Business Research (2020) 13:69–92
https://doi.org/10.1007/s40685-019-0100-z

ORIGINAL RESEARCH

The multi-vehicle stochastic-dynamic inventory routing problem for bike sharing systems

Jan Brinkmann[1] · **Marlin W. Ulmer**[1] · **Dirk C. Mattfeld**[1]

Received: 19 February 2019 / Accepted: 28 May 2019 / Published online: 21 June 2019

Abstract We address the operational management of station-based bike sharing systems (BSSs). In BSSs, users can spontaneously rent and return bikes at any stations in the system. Demand is driven by commuter, shopping, and leisure activities. This demand constitutes a regular pattern of bike usage over the course of the day but also shows a significant short-term uncertainty. Due to the heterogeneity and the uncertainty in demand, stations may run out of bikes or congest during the day. At empty stations, no rental demand can be served. At full stations, no return demand can be served. To avoid unsatisfied demand, providers dynamically relocate bikes between stations in reaction of current shortages or congestion, but also in anticipation of potential future demand. For this real-time decision problem, we present a method that anticipates potential future demands based on historical observations and that coordinates the fleet of vehicles accordingly. We apply our method for two case studies based on real-world data of the BSSs in Minneapolis and San Francisco. We show that our policy outperforms benchmark policies from the literature. Moreover, we analyze how the interplay between anticipation and coordination is essential for the successful operational management of BSSs. Finally, we reveal that the value of coordination and anticipation based on the demand-structure of the BSS under consideration.

✉ Dirk C. Mattfeld
d.mattfeld@tu-bs.de

Jan Brinkmann
winfo@tu-bs.de

Marlin W. Ulmer
m.ulmer@tu-bs.de

[1] Institut für Wirtschaftsinformatik, Technische Universität Braunschweig, Mühlenpfordtstraße 23, 38106 Braunschweig, Germany

Keywords Bike sharing · Dynamic vehicle routing · Inventory routing · Approximate dynamic programming

1 Introduction

In many cities, station-based bike sharing systems (BSSs) have been proven to be a healthy and flexible alternative to individual travel by car and a suitable complement to public transportation (Büttner et al. 2011). In BSSs, customers can spontaneously perform one-way trips between stations. BSSs are often used for commuting as well as for leisure and shopping activities. This leads to imbalances between stations and in the worst case to failed demand in terms of bikes and bike racks causing customer loss on the long run. To account for the imbalances, BSS providers dispatch vehicles to relocate bikes between stations. Research suggests to balance BSSs by means of optimization models based on typical daily demand patterns. Thereby, short-term demand changes, for example, due to weather conditions are largely ignored (Borgnat et al. 2011; Vogel et al. 2011; O'Brien et al. 2014).

To incorporate spontaneous demand, the problem needs to be considered on the level of operational control. A dispatcher dynamically routes a fleet of vehicles relocating bikes between the stations. Subsequent decisions are taken for the vehicles with respect to the number of bikes to relocate at its current station as well as the next station to serve. To minimize the expected amount of unsatisfied demand per day, stations with an immediate demand are reactively chosen to be visited next. The respective model can be described as multi-vehicle stochastic and dynamic inventory routing problem (SDIRP).

Recent research address the operational repositioning of bikes in a BSS. In the previous research, a single vehicle is dynamically routed in the city to react to realized demand and in anticipation of future demand. In this paper, we route a fleet of vehicles. Isolated control approaches as described by Chiariotti et al. (2018), Brinkmann et al. (2019), and Legros (2019) work for a single vehicle, but may lead to insufficient solutions in a multi-vehicle case. For example, several vehicles may follow the same control policy and approach the same station. Missing coordination between vehicles may also cause unnecessarily long trips within the city. In this paper, we present a method that coordinates the fleet of vehicles in the city. Our method draws on the current state of the vehicles and their individual potential to avoid failed demand. Based on the fleet information, the station an individual vehicle is sent to is determined by means of an assignment problem solution. The parts of the assignment solution which are relevant for the immediate decision are implemented while the remaining parts are reevaluated later in the process based on newly observed demand information. To estimate the unmet demand a vehicle can avoid at a specific station, we extend a lookahead method introduced by Brinkmann et al. (2019).

We apply our methods for two case studies based on historical data from the BSSs in San Francisco (CA, USA) and Minneapolis (MN, USA). We show how coordination between the vehicles is essential to enable demand satisfaction.

This article is structured as follows. Relevant literature is reviewed in Sect. 2. The SDIRP is defined in Sect. 3. In Sect. 4, we define our method and the benchmark policies. A computational study is presented in Sect. 5. In Sect. 6, a summary is drawn and an outlook is given.

2 Literature

The literature on inventory routing problems for BSSs is manifold. In the following, we give a brief overview on relevant work. For further literature classifications, we kindly refer the interested reader to Brinkmann et al. (2016, 2019) and Espegren et al. (2016). Here, we consider deterministic-static, stochastic-static, and stochastic-dynamic models.

Deterministic-static models either do not comprise demand or assume that demand is known in advance. Decisions are taken in one decision point and a solution is a set of static routes. Various authors ignore user demand during operations (e.g., Chemla et al. 2013; Raviv et al. 2013; Erdoğan et al. 2014, 2015; Espegren et al. 2016; Szeto et al. 2016; Szeto and Shui 2018). These models assume that relocations are carried out in the night when the system is closed. The goal is to realize target fill levels provided by external information systems. Time-dependent target fill levels for daily relocations are determined by Vogel et al. (2014) and Neumann Saavedra et al. (2015). In other work, deterministic routing solutions are derived to ensure such given fill levels (Contardo et al. 2012; Kloimüllner et al. 2014; Brinkmann et al. 2016; Schuijbroek et al. 2017). Neumann Saavedra et al. (2016) determine target fill levels, relocations, and tours simultaneously.

Stochastic-static models consider stochastic demand, however, optimization is conducted in a single decision point before the start of the BSS-operations. Raviv and Kolka (2013) and Datner et al. (2017) determine initial target fill levels for stations on the basis of stochastic demand. These fill levels serve as input for overnight relocations as described in the previous category. Lu (2016) considers stochastic demand and determines relocations for vehicle tours predefined by external information systems and one week in advance. Stochastic-static inventory routing problems are investigated by Ghosh et al. (2016, 2017) and Yan et al. (2017). Ghosh et al. (2016, 2017) plan for one day in advance. Yan et al. (2017) plan once a week.

To counteract unexpected demand and sudden imbalances, some papers propose dynamic control during the day on basis of latest information. Brinkmann et al. (2015, 2019), Fricker and Gast (2016), Chiariotti et al. (2018), and Legros (2019) propose stochastic-dynamic models for a single vehicle. To the best of our knowledge, multiple vehicles are not considered in the literature on stochastic-dynamic routing for BSSs. Brinkmann et al. (2015) and Fricker and Gast (2016) draw on myopic approaches. Brinkmann et al. (2019) introduce online simulations to incorporating stochastic information and to enable anticipation. The vehicle is routed to the station where the largest amount of expected failed demand in the (near) future can be avoided. To this end, potential future demand is simulated. Chiariotti et al. (2018) and Legros (2019) divide the time horizon into periods of

equal length. Anticipation is enabled by means of target fill levels for every station and period.

In this paper, we extend the work of Brinkmann et al. (2019) to a multi-vehicle model. We present methods to coordinate the multiple vehicles in the service area. To account for the increased computational burden, we modify the anticipation process of Brinkmann et al. (2019) and shift the anticipation to an analytical offline procedure.

3 Multi-vehicle stochastic-dynamic inventory routing for bike sharing systems

In this section, we define the multi-vehicle stochastic-dynamic inventory routing problem for BSSs (SDIRP). The SDIRP is an extension of the single-vehicle model by Brinkmann et al. (2019). First, we formulate the SDIRP's setting in Sect. 3.1. We then model the SDIRP as Markov decision process (MDP) in Sect. 3.2 and present an example in Sect. 3.3. In Sect. 3.4, we briefly analyze the SDIRP's state space dimensionality.

3.1 Problem setting

We consider a BSS with stations $n_i \in N, i > 0$ and a depot n_0. Every station n has an initial fill level f_0^n, i.e., a specific number of bikes, and a maximum capacity c^n. Over a time horizon T, users rent and return bikes altering the fill level of the stations spontaneously. At a non-empty station, rental demand can be served. At a non-full station, return demand can be served. If a demand is served, the station's fill level is decreased, or increased, respectively. If demand fails, a penalty occurs and the user approaches an adjacent station.

Failed demand can be avoided by providing sufficient bikes and free bike racks at every station and every time. To this end, the dispatcher dynamically routes a fleet of transport vehicles $V = \{v_1, \ldots, v_{\max}\}$. We assume the number of vehicles is determined by the tactical management and the drivers are already paid. Every vehicle starts at the depot, is initially empty, and has a capacity of c^v. The travel time between two stations is given by function $\tau(\cdot, \cdot)$. The service time for relocation is τ_r per bike.

Over the course of the day, the vehicles are subsequently sent from station to station to pick up or load bikes. Decision making therefore comprises a combination of routing to a station and loading operation at a station. Since new demand information reveals during travel, the loading operation is to be reconsidered, once a vehicle arrives at a station. The service provider aims on maximizing the users' satisfaction based on the limited resources accessible. Therefore, the goal is to minimize the expected failed demand per day.

3.2 Markov decision process

In the following, we model the problem as an MDP. An MDP is a framework for stochastic-dynamic optimization problems (Puterman 2014). We extend the single-vehicle Markov decision process of Brinkmann et al. (2019). Equation (1) depicts the components of an MDP:

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

In an MDP, a sequence of decision states s_k occurs over time. In a decision state, a decision x is selected leading to a post-decision state s_k^x. Based on post-decision state s_k^x, a transition ω leads to a new decision state s_{k+1}. In the following, we define the MDP components for the SDIRP. The notation is given in Table 1.

3.2.1 Decision states

A decision point $k \in K$ occurs when a vehicle arrives at a station. The associated decision state s_k contains information about the time, the stations, and the fleet. The current point in time is t_k. The fill levels at the stations are $f_k^{n_1}, \ldots, f_k^{nmax}$. Every vehicle v is represented by a vector $(n_k^v, \alpha_k^v, f_k^v)$. Value n_k^v represents the station vehicle v is currently traveling to and value α_k^v depicts the corresponding arrival time. In case that the vehicle just arrives at a station, the arrival time is $\alpha_k^v = t_k$. Value f_k^v represents the current number of bikes loaded on the vehicle.

Table 1 Notation of the Markov decision process

Symbol	Description
$K = (0, \ldots, k_{\max})$	Sequence of decision points
$S = \{s_0, \ldots, s_{\max}\}$	Set of decision states
$X_s = \{x_1, \ldots, x_{\max} \mid x = (\iota, \rho)\}\ ,\ \forall s \in S$	Sets of feasible decisions
$s_k^x = (s_k, x)\ ,\ \forall s \in S, x \in X_s$	Post-decision states
$\omega : S \times X \to S$	Transition function
$T = \{t_0, \ldots, t\max \mid t \in \mathbb{N}\}$	Time horizon
$V = \{v_1, \ldots, v\max \mid v = (n_k^v, \alpha_k^v, f_k^v)\}$	Set of vehicles
$t_k \in T$	Point in time in state s_k
$f_k = \left(f_k^{n_0}, \ldots, f_k^{n_{\max}}\right)$	Stations' fill levels in state s_k
$n_k^v \in N$	Vehicle v's station in state s_k
$\alpha_k^v \in T$	Vehicle v's arrival time at n_k^v
$f_k^v \in \mathbb{N}_0$	Vehicle v's load in state s_k
$\iota^x \in \mathbb{Z}$	Inventory decision
$n^x \in N$	Routing decision
$p : S \times X \to \mathbb{N}_0$	Penalty function
$\Pi = \{\pi_0, \ldots, \pi_{\max} \mid \pi : S \to X\}$	Set of policies

3.2.2 Decisions

A decision is made only about the vehicles currently staying at a station. For vehicle v, decision $x = (\iota, \rho)$ contains an inventory decision $\iota \in \mathbb{Z}$ at station n_k^v the vehicle just arrives, and a next station $\rho \in N$ to serve. If $\iota < 0$, the vehicle picks up $|\iota|$ bikes at the current station. If $\iota > 0$, the vehicle delivers $|\iota|$ bikes.

A decision changes the status of station n_k^v and vehicle v as follows. The fill level of the station n_k^v is stepwise altered to $f_k^n + \iota$ based on the inventory decision. The fill level of the vehicle is altered to $f_k^v - \iota$. The station of the vehicle is altered to ρ and the new arrival time is $t_k + |\iota| \cdot \tau_r + \tau(n_k^v, \rho)$ reflecting time for relocation and traveling.

3.2.3 Transition

The transition ω leads to a new decision point $k+1$ and reflects the vehicles' traveling and a realization of customer demand at every station. The new point in time t_{k+1} is the minimum of all arrival times α. The transition updates the fill levels of the stations due to successfully fulfilled demand. The penalty function $p(s_k, x)$ reflects the expected failed demand between two decision points k and $k+1$ if x is the decision made in s_k. Value $p_\omega(s_k, x)$ represents the penalty based on the realized transition ω.

3.2.4 Objective

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

$$\pi^* = \underset{\pi \in \Pi}{\arg\min}\ \mathbb{E}\left[\sum_{k=0}^{k_{max}} p\big(s_k, \pi(s_k)\big)\Big| s_0\right]. \tag{2}$$

3.3 Example

In Fig. 1, we give an example of the MDP. On the left-hand side, decision state s_k is depicted. The center shows post-decision state s_k^x. The following decision state s_{k+1}

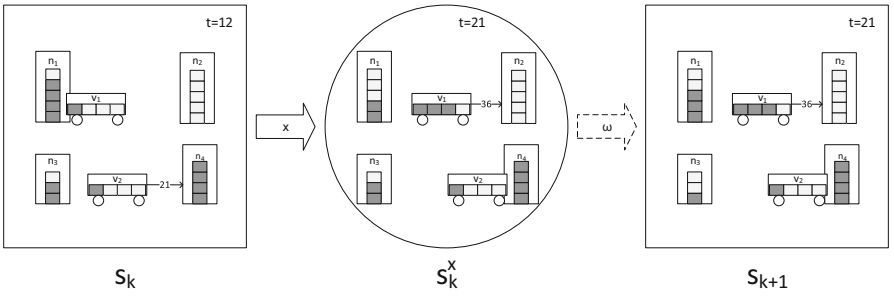

Fig. 1 Exemplary decision state, post-decision state, and resulting decision state

is depicted on the right-hand side. The system has four stations n_1, n_2, n_3, and n_4. The depot is neglected. Two vehicles v_1, v_2 are given. For the stations and vehicles, 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 five racks, station n_3 of three, and station n_4 of four racks. The vehicle capacities are four.

Decision point k is induced due to vehicle v_1's arrival at station n_1 in time $t_k = 12$. In the associated decision state s_k, stations n_1 and n_4 contain $f_k^{n_1} = f_k^{n_4} = 4$ bikes, n_3 contains $f_k^{n_3} = 2$ bikes, and n_2 is empty, $f_k^{n_2} = 0$. Vehicle v_1 is located at $n_k^{v_1} = n_1$. It just arrived at this station. Thus, the arrival time is $\alpha_k^{v_1} = t_k = 12$. The current fill level of vehicle v_1 is $f_k^{v_1} = 1$. Vehicle v_2 travels to station n_4 and arrives in nine time units. It has loaded one bike. Thus, the vector for vehicle v_2 reads $(n_4, 21, 1)$.

The inventory decision is made about how many bikes to pick up at or to deliver to station n_1 by vehicle v_1. The routing decision is made about which station to serve next, or idling at the current station. Here, the applied decision is $x = (-2, n_2)$: picking up two bikes and travel to n_2. Assuming $\tau_r = 2$ units, two pick ups consume four time units. Furthermore, assuming travel time of $\tau(n_1, n_2) = 20$, the decision consumes 24 time units in total. Finally, vehicle v_1 arrives at station n_2 in $t_k + |\text{-}2| \cdot \tau_r + \tau(n_1, n_2) = 12 + 4 + 20 = 36$. The updated vector is $(n_2, 36, 3)$.

The resulting post-decision state s_k^x, including the decisions made, is depicted in the center of Fig. 1. The next decision point $k + 1$ occurs at $t_{k+1} = 21$ because vehicle v_2 arrives at station n_4: $\alpha_{k+1}^{v_2} = t_{k+1} = 21$.

Before the associated decision state s_{k+1} occurs, the transition ω reveals new fill levels and a realization of the penalty function. New fill levels are revealed at n_1 and n_3 due to successful requests. At n_1, the fill level has increased by one. At n_3, the fill level has decreased by one. Station n_3 remains empty and station n_4 remains full. For the purpose of presentation, the customer trips are not depicted in Fig. 1. We assume that one rental demand failed at n_2 and one return demand failed at n_4 resulting in a realized penalty of $p(s_k, x, \omega) = 2$. The resulting new decision state s_{k+1} is shown on the right-hand side of Fig. 1.

3.4 State space dimensionality

An optimal policy as defined in the objective function (2) relies on the Bellman equation (3). It minimizes the penalties over the current and all future decision points:

$$\pi^*(s_k) = \underset{x \in X_{s_k}}{\arg\min}\ \mathbb{E}\left[\sum_{k'=k}^{k_{max}} p(s_{k'}, x) \middle| s_k\right]. \tag{3}$$

Applying the Bellman equation in decision point k results in evaluating future decision points $k + 1, \ldots, k_{\max}$ and the associated decisions $x \in X$. This results in evaluating almost every potential decision state $s \in S$ and is usually computationally intractable even for smaller problem instances. In the following, we demonstrate the state space dimensionality of the SDIRP by investigating an upper bound of the state space's size $|S|$:

(a) A state may occur at any point in time $t = t_0, \ldots, t_{\max}$.
(b) At any point in time, every station fill level is between 0 and maximum capacity c_n.
(c) Further, every vehicle load is between 0 and maximum capacity c_v.
(d) Every vehicle may stay or travel to any station.
(e) Arrival times are arbitrary as well.

The resulting upper bound reads as follow:

$$|S| \leq \underbrace{(t_{\max}+1)}_{\text{(a)}} \cdot \underbrace{\prod_{n\in N}(c_n+1)}_{\text{(b)}} \cdot \underbrace{\prod_{v\in V}(c_v+1)}_{\text{(c)}} \cdot \underbrace{|N|^{|V|}}_{\text{(d)}} \cdot \underbrace{(t_{\max}+1)^{|V|}}_{\text{(e)}} \tag{4}$$

The upper bound ignores the interdependencies between the fill levels of stations (and vehicles) due to the given number of bikes in the system. However, it indicates that calculating optimal policies by means of dynamic programming is computationally intractable even for smaller systems. To this end, we draw on a heuristic in the next section.

4 Coordinated lookahead policy

In this section, we present our solution policy. The method coordinates vehicles within the city and anticipates unmet demand by means of lookahead. We denote our method the coordinated lookahead policy (CLA). The method is an extension of prior work (Brinkmann et al. 2019) in terms of usability in the multi-vehicle case. We first give a motivation for our policy and then describe the methodological details.

4.1 Motivation and policy outline

The SDIRP is dynamic because of the spontaneity in user demand. A suitable method is required to coordinate the fleet of vehicles effectively, reacting to current demand and anticipating potential future demand.

Coordination manifests in the routing decision of the MDP. In the routing decision, the potential movement of the other vehicles needs to be considered. To highlight the importance of coordination, we show a (simplified) example in Fig. 2. The figure consists of three parts depicting different routing decisions for the same situation. Two empty vehicles and two full stations are given. We assume the travel time of vehicle v_1 to station n_1 is 12 min, to station n_2, the travel time is 10 min. The travel times for vehicle v_2 are 25 min to station n_1 and 15 min to station n_2.

In the situation, we need to route v_1. We assume that v_2 will require a routing decision a very short time later. A potentially good decision is depicted in Fig. 2a, sending v_1 to n_1 and v_2 to n_2. The decision leads to relatively short travel times and the vehicles arrive early to balance the stations and to avoid unmet demand.

Without coordination, we observe different decisions. The first (most obvious) decision is depicted in Fig. 2b. If we only focus on the current vehicle and ignoring

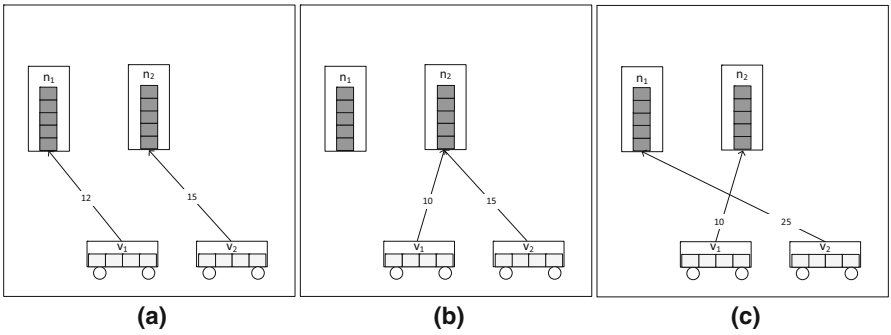

Fig. 2 Example of the impact of coordination

the other vehicles in the decision process, both vehicles may eventually approach the same station. We first send v_1 to n_2 (ignoring v_2) and a short time later, we send v_2 to the same station (ignoring v_1).

However, coordination should not be restricted to avoiding vehicles approaching the same station, as shown in Fig. 2c. In this decision, v_1 is sent to its individual "best" station and a short time later, v_2 is sent a long way to the remaining "best" station. As the example shows, this procedure may be inferior because other vehicles may travel unnecessarily long distances within the city.

Coordinating the fleet allows to effectively assign vehicles to stations avoiding unnecessary long travel. To achieve coordinated routing decisions, our method determines the potential for every vehicle to avoid failed demand for every station. Based on the estimation, it then determines the assignment for vehicles to stations and selects the routing decision accordingly. The method only implements the assignment only for the vehicle under consideration. The assignments for the other vehicles can be reevaluated later in the process based on new observations.

To estimate the potential for a vehicle to avoid unsatisfied demand at the current station and all other stations, anticipation of future demand is necessary. Anticipation indicates stations which are likely to run out of bikes or congests.

In the following, we present the details of the algorithm. We first describe how we estimate the future failed demand for the stations. We then present the procedures to determine inventory and routing decisions.

4.2 Anticipation of failed demand

Our policy requires a measure to calculate the expected failed demand for stations. To this end, we use historical trip data to successively update the fill levels of stations. The updates may lead to accumulations of failed demand for each station. The failed demand measured can then be used in the decision making of our policy. As previous research has shown, there is benefit to limit the horizon (Brinkmann et al. 2019) because it balances the urgency of near-future demand with longer term demand. Thus, we limit the horizon in state s_k to δ minutes.

We calculate the average difference of rented and returned bikes per minute $\Delta_t^n \in \mathbb{R}$ for each station n and minute t. These values can be precomputed based on historical data and allow instant decision making in a real-time decision state. A negative value of Δ_t^n indicates that in minute t at station n more bikes are rented than returned. A positive value indicates that more bikes are returned than rented. For a station n at time t_k with fill level $f_{t_k}^n$, we can then calculate the fill level development over time as follows:

$$f_t^n = \min(\max(f_{t-1}^n + \Delta_{t-1}^n, 0), c^n) \quad \forall t = t_k + 1, \ldots, t_k + \delta. \tag{5}$$

The fill levels are altered minute by minute with respect to the average difference in demand per minute. The fill levels always stay in the range between zero and the maximum capacity. Based on this development, we calculate the cumulated amounts of failed return demand $\gamma^+(n)$ and of failed rental demand $\gamma^-(n)$ for station n. Value $\gamma^+(n)$ is calculated as follows:

$$\gamma^+(n) = \sum_{t=t_k}^{t_k+\delta} \max(f_t^n + \Delta_t^n - c^n, 0). \tag{6}$$

In minute t, a failed return demand occurs if the sum of current fill level and return demand exceeds the station's capacity. Analogously, the failed rental demand can be calculated:

$$\gamma^-(n) = \sum_{t=t_k}^{t_k+\delta} |\min(f_t^n + \Delta_t^n, 0)|. \tag{7}$$

The values can now be used in our inventory decision ι and routing decision ρ of our MDP.

4.3 Inventory decision

In the first step, the CLA determines the inventory decision ι. We reduce the set of inventory decisions to evaluate because the set of potential fill levels is very large and the runtime is limited. Here, we draw on the percentages $\mu_1 = 25\%$ (low), $\mu_2 = 50\%$ (medium), and $\mu_3 = 75\%$ (high) leading to the target fill levels $\mu_1 \cdot c^{n_k^v}$, $\mu_2 \cdot c^{n_k^v}$, and $\mu_3 \cdot c^{n_k^v}$. With respect to the vehicle's capacity c^v and load f_k^v, we may not be able to realize the target fill levels as described. If necessary, we modify the target fill levels and derive the corresponding relocations ι_1, ι_2, and ι_3 according to Eq. (8):

$$\iota_i = \begin{cases} \min\{\mu_i \cdot c^{n_k^v} - f_k^{n_k^v}, f_k^v\} & , \text{if } \mu_i \cdot c^{n_k^v} > f_k^{n_k^v} \\ \max\{\mu_i \cdot c^{n_k^v} - f_k^{n_k^v}, f_k^v - c^v\} & , \text{if } \mu_i \cdot c^{n_k^v} < f_k^{n_k^v} \\ 0 & , \text{else.} \end{cases} \tag{8}$$

The first case occurs when bikes need to be delivered to the station. Then, we limit the number of bikes to the vehicle's load. In the second case, bikes need to be picked up. Here, the equation ensures that the number of loaded bikes does not

exceed the vehicle's capacity. The third case occurs if the station's fill level is equal to the predetermined target fill level and therefore no relocations need to be done.

As an example, we consider a station n_k^v with capacity $c^{n_k^v} = 20$ bikes. The current fill level is $f_k^{n_k^v} = 10$ bikes. The vehicle v has currently loaded $f_k^v = 2$ and a capacity of $c^v = 20$ bikes. The three relocation decisions aim on fill levels of $25\% \cdot 20 = 5$, $50\% \cdot 20 = 10$, and $75\% \cdot 20 = 15$ bikes. The first fill level can be achieved by removing 5 bikes from the station, $\iota_1 = -5$. The second fill level can be achieved by maintaining the fill level, $\iota_2 = 0$. The third fill level cannot be achieved, because only $f_k^{n_k^v} + f_k^v = 12$ bikes are available. Thus, with respect to Eq. (8), the third relocation decision is $\iota_3 = 2$.

For each inventory decision ι_i, we now can determine the expected failed demand at that station as described above. Because the failed demand depends on decision ι_i, we denote the values $\gamma^-(\iota_i, n_k^v)$ and $\gamma^+(\iota_i, n_k^v)$. Based on the cumulated values of $\gamma^-(\iota_i, n_k^v)$ and $\gamma^+(\iota_i, n_k^v)$ for $\iota_i \in \{\iota_1, \iota_2, \iota_3\}$, we select $\iota^\star$ minimizing the sum of failed demand:

$$\iota^\star = \underset{\iota \in \{\iota_1, \iota_2, \iota_3\}}{\arg\min} \{\gamma^-(\iota, n_k^v) + \gamma^+(\iota, n_k^v)\}. \tag{9}$$

4.4 Routing decision

In the second step, the CLA determines the routing decision ρ for the current vehicle in coordination with the other vehicles. In the following, we anticipate the prevented failed demand for pairs of stations and vehicles. We then describe how we coordinate the routing based on these values.

To derive the failed demand a vehicle can prevent at a station, we follow the same procedure as discussed in Sect. 4.2. The only difference is that the vehicles have different arrival times at stations and different fill levels. Mathematically, we determine for every station n the failed demand a specific vehicle v can *prevent* as follows. In time t_k, a vehicle v, which arrives in time α_k^v minutes at station n_k^v, can reach station n at earliest at time $\alpha_k^v + \tau(n_k^v, n)$. Thus, we consider for v only the failed demand at n in the time interval $\mathcal{I}_v^n = \left[\alpha_k^v + \tau(n_k^v, n), t_k + \delta\right]$ where δ depicts the length of the considered horizon. Let $\gamma^-(n, v)$ and $\gamma^+(n, v)$ be the failed rental and failed return demand at stations n occurring in this interval $\mathcal{I}_v^n$. Again, we need to consider the vehicle's capacity c^v and load f_k^v. Then, the prevented failed demand if vehicle v is sent to station n can be calculated as follows:

$$\gamma(n, v) = \max\left\{ \underbrace{\min\left\{\gamma^-(n, v), f_k^v\right\}}_{\text{preventedfailedrentals}}, \underbrace{\min\left\{\gamma^+(n, v), c^v - f_k^v\right\}}_{\text{preventedfailedreturns}} \right\}. \tag{10}$$

Figure 3 provides an example. The lookahead starts at 14:30 h and is limited to $\delta = 360$ min. We successively update the station's fill level as described and observe the fill level and cumulated failed demand at station n. The fill level is represented by the solid line. We see that the fill level decreases due to rental demand. Just before 16:00 h, the station runs out of bikes. From that time on, the

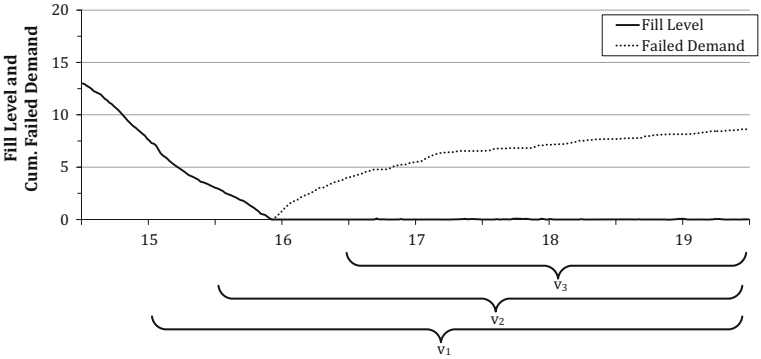

Fig. 3 Observed fill level and cumulated failed demand in an exemplary lookahead

cumulated failed demand, indicated by the dashed line, increases because rental demand cannot be satisfied any more. If vehicles v_1 or v_2 are sent to n, they arrive earliest at 15.00 h, or 15.30 h, respectively. Both vehicles arrive early enough before any demand fails. Vehicle v_3 arrives earliest at 16.30 h. This results in failed demand of about 3.5 rentals between 16:00 and 16:30 h. Given the three vehicles have the same capacity and load, the values for v_1 and v_2 are the same and both are higher than the value for v_3: $\gamma(n, v_1) = \gamma(n, v_2) > \gamma(n, v_3)$.

Only considering the demand at station n in the example in Fig. 3, it is beneficial to send either v_1 or v_2 to this station. However, looking at the demand at all stations, it might be beneficial to send v_3 to n if v_1 and v_2 can prevent more demand at other stations. To consider these interdependencies, we assign every vehicle to a station. Although we only decide about the routing for the current vehicle, we make a preliminary, tentative decision for every other vehicle as well.

To coordinate the dispatching of the vehicles, we have to solve an assignment problem. The goal is to maximize the prevented failed demand over all stations. The constraint is that every vehicle is assigned to one station and every station is assigned at most to one vehicle. In the following, we formally define the assignment problem.

We define decision variables $y_{ij} \in \{0, 1\}\ \forall n_i \in N, v_j \in V$ to indicate the assignments of stations to vehicles in Eq. (11):

$$y_{ij} = \begin{cases} 1, \text{ if station} n_i \text{is assigned to vehicle} v_j \\ 0, \text{ else} \end{cases} \tag{11}$$

Value $\gamma(n_i, v_j)$ indicates the amount of prevented failed demand at station n_i if it is assigned to vehicle v_j. Then, the problem can be defined as follows:

$$\max \sum_{i=1}^{|N|} \sum_{j=1}^{|V|} y_{ij} \cdot \gamma(n_i, v_j) \tag{12}$$

subject to

$$\sum_{j=1}^{|V|} y_{ij} \leq 1 \quad \forall n_i \in N \tag{13}$$

$$\sum_{i=1}^{|N|} y_{ij} = 1 \quad \forall v_j \in V \tag{14}$$

In the objective function (12), the amount of prevented failed demands over all stations and vehicles is maximized. In constraints (13) and (14), we ensure that every station is assigned to at most one vehicle, or every vehicle is assigned to one station, respectively.

We apply the heuristic matrix maximum approach to solve the assignment problem: The values of $\gamma(n, v)$ for all vehicles and stations lead to a matrix Γ. Each row indicates a station and each column a vehicle. Let v be the vehicle we need to route in the current decision state. We iteratively search the entry with the maximum in Γ. Let $\gamma(n_i, v_j)$ be this entry. If $v_j = v$, we stop the procedure and send vehicle v to station n_i: $\rho^\star = n_i$. Else, if $v_j \neq v$, we remove row i and column j of matrix Γ. We remove row i to ensure that we do not route our vehicle to n_i because vehicle v_j is the best assignment for this station. We remove column j to ensure that v_j is not sent to another station for the same reason. We repeat the procedure until the maximum $\gamma(n_i, v_j)$ refers to vehicle v, i.e., $v = v_j$. Then, we choose n_i to be vehicle v's next station: $\rho^\star = n_i$.

Notably, even though the approach is greedy, it performs comparably well. Experiments show that solving the deterministic-static assignment problem to optimality even decreases the solution quality of the SDIRP (compare Appendix A.2). As for many dynamic problems, these results confirm the observation that solving static subproblems to optimality in a dynamic context is often not beneficial (Powell et al. 2000; Maggioni and Wallace 2012). One reason is that optimal solutions for static problems usually do not yield any flexibility to react to future information changes, as for the SDIRP, changes in Γ.

Eventually, the CLA selects the inventory decision $\iota^\star$ of Sect. 4.3 and the routing decision $\rho^\star$ of Sect. 4.4.

4.5 Benchmark policies and tuning

In the following, we define our benchmark policies. Our CLA draws on two components: anticipation of future developments and coordination of the fleet. We design our benchmark policies with respect to these components.

The first benchmark policy draws on the lookahead as described in Sect. 4 but ignores coordination of the vehicles. We denote this policy CLA-NC for no coordination. This policy ignores information on other vehicles and sends vehicle v to the station where this vehicle can prevent as much demand as possible:

$$\rho^\star = \underset{n \in N}{\arg\max}\ \gamma(n, v). \tag{15}$$

The second benchmark ignores anticipation but draws on safety buffers to determine if a station needs to be visited (Brinkmann et al. 2015). We denote this policy as the short-term relocation policy (STR). A threshold percentage β of the stations' capacity is defined a priori. If a station n has less than $\beta \cdot c^n$ bikes or free bike racks, the safety buffer is violated and the station is denoted to be imbalanced. The inventory decision ι is determined as the difference between the current fill level and the safety buffer. For balanced stations, the inventory decision is $\iota = 0$. Routing decision ρ selects the nearest imbalanced station. Stations where another vehicle is currently located or approaching are excluded from consideration. The third benchmark policy STR-NC follows the same procedure but ignores coordination. Thus, several vehicles may be sent to the same imbalanced station. Policy STR-NC can be seen as the basis benchmark policy because it neither considers anticipation nor coordination.

We tune the policies as follows. For the CLA and the CLA-NC, we test horizons of $\delta \in \{60, 120, \ldots, 720\}$. For STR and STR-NC, we select $\beta \in \{0.1, 0.2, \ldots, 0.5\}$. For every instance setting, we run all parametrizations over 1, 000 test days and select the best δ and β for our experiments. The parameters can be found in Tables 2 and 3 in Appendix A.2.

5 Computational studies

In this section, we present our computational studies. We first describe the instance settings based on real-world data from the BSSs of Minneapolis and San Francisco. We then compare the results of the different policies. Finally, we analyze the impact of our coordination method in detail. The experiments are performed on an Intel Core i5-3470 with 3.2 GHz and 32GB RAM. The implementation bases on Java 8u121.

5.1 Instances

In this section, we describe the instances for the SDIRP based on real-world data. We draw on the data of the BSSs in Minneapolis (MN, USA, Nice Ride 2016) and San Francisco (CA, USA, Ford GoBike 2017). The providers frequently publish data sets providing information on trips. Every trip comprises a rental and return station and the corresponding rental and return times. The data sets are preprocessed as proposed by Vogel et al. (2011):

We only draw on week days because the customer demand significantly differs on weekends. We further remove all trips that start and end at the same station and take less than 1 min. In these cases, we assume a user has returned the bike immediately due to some defect and, thus, no real trip takes place. We draw on 88 working days in the summer months from July to October in 2015 for Minneapolis. The preprocessed data set consists of 197,726 trips occurred at 169 stations. The station capacities differ between 15 and 35 bike racks. 2246 trips take place per day on average. Ford GoBike BSS consists of three disjoint subsystems East Bay, San Francisco, and San José. We only draw on the San Francisco subsystem. For

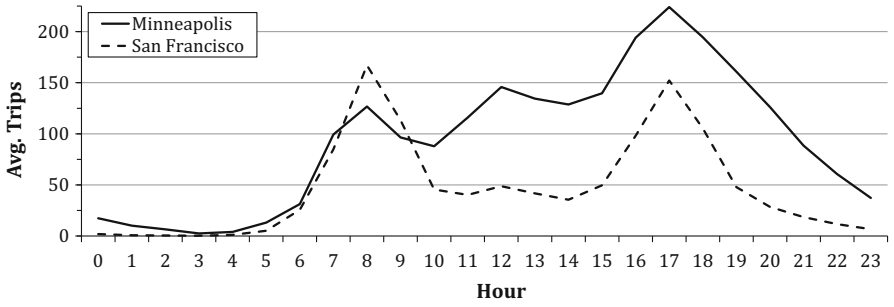

Fig. 4 Temporal distributions of trips of Minneapolis' and San Francisco's BSS

San Francisco, we select 89 working days in the summer months July to October in 2014. The data set consists of 100,728 trips after cleaning. The 35 stations have capacities between 15 and 27. 1131 trips take place per day.

We use data sets as foundation for the average differences of rented and returned bikes per station and minute. In our simulation, we discretize one day into $24 \cdot 60 = 1440$ min. Therefore, time horizon T comprises 1440 individual points in time t. Then, we determine $\Delta_t^n, \forall n \in N, t \in T$ as the average differences of rented and returned bikes per minute and station.

Figure 4 depicts the temporal distributions of trips in the course of the day. The solid line refers to the average number of trips in Minneapolis, the dashed line refers to San Francisco. Both BSSs have peaks in the morning, around noon, and in the evening. In Minneapolis, the second and third peak are higher than the previous ones. In San Francisco, the amplitudes of the morning and evening peak are more equal. The noon peak is much lower. In both systems, the morning and evening peaks are likely to represent commuters. In the morning, commuters rent bikes in residential areas and return them in working areas. In the evening, bikes mainly are rented in working areas and returned in residential areas. The noon peak is likely due to leisure and shopping activities. As stated by O'Brien et al. (2014), leisure usage is significant in Minneapolis.

The preprocessed data sets serve as foundation for the actual instances. For each test day, we randomly draw trips with replacement from the associated data sets. We draw 2246 trips for Minneapolis, and 1126 for San Francisco. The combination of real trips from different days lead to new, artificial days. Further, we assume the ratio of bike racks over all stations and bikes within the system to be 2:1 (which is a common ratio in practice, O'Brien et al. 2014). In the beginning of the time horizon, we distribute the bikes randomly over all stations. Here, we use a uniform distribution. For each instance setting, we generate 1000 test days following this procedure. We test every instance setting for every BSS and for fleet sizes between 1 and 4 vehicles. Every vehicle has a capacity of 20 bikes. The vehicles travel on Euclidean distances between the stations with a speed of 15 km/h. This relatively slow speed accounts for potentially longer travel distances due to the road network. We assume a relocation time of 2 min per bike, i.e., $\tau_r = 2$.

In Appendix A.1, we go into detail with the user behavior we have implemented in case of failing demand.

5.2 Analysis

Every combination of instance, fleet size, and policy demands a special different tuning. In Appendix A.2, we provide information on the tuning parameters and the achieved failed demand of the solutions.

Here, we compare the solution qualities of the four policies. To this end, we calculate the improvement ratio of a policy π to the basic benchmark STR-NC for every instance setting. Let $\mathcal{Q}(\pi)$ denote the average failed demand of policy π. The improvement ratio is then calculated as:

$$1 - \frac{\mathcal{Q}(\pi)}{\mathcal{Q}(\text{STR-NC})}. \tag{16}$$

The improvements of policies STR, CLA-NC, and CLA over STR-NC are shown in Fig. 5. The *x*-axis shows the policies. The *y*-axis shows the improvement ratio. The dark bars refer to Minneapolis and the light bars refer to San Francisco. We observe that all three policies significantly improve the solution quality compared to the basic benchmark policy STR-NC. The proposed policy CLA achieves the highest solution quality. Also STR achieves significant improvements against STR-NC. Thus, even without anticipation, coordination is valuable. We further observe very significant improvements for both CLA-NC and CLA up to 55.0%, or 65.3%, respectively. We also see that anticipation is more important for San Francisco while coordination is more important for Minneapolis: the difference between STR and CLA is significantly higher for San Francisco while the difference between CLA-NC and CLA is significantly higher for Minneapolis. We will analyze these observations in more detail in the following.

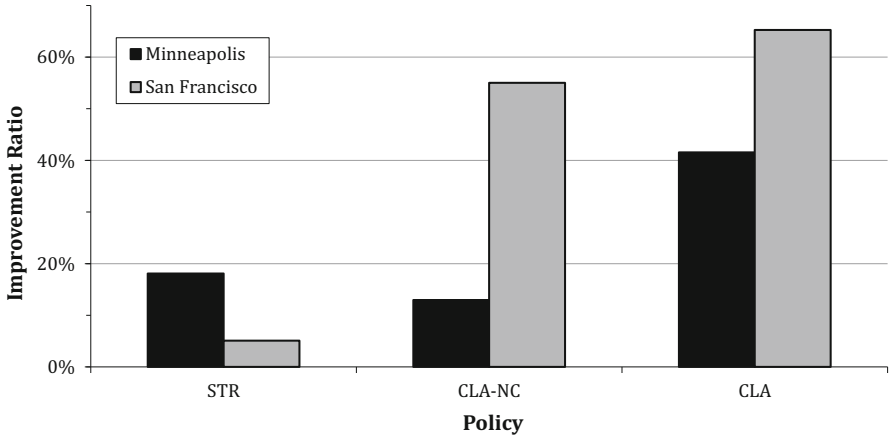

Fig. 5 Average improvement ratio of the policies compared to STR-NC

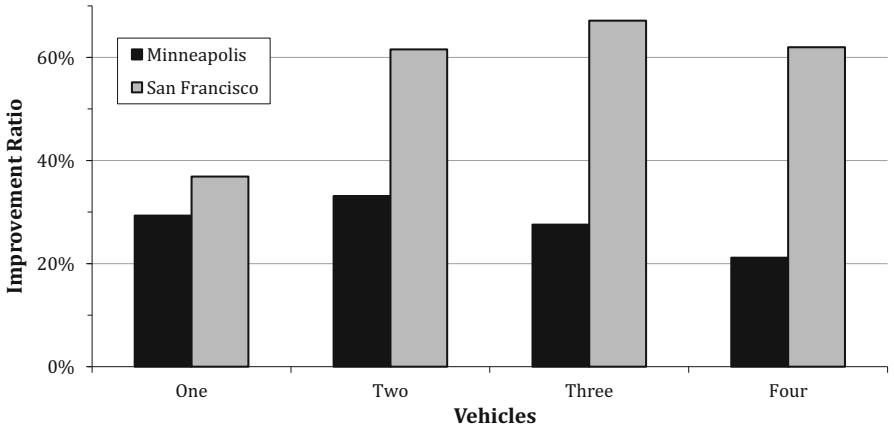

Fig. 6 Impact of anticipation: improvement ratio of CLA compared to the STR

First, we analyze how anticipation impacts the solution quality. To this end, we calculate the improvement ratio of CLA compared to STR for both BSSs and for fleet sizes of 1–4. Both policies coordinate their fleet. The improvement ratio is shown in Fig. 6. We observe that for every instance setting, an improvement of at least 20% can be achieved. Further, the improvement is significantly higher in San Francisco. As mentioned before, San Francisco shows a stronger commuter behavior than Minneapolis. Thus, trips are more predictable for San Francisco and anticipation becomes more reliable. This explains the high improvement ratios for San Francisco.

We now analyze the impact of fleet coordination. To this end, we calculate the improvement ratio of CLA compared to CLA-NC. Following the structure of Fig. 6, the results are shown in Fig. 7. We omit the results for tests with a single

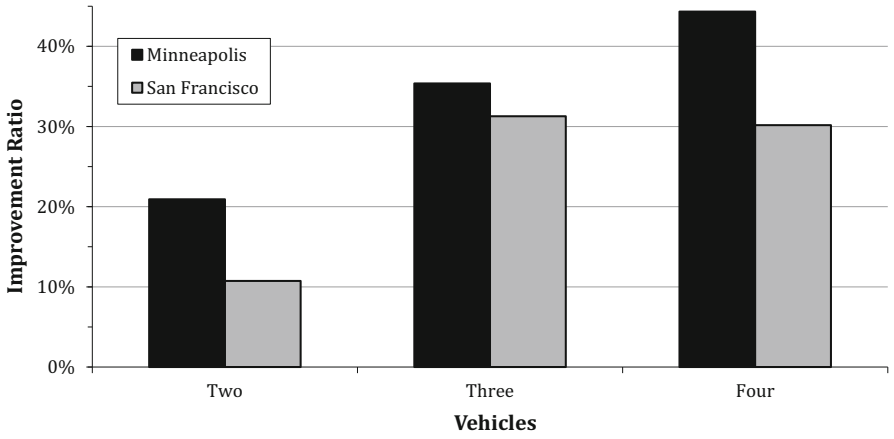

Fig. 7 Impact of coordination: improvement ratio of CLA compared to CLA-NC

vehicle because in that case coordination does not effect decision making. In Minneapolis, the coordinated dispatching achieves improvement ratios of 20.9%, 35.4%, and 44.3% for 2, 3, and 4 vehicles. In San Francisco, the improvement ratios are 10.7%, 31.3%, and 30.2%. As assumed, we observe that coordination becomes more important with increasing number of vehicles. We also observe that for Minneapolis coordination is more important than for San Francisco. The BSS of San Francisco is dominated by commuters: many customers demand rentals and returns at a distinct set of stations at specific times. Thus, the impact of sending vehicles in the same direction without coordination is not as severe compared to Minneapolis where demands are more equally distributed and coordination is crucial.

5.3 Degree of coordination

In our main experiments, we have shown that coordination is essential for successful operational control of the fleet. In the following, we analyze the impact of coordination in more detail. The CLA solves an assignment problem but implements the assignment solution for the current vehicle only. The assignment problem incorporates the potential of all vehicles in the routing decision for the current vehicle. The only partial implementation of the assignment solutions maintains flexibility to react to realized demand later in the process. To show the importance of the assignment problem and the partial implementation, we run experiments with two different coordination degrees:

- The first degree is similar to CLA-NC choosing the station with the highest priority for the vehicle. However, we omit stations where other vehicles are currently traveling to. This method ignores the different potentials of all vehicles.
- The second degree is to solve the assignment problem and to completely commit all vehicles to the solution. That means that the routing decision for other vehicles may already be taken before the vehicles arrive at the next station. This method ignores the value of reconsidering assignment decisions based on new information.

We test the different degrees for both systems and for 2, 3, and 4 vehicles. We benchmark the results with the CLA-NC. The improvements for the different degrees are depicted in Fig. 8. The *x*-axis depicts the different degrees of coordination. Our CLA as described in Sect. 4.4 is indicated by 'Partial Assignment'. The *y*-axis shows the improvement over the no coordination policy CLA-NC for the systems of San Francisco and Minneapolis.

We observe that all three coordination methods add significant benefit in comparison to solutions without coordination. CLA with partial assignments outperforms the other coordination degrees. Particularly, committing to the complete assignment decision for all vehicles lead to inferior results. This confirms that the high volatility in customer demand impedes long-term planning for both systems.

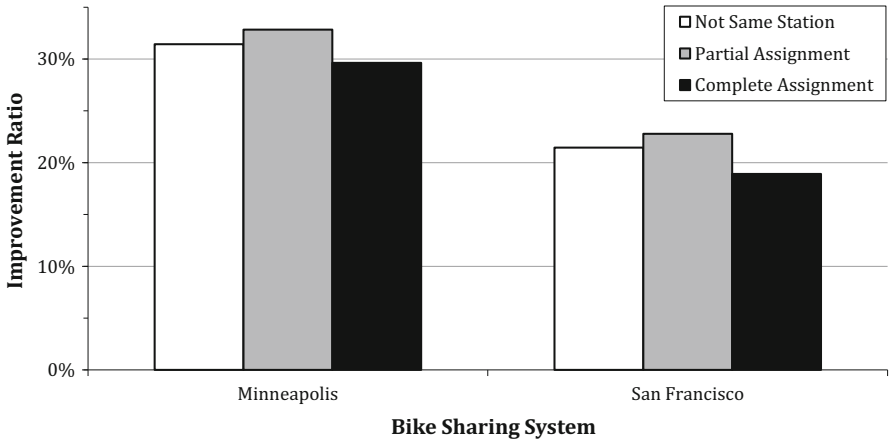

Fig. 8 Improvement of the different degrees of coordination over CLA-NC without coordination

6 Conclusion

In this paper, we have addressed the management of bike sharing systems on the operational level. The service provider uses a fleet of vehicles to relocate bikes between stations with the objective of minimizing failed demand. We have modeled the stochastic-dynamic inventory routing problem as Markov decision process and presented a decision policy allowing for coordination and anticipation. The policy anticipates future demand development based on historical data. Based on this development, it relocates bikes and coordinates the vehicles accordingly. We have tested our policy for two real-world bike sharing systems in Minneapolis and San Francisco. Our results show that both anticipation and coordination are essential for obtaining high customer satisfaction.

In future research, model and method could be further extended. The model may integrate the scheduling of drivers based on the workload. This scheduling is challenging because of the decisions' interdependencies in stochastic, dynamic decision problems. Scheduling decisions for one hour affect failed demand in all future hours. The method may be extended to integrate potential routing developments as well as potential trips between the stations. Integrating routing decisions is challenging because it either requires extensive simulations with a runtime-efficient routing policy or the solution of a stochastic and continuous time inventory routing problem. Furthermore, our results show that spontaneous demand realizations impede long-term planning. Thus, a method may be developed that focuses more on future routing flexibility than on explicit future routing decisions.

Acknowledgements The authors thank the editors and the two anonymous reviewers for their helpful suggestions and comments in preparation of this work.

A Appendix

In the Appendix, we present details of the customer behavior model and results for the individual instances.

A.1 User behavior

As aforementioned, a user approaches a neighboring station if his demand fails. We assume that information on other stations' fill levels is provided by a screen at the station terminal. In this section, we formally define how users select alternative stations in case of failed demand.

Let n_o and n_d be the user's origin and destination stations, and let f^n be the current fill level of any station n. If the rental demand fails at n_o, he approaches the neighboring station n_{o_2} which is closest to the origin and is neither empty nor extends the travel time to the destination according to Formula (17):

$$n_{o_2} = \underset{n \in N}{\arg\min} \{\tau(n_o, n) \mid 0 < f^n \ \wedge \ \tau(n, n_d) \leq \tau(n_o, n_d)\}. \tag{17}$$

If the return demand fails at n_d, he approaches the neighboring station n_{d_2} which is closest to the destination and is not full:

$$n_{d_2} = \underset{n \in N}{\arg\min} \{\tau(n_d, n) \mid f^n < c^n\}. \tag{18}$$

We are aware that in a worst case scenario, a user may approach stations to infinity. However, we also have to state that in the computational studies we did not observe failing demand two times in a row.

A.2 Results

To determine the safety buffer β for STR and STR-NC as well as the lookahead horizon δ for CLA and CLA-NC, we run experiments with $\beta \in \{0.1, 0.2, \ldots, 0.5\}$ and $\delta \in \{60, 120, \ldots, 720\}$ for 1000 non-concatenated test days. For every setting, i.e., BSS and vehicle fleet size, we individually determine parameters. For every setting, the best parameter's results are presented in Table 2 for Minneapolis, or in Table 3 for San Francisco, respectively.

The column 'Policy' indicates the policy and the corresponding parameter in brackets. 'Vehicles' indicates the fleet size. The column 'Coordination' points out

Table 2 Minneapolis results

Policy –	Vehicles –	Coordination –	Failed demand 259.67	CPU time [s] –
STR(0.1)	1	–	139.11	0.14
STR(0.2)	2	–	86.80	0.14
STR(0.2)	3	–	67.01	0.15
STR(0.2)	4	–	57.05	0.15
STR(0.2)	2	Not same station	79.77	0.16
STR(0.2)	3	Not same station	52.77	0.18
STR(0.3)	4	Not same station	40.17	0.28
CLA(240)	1	–	98.33	0.19
CLA(240)	2	–	67.48	0.49
CLA(240)	3	–	59.15	0.90
CLA(240)	4	–	56.89	1.21
CLA(240)	2	Not same station	54.00	0.49
CLA(300)	3	Not same station	39.09	1.05
CLA(420)	4	Not same station	32.76	2.20
CLA(240)	2	Partial	53.36	0.50
CLA(300)	3	Partial	38.23	1.09
CLA(420)	4	Partial	31.68	2.29
CLA(240)	2	Complete	55.24	0.54
CLA(360)	3	Complete	40.07	1.37
CLA(420)	4	Complete	33.84	2.40
CLA(180)	2	Optimal static	57.10	0.52
CLA(240)	3	Optimal static	41.34	1.14
CLA(300)	4	Optimal static	34.86	2.16

how coordination is realized. Here, a dash indicates that no coordination takes place. 'Failed Demand' depicts the corresponding average number over all test days. In the last column, the average CPU time for one test day is given in seconds. The first rows refer to the case if no relocations are realized.

We further present results for the experiments where we solve the static assignment problem to optimality. To this end, in every decision point, we solve the assignment problem by means of the Hungarian Algorithm (Stern 2012). We observe that optimal solutions of this static subproblem do not yield any benefit but even decreases solution quality. This observations is in alignment with other

Table 3 San Francisco results

Policy –	Vehicles –	Coordination –	Failed demand 344.62	CPU time [sec]
STR(0.2)	1	–	210.96	0.04
STR(0.2)	2	–	147.95	0.05
STR(0.3)	3	–	112.68	0.05
STR(0.3)	4	–	94.84	0.05
STR(0.2)	2	Not same station	147.68	0.05
STR(0.3)	3	Not same station	106.09	0.05
STR(0.3)	4	Not same station	83.79	0.06
CLA(300)	1	–	133.15	0.06
CLA(300)	2	–	63.62	0.17
CLA(300)	3	–	50.75	0.33
CLA(240)	4	–	45.55	0.39
CLA(300)	2	Not same station	57.33	0.16
CLA(360)	3	Not same station	35.76	0.32
CLA(300)	4	Not same station	32.52	0.45
CLA(360)	2	Partial	56.79	0.19
CLA(300)	3	Partial	34.88	0.29
CLA(420)	4	Partial	31.82	0.46
CLA(300)	2	Complete	58.86	0.17
CLA(360)	3	Complete	37.16	0.35
CLA(360)	4	Complete	33.65	0.54
CLA(300)	2	Optimal static	62.32	0.18
CLA(300)	3	Optimal static	39.14	0.32
CLA(300)	4	Optimal static	36.33	0.49

research on stochastic and dynamic decision problems (Maggioni and Wallace 2012; Powell et al. 2000).

References

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

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

Brinkmann, J., M.W. Ulmer, and D.C. Mattfeld. 2016. Inventory routing for bikes sharing systems. *Transportation Research Procedia* 19: 316–327.

Brinkmann, J., M.W. Ulmer, and D.C. Mattfeld. 2019. Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. *Computers & Operations Research* 106: 260–279.

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

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

Chiariotti, F., C. Pielli, A. Zanella, and M. Zorzi. 2018. A dynamic approach to rebalancing bike-sharing systems. *Sensors* 18 (2): E512.

Contardo, C., C. Morency, and L.M. Rousseau. 2012. Balancing a Dynamic Public Bike-Sharing System. CIRRELT-2012-09, https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf. Accessed 8 Dec 2014.

Datner, S., T. Raviv, M. Tzur, and D. Chemla. 2017. Setting inventory levels in a bike sharing network. *Transportation Science* 53 (1): 62–76.

Erdoğan, G., G. Laporte, and R. Wolfler Calvo. 2014. The static bicycle relocation problem with demand intervals. *European Journal of Operational Research* 238 (2): 451–457.

Erdoğan, G., M. Battarra, and R.W. Calvo. 2015. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research* 245 (3): 667–679.

Espegren, H.M., J. Kristianslund, H. Andersson, and K. Fagerholt. 2016. The Static Bicycle Repositioning Problem - Literature Survey and New Formulation. In: Computational Logistics, Lecture Notes in Computer Science, Springer, pp 9855:337–351

Ford GoBike. 2017. Ford Go Bike San Francisco. https://www.fordgobike.com/. Accessed 08 Dec 2017.

Fricker, C., and N. Gast. 2016. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics* 5 (3): 261–291.

Ghosh, S., M.A. Trick, and P. Varakantham. 2016. Robust Repositioning to Counter Unpredictable Demand in Bike Sharing Systems. Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16).

Ghosh, S., P. Varakantham, Y. Adulyasak, and P. Jaillet. 2017. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research* 58: 387–430.

Kloimüllner, C., P. Papazek, B. Hu, and G.R. Raidl. 2014. Balancing Bicycle Sharing Systems: An Approach for the Dynamic Case. In: Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science. Springer, Berlin, pp 8600:73–84.

Legros, B. 2019. Dynamic repositioning strategy in a bike-sharing system; how to prioritize and how to rebalance a bike station. *European Journal of Operational Research* 272 (2): 740–753.

Lu, C.C. 2016. Robust multi-period fleet allocation models for bike-sharing systems. *Networks and Spatial Economics* 16 (1): 61–82.

Maggioni, F., and S.W. Wallace. 2012. Analyzing the quality of the expected value solution in stochastic programming. *Annals of Operations Research* 200 (1): 37–54.

Neumann Saavedra, B.A., P. Vogel, and D.C. Mattfeld. 2015. Anticipatory service network design of bike sharing systems. *Transportation Research Procedia* 10: 355–363.

Neumann Saavedra, B.A., T.G. Crainic, B. Gendron, D.C. Mattfeld, and M. Römer. 2016. Service Network Design of Bike Sharing Systems with Resource Constraints. In: Computational Logistics, Lecture Notes in Computer Science. Springer, Berlin, pp 9855:352–366.

Nice Ride. 2016. Nice Ride Minneapolis (MN/USA). https://www.niceridemn.org/. Accessed 05 Oct 2016.

O'Brien, O., J. Cheshire, and M. Batty. 2014. Mining bicycle sharing data for generating insights into sustainable transport systems. *Journal of Transport Geography* 34: 262–273.

Powell, W.B., M.T. Towns, and A. Marar. 2000. On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance. *Transportation Science* 34 (1): 67–85. https://doi.org/10.1287/trsc.34.1.67.12283.

Puterman, M.L. 2014. *Markov decision processes: discrete stochastic dynamic programming*, 2nd ed. Hoboken: Wiley.

Raviv, T., and O. Kolka. 2013. Optimal inventory management of a bike-sharing station. *IIE Transactions* 45 (10): 1077–1093.

Raviv, T., M. Tzur, and I.A. Forma. 2013. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics* 2 (3): 187–229.

Schuijbroek, J., R.C. Hampshire, and W.J. van Hoeve. 2017. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research* 257 (3): 992–1004.

Stern, K.L. 2012. Hungarian Algorithm. GitHub, Inc., https://github.com/KevinStern/software-and-algorithms/blob/master/src/main/java/blogspot/software_and_algorithms/stern_library/optimization/HungarianAlgorithm.java. Accessed 10 Oct 2018.

Szeto, W.Y., and C.S. Shui. 2018. Exact loading and unloading strategies for the static multi-vehicle bike repositioning problem. *Transportation Research Part B: Methological* 109: 176–211.

Szeto, W.Y., Y. Liu, and S.C. Ho. 2016. Chemical reaction optimization for solving a static bike repositioning problem. *Transportation Research Part D: Transport and Environment* 47: 104–135.

Vogel, P., T. Greiser, and D.C. Mattfeld. 2011. Understanding bike-sharing systems using data mining: exploring activity patterns. *Procedia-Social and Behavioral Sciences* 20: 514–523.

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

Yan, S., J.R. Lin, Y.C. Chen, and F.R. Xie. 2017. Rental bike location and allocation under stochastic demands. *Computers & Industrial Engineering* 107: 1–11.