Computers and Operations Research 106 (2019) 260–279

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Dynamic Lookahead Policies for Stochastic-Dynamic Inventory Routing in Bike Sharing Systems

Jan Brinkmann*, Marlin W. Ulmer, Dirk C. Mattfeld

Technische Universität Braunschweig, Braunschweig 38106, Germany

ARTICLE INFO

Article history:
Received 15 June 2017
Revised 18 May 2018
Accepted 4 June 2018
Available online 15 June 2018

Keywords:
Bike sharing
Approximate dynamic programming
Dynamic vehicle routing
Lookahead policies
Value function approximation
Policy search

ABSTRACT

We present the stochastic-dynamic inventory routing problem for bike sharing systems (SDIRP). The objective of the SDIRP is to avoid unsatisfied demand by dynamically relocating bikes during the day. To anticipate potential future demands in the current inventory decisions, we present a dynamic lookahead policy (DLA). The policy simulates future demand over a predefined horizon. Because the heterogeneous demand patterns over the course of the day, the DLA horizons are time-dependent and autonomously parametrized by means of value function approximation, a method of approximate dynamic programming. We compare the DLA with conventional relocation strategies from the literature and lookahead policies with static horizons. Our study based on real-world data by the bike sharing system of Minneapolis (Minnesota, USA) reveals the benefits of both anticipation by lookaheads as well as the time-dependent horizons of the DLA. We additionally show how the DLA is able to autonomously adapt to the demand patterns.

1. Introduction

We consider a station-based bike sharing system (BSS) where users can rent and return bikes spontaneously. Stations are distributed in the city. BSSs are particularly used by commuters and/or as a complement to public transportation. We typically observe one-way trips where rental and return demands do not occur at the same station. Further, rental and return demands are uncertain and subject to heterogeneous spatial-temporal patterns (Büttner et al., 2011). The success of BSSs strongly depends on two factors: density of stations within the city and a reliable availability of bikes and free bike racks any time and at any station (Gauthier et al., 2013). At an empty station, rental demands cannot be served because of the absence of bikes. At a full station, return demands cannot be served because all bike racks are used. It is mandatory to return a bike at a station to indicate that the trip has ended and that the bike is available again. Thus, the users need to approach neighboring stations. To allow a sufficient number of racks and bikes at every station and time, providers dynamically dispatch transport vehicles to relocate bikes between stations.

With the advent of IT-supported station-based BSSs, it quickly became apparent that relocating bikes is mandatory in order to satisfy user demand. The satisfied user demand in terms of bike trips performed is recorded in great detail for invoicing. The provision of this anonymized data to the public has attracted data analysis as well as optimization research with the goal of either maximizing service level or minimizing relocation costs. Data analysis provides insights into demand patterns (Borgnat et al., 2011; O'Brien et al., 2014; Vogel et al., 2011). Further, detailed data of user bike trips is aggregated into typical bike flows (Vogel et al., 2017). Optimization counteracts these flows by means of bike relocations. Static deterministic models assume bike relocations during periods of non-activity, i.e., over night. Here, flows are incorporated in order to determine appropriate distributions of bikes as starting point for the next morning (we refer to Espegren et al., 2016 for a survey). It must be noted that over-night activities can at best empty or fill a station to its number of bike racks provided. This maximum number of bikes typically does not suffice in order to satisfy for instance commuting demand over the day. Therefore, periodic deterministic models aim at optimizing relocation activities over the course of the day (Brinkmann et al., 2016). A solution derived from such a model, however, may at best serve for the determination of regular day to day relocation activities in terms of service network design. Static models, based on aggregated historical data, will not be able to accurately depict future bike relocations in the real-world. To this end, an operational control is needed to address realized user demand on short notice.

The problem at hand is an inventory routing problem (IRP) and contains both stochasticity and dynamism (Coelho et al., 2014b). As

* Corresponding author.
E-mail addresses: j.brinkmann@tu-braunschweig.de (J. Brinkmann), m.ulmer@tu-braunschweig.de (M.W. Ulmer), d.mattfeld@tu-braunschweig.de (D.C. Mattfeld).

https://doi.org/10.1016/j.cor.2018.06.004
0305-0548/

in IRPs, customers consume certain commodities (bikes and racks) provided by vehicles. In contrast to conventional IRPs, the numbers of bikes within the system and bike racks at every station are fixed. Further, the availabilities of bikes and free bike racks at every station are interdependent since every bike within a station blocks a bike rack. Thus, two dependent resources, namely bikes and free bike racks, need to be balanced: Both, not enough and too many bikes (in the sense of not enough bike racks) lead to failed demands and, in this way, to penalties. Another challenge for the considered problem is that demands are uncertain and subject to a heterogeneous spatio-temporal stochastic pattern. This results in a stochastic IRP. Finally, due to the high variability of demand, sudden imbalances may occur on short notice. Thus, dynamic decision making is applied to take advantage of revealed information and to react to sudden imbalances. As a result, we consider a stochastic-dynamic inventory routing problem (SDIRP) for bike sharing systems (Brinkmann et al., 2015). In the SDIRP, a vehicle and a set of capacitated stations with initial fill levels are given. Over the day, users stochastically demand rentals or returns of bikes at stations. To provide a sufficient number of bikes and free bike racks at every station, the vehicle is dynamically dispatched to transport bikes between stations. Decisions are made about the inventory of the station a vehicle is currently located at and about the next station the vehicle visits. A demand *fails* if a bike (to rent) or bike rack (to return a bike) is not available at the station and the time the user demands. From the operational view, vehicles and drivers are paid. Thus, we neglect transportation costs. The goal is to minimize the expected amount of unsatisfied demand.

For the SDIRP, we provide a stochastic and dynamic optimization approach to alleviate unsatisfied user demand. The approach subsequently decides on the relocations of bikes based on realized and expected future user demand. Stations requiring relocation activities are subsequently determined by lookahead policies (LAs). LAs simulate future user demand starting in the system's current state. Future user demand is sampled from historical data. In this way, the dynamic online control is augmented with information gained from a reasonable lookahead.

One challenge for LAs applied in dynamic routing problems is the determination of suitable lookahead horizons. Short horizons of one or two hours may not be able to capture many important future developments while long horizons over several hours may lead to a significant discrepancy between simulated and actually realized outcome (Ghiani et al., 2009; Voccia et al., 2017). Thus, we experience a tradeoff between amount of information and its accuracy. Both too short and too long horizons can lead to inferior decision making. Furthermore, suitable LA horizons are dependent on the structure of the underlying data - in our case, the user demand. Ulmer (2017) shows that more predictable demand patterns require different horizons than demand patterns blurred by statistical noise. Thus, this phenomenon is of particular importance for the SDIRP because the demand patterns vary over the day. As an example, demand patterns induced by commuter behavior during rush hours differ from demand patterns around lunchtime in both volume and noise.

To account for this challenge, we develop dynamic LAs (DLAs) with time-dependent lookahead horizons. The horizons per hour are determined a priori by means of value function approximation (VFA), a method of approximate dynamic programming (ADP, Powell, 2011). The VFA approximates the DLA's outcome for each point of time and potential horizon and selects the DLA-parametrization leading to the smallest amount of failed demands. In comprehensive computational studies based on real-world data of the BSS in Minneapolis (Minnesota, USA, MN, 2016), we show that the VFA-parametrized DLA significantly outperforms LAs with static horizons and manually parametrized DLAs as well as benchmark policies from the literature. In a detailed analysis, we show how the high solution quality is enabled by the DLA's adaption to the demand patterns.

Our contributions are as follows. We present a comprehensive Markov decision process-model for the stochastic-dynamic inventory routing problem for station-based BSSs. For the SDIRP, we develop anticipatory methods autonomously adapting to the demand pattern of the historical data and significantly reducing the amount of unsatisfied demand compared to benchmark policies from the literature. Methodologically, we present a new and general method of using VFA to determine suitable horizons and to assign a horizon to a subset of states. This can be seen as a non-parametric policy search. Given a set of potential horizons, this method allows a state-dependent policy selection based on a set of state parameters. A non-parametric policy search may provide value particularly for stochastic-dynamic decision problems of high complexity as often experienced in the fields of transportation and routing.

The remainder of this paper is structured as follows. In Section 2, we present the related literature. The SDIRP is presented in Section 3. In Section 4, we define the DLA and the VFA. The computational studies and the detailed analysis of the results are presented in Section 5. The paper concludes with a summary and an outlook in Section 6.

2. Literature

In this section, we present the related literature from the fields of bike sharing systems and stochastic inventory routing. We first present a classification for the analysis and then embed the relevant literature.

2.1. Classification

In the following, we develop a classification to compare our work and highlight our contribution. The classification is used in Table 1.

The SDIRP is stochastic and dynamic as defined by Kall and Wallace (1994). The problem is stochastic because the demand is not known in advance and follows a stochastic distribution. It is dynamic because subsequent decisions are made over a planning horizon. Thus, we analyze literature with respect to *Stochasticity* and *Dynamism*. A "✓" in the corresponding column indicates that the work considers a stochastic or dynamic problem. For the SDIRP, demand is stochastic and occurs over the course of the day. Some other works on BSSs consider relocation routing without any stochastic or deterministic user demand. We highlight work considering demand in the *Demand*-column. In the SDIRP, the demand is continuously revealed over time while for many inventory routing problems, demand is revealed once per period or day. We highlight work considering continuous demand in the *Continuity*-column. Work without demand have an "n/a"-entry in the *Continuity*-column. Work where demands are realized over the course of the day but only in time steps (for example in hours), is indicated with "(✓)". While for the SDIRP inventory decisions need to be made, other works on BSSs do not consider inventory levels but just determine stops at imbalanced stations. We indicate work addressing inventory decisions in the *Inventory*-column. Beside the inventory decisions, the SDIRP also requires a routing decision. Work considering routing is indicated by a "✓" in the *Routing*-column. Finally, our proposed DLA method anticipates future demands in both inventory and routing decision. We highlight work anticipating stochastic demand in the *Anticipation*-column. Notably, anticipation is only applicable in a stochastic problem. Thus, all deterministic problems have an "n/a"-entry in the *Anticipation*-column. In the following, we use the classification to describe the work on BSSs and stochastic inventory routing.

Table 1
Literature classification.

		Stochasticity	Dynamism	Demand	Continuity	Inventory	Routing	Anticipation
Bike sharing	Contardo et al. (2012)			✓	✓	✓	✓	n/a
	Chemla et al. (2013)				n/a	✓	✓	n/a
	Raviv et al. (2013)				n/a	✓	✓	n/a
	Kloimüllner et al. (2014)			✓	✓	✓	✓	n/a
	Erdoğan et al. (2015, 2014)				n/a	✓	✓	n/a
	Vogel et al. (2014)			✓	(✓)	✓		n/a
	Kloimüllner et al. (2015)				n/a		✓	n/a
	Neumann Saavedra et al. (2015)			✓	(✓)	✓		n/a
	Brinkmann et al. (2016)			✓	(✓)	✓	✓	n/a
	Espegren et al. (2016)				n/a	✓	✓	n/a
	Neumann Saavedra et al. (2016)			✓	(✓)	✓	✓	n/a
	Schuijbroek et al. (2017)				n/a	✓	✓	n/a
	Fricker and Gast (2016)	✓		✓		✓		
	Lu (2016)	✓		✓	(✓)	✓		✓
	Yan et al. (2017)	✓		✓	(✓)	✓	✓	
	Brinkmann et al. (2015)	✓	✓	✓	✓	✓	✓	
	Ghosh et al. (2017)	✓		✓	✓	✓	✓	
IRP	Godfrey and Powell (2002)	✓	✓	✓		✓	✓	✓
	Adelman (2004)	✓	✓	✓		✓	✓	✓
	Toriello et al. (2010)			✓		✓	✓	n/a
	Bertazzi et al. (2013)	✓	✓	✓		✓	✓	✓
	Papageorgiou et al. (2014)			✓		✓	✓	n/a
	Coelho et al. (2014a)	✓	✓	✓		✓	✓	✓
	SDIRP, DLA	✓	✓	✓	✓	✓	✓	✓

2.2. Bike sharing systems

Works on BSSs mainly focus on static and deterministic problems. In these problems, relocation operations often take place over night where no demand is assumed. The objective is to find a routing to reach satisfying fill levels at stations. Examples are works by Chemla et al. (2013), Raviv et al. (2013), Erdoğan et al. (2014), Erdoğan et al. (2015), Kloimüllner et al. (2015), Espegren et al. (2016) and Schuijbroek et al. (2017). To achieve a suitable routing, either mixed-integer programming methods or metaheuristics are developed.

Other static and deterministic works develop master tours to meet typical daily user demand, for example, Contardo et al. (2012), Kloimüllner et al. (2014), Neumann Saavedra et al. (2015) and Neumann Saavedra et al. (2016). The demand is assumed to be known and time-dependent. The relocation problems become time-dependent as well. These problems are still static because determining the routing and the relocation operations are made in a single decision point. Contardo et al. (2012) use mixed-integer programming to determine suitable inventory and routing decisions. Kloimüllner et al. (2014) apply metaheuristics to find a suitable routing and inventory solution avoiding as much failed demand as possible. Vogel et al. (2014) and Neumann Saavedra et al. (2016, 2015) relax the routing problem into a transport problem and solve this problem by means of matheuristics.

There are a few papers considering stochastic demand in a static context (Fricker and Gast, 2016; Ghosh et al., 2017; Lu, 2016; Yan et al., 2017). These works aim on developing suitable routes given uncertainty in the user demand. Fricker and Gast (2016) consider a stochastic system in a steady state and calculate the percentage of critical stations where demand may not be fulfilled. They analyze how incentives are able to avoid failed demand. Lu (2016) address a problem with stochastic demands. They use a time-space network of bike flows to determine robust solutions given a set of potential demand realizations. Yan et al. (2017) also use a time-space network and address the stochasticity of the problem with a threshold-based heuristic.

The two works on BSSs closest to the work presented in this article are presented by Brinkmann et al. (2015) and Ghosh et al. (2017). Both problems address the stochastic and dynamic relocations of bikes. Brinkmann et al. (2015) present a policy function approximation based on safety buffers. The vehicle is sent to the closest stations where the safety buffer is violated. We use this policy as a benchmark in our computational study. Ghosh et al. (2017) reduce their problem by means of aggregation. Both stations and trips are aggregated. Further, future stochastic demand is not considered. To approach the simplified static problem, a mixed-integer linear program is developed and solved by means of dual decomposition on a rolling horizon. The runtimes per decision point comprise several minutes. A real-time implementation is therefore challenging.

2.3. Inventory routing

The SDIRP is further related to work on stochastic inventory routing presented in this section.

Godfrey and Powell (2002) address a stochastic and dynamic resource allocation problem where stochastic demand is revealed every period. A relocation of resources requires one period. They apply a VFA to anticipate potential future demand in their decision making. In the SDIRP, demand is continuously revealed. To capture this demand, a detailed simulation is required. Adelman (2004) presents a problem where a set of customers needs to be served over a set of days. Each customer has a stochastic demand per day. For each day, a routing and inventory decision is determined. For anticipation of future routing costs, the authors present a VFA based on dual relaxations. A VFA is also applied by Toriello et al. (2010) and Papageorgiou et al. (2014) for deterministic inventory routing problems.

Bertazzi et al. (2013) apply a rollout algorithm (RA) to a similar stochastic dynamic inventory routing problem. The idea of an RA is to look ahead into the future and use a base policy within the lookahead. While RAs often draw on simulation for the lookahead, Bertazzi et al. (2013) use the solution of a mixed-integer program based on average values. The advantage of RAs is that both potential future stochastic demand as well as potential future decisions can be anticipated. However, RAs require a significant amount of runtime, a scarce resource in real-time decision making. In our

computational study, we compare our DLA with an RA. We show that the RA is not able to achieve competitive results within reasonable runtime. Coelho et al. (2014a) address a stochastic and dynamic inventory routing problems where a route through a set of customers needs to be determined every day. Therefore, they solve the deterministic problem based on average demand over a limited time horizon. This procedure is similar to the DLA, but instead of using average demand, we draw on simulation. Further, we vary the lookahead horizon over the decision states and determine suitable horizons by means of VFA.

Additionally, the relocation process in BSSs in related to empty container management (ECM). Containers are used in maritime (Erera et al., 2009) as well as road-based transport (Shintani et al., 2007; Song and Carter, 2009) to wrap commodities. When a commodity is picked up, an empty container is used. When it is delivered, the container becomes empty. Due to asymmetric pick-ups and deliveries, empty containers need to be relocated. However, the uncertainty in ECM is much lower than in stochastic and dynamic IRP. Thus, we neglect this domain in our classification.

In essence, our method DLA is the first anticipating continuous stochastic demand changes in dynamic inventory and routing decisions for BSSs.

3. Stochastic-dynamic inventory routing for bike sharing systems

In this section, we define the SDIRP (Brinkmann et al., 2015). We first formulate the SDIRP and the notation in Section 3.1. In Section 3.2, we model the SDIRP as Markov decision process. An example is presented in Section 3.3.

3.1. Formulation

We consider a BSS consisting of a depot n_0 and stations $n_i \in N$, $i > 0$: $N = \{n_0, \dots, n_{\max}\}$. User demand is reflected in stochastic rental and return demand for bikes over a time horizon $T = (t_0, \dots, t_{\max})$ which is discretized into points in time t. Every station has a capacity $c(\cdot)$. A rental demand is successful if the associated station is not empty. A return demand is successful if the associated station is not full. A successful demand impacts the associated station's fill level. If a demand fails, the user is sent to the nearest station that can serve the demand.[1] A transport vehicle relocates bikes between stations. The fleet size is fixed. Because driver wages account for a majority of costs, driving costs are neglected. For safety reasons, dispatcher and driver do not communicate during travel. Thus, diversions from a current destination are not possible while the vehicle is on the road. The vehicle starts and ends its tour at the depot. The vehicle capacity is given by c_v. The vehicle travels with constant speed leading to travel times between two stations of $\tau(\cdot, \cdot)$. Every relocation operation consumes a service time of τ_r per bike. Time for parking is neglected. In order to offer a reliable system and to maximize the users' satisfaction, the objective for the dispatcher is to route the vehicle in a way that as few demand fails as possible.

3.2. Markov decision process

The SDIRP can be modeled as a Markov decision process (MDP, Puterman, 2014). The components of an MDP are depicted in Eq. (1):

$$s_k \xrightarrow{x} s_k^x \xrightarrow{\omega} s_{k+1}. \tag{1}$$

Table 2
Notation of the Markov decision process.

Symbol	Description
$K = (0, \dots, k_{\max})$	Sequence of decision points
$S = \{s_0, \dots, s_{\max}\}$	Set of decision states
$X_s = \{x_1, \dots, x_{\max} \mid x = (\iota^x, n^x)\}\ ,\ \forall s \in S$	Sets of feasible decisions
$s_k^x = (s_k, x)\ ,\ \forall s \in S, x \in X_s$	Post-decision states
ω: $S \times X \to S$	Transition function
$t_k \in T$	Point in time in state s_k
$f_k^v \in \mathbb{N}_0$	Vehicle's load in time t_k
$n_k^v \in N$	Vehicle's station in time t_k
$f_k = \left(f_k^{n_0}, \dots, f_k^{n_{\max}}\right)$	Stations' fill levels in time t_k
$\iota^x \in \mathbb{Z}$	Inventory decision
$n^x \in N$	Routing decision
$p : S \times X \to \mathbb{N}_0$	Penalty function
$\Pi = \{\pi_0, \dots, \pi_{\max} \mid \pi : S \to X\}$	Set of policies

In an MDP, decisions are made for a number of decision points k. An associated decision state s_k defines the parameters on which a decision $x \in X_{s_k}$ is made. The deterministic post-decision state s_k^x represents the combination of decision state and decision. Before the next decision point occurs, a stochastic transition function ω alters state parameters.

The MDP's notation for the SDIRP is described in Table 2. In the SDIRP, a decision point k occurs when the vehicle arrives at a station. A decision state $s_k \in S$ comprises the point in time t_k, the stations' fill levels $f_k = \left(f_k^{n_0}, \dots, f_k^{n_{\max}}\right)$, the vehicle's current station n_k^v, and the vehicle load f_k^v:

$$s_k = \left(t_k, f_k, n_k^v, f_k^v\right). \tag{2}$$

The current fill level of a station n is f_k^n. All fill levels are summarized in f_k. The station where the vehicle currently is located is denoted by n_k^v. The number of bikes loaded on the vehicle in time t_k is given by f_k^v.

Every decision $x = (\iota^x, n^x)$ contains an inventory decision $\iota^x \in \mathbb{Z}$ at the current station and a next station $n^x \in N$ to visit. If $\iota^x < 0$, the vehicle picks up bikes at the current station. If $\iota^x > 0$, the vehicle delivers bikes. The overall time for relocation is $|\iota^x| \cdot \tau_r$. The travel time between stations is given by $\tau(n_i, n_j)$ for any two stations $n_i, n_j \in N$. In the post-decision state $s_k^x = (s_k, x)$, the fill levels have been stepwise[2] adapted according to relocations and the vehicle has traveled to the next station. Before the next decision state occurs, altered fill levels are revealed by the stochastic transition $\omega(\cdot, \cdot)$ due to successful demands. However, some demands may have failed. The penalty function $p(s_k, x)$ reflects the expected amount of failed demand between two decision states given post-decision state s_k^x. The penalty function $p(s_k, x, \omega)$ reflects the amount of failed demand based on realization ω.

At the end of the time horizon, the vehicle returns to the depot. Thus, the final decision point $k_{\max}$ occurs if returning to the depot is the only feasible decision

$$k = k_{\max} \quad \Leftrightarrow \quad X_{s_k} = \{(0, n_0)\}. \tag{3}$$

A solution for the SDIRP is a policy $\pi \in \Pi$ assigning every state to a decision. The optimal policy π^* minimizes the expected amount of unsatisfied demand. Let s_0 be the initial decision state. The objective is to identify an optimal policy $\pi^* \in \Pi$ leading to the minimum expected amount of unsatisfied demand:

$$\pi^* = \arg\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=0}^{k_{max}} p\left(s_k, \pi\left(s_k\right)\right) \middle| s_0\right]. \tag{4}$$

[1] For a detailed depiction of this process, the interested reader is referred to Appendix D.

[2] Because one relocation operation takes τ_r minutes per bike, the adaption is not instantaneous.

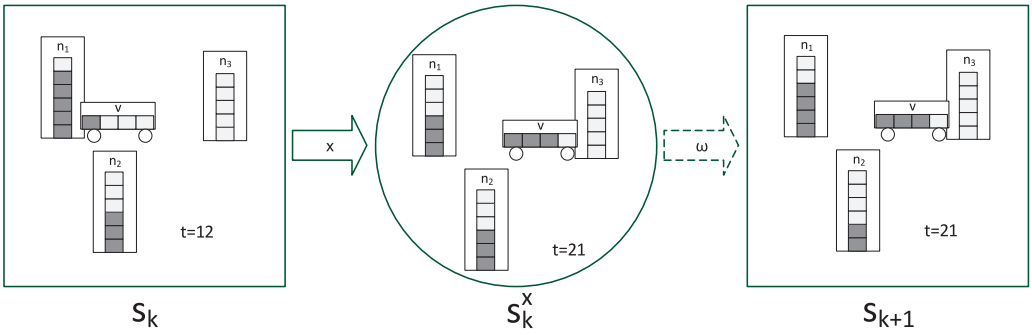

Fig. 1. Exemplary decision state, post-decision state, and resulting decision state.

3.3. Example

Fig. 1 presents an example for the MDP. On the left-hand side, decision point s_k is depicted. The center shows post-decision state s_k^x. The following decision state s_{k+1} is depicted on the right-hand side. The system has three stations. For stations n_1, n_2, n_3, and vehicle v, light boxes represent empty bike racks, dark boxes represent bike racks filled with a bike. Stations n_1 and n_2 have a capacity of 6 racks, station n_3 of 5 racks. The vehicle capacity is 4. In state s_k, occurring at time $t_k = 12$, n_1 contains $f_k^{n_1} = 5$ bikes, n_2 contains $f_k^{n_2} = 3$ bikes, and n_3 is empty, $f_k^{n_3} = 0$. The vehicle is located at $n_k^v = n_1$ and has one bike loaded, $f_k^v = 1$. The inventory decision ι^x is made about how many bikes to pick up at or deliver to station n_1. The routing decision n^x is made about which station to serve next, or idling at the current station. Here, the applied decision $x = (-2, n_3)$ is to pick up two bikes and travel to n_3. The resulting post-decision state s_k^x, including the decisions made, is depicted in the center of Fig. 1. Given $\tau_r = 2$ min, two pick-ups consume four minutes of time. Furthermore, given $\tau(n_1, n_3) = 5$, the next decision point $k+1$ occurs in $t_{k+1} = 12 + 4 + 5 = 21$. The stochastic transition ω reveals new fill levels and a realization of the penalty function. At n_1, the fill level has increased by one due to one successful return. At n_2, the fill level has decreased by one due to one successful rental. Station n_3 remains empty. For the purpose of presentation, the user trips are not depicted in Fig. 1. We assume that one rental demand failed at station n_3 resulting in a penalty of $p(s_k, x, \omega) = 1$. The associated user will soon approach n_2 since it is the nearest non-empty station. The resulting new decision state $s_{k+1} = \omega(s_k, x)$ is shown on the right side of Fig. 1.

3.4. Challenges

According to the Bellman equation (Bellman, 1957), in every decision state s_k, π^* returns the optimal decision $\pi^*(s_k) \in X_{s_k}$. It minimizes the expected number of unsatisfied future demand, i.e., the expected penalty:

$$\pi^*(s_k) = \arg\min_{x \in X_{s_k}} \mathbb{E}\left[\sum_{k'=k}^{k_{max}} p(s_{k'}, x) \Big| s_k\right]. \tag{5}$$

The expectations on future unsatisfied demand can be determined recursively by dynamic programming and backwards induction. To do this, perfect information on the decision tree is required. Gathering this amount of information is hardly possible due to the three curses of dimensionality (Powell, 2011):

The state space: The number $|S|$ of states $s \in S$ grows with the state parameters' attributes. Let $|T|$ be the number of points in time. The stations' fill levels and the vehicle load can occur in any combination. The vehicle can be located at every $n \in N$. Then, $|S| \leq |T| \cdot \prod_{n \in N} (c(n) + 1) \cdot (c_v + 1) \cdot |N|$.

The decision space: For every state $s \in S$, a set of feasible decisions X_s is given. Decisions are about realizing relocations at the vehicle's current station and to go to a next station $n \in N$. Therefore, $|X_s| \leq (c_v + 1) \cdot |N|$, where $c_v + 1$ is an upper bound on all feasible relocation operations and $|N|$ is the number of feasible next stations.

The outcome space: The transition function ω depicts demands by mapping a decision state and a decision to a subsequent decision state. The number of feasible transitions is (nearly) unbounded since in any point in time, any demands may occur.

The decision tree's size is approximately the Cartesian product of the state space, the decision space, and the outcome space. Identifying π^* analytically is not possible. Therefore, we introduce lookahead policies (LAs) drawing on online simulation in Section 4.

4. Dynamic lookahead policies

In this section, we motivate and define dynamic lookahead policies (DLAs) to solve the SDIRP. We first give an overview on the steps taken. We then define the dynamic lookahead in detail as well as the value function approximation (VFA) to tune the DLA.

4.1. Overview

In the following, we give an overview on the functionality of the DLA to prepare the definition of the method. The DLA consists of two parts. The first part is the lookahead as foundation of the DLA. The second part is the parametrization of the DLA by means of VFA. The first part is described by Fig. 2. The second part is described by Fig. 3.

We start with the procedure of the dynamic lookahead as depicted in Fig. 2. Given a state in the MDP, the DLA simulates into the future to derive a decision. This is conducted by simulating user demand online over a predefined horizon. Based on the outcome of the simulation, the inventory decision at the current station as well as the routing decision to the next station are determined. More specific, the DLA selects the inventory decision leading to the smallest amount of unsatisfied demand at the current station within the horizon. Further, the vehicle is routed to the station where the largest amount of unsatisfied demand can be avoided within the horizon. Beside the simulated demand, this amount also depends on the number of loaded bikes on the vehicle and the travel time between the vehicle's current location and the potential next station.

As aforementioned, different horizons may be suitable with respect to the time of the day and the corresponding demand pattern. Thus, the DLA draws on different horizons for different periods of the day. In our setting, we determine an individual horizon for periods of one hour length. Given 24 hours of the day, a DLA

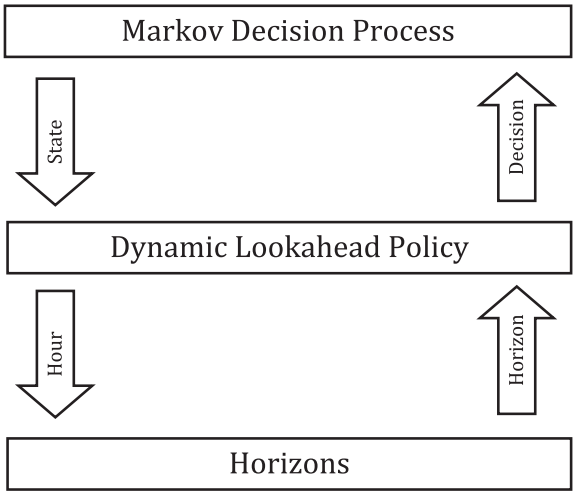

Fig. 2. Overview of the dynamic lookahead.

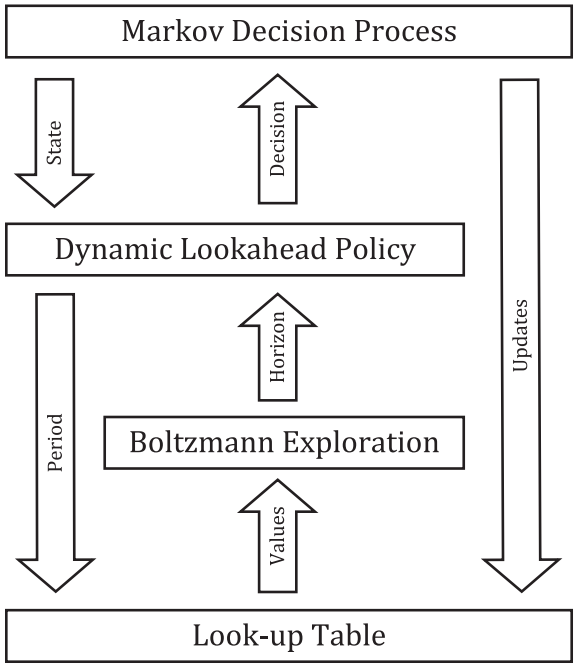

Fig. 3. Approximation run of the value function approximation.

can therefore be parametrized by a vector of 24 horizons, i.e., one horizon per hour.

In Fig. 2, the horizons are externally given and stored in the "Horizons"-black box. In a state, the lookahead accesses this black box to obtain the corresponding horizon for the current hour. However, to successfully apply the lookahead, the horizons per hour need to be determined. This determination of suitable horizons is challenging for two reasons. First, the horizons depend on the demand patterns. These patterns are complex and change over the day. An analytical determination of the horizons is challenging. Second, a decision at one point of time impacts the future developments of the MDP. Thus, the horizons cannot be determined in isolation but only in combination. To this end, we use VFA. The VFA draws on offline simulations of the MDP to approximate the unsatisfied demands for each hour-horizon combination. The approximation for each combination is stored in a look-up table. To visualize the procedure, we extend Fig. 2 to Fig. 3. Fig. 3 shows the procedure for one approximation run of the MDP. In an approximation run, the MDP is simulated. Within a state of the simulation, the lookahead requests the horizon for the hour. The procedure either returns the currently "best" horizon for the corresponding hour or a different horizon based on Boltzmann exploration. After each simulation run, the observed values are used to update the current approximation in the look-up table. Subsequently, the approximation becomes more accurate and the policy improves.

In the remainder of this section, we define the two components of the DLA in detail. First, we describe the lookahead procedure in Section 4.2. We then describe how the DLA is parametrized by the VFA in Section 4.3.

4.2. Lookahead algorithm

In the following, we describe the lookahead as shown in Fig. 2. For the algorithmic procedure, we refer to Appendix B.1. We assume that we are in a decision state s_k and the horizon δ of the lookahead is predetermined externally.

The lookahead is used to determine the two parts of the decision $x = (\iota^x, n^x)$. We recall ι^x as the inventory decision determining the amount of bikes delivered to or picked up at the current station n_k^v. Parameter n^x describes the next station the vehicle visits.

4.2.1. Inventory decision

The lookahead first determines ι^x by analyzing (at most) three potential inventory decisions with respect to the fill level of station n_k^v. We consider three potential percentages leading to three different target fill levels: low ($\mu_1 = 25\%$ of station's capacity), medium ($\mu_2 = 50\%$), or high ($\mu_3 = 75\%$). We denote the associated inventory decisions ι_1, ι_2, ι_3 and the resulting fill levels $\mu_1 \cdot c(n_k^v)$, $\mu_2 \cdot c(n_k^v)$, $\mu_3 \cdot c(n_k^v)$. There may be states where a fill level cannot be fully reached because we additionally have to consider the station's fill level $f_k^{n_k^v}$, the vehicle load f_k^v, and the vehicle capacity c_v. We then modify the inventory decisions in accordance to Eq. (6):

$$\iota_i = \begin{cases} \min\{\mu_i \cdot c(n_k^v) - f_k^{n_k^v}, f_k^v\}, & \text{if } \mu_i \cdot c(n_k^v) > f_k^{n_k^v} \\ \max\{\mu_i \cdot c(n_k^v) - f_k^{n_k^v}, c_v - f_k^v\}, & \text{if } \mu_i \cdot c(n_k^v) < f_k^{n_k^v} \\ 0, & \text{else.} \end{cases} \tag{6}$$

The number of bikes to deliver is limited to the number of bikes currently loaded on the vehicle. The number of bikes to pick up is limited to the number of bikes that can additionally be loaded onto the vehicle. The first case of Eq. (6) occurs if bikes need to be delivered to realize the target fill level. The second case occurs if bikes need to be picked up to realize the target fill level. The third case occurs if the current station's fill level is equal to the target fill level.

Starting from these three potential fill levels, the algorithm repeatedly simulates future realizations of demand over the horizon δ. These simulations are consecutive realizations of the MDP's transition function and follow the event handling procedure described in Appendix D. Based on the simulations, we can approximate the impact of the inventory decisions ι_i.

According to preliminary tests, we set the number of simulation runs per fill level to 32. The inventory decision ι^x is set to the one of the three leading to the least amount of unsatisfied demand as the sum of failed rental and failed return demands at this station. To this end, the lookahead counts the numbers of failed rentals $\gamma_{\iota,n}^-$ and returns $\gamma_{\iota,n}^+$ for every ι and for every station $n \in N$ and determines the average over 32 simulations j. Let k^j be the first decision point in simulation j and $k_{\max}^j$ be the last. The failed rentals and returns in simulation j are depicted by $p_j^-(\cdot,\cdot)$, or $p_j^+(\cdot,\cdot)$, respectively.[3] Then, we can determine $\gamma^-(\iota,n), \gamma^+(\iota,n), \forall \iota, n \in N$ accord-

[3] For a detailed depiction of this process, we again refer to Appendix D.

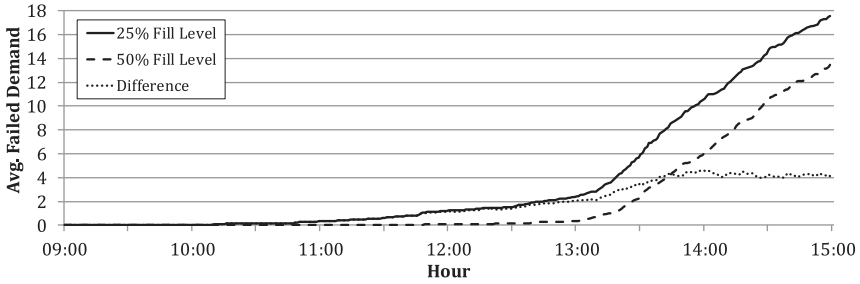

Fig. 4. Observed cumulated failed demand in an exemplary lookahead.

ing to Eq. (7) and Eq. (8):

$$\gamma^-_{\iota,n} = \frac{1}{32} \cdot \sum_{j=1}^{32} \sum_{k^j=k}^{k^j_{\max}} p^-_j(s_{k^j}, n), \tag{7}$$

$$\gamma^+_{\iota,n} = \frac{1}{32} \cdot \sum_{j=1}^{32} \sum_{k^j=k}^{k^j_{\max}} p^+_j(s_{k^j}, n). \tag{8}$$

Finally, we select the inventory decision ι^x minimizing the failed rentals and returns at the current station n^v_k:

$$\iota^x = \underset{\iota \in \{\iota_1, \iota_2, \iota_3\}}{\arg\min} \{\gamma^-_{\iota,n^v_k} + \gamma^+_{\iota,n^v_k}\}. \tag{9}$$

An example is given by Fig. 4. The decision point k occurs at 09:00. We assume a lookahead horizon of $\delta = 360$ minutes. Thus, we analyze the failed demand at the current station until 15:00. The solid and dashed lines refer to the failed demand caused by inventory decisions of 25% and 50% fill level. For the sake of simplicity, we omit the fill level of 75% in this example. The dotted line depicts the difference of failed demand between the solid and dashed lines. According to the lookahead, until 10:00 both realized fill levels can serve the demand. Then, a we observe a slight increase in failed demand in the case of 25% fill level due to rental demand. The fill level of 50% is suitable until 13:00. From 14:00 on, the difference of failed demand is more or less constant. The observations are captured by $\gamma^-_{\iota_1,n^v_k} > \gamma^-_{\iota_2,n^v_k}$. Thus, we select the decision leading to the fill level of 50%, $\iota^x = \iota_2$.

Fig. 4 gives an impression of the impact of horizon δ. If $\delta \leq 60$, the lookahead would be limited to 10:00. Until this time, no demand fails for both decisions. Thus, the decisions cannot be differentiated. If $\delta \geq 300$, the difference of failed demand is constant. Thus, the lookahead does not provide more information if it simulates further than 14:00.

4.2.2. Routing decision

Based on the inventory decision ι^x, the lookahead then determines the next station to visit, again, by means of online simulation over the horizon δ. The algorithm selects the station n where relocations can prevent the largest amount of unsatisfied demand. To this end, the algorithm tracks all failed rentals and returns at every station $n \in N$ over the horizon δ. Here, we draw on the simulations originally conducted for the inventory decisions. When approximating the failed rentals and returns, $\gamma^-_{\iota^x,n}, \gamma^+_{\iota^x,n}, \forall n \in N \setminus \{n^v_k\}$, the inventory decision ι^x has been realized at the current station n^v_k. In the case of failed demand, the demand at a station is subject to the fill levels of neighboring stations (Rudloff and Lackner, 2014). Therefore, considering the adapted fill level of n^v_k achieves more accurate approximations.

Given a specific station n, the algorithm considers the travel time $\tau(n^v_k, n)$ between the current station n^v_k and n. Failed demands in the time before the vehicle's arrival at station n are ignored because they cannot be prevented. Thus, $\gamma^-_{\iota^x,n}$ and $\gamma^+_{\iota^x,n}$ represent the amount of failed demand in the time interval $\left[t_k + \tau(n^v_k, n), t_k + \delta\right]$ (compare Eqs. 7 and 8).

Beside the expected failed rentals $\gamma^-_{\iota^x,n}$ and returns $\gamma^+_{\iota^x,n}$ observed in the simulations, we further have to consider the vehicle's current fill level f^t_v in combination with inventory decision ι^x that will be conducted at n^v_k. To prevent rentals from failing, bikes need to be delivered. The vehicle is able to deliver at most $f^t_v - \iota^x$ bikes. Thus, the number of prevented failed rentals is the minimum of $\gamma^-_{\iota^x,n}$ and $f^t_v - \iota^x$. To prevent returns from failing, bikes need to be picked up. The vehicle is able to pick up at most $c_v - f^t_v - \iota^x$ bikes. Thus, the number of prevented failed returns is the minimum of $\gamma^+_{\iota^x,n}$ and $c_v - f^t_v - \iota^x$. The number of potentially prevented demand is the maximum of potentially prevented failed rentals and failed returns as shown in Formula (10):

$$n^x = \underset{n \in N}{\arg\max} \left\{ \underbrace{\min\left\{\gamma^-_{\iota^x,n}, f^t_v - \iota^x\right\}}_{\text{prevented failed rentals}}, \underbrace{\min\left\{\gamma^+_{\iota^x,n}, c_v - f^t_v - \iota^x\right\}}_{\text{prevented failed returns}} \right\}. \tag{10}$$

Eventually, (ι^x, n^x) is the decision made by DLA in s_k.

Fig. 5 provides an example. The vehicle stays at a station in the same district as n_1. Station n_2 is located at the other side of the city. According to the lookahead, the development of cumulated failed rental demand is more or less the same. Station n_1 can be reached at 16:15. At this time, only a small amount of demand has failed. Given that the vehicle's load is sufficient, all future demand can be served. Station n_2 can be reached at 16:45. At this time, much more demand has failed. This failed demand cannot be saved. These observations are reflected by $\gamma^-_{\iota^x,n_1} > \gamma^-_{\iota^x,n_2}$. Thus, the vehicle's next station is $n^x = n_1$.

Notably, our method operates directly on the MDP's decision state. Thus, we only select the inventory at the current station and the next location to visit. There are dynamic vehicle routing methods extending the decision space to a set of tentative route plans (Ulmer et al., 2017b). A route plan is a sequence of future decisions, for example, stops at customers. Instead of only the next customer to visit, a route plan is evaluated and selected by these methods in every state. The implemented decision can then be extracted from this plan. The advantage of route plans is that they allow to incorporate a sequence of potential future decisions into the evaluation of the current decision. However, as we show in Section 5.6, incorporating potential route plans, i.e., a sequence of additional inventory and routing decisions, by means of a rollout algorithm is not advantageous because of the highly stochastic and dynamic nature of the SDIRP.

4.3. Dynamic lookahead horizons

As motivated in the introduction, suitable lookahead horizons may depend on the demand pattern and may therefore change

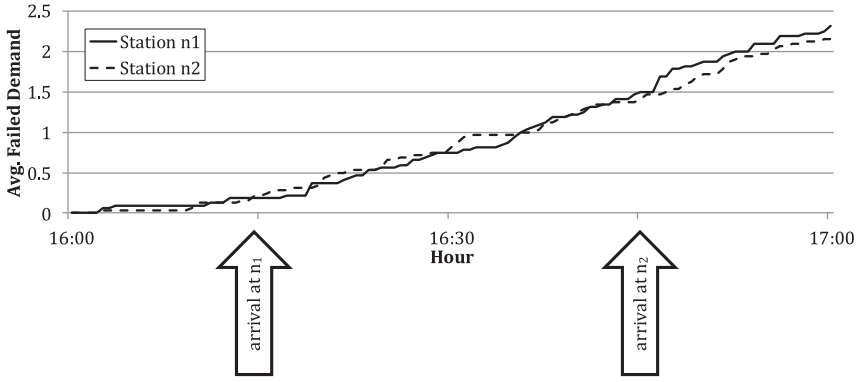

Fig. 5. Observed cumulated failed demand in an exemplary lookahead.

over the course of the day. In this section, we describe how we define and determine a DLA with time-dependent horizons.

4.3.1. Definition: dynamic lookahead

To vary the lookahead horizons, we subdivide the overall time horizon into a set of hours $\mathrm{P} = \{\rho_0, \ldots, \rho_{\max}\}$. Then, we can assign every point of time, and therefore every decision state, to an hour. We consider a set of lookahead horizons $\Delta = \{\delta_0, \ldots, \delta_{\max}\}$. This set contains seven potential horizons of $\delta_0 = 0$ minutes, $\delta_1 = 60$ minutes, up to $\delta_6 = 360$ minutes. In the special case of δ_0, no simulations are conducted. Because in that case, no decisions could be derived, the lookahead is replaced by a conventional heuristic based on safety buffers of 20%. This heuristic is described in more detail in Section 5.2.

In essence, a DLA-parametrization can be defined by a sequence of horizons:

$$(\delta^{\rho_0}, \delta^{\rho_1}, \ldots, \delta^{\rho_{\max}}) \in \Delta^{|\mathrm{P}|}. \tag{11}$$

In hours 0 – 18, we can select every simulation horizon. From the hour 19, we cannot select the long simulation horizons since it would exceed the time horizon. Given $|\mathrm{P}| = 24$ hours and $|\Delta| = 7$ horizons, we achieve approximately $7^{19} \cdot 6! > 8.2 \cdot 10^{18}$ different parametrizations. The challenge is to find the parametrization leading to the lowest amount of expected unsatisfied demand.

4.3.2. Value function approximation

As we show in Appendix C, an manual derivation of the parametrization is challenging. Thus, we determine the parametrization by value function approximation (VFA). This procedure can be seen as non-parametric policy search. *Non-parametric* policy search focuses on identifying parameters of a policy without assuming any functional dependency between state attributes and decisions to make. For an overview on *parametric* policy search, we refer to Powell (2011).

Technically, we aim on applying a value function v. Given an hour $\rho_i \in \mathrm{P}$ and an horizon $\delta_j \in \Delta$, the value $v(\rho_i, \delta_j)$ is defined as the expected number of unsatisfied demand until the end of the day. The left term of Eq. (12) refers to the expected failed demand in the current hour ρ_i in which horizon δ_j is selected. $\overline{k}^{\rho_i}$ and $\underline{k}^{\rho_i}$ refer to the first and last decision point of hour ρ_i. The right term refers to the expected failed demand in future hours, if hour ρ_i's last decision point $\underline{k}^{\rho_i}$ is not the final decision point $k_{\max}$.

Table 3
Notation of the value function approximation.

Symbol	Description
$\mathrm{P} = \{\rho_0, \ldots, \rho_{\max}\}$	Set of hours
$\Delta = \{\delta_0, \ldots, \delta_{\max}\}$	Set of lookahead horizons
$\overline{k}^{\rho} \in K$	Hour ρ's first decision point
$\underline{k}^{\rho} \in K$	Hour ρ's last decision point
$v : \mathrm{P} \times \Delta \to \mathbb{R}_0^+$	Value function
$\tilde{v} : \mathrm{P} \times \Delta \to \mathbb{R}_0^+$	Approximated value function
$\alpha : \mathrm{P} \times \Delta \to \mathbb{N}_0$	Occurrences of hour/horizon pairs

$$v(\rho_i, \delta_j) = \underbrace{\mathbb{E}\left[\sum_{k=\overline{k}^{\rho_i}}^{\underline{k}^{\rho_i}} p\left(s_k, \pi^{\delta_j}(s_k)\right) \middle| s_{k_{\rho_i}}\right]}_{\text{current hour}} + \underbrace{\begin{cases} \min_{\delta \in \Delta} v(\rho_{i+1}, \delta) & \text{, if } \underline{k}^{\rho_i} \neq k_{\max} \\ 0 & \text{, else} \end{cases}}_{\text{future hours}} \tag{12}$$

Notably, the selection of lookahead horizons in subsequent hours has an impact on the DLA's performance. Further, the benefit of selecting a lookahead horizon may pay off very late. For instance, lookahead horizon δ_i may lead to less unsatisfied demand than δ_j in the current hour ρ. Still, horizon δ_j might be preferred if it leads to less unsatisfied demand in the future hours. This is reflected by $v(\rho, \delta_j) < v(\rho, \delta_i)$.

We are not able to determine the exact value function v due to the curses of dimensionality (compare Section 3.4). Thus, we draw on an approximated value function $\tilde{v} \approx v$. VFA approximates the value function by means of offline simulations (Powell, 2011). Within the simulations of the MDP, the VFA selects DLA-parametrizations based on the approximated values. To avoid local optima, we draw on Boltzmann exploration, balancing exploitation and exploration by selecting potentially inferior horizons with a certain probability (Powell, 2011; Rothlauf, 2011).

The notation of the VFA is shown in Table 3. VFA comprises two phases. In the approximation phase, the value function $\tilde{v}$ is approximated. In the application phase, the value function is used for real-time decision making. We will describe the two phases in the following.

4.3.2.1. *Approximation phase*.

Fig. 3 provides an overview on the approximation phase. The formal procedure is given in Appendix B.2. The approximation of v bases on a training sets of trips. For every set of trips, a trajectory of the MDP is traversed. Decisions are made by an DLA. To select this DLA's parametriza-

tion, the MDP commits the current hour ρ to the look-up table. The look-up table is a data structure storing the values for every hour and horizon. The values $\tilde{v}(\rho, \delta), \forall \delta \in \Delta$ are returned to the Boltzmann exploration which selects the DLA's horizon for the current hour. Unlike the Bellman equation, the Boltzmann exploration does not always choose the horizon comprising the minimum value but may force exploration. We refer the interested reader to Appendix A providing detailed information on the Boltzmann exploration. When the final decision state in one simulation run occurs, the realized unsatisfied demand is used to update the values. Let δ^ρ be the lookahead horizon selected in hour ρ, let $\alpha(\rho, \delta^\rho)$ be the number of selections of δ^ρ in ρ over all trajectories, and let $\widehat{\gamma}(\rho)$ be the amount of unsatisfied demand observed starting in hour ρ. Then, the selected lookahead horizons' approximated values are updated as follows:

$$\tilde{v}(\rho, \delta^\rho) := \underbrace{\frac{\alpha(\rho, \delta^\rho) - 1}{\alpha(\rho, \delta^\rho)} \cdot \tilde{v}(\rho, \delta^\rho)}_{\text{old approximation}} + \underbrace{\frac{1}{\alpha(\rho, \delta^\rho)} \cdot \widehat{\gamma}(\rho)}_{\text{new observation}}. \tag{13}$$

As a termination criterion, we analyze the development of the parametrizations provided by both Boltzmann and Bellman. If both provide the same unaltered parametrization for successive 1,000 trajectories, we assume that the procedure converged (In our computational study that is the case after 34,649 simulation runs). As soon as this termination criterion occurs, the approximation phase ends.

4.3.2.2. ***Application phase.*** In the application phase, we select the best parametrization $(\delta^{\rho_0}, \delta^{\rho_1}, \ldots, \delta^{\rho_{\max}})$ for our DLA. More specific, we apply the approximate Bellman Eq. (14) to select the lookahead horizons minimizing the expected number of future failed demands:

$$\delta^\rho := \arg\min_{\delta \in \Delta} \tilde{v}(\rho, \delta)\ ,\ \forall \rho \in \mathrm{P}. \tag{14}$$

We denote this parametrization DLA(VFA).

5. Computational studies

In this section, we analyze the performance of the DLA. We show that the DLA is able to reduce the amount of unsatisfied demand compared to benchmark policies. We specifically show the value of the DLA's autonomous adaption to the demand pattern and how the horizons reflect the varying demand pattern over the day. We also show that integrating routing decisions in our simulations is challenging. Finally, we show how the DLA's advantages decline when we systematically increase the percentage of noise in the user demand.

In the following, we present the instances based on real-word data of the bike-sharing system of Minneapolis in Section 5.1. We then define the benchmark policies in Section 5.2. We compare the results in Section 5.3 and show the impact of anticipation in Section 5.4. We analyze the structure of the DLA in Section 5.5 and provide results of a rollout algorithm integrating routing decisions in the simulation in Section 5.6. Finally, in Section 5.7, we show how the advantages of the DLA decrease when the demand pattern is dissolved in noise.

5.1. Instances

We draw on real-world data offered by Minneapolis' (Minnesota, USA) bike sharing system "Nice Ride MN" (MN, 2016). The data set comprises 483,229 trips recorded in the year 2015. We follow the preprocessing steps proposed by Vogel et al. (2011). We focus on trips occurred between June and September, where the user activity intensity is highest. Since the demand patterns of working days and weekends differ significantly, we select only trips occurred between Mondays and Fridays. Further, we removed all stations (and associated trips) where only a few trips have been recorded or which were installed or removed from the BSS in the considered time period. The resulting data set comprises 88 working days, 169 stations, and 197,726 trips. The stations' capacities differ between 15 and 35. The number of bikes within the BSS is 1510 which corresponds to a bike racks and bikes ratio of 2: 1. On average, approximately 2246 trips occur per day. By randomly drawing trips with replacement from the set, we can create $197,726^{2,246} > 9.1 \cdot 10^{11,895}$ different subsets preserving the spatio-temporal pattern. A subset represents a realization of a working day and serves as an instance for the SDIRP. Fig. 6 depicts the temporal distribution of trips per hour in the course of the day. We observe two peaks at hours 8 and 17 due to commuters. An additional peak appears at noon, presumably due to leisure activities. For the initial fill levels, we distribute the bikes over all stations equally.

Between stations, we consider Euclidean distances. The vehicle as well as cyclists move with a constant speed of $15\frac{\mathrm{km}}{\mathrm{h}}$. This relatively conservative speed selection reflects travel through a street network, traffic, and time for parking. Pedestrians move with $5\frac{\mathrm{km}}{\mathrm{h}}$. A relocation operation consumes a service time of 2 minutes per bike. The vehicle capacity is 10 bikes.

Based on these parameters, we simulate 10,000 non-concatenated working days to evaluate the policies.

5.2. Benchmark policies

We define two benchmark policies to analyze the impacts of lookaheads in general and the advantages of time-dependent LA horizons. To analyze the impact of incorporating historical data by means of lookaheads, we define a conventional short-term relocation policy (STR) based on safety buffers as suggested in Coelho et al. (2014a) and Brinkmann et al. (2015). For the STR, percentual safety buffers β for bikes and free bike racks are defined for every station. If a station's fill level is between both safety buffers, we assume the station to be balanced. If one of the safety buffers is violated, we assume the station to be imbalanced. In every decision state, the vehicle will serve the nearest imbalanced station to balance as many stations as possible. A pseudo code formulation of STR is provided in Appendix B.4. For tuning, we vary STR(β) with $\beta \in \{0.1, \ldots, 0.5\}$.

To analyze the benefits of time-dependent lookahead horizons, we compare the DLA to lookahead policies with static horizon (SLA). Thus, SLA is a special case of DLA with the horizon δ identical for every hour of the day. We define a set of SLAs with SLA(δ) denoting the SLA of $\delta \in \{60, \ldots, 360\}$ minutes horizon.

5.3. Results

The experiments are conducted on an Intel Core i5-3470 with 3.2GHz and 32GB RAM. The implementation bases on Java 8.0u121. The individual results are depicted in Appendix E. We compare the average failed demands achieved by DLA(VFA) with the SLAs and with the best tuning of the STR, provided by 20% safety buffers, in Fig. 7.

Policy STR(0.2) achieves 145.126 failed demands on average. We observe that DLA and SLAs outperform the STR significantly. The SLA leads to 97.091 failed demands given simulation horizons of 180 minutes. The DLA reduces the failed demands even to 85.405. Thus, the anticipation of the lookaheads is highly beneficial.

Notably is the varying performance of the SLAs. As aforementioned, we experience a tradeoff between too short and too long horizons. For horizons of only 60 minutes, the improvement is relatively low. For longer horizons of more than 180 minutes, the

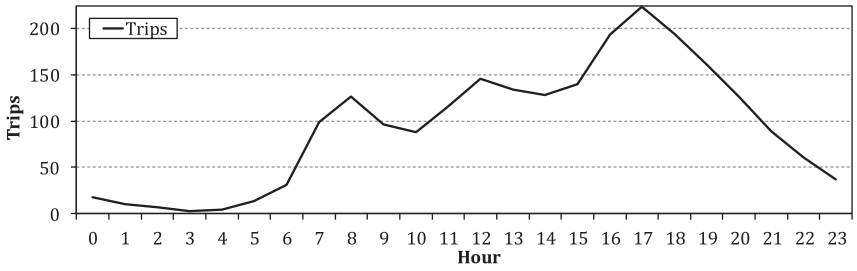

Fig. 6. Temporal distribution of trips of Minneapolis' BSS "Nice Ride MD".

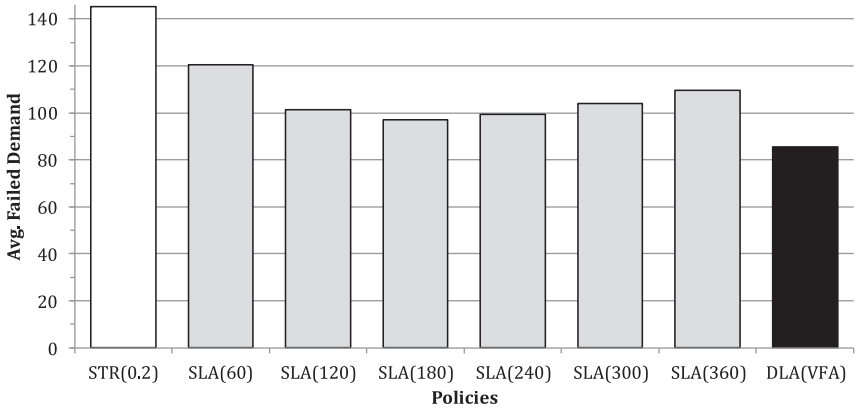

Fig. 7. Results of short-term relocation policy and lookahead policies.

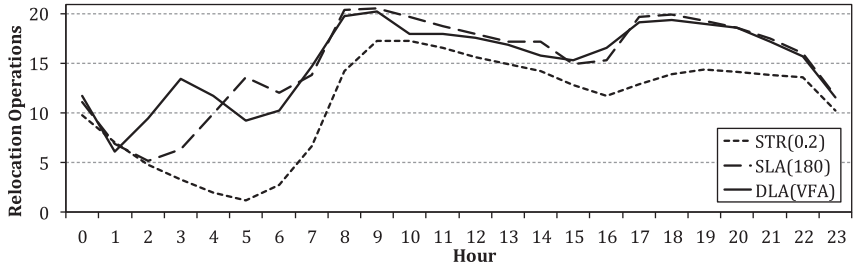

Fig. 8. Relocations per hour.

improvement decreases as well. This confirms the observation of Voccia et al. (2017) that long horizons may lead to inferior decision making.

5.4. The value of anticipation

In the following, we analyze how anticipation by the LA changes decision making compared to STR. To this end, we analyze when relocations are conducted and when failed demand is observed. We show that the LAs shift workload in hours with less demand to prepare for hours with increasing demands. To this end, we analyze the amount of relocated bikes over the course of the day for DLA(VFA), SLA(180), and STR(0.2) as depicted in Fig. 8. On the abscissa, the time in hours is depicted. The ordinate shows the average numbers of relocated bikes for the three policies. We observe that in the time between hours 2 and 7, the LAs relocate many bikes while STR conducts only a few relocations. Comparing the relocations to the average number of trips in this time span depicted in Fig. 6, we see that the LAs prepare for the increase in demand of the morning peak around hour 8. This preparation then leads to less unsatisfied demand around that time as shown in Fig. 9. The ordinate depicts the average numbers of failed rentals and returns per hour. For all three policies, we observe an increase around hour 8 and later around hours 17 and 18 reflecting the high demands in peak hours. Nevertheless, due to the aforementioned preparations of the LAs, the peaks are substantially lower than for the STR.

5.5. Horizons

In this section, we analyze the horizons of the DLA and show how they reflect the demand pattern of the instances.

The LA horizons are shown in Fig. 10. The ordinates depict the current hour. The abscissas depict the lengths of the horizons. The left-hand side shows the horizons for SLA(180). Since here the horizons are static, we observe a constant horizon of 180 minutes, i.e., three hours. The DLA's horizons selected by VFA are depicted on the right-hand side. We observe two significant peaks in the DLA horizons. One peak is around hour 8, another around hour 17. We further see a small peak around hours 11 and 12. The DLA only conducts simulations up to these peaks. Thus, the horizons reflect the temporal pattern of trips depicted in Fig. 6. In the early morning, the simulations comprise the hours until the first commuter peak. In the following hours, the simulations last until the lunchtime peak around noon. Then, the simulations comprise the hours until the second commuter peak. All in all, the VFA learns to simulate up to but not further than the peak hours. A simulation up to the commuter peaks is necessary to anticipate the

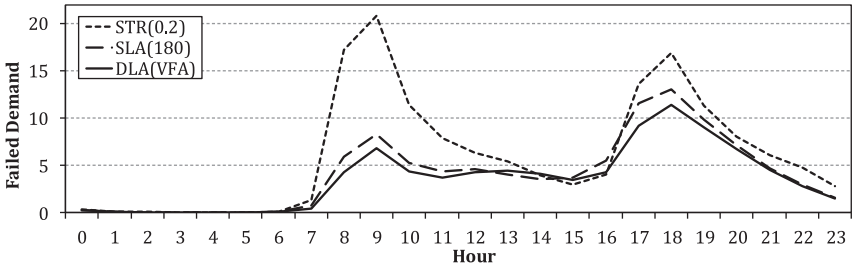

Fig. 9. Failed demand per hour.

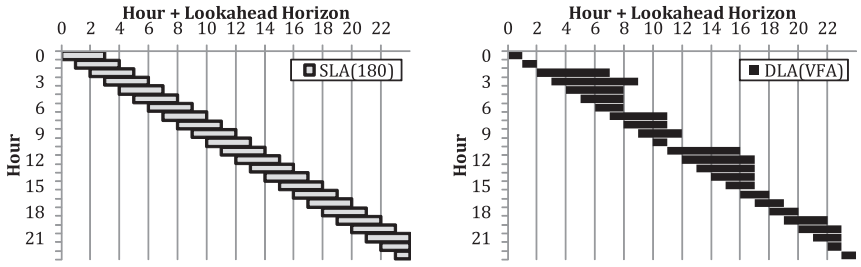

Fig. 10. SLA(180) and DLA(VFA) horizons.

Table 4
Rollout algorithm's results: avg. failed demand and standard error.

	Simulation runs		
Horizon	16	32	64
60	211.93 (±3.37)	188.59 (±3.35)	165.67 (±3.04)
120	190.52 (±3.30)	195.54 (±3.76)	164.63 (±3.37)
180	207.76 (±3.42)	209.58 (±5.35)	172.04 (±3.08)
240	205.12 (±4.13)	209.37 (±4.48)	168.19 (±3.24)
300	214.66 (±4.02)	225.22 (±5.13)	177.08 (±4.01)
360	211.97 (±3.87)	223.87 (±4.89)	177.33 (±3.40)

Table 5
Rollout algorithm's runtimes: maximum and overall runtime in minutes.

	Simulation runs					
	16		32		64	
Horizon	*Max.*	*Compl.*	*Max.*	*Compl.*	*Max.*	*Compl.*
60	0.36	17.65	0.68	41.92	1.24	87.11
120	0.51	26.07	0.92	54.72	2.15	109.66
180	0.73	31.62	1.34	67.57	2.73	134.55
240	0.94	38.18	2.00	81.44	3.89	168.04
300	1.24	43.42	2.60	100.22	4.81	203.77
360	1.55	54.80	2.97	120.55	6.01	244.70

commuters' demands. In these hours, the numbers of demands are high and the transitions are therefore highly stochastic. An anticipation further than these hours leads therefore to inaccuracy and to inferior decisions. VFA is able to capture these phenomena adapting the horizons to the demand pattern and makes decisions on accurate information.

Given the knowledge that the temporal aspect of the demand pattern can explain the simulation horizons, we parametrize the DLA manually. In Appendix C, we define two manual-parametrized DLAs and show how the manual parametrization leads to inferior results.

5.6. Simulation of routing decisions

The presented lookahead methods ignore any routing within the simulations. However, anticipation of future routing in the evaluation of current decisions may provide additional benefit. In this section, we analyze how the integration of routing decisions impacts the performance of our lookaheads. To this end, we draw on the concept of a post-decision state rollout algorithm (for an overview of rollout algorithms, we refer to Goodson et al., 2017). The idea of a rollout algorithm is to evaluate post-decision states by using a base policy to approximate the expected future value of a decision. Rollout algorithms for inventory routing are, for example, presented by Bertazzi et al. (2013). As Ulmer et al. (2017a) show, the rollout improves the base policy given that the expected value of the base policy is known or given a sufficient number of simulation runs.

To apply a rollout algorithm to the SDIRP, a base policy needs to be selected. Because the runtime per simulation run is highly limited, we are not able to use the DLA or an SLA. Thus, we use STR(0.2) as base policy. Within our simulation runs, inventory and routing decisions are made by STR(0.2). The obtained penalties are then used to decide in the current decision state.

In our MDP, we have to make an inventory and a routing decision in every decision state. We still limit the inventory decisions to 25, 50, and 75% of the current station's capacity and consider every station (expect for the current) to be the vehicle's next station. Thus, we have to evaluate each combination of inventory and routing decision leading to $3 \cdot (169 - 1) = 504$ post-decision states. Every post-decision state is evaluated over 16, 32, or 64 simulation runs. Following the idea of the SLA, the horizons in the simulations are limited to $\delta \in \{60, \ldots, 360\}$ minutes. This leads to $3 \cdot 6 = 18$ different tunings for the rollout algorithm. Because the rollout requires extensive simulations, we only run 100 test days. To allow an evaluation of the significance of our results, we also provide the standard errors. The results are depicted in Table 4. We observe a slight increase in solution quality with an increasing number of simulation runs. The result for DLA(VFA) and STR(0.2) are 85.405, or 145.126, respectively. Thus, even with 64 simulations runs, the rollout algorithm is not able to achieve the results of the base policy. This indicates that the SDIRP is highly stochastic requiring a

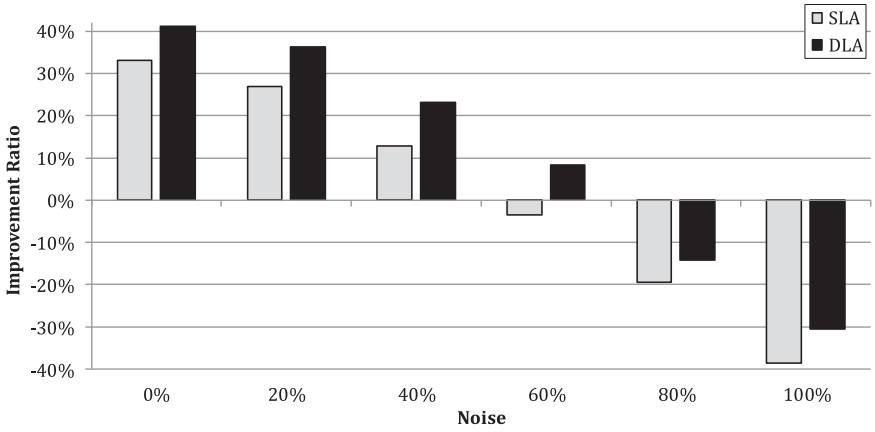

Fig. 11. Improvement of DLA and SLA compared to STR for varying amounts of noise.

significant number of simulation runs to obtain comparable results. However, already with 64 simulation runs, the runtimes are significant as shown in Table 5. The table shows the runtimes in minutes. The first entry reflects the maximum runtime observed in a decision point over all 100 runs. The second entry represents the average runtimes for the complete MDPs over all 100 runs. As expected, we observe a increased runtime with increasing horizon lengths and increasing numbers of simulation runs. The highest runtime observed for a horizon of 120 minutes and 64 simulation runs (which offers the best solution quality among the 18 parametrizations) exceeds 2 minutes. Thus, real-time decision making is not possible for this policy.

In summary, an integration of routing decisions in the simulations does not improve the performance of the algorithm but even provides inferior results. Furthermore, the runtime required for simulations is significant.

5.7. Limitations of the DLA

In this final section of the computational studies, we analyze the limitations of the DLA. We show that with a decrease in the demand pattern's distinction and an increase in noise, the advantages of the DLA are diminished. We show that if the impact of noise becomes too significant, anticipation is less possible and decision making based on conventional safety buffers is advantageous. To this end, we systematically construct instance settings differing in the demand pattern's distinction and the amount of noise. We focus on the point of times the demands occur. The amount of noise is tuned by parameter $\lambda \in [0, 1]$. Given λ, we modify a percentage of λ of the sampled original trips based on historical data. For this percentage, we maintain origin, destination, and ride time of the observation but randomly draw the point of time the trip occurred from a uniform distribution. Thus, we add noise in the point of time the trips occur. Parameter $\lambda = 0$ results in the original instance settings with the commuter peaks. Parameter $\lambda = 1$ results in trips equally distributed over time. For instance settings of $\lambda \in \{0, 0.2, \ldots, 1\}$, we now compare the performances of DLA and SLA with STR. We must note that for DLA and every λ an approximation of the value function has been applied. The best SLA's lookahead horizon is 180 minutes for $\lambda \leq 0.2$, 120 minutes for $\lambda = 0.4$, and 60 minutes for $\lambda \geq 0.6$. The best safety buffer percentage for STR remains 0.2 for all λ.

The average improvements of the DLA and SLA are depicted in Fig. 11. On the abscissa, the percentage λ is shown. The ordinate depicts the improvement to STR(0.2). We observe a constant decrease with increasing noise. Given a noise of $\lambda \geq 0.8$, STR(0.2) even outperforms the DLA. Thus, the DLA performs well when a heterogeneous demand pattern can be experienced. In cases where demand becomes more and more random, the anticipation is impeded.

6. Conclusion

In this paper, we have presented the stochastic-dynamic inventory routing problem for bike sharing systems (SDIRP). The objective of the SDIRP is to avoid unsatisfied demand by dynamically relocating bikes. To anticipate potential future demands in the current inventory decisions, we have presented a dynamic lookahead policy (DLA). The policy uses online simulations to look ahead for a predefined horizon. Because the demand pattern changes over the course of the day, the horizons are time-dependent and autonomously parametrized by means of value function approximation. Comparisons with conventional policies from the literature and LAs with static horizons reveal the benefits of both anticipation by the lookahead and time-dependent horizons of the DLA. We have further shown how the DLA is able to autonomously adapt to the demand pattern.

Future research may focus on both model and method. For the SDIRP, a fleet of vehicles may be considered. In the consideration of fleets, two additional challenges arise. First, for a fleet, the vehicles need to be coordinated to avoid simultaneous service of stations by different vehicles and to enable many suitable operations in the entire city. Second, decision states in the Markov decision process occur more frequently. Thus, the runtime available for simulations may be further restricted.

The proposed method may further be transferred to larger instance settings or instance settings with different demand patterns. The method may be extended by making the horizons dependent on additional state parameters like vehicle locations. The method of non-parametric policy search may further be generalized by assigning individual policies to a set of states dependent on the states' characteristics.

Acknowledgments

The authors thank Ninja Soeffker, Barrett W. Thomas, the associate editor, and the two anonymous reviewers for their helpful suggestions and comments in preparation of this work.

Appendix A. Boltzmann exploration

Exploration of the search space can be supported by occasionally deviating from the currently found best solution. Thus, we apply Boltzmann exploration in our VFA using the additional notation in Table A.6. In hour ρ, every lookahead horizon δ_i is assigned a probability

Table A.6
Notation of the Boltzmann exploration.

Symbol	Description
$\eta \in \mathbb{N}_0$	Number of traversed trajectories
$\epsilon : \mathrm{P} \to \mathbb{R}_0^+$	Coefficient to control exploitation and exploration
$\phi : \mathrm{P} \times \Delta \to \mathbb{R}^+$	Probabilities of choosing a lookahead horizon

$\phi(\rho, \delta_i)$ to be selected. The probability depends on the value $\tilde{v}(\rho, \delta_i)$ and a coefficient $\epsilon(\rho)$ controlling exploration and exploitation:

$$\phi(\rho, \delta_j) = \frac{\left(\exp\left(\epsilon(\rho) \cdot \tilde{v}(\rho, \delta_j)\right)\right)^{-1}}{\sum_{\delta \in \Delta}\left(\exp\left(\epsilon(\rho) \cdot \tilde{v}(\rho, \delta)\right)\right)^{-1}}. \tag{A.1}$$

According to Eq. (A.1), linearly decreasing input of the exponential function leads to exponentially increasing probabilities. The coefficient, as defined in Eq. (A.2), manipulates the values. It depends on the number of traversed trajectories η and on the difference of maximum and minimum values:

$$\epsilon(\rho) = \begin{cases} 0, & \text{if } \max_{\delta \in \Delta} \tilde{v}(\rho, \delta) = \min_{\delta \in \Delta} \tilde{v}(\rho, \delta) \\ \frac{\eta \cdot 0.01}{\max_{\delta \in \Delta} \tilde{v}(\rho, \delta) - \min_{\delta \in \Delta} \tilde{v}(\rho, \delta)}, & \text{else.} \end{cases} \tag{A.2}$$

With ongoing trajectories, η linearly grows and so does the input of the exponential function if the value remains constant. Thus, exploration is allowed in the beginning of the approximation phase. The procedure is more and more forced to exploitation since the lowest values will dominate even if the difference between the lowest and the second lowest is small. Further, the difference of maximum and minimum values has an impact on the probability. If the difference is large, the exponential function's input is small and thus exploration will be allowed. In this way, inferior lookahead horizons will likely be selected to firm the approximation. If all values are the same, all probabilities will be identical. For more details, we refer to Appendix B.3. A lookahead horizon is selected with respect to the probabilities and a random variable. The selected lookahead horizon is applied in the MDP's current hour.

Appendix B. Algorithms

In this section, we provide formal definitions of the dynamic lookahead policy, the VFA-guided non-parametric policy search, the Boltzmann exploration, and the short-term relocation policy.

B1. Dynamic lookahead policy

In this section, we present the algorithmic definition of a lookahead policy.

The DLA receives a decision state $s = (t, f_k, n_k^v, f_k^v)$ and returns an inventory decision ι^x and a routing decision n^x. In the for-loop between lines 2 – 28, we evaluate the three fill level percentages μ_i (low, medium, and high). The associated inventory decisions ι_i are determined in lines 3 –10 with respect to the station's fill level $f_k^{n_k^v}$ and the vehicle's load f_k^v and capacity c_v. Before the simulations are started, we initialize the numbers of failed rentals $\gamma_{\iota_i,n}^-$ and failed returns $\gamma_{\iota_i,n}^+$ in lines 11 – 14. For every ι_i, we traverse 32 simulations j in lines 16 – 27. Since we aim on evaluating inventory decisions, we apply ι_i to the current decision state s in line 17. To determine the initial decision state s_{kj} of simulation j, we further apply the transition function ω to sample demand (see Sections 3.2 and 3.3). Every simulation j has a number of decision points processed in lines 19 – 26. In lines 21 and 24, we steadily update the approximated failed rentals and returns γ^-, γ^+. In line 25, we update the simulation's decision state. In line 29, we select the inventory decision ι^x, minimizing the expected failed demand at the current station. In line 29, we select the station where the vehicle can prevent as most failed demand. The decisions are returned in line 31.

Algorithm 1: Dynamic lookahead policy.

Input : $(t, f_k, n_k^v, f_k^v) = s \in S$
Output : $(\iota^x, n^x) = x \in X$
1 **for all** $\mu_i \in \{\mu_1, \mu_2, \mu_3\}$; // For all target fill level percentages
2 **do**
3 $\quad \iota_i \leftarrow 0$
4 $\quad$ **if** $\mu_i \cdot c(n_k^v) > f_k^{n_k^v}$; // If fill level lower than target
5 $\quad$ **then**
6 $\quad\quad \iota_i \leftarrow \min\{\mu_i \cdot c(n_k^v) - f_k^{n_k^v}, f_k^v\}$; // Define inventory decision
7 $\quad$ **else if** $\mu_i \cdot c(n_k^v) < f_k^{n_k^v}$; // If fill level higher than target
8 $\quad$ **then**
9 $\quad\quad \iota_i \leftarrow \max\{\mu_i \cdot c(n_k^v) - f_k^{n_k^v}, c_v - f_k^v\}$; // Define inventory decision
10 $\quad$ **end**
11 $\quad$ **for all** $n \in N$ **do**
12 $\quad\quad \gamma^-_{\iota_i,n} \leftarrow 0$
13 $\quad\quad \gamma^+_{\iota_i,n} \leftarrow 0$
14 $\quad$ **end**
15 $\quad$ **for all** $j \in \{1, \ldots, 32\}$; // For 32 simulations
16 $\quad$ **do**
17 $\quad\quad s_{kj} \leftarrow \omega(s, (\iota_i, n_k^v))$; // Apply inventory decision
18 $\quad\quad$ **while** $k^j \neq k^j_{\boldsymbol{max}}$; // Until end of lookahead horizon
19 $\quad\quad$ **do**
20 $\quad\quad\quad$ **for all** $n \in N$; // For all stations update failed demands
21 $\quad\quad\quad$ **do**
22 $\quad\quad\quad\quad \gamma^-_{\iota_i,n} \leftarrow \gamma^-_{\iota_i,n} + \frac{1}{32} \cdot p_j^-(s_{kj}, n)$
23 $\quad\quad\quad\quad \gamma^+_{\iota_i,n} \leftarrow \gamma^+_{\iota_i,n} + \frac{1}{32} \cdot p_j^+(s_{kj}, n)$
24 $\quad\quad\quad$ **end**
25 $\quad\quad\quad s_{kj} \leftarrow \omega\left(s_{kj}, \left(0, n_k^v\right)\right)$; // Apply no further decisions
26 $\quad\quad$ **end**
27 $\quad$ **end**
28 **end**
29 $\iota^x \leftarrow \arg\, min_{\iota \in \{\iota_1, \iota_2, \iota_3\}} \gamma^-(\iota, n_k^v) + \gamma^+(\iota, n_k^v)$; // Select inventory decision
30 $n^x \leftarrow \arg\, max_{n \in N} \max\{\min\{\gamma^-(\iota^x, n), f_v^t - \iota^x\}, \min\{\gamma^+(\iota^x, n), c_v - f_v^t - \iota^x\}\}$; // Select next station
31 **return** (ι^x, n^x)

B2. VFA-guided non-parametric policy search

The VFA-guided non-parametric policy search approximates the values $\tilde{v}$ of combinations of hours ρ and lookahead horizons δ. Table 3 provides an overview on all sets, objects, and functions of the VFA-guided non-parametric policy search.

We initialize the approximated values $\tilde{v}$ and numbers of horizon selections α in lines 2 – 5. In lines 6 – 31, we traverse trajectories until a termination criterion occurs. At first, the simulation horizons, defining the DLA, are selected by the Boltzmann exploration in lines 8 – 11. For every hour $\rho \in$ P, we select one simulation horizon δ^ρ. In line 12, we initialize the first decision point k. To this end, in lines 14 – 17, we randomly distribute bikes over all stations. The ratio of bike racks and bikes corresponds to 2: 1. In other words, the total number of bikes is equal to 50% of the sum of stations' capacities: $\frac{1}{2} \cdot \sum_{n \in N} c(n)$. Then, the initial decision state s_o is initialized in line 20. In lines 22 – 26, decision points of the trajectory's MDP are processed. In line 23, we determine the hour i. Every hour has a length of 60 minutes. In line 23, we apply the lookahead policy with the selected horizon δ^{ρ_i} for hour ρ_i to the current decision state s_k. In lines 28 – 30, the approximated values $\tilde{v}$ are updated according to the observed failed demands in the current trajectory. In fact, we determine the averages over all combinations of hours and horizons. Finally, the approximated values $\tilde{v}$ are returned in line 32.

Algorithm 2: VFA-guided non-parametric policy search.

Input : P, Δ
Output : $\tilde{\nu}$

1 **for all** $\rho \in P, \delta \in \Delta$; // Initialize value function
2 **do**
3 $\quad \alpha(\rho, \delta) \leftarrow 0$
4 $\quad \tilde{\nu}(\rho, \delta) \leftarrow 0$
5 **end**
6 **repeat**
7 $\quad$ **for all** $\rho \in P$; // For all hours select lookahead horizon
8 $\quad$ **do**
9 $\quad\quad \delta^{\rho} \leftarrow \text{boltzmann}(\rho)$; // Boltzmann exploration selects horizon
10 $\quad\quad \alpha(\rho, \delta^{\rho}) \leftarrow \alpha(\rho, \delta^{\rho}) + 1$
11 $\quad$ **end**
12 $\quad k \leftarrow 0$; // Initialize first decision point
13 $\quad$ **for all** $b \in \{1, \ldots, \frac{1}{2} \cdot \sum_{n \in N} c(n)\}$; // Distribute bikes initially
14 $\quad$ **do**
15 $\quad\quad n \leftarrow \text{random}\{n_1, \ldots, n_{\max}\}$; // Select random station
16 $\quad\quad f_k^n \leftarrow f_k^n + 1$; // Increase selected stations fill level
17 $\quad$ **end**
18 $\quad n_k^v \leftarrow n_0$; // Vehicle starts at the depot
19 $\quad f_k^v \leftarrow 0$; // Vehicle initial load initially is void
20 $\quad s_0 \leftarrow \left(k, f_k, n_k^v, f_k^v\right)$; // Initialize first decision state
21 $\quad$ **while** $k \neq k_{max}$; // Until final decision point
22 $\quad$ **do**
23 $\quad\quad i \leftarrow \lfloor \frac{t_k}{60} \rfloor$; // Determine hour
24 $\quad\quad s_{k+1} \leftarrow \omega\left(s_k, \pi^{\delta^{\rho_i}}(s_k)\right)$; // Apply DLAs decision and transition
25 $\quad\quad k \leftarrow k + 1$; // Go to next decision point
26 $\quad$ **end**
27 $\quad$ **for all** $\rho \in P$; // For all hours update values
28 $\quad$ **do**
29 $\quad\quad \tilde{\nu}(\rho, \delta^{\rho}) \leftarrow \frac{\alpha(\rho, \delta^{\rho}) - 1}{\alpha(\rho, \delta^{\rho})} \cdot \tilde{\nu}(\rho, \delta^{\rho}) + \frac{1}{\alpha(\rho, \delta^{\rho})} \cdot \sum_{k=\bar{k}^{\rho}}^{k_{\max}} p\left(s_k, \pi^{\delta^{\rho}}(s_k)\right)$
30 $\quad$ **end**
31 **until** *STOP*;
32 **return** $\tilde{\nu}$

B3. Boltzmann exploration

The Boltzmann exploration selects a lookahead horizon δ for an hour ρ. In lines 1 – 4, we define the parameter ϵ controlling exploration and exploitation dependent on the number of trajectories η of the VFA and lowest and highest approximated values. In lines 6 – 8, every horizon δ_i is assigned a probability $\phi(\rho, \delta_i) \in \mathbb{R}$ in the interval [0,1] dependent on the values. In lines 12 – 16, we implement a roulette wheel selecting a horizon with respect to the probabilities.

Algorithm 3: Boltzmann exploration.

Input : $\rho \in \mathrm{P}$
Output : $\delta \in \Delta$
$\epsilon(\rho) \leftarrow 0$; // Controls exploitation and exploration
if $\max_{\delta \in \Delta} \tilde{v}(\rho, \delta) \neq \min_{\delta \in \Delta} \tilde{v}(\rho, \delta)$ **then**
  $\epsilon(\rho) \leftarrow \frac{\eta \cdot 0.01}{\max_{\delta \in \Delta} \tilde{v}(\rho,\delta) - \min_{\delta \in \Delta} \tilde{v}(\rho,\delta)}$
end
for all $\delta_i \in \Delta$; // For all horizons, determine selection probability
do
  $\phi(\rho, \delta_i) \leftarrow \frac{\left(\exp\left(\epsilon(\rho) \cdot \tilde{v}(\rho, \delta_i)\right)\right)^{-1}}{\sum_{\delta \in \Delta} \left(\exp\left(\epsilon(\rho) \cdot \tilde{v}(\rho, \delta)\right)\right)^{-1}}$
end
$w \leftarrow 0$
$r \leftarrow \mathrm{random}(0, 1)$
$i \leftarrow 0$
while $w \leq r$; // Roulette wheel selection
do
  $w \leftarrow w + \phi(\rho, \delta_i)$
  $i \leftarrow i + 1$
end
return δ_i

B4. Short-term relocation policy

The STR returns an inventory decision ι^x and a routing decision n^x with respect to the decision state $s = (t, f_k, n_k^v, f_k^v)$. In lines 1 – 8, we detect if the current station n_k^v has a suitable numbers of bikes (line 3) and free bike racks (line 6) according to the safety buffer β. If a safety buffer is violated, we define an inventory decision ι^x with respect to the vehicle's load f_t^v and capacity c_v. In lines 9 – 18, we detect which other stations violate the safety buffers and if the vehicle could pick up or deliver bikes. In line 19, we choose the nearest station n^x demanding relocations. The decisions are returned in line 20.

Algorithm 4: Short-term relocation policy.

Input : $(t, f_k, n_k^v, f_k^v) = s \in S$
Output : $(\iota^x, n^x) = x \in X$
$\iota^x \leftarrow 0$
if $f_k^{n_k^v} < \lceil \beta \cdot c(n_k^v) \rceil$; // If not enough bikes
then
  $\iota^x \leftarrow \min\left\{\lceil \beta \cdot c(n_k^v) \rceil - f_k^v,\ f_k^v\right\}$
else if $\lceil (1 - \beta) \cdot c(n_k^v) \rceil < f_k^{n_k^v}$; // If too many bikes
then
  $\iota^x \leftarrow \max\left\{\left\lceil (1 - \beta) \cdot c(n_k^v) - f_k^{n_k^v} \right\rceil,\ f_k^v - c_v\right\}$
end
for all $n \in N$ **do**
  $\sigma(n) \leftarrow 0$
  if $f_k^n < \lceil \beta \cdot c(n) \rceil \ \wedge\ 0 < f_k^v - \iota^x$; // If not enough bikes
  then
    $\sigma(n) \leftarrow \frac{1}{\tau(n_k^v, n)}$
  else if $\lceil (1 - \beta) \cdot c(n) \rceil < f_k^n \ \wedge\ f_k^v - \iota^x < c_v$; // If too many bikes
  then
    $\sigma(n) \leftarrow \frac{1}{\tau(n_k^v, n)}$
  end
end
$n^x = \arg\, max_{n \in N} \sigma(n)$
return (ι^x, n^x)

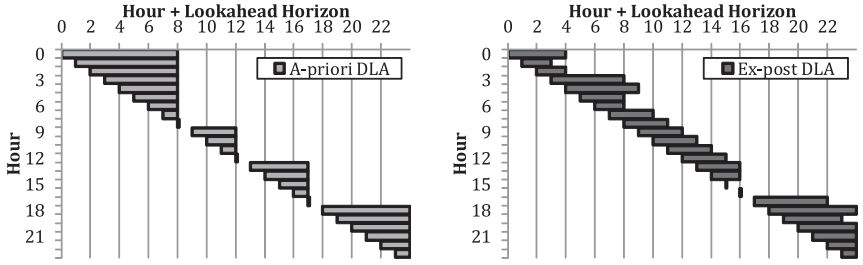

Fig. C.12. A-priori DLA's and ex-post DLA's lookahead horizons.

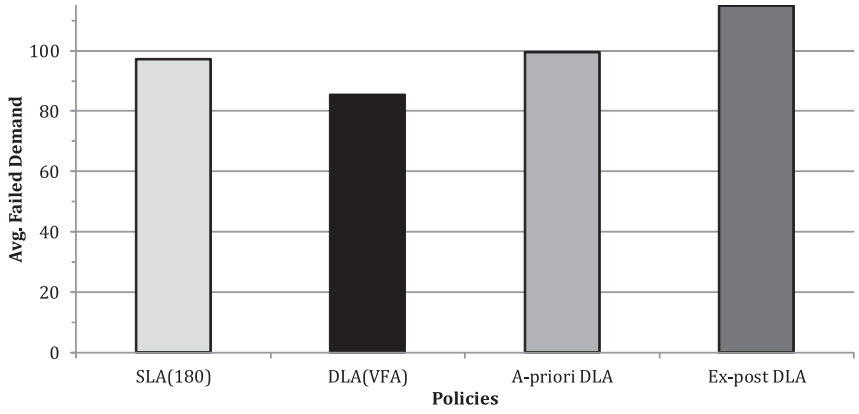

Fig. C.13. Results of the lookahead policies.

Appendix C. Manual-parametrized dynamic lookahead policies

In this section, we manually parametrize two DLAs using the knowledge gained in the computational studies. In Section 5.5, we found out that DLA(VFA) mostly does not simulate ahead the demand peaks. Thus, we manually define the A-priori DLA, simulating exactly until the hour of the next peak. In a peak hour, we do not simulate and apply STR(0.2) instead. We further observed that the policies' performances heavily differ in distinct hours. Thus, we manually define the Ex-post DLA, selecting in every hour the lookahead horizon of the best performing SLA or STR, respectively. Fig. C.12 depicts the lookahead horizons of both manually parametrized DLAs.

Fig. C.13 depicts the average failed demands achieved by SLA(180) and the DLAs. A-priori DLA achieves 99.577 failed demands and, thus, performs slightly worse than SLA(180). Ex-post DLA achieves 114.931 failed demands and provides the lowest solution quality in this collection.

Both manual-parametrized DLAs perform much worse than DLA(VFA). For A-priori DLA, we only consider the temporal aspect of the demand pattern. Thus, we neglect the spatial aspect. VFA considers both aspects and, thus, provides a higher solution quality. For Ex-post DLA, we allow the fallacy that the failed demands in a hour are due to the selected lookahead horizon in this certain hour. In fact, a decision often pays off late. Thus, an hour's failed demands are due to the decisions of the previous hours.

Appendix D. Handling demands

In this section, we describe the process of handling rental and return demands between two decision points k and $k+1$.

The process is depicted by Fig. D.14. Demands are in a queue ordered by the associated point in time. We handle all demands in the time interval $[t_k, t_{k+1}]$. The demand with the earliest point in time t is handled first. We first have to distinguish rental and return demands.

If it is a rental demand and a bike is available at the corresponding station n, the user can rent a bike. Let n_i be the destination station of the user. He or she will approach to return the bike in time $t + \tau(n, n_i)$ at n_i. Thus, a return demand is inserted in the queue. If no bike is available at n, the rental demand fails. The user will approach the nearest non-empty station n_j.[4] Thus, we insert a new rental demand at n_j in time $t + \tau(n, n_j) \cdot 3$ in the queue.[5] Since one demand has failed, we increase the penalties between the two decision states by one.

If it is a return demand and a free bike rack is available at the corresponding station n, the user can return the bike. In this case, the trip ends. If no free bike rack is available at n, the user will try to return the bike at the nearest non-full station n_j. Thus, we insert a new return demand at n_j in time $t + \tau(n, n_j)$ in the queue. Again, since one demand has failed, we increase the penalties between the two decision states by one.

The process is repeated until the queue is empty or the queue's first demand takes place after the next decision point $k+1$, or time t_{k+1}, respectively.

[4] We assume that the stations are equipped with a terminal, providing real-time information on the stations' fill levels.

[5] Function τ refers to $15\,\frac{\text{km}}{\text{h}}$. We assume pedestrians to move with a speed of $5\,\frac{\text{km}}{\text{h}}$. Thus, the travel time is multiplied by 3.

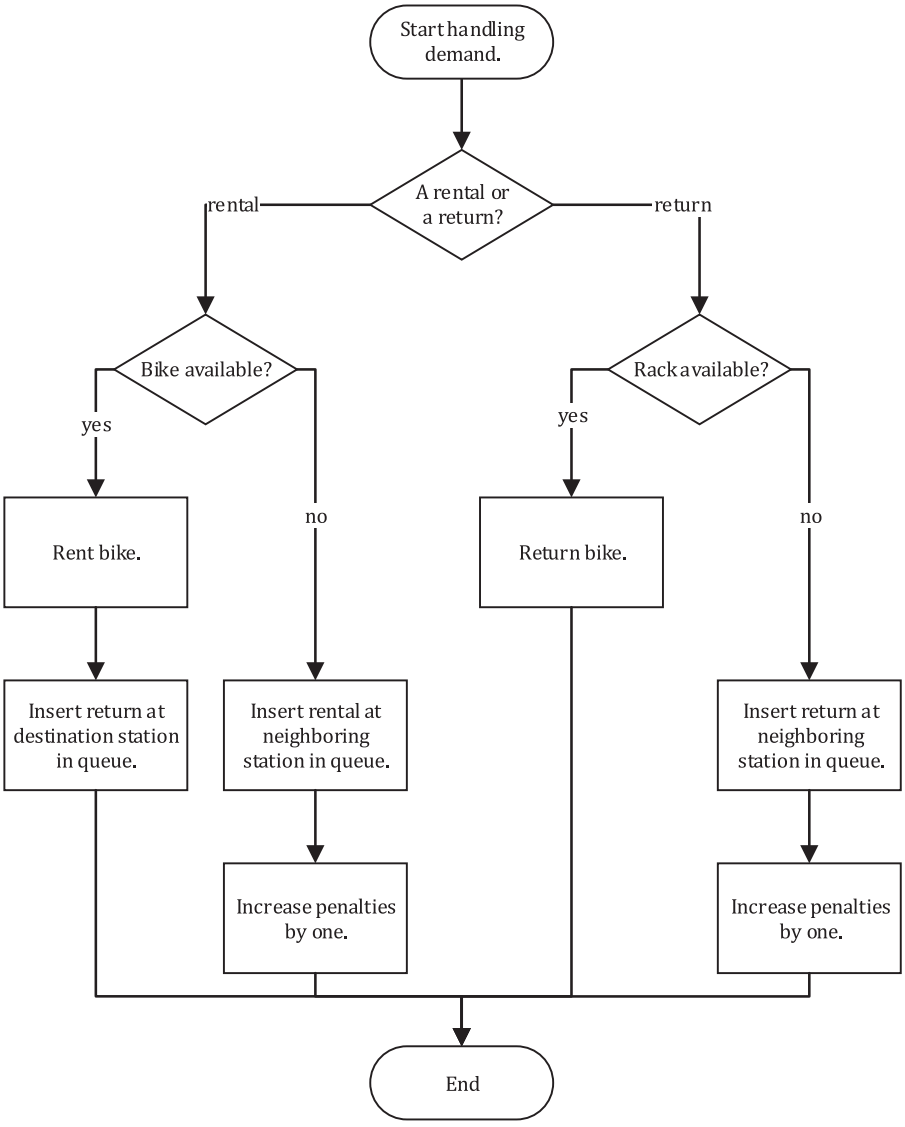

Fig. D.14. Handling demands.

Appendix E. Results

In this section, we present further details on results achieved by our policies.

Table E.7 depicts the average number of trips and results by STR(0.2), SLA(180), and DLA(VFA) for every hour of the day. Σp indicates the average numbers of failed demands in the associated hour. $\Sigma|\iota|$ shows the average numbers of relocations. The numbers are also shown in Figs. 6, 8, and 9 (Section 5).

A comprehensive overview on results by all policies and parameters is provided by Tables E.8, E.9 and E.10. Here, we separate the numbers of failed rentals and returns. We observe that always more rentals than returns fail. The user detours, occurring if a demand fails and the user tries a neighboring station, are alternative measures for the solution quality. On average, a failed demand results in a detour of 4.0 – 4.5 minutes. Here, the policies are very similar. Regarding the STRs, the higher the safety buffer, the more stations are defined as imbalanced, resulting in more relocated bikes and more served stations. For all STRs, we observe significantly more served stations than for all LAs. This is due to STRs' myopic routing decisions. For the LAs, we experience a significantly higher number of relocated bikes per station compared to the STRs. This is due to the LAs' inventory decisions, drawing on expected failed demands. Regarding the CPU times, we distinguish the time for solving one decision point k and a sequence K of decision points (representing one complete MDP, or one working day, respectively). The CPU time per k is of interest, since in a real-world application decisions are made in real-time. Thus, a decision support system needs to return a decision quickly when a vehicle arrives at a station. The numbers refer to the maximum CPU time observed for one k over all 10,000 instances solved. For the STRs, the CPU time is negligible. The longer the SLAs' lookahead horizons, the higher the CPU times. Nevertheless, SLA(360)'s maximum CPU time per decision point is slightly higher than one second. The DLAs always return decisions in less than $\frac{3}{4}$ seconds. From this perspective, all policies are suitable for real-world decision making. The CPU time for one sequence K of decision points is relevant in the evaluation of policies and parameters. We observe high CPU times demanding high computational effort to gain reliable results.

Table E.7
Detailed Results.

		do nothing	STR(0.2)		SLA(180)		DLA(VFA)	
Hour	Trips	Σp	Σp	$\Sigma \lvert\iota\rvert$	Σp	$\Sigma \lvert\iota\rvert$	Σp	$\Sigma \lvert\iota\rvert$
0	17.458	0.585	0.327	9.756	0.239	11.131	0.243	11.707
1	10.220	0.216	0.105	7.051	0.059	6.858	0.067	6.079
2	6.524	0.175	0.059	4.770	0.031	5.147	0.039	9.467
3	2.595	0.064	0.014	3.254	0.011	6.324	0.014	13.479
4	4.053	0.038	0.006	1.987	0.004	9.969	0.006	11.756
5	13.136	0.136	0.011	1.210	0.011	13.644	0.015	9.217
6	31.218	0.448	0.055	2.708	0.048	12.041	0.042	10.273
7	99.282	4.182	1.328	6.619	0.743	13.835	0.386	14.727
8	126.756	27.843	17.207	14.202	5.906	20.425	4.236	19.790
9	96.443	31.160	20.845	17.279	8.273	20.549	6.825	20.251
10	87.894	19.159	11.360	17.260	5.217	19.718	4.377	17.976
11	115.905	16.433	7.891	16.562	4.376	18.765	3.726	17.975
12	145.760	15.862	6.273	15.637	4.610	18.007	4.292	17.612
13	134.456	14.839	5.395	14.971	4.049	17.224	4.386	16.871
14	128.645	12.251	3.923	14.253	3.561	17.240	4.060	15.824
15	139.636	10.130	2.931	12.842	3.649	14.904	3.419	15.369
16	194.032	10.792	4.002	11.740	5.523	15.361	4.239	16.590
17	223.985	17.291	13.612	12.913	11.570	19.723	9.145	19.199
18	194.556	21.277	16.868	13.940	13.026	19.941	11.405	19.417
19	160.824	16.884	11.298	14.371	9.858	19.304	9.047	18.989
20	126.248	13.288	8.038	14.131	7.127	18.619	6.725	18.618
21	88.493	11.063	6.065	13.817	4.708	17.524	4.479	17.203
22	60.674	9.379	4.734	13.615	2.959	16.016	2.759	15.713
23	37.208	6.300	2.784	10.230	1.533	11.749	1.475	11.549
0 – 23	2,246.000	259.794	145.126	265.118	97.091	364.016	85.405	365.649

Table E.8
Short-term Relocation Policies' Results.

	do nothing	STR(0.1)	STR(0.2)	STR(0.3)	STR(0.4)	STR(0.5)
Failed Demands	259.794	146.699	145.126	166.358	187.945	210.758
Rentals	138.139	80.237	81.716	91.268	103.009	117.995
Returns	121.655	66.462	63.410	75.090	84.935	92.763
User Detours [min]	1,110.196	631.163	635.423	722.527	829.983	944.189
Detour per Failed Demand [min]	4.273	4.302	4.378	4.343	4.416	4.480
Relocated Bikes	–	180.721	265.118	371.453	433.500	471.698
Served Stations	–	128.395	146.264	170.902	179.553	196.967
Relocations per Served Station	–	1.408	1.813	2.173	2.414	2.395
Max. CPU Time per k [sec]	–	0.005	0.005	0.005	0.009	0.010
Avg. CPU Time per K [sec]	–	0.139	0.142	0.146	0.150	0.157

Table E.9
Static Lookahead Policies' Results.

	SLA(60)	SLA(120)	SLA(180)	SLA(240)	SLA(300)	SLA(360)
Failed Demands	120.440	101.433	97.091	99.272	104.048	109.625
Rentals	65.303	55.374	53.515	54.687	57.220	60.259
Returns	55.138	46.059	43.576	44.586	46.828	49.365
User Detours [min]	492.214	417.963	405.324	418.354	442.158	472.246
Detour per Failed Demand [min]	4.087	4.121	4.175	4.214	4.250	4.308
Relocated Bikes	335.682	361.313	364.016	353.209	339.556	323.022
Served Stations	48.468	49.279	49.970	49.259	47.943	45.821
Relocations per Served Station	6.926	7.332	7.285	7.170	7.083	7.050
Max. CPU Time per k [sec]	0.265	0.344	0.500	0.702	0.843	1.030
Avg. CPU Time per K [sec]	7.498	8.894	12.386	17.541	24.357	32.720

Table E.10
Dynamic Lookahead Policies' Results.

	DLA(VFA)	A-priori DLA	Ex-post DLA
Failed Demands	85.405	99.577	114.931
Rentals	46.407	54.681	68.484
Returns	38.998	44.896	46.447
User Detours [min]	360.446	425.459	484.571
Detour per Failed Demand [min]	4.220	4.273	4.216
Relocated Bikes	365.649	362.644	375.893
Served Stations	54.756	62.969	57.220
Relocations per Served Station	6.678	5.759	6.569
Max. CPU Time per k [sec]	0.702	0.809	0.702
Avg. CPU Time per K [sec]	12.809	12.469	12.004

References

Adelman, D., 2004. A price-Directed approach to stochastic inventory/routing. Oper. Res. 52 (4), 499–514.

Bellman, R., 1957. A Markovian Decision Process. Technical Report. DTIC Document.

Bertazzi, L., Bosco, A., Guerriero, F., Lagana, D., 2013. A stochastic inventory routing problem with stock-out. Transp. Res. Part C 27, 89–107.

Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-B., Fleury, E., 2011. Shared bicycles in a city: a signal processing and data analysis perspective. Adv. Complex Syst. 14 (3), 415–438.

Brinkmann, J., Ulmer, M.W., Mattfeld, D.C., 2015. Short-term strategies for stochastic inventory routing in bike sharing systems. Transp. Res. Procedia 10, 364–373.

Brinkmann, J., Ulmer, M.W., Mattfeld, D.C., 2016. Inventory routing for bikes sharing systems. Transp. Res. Procedia 10, 316–327.

Büttner, J., Mlasowsky, H., Birkholz, T., et al., 2011. Optimising Bike Sharing in European Cities - A Handbook. OBIS Project.

Chemla, D., Meunier, F., Wolfler Calvo, R., 2013. Bike sharing systems: solving the static rebalancing problem. Discr. Optim. 10 (2), 120–146.

Coelho, L.C., Cordeau, J.-F., Laporte, G., 2014. Heuristics for dynamic and stochastic inventory-routing. Comput. Oper. Res. 52 (A), 55–67.

Coelho, L.C., Cordeau, J.-F., Laporte, G., 2014. Thirty years of inventory routing. Transp. Sci. 48 (1), 1–19.

Contardo, C., Morency, C., Rousseau, L.-M., 2012. Balancing a Dynamic Public Bike-Sharing System. CIRRELT-2012-09. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf (2014-12-08).

Erdoğan, G., Battarra, M., Calvo, R.W., 2015. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. Eur. J. Oper. Res. 245(3), 667–679.

Erdoğan, G., Laporte, G., Calvo, R.W., 2014. The static bicycle relocation problem with demand intervals. Eur. J. Oper. Res. 238(2), 451–457.

Erera, A.L., Morales, J.C., Savelsbergh, M.W.P., 2009. Robust optimization for empty repositioning problems. Oper. Res. 57 (2), 468–483.

Espegren, H.M., Kristianslund, J., Andersson, H., Fagerholt, K., 2016. The static bicycle repositioning problem - literature survey and new formulation. In: Computational Logistics. In: Lecture Notes in Computer Science. Springer, pp. 9855:337–351.

Fricker, C., Gast, N., 2016. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. EURO J. Transp. Logist. 5 (3), 261–291.

Gauthier, A., Hughes, C., Kost, C., Li, S., Linke, C., Lotshaw, S., Mason, J., Pardo, C., Rasore, C., Schroeder, B., Treviño, X., 2013. The Bike-Share Planning Guide. Technical Report. Institute for Transportation and Development Policy. https://www.itdp.org/wp-content/uploads/2014/07/ITDP_Bike_Share_Planning_Guide.pdf (2014-07-31).

Ghiani, G., Manni, E., Quaranta, A., Triki, C., 2009. Anticipatory algorithms for same–day courier dispatching. Transp. Res. Part E 45 (1), 96–106.

Ghosh, S., Varakantham, P., Adulyasak, Y., Jaillet, P., 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. J. Artif. Intell. Res. 58, 387–430.

Godfrey, G.A., Powell, W.B., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, II: multiperiod travel times. Transp. Sci. 36 (1), 40–54.

Goodson, J.C., Thomas, B.W., Ohlmann, J.W., 2017. A rollout algorithm framework for heuristic solutions to finite-Horizon stochastic dynamic programs. Eur. J. Oper. Res. 258 (1), 216–229.

Kall, P., Wallace, S.W., 1994. Stochastic Programming. John Wiley & Sons.

Kloimüllner, C., Papazek, P., Hu, B., Raidl, G.R., 2014. Balancing bicycle sharing systems: an approach for the dynamic case. In: Evolutionary Computation in Combinatorial Optimization. In: Lecture Notes in Computer Science. Springer, pp. 8600:73–84.

Kloimüllner, C., Papazek, P., Hu, B., Raidl, G.R., 2015. A cluster-first route-second approach for balancing bicycle sharing systems. In: Computer Aided Systems Theory - EUROCAST 2015. In: Lecture Notes in Computer Science. Springer, pp. 9520:439–446.

Lu, C.-C., 2016. Robust multi-period fleet allocation models for bike-Sharing systems. Netw. Spat. Econ. 16 (1), 61–82.

MN, N. R., 2016. Nice Ride Minneapolis (MN/USA). https://www.niceridemn.org/ (2016-10-05).

Neumann Saavedra, B.A., Crainic, T.G., Gendron, B., Mattfeld, D.C., Römer, M., 2016. Service network design of bike sharing systems with resource constraints. In: Computational Logistics. In: Lecture Notes in Computer Science. Springer, pp. 9855:352–366.

Neumann Saavedra, B.A., Vogel, P., Mattfeld, D.C., 2015. Anticipatory service network design of bike sharing systems. Transp. Res. Procedia 10, 355–363.

O'Brien, O., Cheshire, J., Batty, M., 2014. Mining bicycle sharing data for generating insights into sustainable transport systems. J. Transp. Geogr. 34, 262–273.

Papageorgiou, D.J., Cheon, M.-S., Nemhauser, G., Sokol, J., 2014. Approximate dynamic programming for a class of long-Horizon maritime inventory routing problems. Transp. Sci. 49 (4), 870–885.

Powell, W.B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, second ed. John Wiley & Sons.

Puterman, M.L., 2014. Markov Decision Processes: Discrete Stochastic Dynamic Programming, second ed. John Wiley & Sons.

Raviv, T., Tzur, M., Forma, I.A., 2013. Static repositioning in a bike-sharing system: models and solution approaches. EURO J. Transp. Logist. 2 (3), 187–229.

Rothlauf, F., 2011. Design of Modern Heuristics: Priciples and Applications. Springer.

Rudloff, C., Lackner, B., 2014. Modeling demand for bikesharing systems – neighboring stations as source for demand and reason for structural breaks. Transp. Res. Rec. (2430) 1–11.

Schuijbroek, J., Hampshire, R.C., van Hoeve, W.-J., 2017. Inventory rebalancing and vehicle routing in bike sharing systems. Eur. J. Oper. Res. 257 (3), 992–1004.

Shintani, K., Imai, A., Nishimura, E., Papadimitriou, S., 2007. The container shipping network design problem with empty container repositioning. Transp. Res. Part E 43 (1), 39–59.

Song, D.-P., Carter, J., 2009. Empty container repositioning in liner shipping. Maritime Policy Manage. 36 (4), 291–307.

Toriello, A., Nemhauser, G., Savelsbergh, M.W.P., 2010. Decomposing inventory routing problems with approximate value functions. Nav. Res. Logist. 57 (8), 718–727.

Ulmer, M. W., 2017. Horizontal combinations of online and offline approximate dynamic programming for stochastic dynamic vehicle routing. Working Paper. Submitted for publication.

Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Hennig, M., 2017. Offline-Online approximate dynamic programming for dynamic vehicle routing with stochastic requests. Transp. Sci..

Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., Thomas, B. W., 2017b. Dynamic vehicle routing: Literature review and modeling framework. Working Paper. Submitted for publication.

Voccia, S.A., Campbell, A.M., Thomas, B.W., 2017. The same-day delivery problem for online purchases. Transp. Sci..

Vogel, P., Ehmke, J.F., Mattfeld, D.C., 2017. Cost-efficient allocation of bikes to stations in bike sharing systems. In: Computational Logistics. In: Lecture Notes in Computer Science. Springer, pp. 10572:498–512.

Vogel, P., Greiser, T., Mattfeld, D.C., 2011. Understanding bike-sharing systems using data mining: exploring activity patterns. Procedia Soc. Behav. Sci. 20, 514–523.

Vogel, P., Neumann Saavedra, B.A., Mattfeld, D.C., 2014. A hybrid metaheuristic to solve the resource allocation problem in bike sharing systems. In: Hybrid Metaheuristics. In: Lecture Notes in Computer Science. Springer, pp. 8457:16–29.

Yan, S., Lin, J.-R., Chen, Y.-C., Xie, F.-R., 2017. Rental bike location and allocation under stochastic demands. Comput. Ind. Eng. 107, 1–11.