning is needed. Because the suitability of a partitioning is highly problem-specific, we have proposed an adaptive state space partitioning (ASSP). This procedure adaptively generates a state space partitioning by iterating between the VFA and the state space partitioning. In the VFA, the current partitioning is used to store the values and guide the optimization. In this context, potential state outliers are penalized due to the possible approximation inaccuracy.

We have shown the advantages and analyzed the functionality of ASSP for a dynamic vehicle routing problem from the literature. In a computational study, we have shown that ASSP outperforms benchmark policies and is able to iteratively adapt to the problem specifics. Furthermore, ASSP is able to reduce the required size of the state space representation substantially. In a detailed analysis, we depicted the development of the state space partitioning and highlighted the impact of the ACT.

Future research may tackle both method and problem complexity. As our analysis indicates, the ACT has a significant impact on the solution quality and the state space representation structure over the iterations of the ASSP. Future research may aim on automatically altering the ACT-factor with respect to the approximation process to shift from exploration to exploitation over the iterations. If non-numerical state features are used, our approach can still be applied if a suitable distance measure for the clustering can be found. Also, ASSP may be analyzed for different clustering methods and different distance measures should be tested in future work for their suitability for both problem and approach as other approaches may provide different solution structures and solution qualities. Furthermore, as our proof of concept indicates, ASSP is able to obtain anticipatory solutions with a very efficient state space representation compared to conventional representations. In general, many complex decision-making problems that require fast decision-making could benefit from approaches applying offline simulation to conduct the learning of values. The proposed approach may therefore suitable to be transferred to problems of higher state space complexity, for example with fleets of vehicles or customer time-windows. In that case, a set of routes has to be considered when making a decision and each customer could be served by different drivers or rejected. Also, it can be assumed that many more customers will request service resulting in a higher number of decision points during the decision horizon. The number of decision points is not relevant for the success of the approach which makes it very flexible to apply. Another problem extension in the direction of more realistic assumptions would be to consider additional sources of stochasticity, for example in the form of stochastic travel times. Then, current routing plans can violate the time limit and the solution approach would have to learn how to maintain a certain time buffer in order to arrive in time. Also, different customers may provide a different revenue to the service provider which would change the objective function. While most of the mentioned problem extensions affect the MDP, it could also be expected that, with the rise of modern technologies, the driver statuses can be checked automatically which would relax the assumption of decisions only being made upon the completion of service at a customer location.

References

Bellman R (1957) A Markovian decision process. Technical report, DTIC document

Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Scientific, Belmont

Ikonen E, Selek I, Najim K (2016) Process control using finite Markov chains with iterative clustering. Comput Chem Eng 93:293–308

Jarke M (2014) Interview with Michael Feindt on prescriptive big data analytics. Bus Inf Syst Eng 6(5):301–302

Jin Z, Liu W, Jin J (2009) Partitioning the state space by critical states. In: Fourth international conference on bio-inspired computing, 2009. BIC-TA'09. IEEE, pp 1–7

Kishima Y, Kurashige K (2013) Reduction of state space in reinforcement learning by sensor selection. Artif Life Robot 18(1–2):7–14

Kowalczyk M, Buxmann P (2014) Big data and information processing in organizational decision processes. Bus Inf Syst Eng 6(5):267–278

Larsen A, Madsen OBG, Solomon MM (2002) Partially dynamic vehicle routing-models and algorithms. J Oper Res Soc 53(6):637–646

Lee IS, Lau HY (2004) Adaptive state space partitioning for reinforcement learning. Eng Appl Artif Intell 17(6):577–588

Powell WB (2011) Approximate dynamic programming: solving the curses of dimensionality, vol 842. Wiley series in probability and statistics. Wiley, New York

Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley, New York

Sarkar S, Subramaniam A, Neogi R (2000) Study of methods for model reduction in transition systems. In: 2000 IEEE

international conference on systems, man, and cybernetics, vol 1. IEEE, pp 172–176

Savelsbergh M, Van Woensel T (2016) 50th anniversary invited article – city logistics: challenges and opportunities. Transp Sci 50(2):579–590

Sherstov AA, Stone P (2005) Function approximation via tile coding: automating parameter choice. In: International symposium on abstraction, reformulation, and approximation. Springer, Berlin, pp 194–205

Soeffker N, Ulmer MW, Mattfeld DC (2016) Problem-specific state space partitioning for dynamic vehicle routing problems. In: Nissen V, Stelzer D, Straßburger S, Fischer D (eds) Proceedings of Multikonferenz Wirtschaftsinformatik (MKWI) 2016. Universitätsverlag Ilmenau, Ilmenau, pp 229–240

Speranza MG (2018) Trends in transportation and logistics. Eur J Oper Res 264(3):830–836

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction, vol 1. MIT Press, Cambridge

Thomas BW (2007) Waiting strategies for anticipating service requests from known customer locations. Transp Sci 41(3):319–331

Ulmer MW, Brinkmann J, Mattfeld DC (2015) Anticipatory planning for courier, express and parcel services. In: Dethloff J, Haasis HD, Kopfer H, Kotzab H, Schönberger J (eds) Logistics management. Springer, Cham, pp 313–324

Ulmer MW, Mattfeld DC, Köster F (2017) Budgeting time for dynamic vehicle routing with stochastic customer requests. Transp Sci 52(1):20–37

Ulmer MW, Soeffker N, Mattfeld DC (2018) Value function approximation for dynamic multi-period vehicle routing. Eur J Oper Res 269(3):883–899

Whiteson S, Taylor ME, Stone P (2007) Adaptive tile coding for value function approximation. Computer Science Department, University of Texas at Austin, Austin