European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

A column generation heuristic for the dynamic bicycle rebalancing problem

Marte D. Gleditsch, Kristine Hagen, Henrik Andersson, Steffen J. Bakker*, Kjetil Fagerholt

Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz veg 3, Trondheim NO-7491, Norway

ARTICLE INFO

Article history:
Received 28 February 2022
Accepted 5 July 2022
Available online xxx

Keywords:
Transportation
Bicycle sharing
Dynamic rebalancing
Column generation heuristic
Simulation

ABSTRACT

Public bicycle sharing systems are becoming an essential part of the future urban mobility system. Real-time monitoring of the system state through sensors on bicycles and/or stations gives possibilities for advanced coordination of the system. In this paper, we consider the dynamic bicycle rebalancing problem, where bicycles are re-positioned by service vehicles to prevent stations from becoming completely full or empty, and so satisfying the demand for bicycles or locks. We solve the problem in a rolling horizon fashion with dynamic deterministic bicycle rebalancing subproblems (DDBRS) at the decision epochs. To solve the DDBRS within a few seconds in real-time, we propose a novel column generation heuristic (CGH). The CGH is tested within a simulation framework based on real data from the bicycle sharing system in Oslo. We show that the CGH is able to solve large real-life instances with computational times that are suitable for actual operation and that it provides significantly improved solutions compared with current planning practice. We also perform a number of tests to analyze the effect of changing the number of bicycles and locks in the system, as well as adding extra service vehicles. The case company is now making preparations to implement an optimization-based decision support system based on the CGH proposed in this paper.

© 2022 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

The amount of private motorized traffic is increasing in cities all over the world. This leads to, amongst others, traffic congestion and environmental pollution. Sustainable transportation methods, such as bicycle sharing systems (Laporte, Meunier, & Wolfler Calvo, 2018; Shui & Szeto, 2020), will therefore play an increasing role in future urban mobility systems (Kaspi, Raviv, & Ulmer, 2022). A Bicycle Sharing System (BSS) allows users to pick up a bicycle at a station, ride it to their destination, and lock it at a nearby station.

BSSs have already become an essential part of cities' public transport systems. Over the last twenty years, the number of BSSs has increased from 10 to around 2000 systems (The Meddin Bike-sharing World Map, 2021). During recent years, many of these systems have equipped their bicycles and/or stations with sensors, giving opportunities in terms of real-time tracking and coordination of the system. An important operational challenge for most BSSs is that stations, where the bicycles are picked up and dropped off, regularly get empty (i.e., no bicycles to pick up) or full (i.e., no locks to drop off the bicycle) due to customer interactions. This leads to an imbalanced system with missed demand and poor user experience as a result.

In this paper we investigate the operational planning problem of rebalancing bicycles during the day using dedicated service vehicles and real-time data. Bicycle rebalancing involves making routing and inventory management decisions (Gammelli et al., 2022). It can be categorized into static and dynamic rebalancing, where the former typically happens over night and the latter happens throughout the day. The main difference is that for the dynamic case, user demand is considered during the rebalancing, making the modelling more complicated. The static rebalancing problem has attracted a lot of research (Bulhões, Subramanian, Erdoğan, & Laporte, 2018; Chemla, Meunier, & Wolfler Calvo, 2013; Erdogan, Battarra, & Wolfler Calvo, 2015; Erdoğan, Laporte, & Wolfler Calvo, 2014; Espegren, Kristianslund, Andersson, & Fagerholt, 2016; Maggioni, Cagnolari, Bertazzi, & Wallace, 2019; Raviv, Tzur, & Forma, 2013; Schuijbroek, Hampshire, & van Hoeve, 2017). It is an important problem for bike sharing systems in big cities that need to have a good initial state at the beginning of the day, or when rebalancing throughout the day is challenging due to traffic issues. Nevertheless, having a balanced system throughout the

* Corresponding author.
E-mail address: steffen.bakker@ntnu.no (S.J. Bakker).

https://doi.org/10.1016/j.ejor.2022.07.004
0377-2217/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

Please cite this article as: M.D. Gleditsch, K. Hagen, H. Andersson et al., A column generation heuristic for the dynamic bicycle rebalancing problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2022.07.004

day is crucial for a well-functioning BSS, being the main motivation to study the dynamic bicycle rebalancing problem (DBRP) in this paper.

The DBRP is a complex problem which has received increased attention in the literature. We present a brief overview over the most crucial elements and refer the interested reader to Brinkmann, Ulmer, & Mattfeld (2019) and Wang & Szeto (2021) for a more extensive classification of DBRP literature.

In the DBRP, demand for bicycles and locks is not known beforehand, and the system changes over time as users pick up and deliver bicycles. The literature deals with this complexity in different ways. First, instead of making routing and inventory decisions in an integrated way, they can be decomposed and solved sequentially (Regue & Recker, 2014). Moreover, it is common to determine offline routing and rebalancing decisions for a long time horizon, allowing for longer computational times (Ghosh, Varakantham, Adulyasak, & Jaillet, 2017). Furthermore, it is common to break down the DBRP into more manageable subproblems, considering shorter planning horizons, and solve these in a rolling horizon fashion (Shui & Szeto, 2018). These subproblems can then be modelled using continuous- or discrete time formulations. For the latter, a time-space network flow model (Contardo, Morency, & Rousseau, 2012; Pfrommer, Warrington, Schildbach, & Morari, 2014; Zhang, Yu, Desai, Lau, & Srivathsan, 2017) is usually used. Most DBRPs minimize imbalances in demand satisfaction (Brinkmann et al., 2019; Legros, 2019; Shui & Szeto, 2018), but other measures, such as travel costs, can also be formulated in the objective function (Dell'Amico, Iori, Novellani, & Stützle, 2016).

We can further categorize the DBRP based on the consideration of service vehicles. Nair, Miller-Hooks, Hampshire, & Bušić (2012) calculate the need for rebalancing within considering vehicles explicitly, while (Brinkmann et al., 2019; Legros, 2019; Pfrommer et al., 2014; Regue & Recker, 2014) do the rebalancing for a single vehicle. For larger systems, the interaction between the routes of individual service vehicles can become a major challenge. A typical approach is then to allocate individual service vehicles to predefined service areas (Fu, Zhu, Ma, & Liu, 2021), but this can lead to sub-optimal solutions by design. As opposed to the majority of the literature, we consider multiple vehicles that perform rebalancing in an integrated way. This should be the preferred approach for larger systems. To the extend of our knowledge, Brinkmann, Ulmer, & Mattfeld (2020) is the only work that also does this by extending the work from Brinkmann et al. (2019) to allow for coordinating multiple service vehicles. They view the DBRP as a Markov decision process and embrace methods from approximate dynamic programming (Powell, 2007) to solve it. The constructed lookahead policies mainly focus on which rebalancing actions to perform at the current station and to which station to go next. Our approach, on the other hand, explicitly models the future routing and rebalancing decisions of all vehicles by means of a mixed integer programming (MIP) model and a rolling horizon approach. As opposed to the literature, we use a continuous time formulation for the subproblems. The instances that we solve have similar dimensions as to what is presented in Brinkmann et al. (2020).

When considering solution approaches for the (sub-)problems, we observe that exact methods, such as Branch-and-Cut (Erdoğan et al., 2014), are not suitable for the operational DBRP due to high solution times and limitations on the problem size that can be solved. Therefore, various heuristic approaches, including Large Neighbourhood Search (Ho & Szeto, 2017); Enhanced Artificial Bee Colony (Shui & Szeto, 2018); Variable Neighbourhood Descent/Search and Greedy Randomized Adaptive Search Procedure (Rainer-Harbach, Papazek, Raidl, Hu, & Kloimüllner, 2015), have been investigated. Nevertheless, there are still many promising approaches that have not been tested, for example heuristics based on column generation. These have shown to provide good results on other vehicle routing problems (Vadseth, Andersson, & Stålhane, 2021; Yuan, Cattaruzza, Ogier, Semet, & Vigo, 2021).

Finally, we observe that the majority of works focus on the (computational) performance of the solution approach only in the context of the formulated optimization model. However, ideally, these methods should be tested in real-life bicycle sharing systems. As such, there have been several studies that have developed a simulator, mimicking real-life BSSs, to test their approaches (Ghosh et al., 2017; Regue & Recker, 2014; Shu, Chou, Liu, Teo, & Wang, 2013).

To solve the DBRP in a practical setting, we propose a solution approach where we iteratively solve a subproblem with a relatively short planning horizon in a rolling horizon fashion at each decision epoch. Since we consider such a short planning horizon, we assume this subproblem to be deterministic. The main contributions of this paper are: (1) the development of a new MIP model for this problem; (2) a column generation heuristic to solve the problem in reasonable time; (3) the simulation of this approach in a detailed discrete-event simulator for solving the DBRP using real-life data from the BSS in Oslo, Norway, showing that solution time is a crucial aspect for good real-time performance; and (4) using these elements to present various managerial insights.

The remainder of this paper is structured as follows: Sections 2 and 3 provide a description and mathematical formulation, respectively, of the dynamic deterministic bicycle rebalancing subproblem. We solve this problem using the column generation heuristic described in Section 4 and test its performance using a discrete-event simulator described in Section 5. We present an extensive computational study in Section 6, before concluding in Section 7.

2. Problem description

The DBRP is an inherently dynamic and stochastic problem. Demand for bicycles and locks is not known beforehand, and the system changes over time as users pick up and deliver bicycles. A common technique for solving large-scale discrete-time multistage stochastic problems is to approximate the problem by a series of smaller subproblems and solve these at each decision epoch in a rolling horizon fashion (Shui & Szeto, 2018). Decision epochs can be defined in terms of an event, such as the arrival of a service vehicle at a station, or a fixed time interval. The generated, overlapping, subproblems have a much shorter planning horizon than the original problem.

Fig. 1 illustrates this rolling horizon approach. Preliminary testing has shown that allowing for multiple visits is not efficient when considering a short planning horizon. Hence, with a sufficiently short planning horizon in the subproblem, the demand for bicycles and locks can be assumed to be deterministic and stations can be visited at most once. This leads to the dynamic deterministic bicycle rebalancing subproblem (DDBRS), which we will refer to as *the subproblem*.

The DDBRS considers a bicycle sharing system consisting of a fleet of bicycles, capacitated stations and a heterogeneous fleet of capacitated service vehicles. At each station, there is a known demand for bicycles and locks for the considered planning horizon (e.g., 30 minutes). This demand typically fluctuates throughout a day. The distribution of bicycles in the system depends on the pick-ups and deliveries of bicycles by the users, as well as rebalancing decisions from the BSS operator. The event where a bicycle-demanding customer arrives at an empty station is referred to as a *starvation*, while the event where a lock-demanding customer arrives at a full station is referred to as a *congestion*. These events are collectively known as *violations*. To meet future demand beyond the planning horizon of the subproblem, each station has a

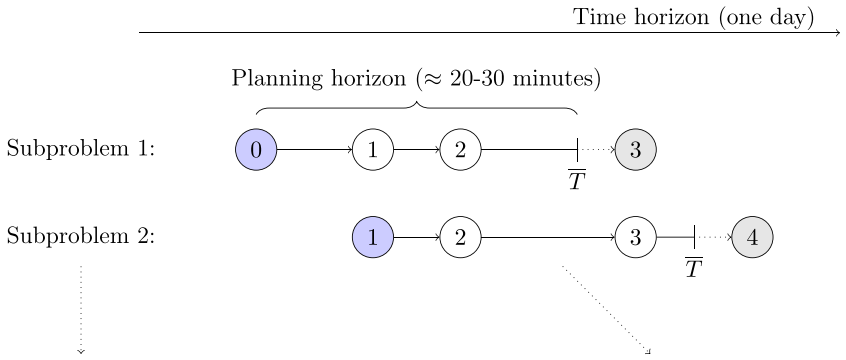


Fig. 1. Illustration of a rolling horizon procedure for a single service vehicle. Stations are represented by nodes in a path.

target inventory level (number of parked bicycles) at the end of the planning horizon, defined as the station's *target state*. The difference between a station's load at the end of the horizon and its target state is defined as a *deviation*.

The DDBRS consists of, for a given planning horizon, determining (1) the route of each of the service vehicles (i.e., the sequence of stations to visit), and (2) how many bicycles to pick up and deliver at each station along these routes. The objective is to minimize a weighted sum of the total number of violations and deviations, by performing rebalancing actions with the available service vehicles during the considered planning horizon. We assume that the initial state of the problem, described by the number of bicycles at each station and the service vehicles' location and number of loaded bicycles, is known. The *net demand* at a station, defined as the number of customers requesting a lock minus the number of customers requesting a bicycle (every time unit), can be estimated based on historical data. Stations are defined as *delivery* or *pickup* stations, depending on whether they have a positive or negative net demand in the following planning horizon, respectively. Note that a station can be a pickup station during one period of the day and a delivery station during another. The service vehicles' driving times between stations are assumed to be known and include a fixed parking time which can depend on the location as well as the service vehicle. The bicycle handling time is proportional to the number of bicycles handled, regardless of the nature of the actions.

As explained more in Section 5, it should be emphasized that the DDBRS is to be solved at every decision epoch, e.g., whenever a service vehicle arrives at a station. This also means that we are in reality only concerned about the decision about (1) how many bicycles should this particular service vehicle pick up or deliver at the current station, and (2) which station to go to next. However, in order not to be too myopic, the subproblem also includes the decisions about future stations to visit for the whole fleet of service vehicles. To prevent further shortsightedness, we also consider the deviations from the target state and incentivize service vehicles to start on a trip that exceeds the relatively short planning horizon considered in the subproblem, as illustrated in Fig. 1. This means that the planning horizon indicates the latest time vehicles can start a new trip.

3. Model formulation

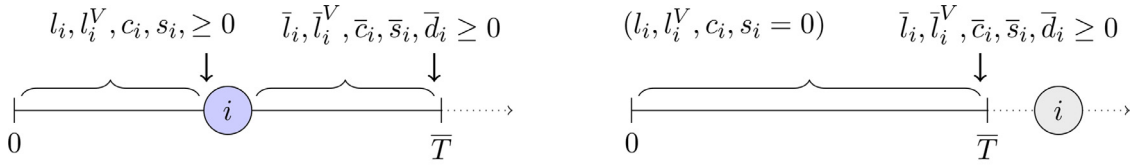
In the following, we present the notation and the mathematical formulation for the DDBRS. A summary of all notation is presented in Appendix A.

3.1. Notation

We consider a set of stations $\mathcal{S}$ with elements indexed by i and j, where each station has a given number of bicycle locks that are represented by the capacity Q_i. Moreover, we have a set of service vehicles $\mathcal{V}$, where the capacity Q^V determines the number of bicycles that can be loaded on each vehicle. The problem is solved over a planning horizon $\overline{T}$, indicating the latest time service vehicles can start a trip to a new station. Each service vehicle starts and ends its route at the origin and destination nodes o and d, representing the initial and final location, respectively. The parameters L_i and L_v^V define the initial number of bicycles at station i and on service vehicle v, respectively. The target state at the end of the planning horizon at station i is denoted $\overline{L}_i^T$. The driving time between stations i and j, including the parking time at station j, is denoted by T_{ij}^D, and T^H is the handling time per bicycle for loading and unloading onto/from the service vehicles. The customer demand D_i represents the net demand per time unit at station i. A positive demand indicates demand for locks, and a negative demand indicates demand for bicycles. Stations with positive demand are denoted as pickup stations ($\mathcal{S}^L$), and stations with negative demand are denoted as delivery stations ($\mathcal{S}^U$). We use the convention that M denotes a large number whose minimum value can be deduced from the constraint where it appears.

Let x_{ijv} be a binary variable which equals 1 if service vehicle v travels directly from station i to station j, and 0 otherwise. Let z_i be a binary variable which is equal to 1 if station i is visited during the planning horizon, and 0 otherwise. The continuous variable t_i takes the time when station i is visited, and equals zero when no visit happens. The integer variables q_{iv}^L and q_{iv}^U keep track of how many bicycles are loaded from and unloaded to station i by service vehicle v, respectively. The number of bicycles at station i just *before* being visited is denoted by l_i, this variable equals zero when the station is not visited during the planning horizon. The number of bicycles on service vehicle v just *after* the service vehicle has visited station i is denoted by l_{iv}^V. Moreover, we introduce two variables for violations when a station visit occurs during the planning horizon: the accumulated congestions and starvations, denoted by c_i and s_i, respectively. Now, let $\overline{l}_i, \overline{l}_v^V, \overline{c}_i$ and $\overline{s}_i$ be defined similarly as their counterparts without the bar, with the only difference that they represent the values at the end of the planning horizon $\overline{T}$.

The variable $\overline{d}_i$ represents the deviation at station i at the end of the planning horizon, i.e., the difference between the number of bicycles at station i and its target state $\overline{L}_i$. A visualization over these variable definition is given in Fig. 2. Finally, we force each route to start on a trip that ends after the planning horizon. A reward r_i that depends on the deviation at the stations can be obtained, but a cost is also incurred for the extra driving time t_v^R. Binary variables y_v^L and y_v^U are equal to 1 when vehicle v is within a range of I units from being empty or full at the end of the planning horizon and will have to perform, respectively, a loading or unloading action next.



(a) Visit during planning horizon ($z_i = 1$)

(b) No visit during planning horizon ($z_i = 0$)

Fig. 2. Definition of loading, congestion, starvation and deviation variables depending on whether a station is visited or not.

3.2. Objective function

The objective of the DDBRS is presented in (1), and minimizes a weighted sum of violations (before and at the end of the planning horizon) and deviations. Moreover, we give a reward for starting on a new visit that ends after the planning horizon, but penalize the extra driving time. We use ω with superscript to denote the weights where V is violations, D deviations, $R+$ rewards and $R-$ extra driving time.

$$\min \quad \sum_{i \in \mathcal{S}} \left[\omega^V (s_i + c_i + \bar{s}_i + \bar{c}_i) + \omega^D \bar{d}_i - \omega^{R+} r_i \right] + \sum_{v \in \mathcal{V}} \omega^{R-} t_v^R \tag{1}$$

3.3. Constraints

Degree. Constraints (2a) and (2b) force each service vehicle to start and end in the origin and destination respectively. Constraints (2c) set a nodal balance, ensuring that if a vehicle visits a station, it also leaves that station. Constraints (2d) enforce that each station can be visited at most once. Moreover, the binary variable z_i, capturing whether a station sees a visit during the planning horizon, is defined in constraints (2e).

$$\sum_{j \in \mathcal{S}} x_{ojv} = 1 \quad v \in \mathcal{V} \tag{2a}$$

$$\sum_{i \in \mathcal{S}} x_{idv} = 1 \quad v \in \mathcal{V} \tag{2b}$$

$$\sum_{i \in \mathcal{S}} x_{ijv} - \sum_{k \in \mathcal{S}} x_{jkv} = 0 \quad j \in \mathcal{S}, v \in \mathcal{V} \tag{2c}$$

$$\sum_{j \in \mathcal{S}} \sum_{v \in \mathcal{V}} x_{ijv} \leq 1 \quad i \in \mathcal{S} \tag{2d}$$

$$z_i = \sum_{j \in \mathcal{S}} \sum_{v \in \mathcal{V}} x_{ijv} \quad i \in \mathcal{S} \tag{2e}$$

Timing. If a service vehicle drives from station i to j, then constraints (3a) set the start time of visit j equal to the start time of visit i plus the handling time at station i and driving time from i to j. Constraints (3b) and (3c) enforce bounds on the visit times and make sure that the last visit is always after the end of the planning horizon.

$$t_i + T^H \sum_{v \in \mathcal{V}} \left(q_{iv}^L + q_{iv}^U \right) + T_{ij}^D \leq t_j + M \left(1 - \sum_{v \in \mathcal{V}} x_{ijv} \right) \quad i, j \in \mathcal{S} \tag{3a}$$

$$\overline{T} \sum_{v \in \mathcal{V}} x_{idv} \leq t_i \leq \overline{T} + M \sum_{v \in \mathcal{V}} x_{idv} \quad i \in \mathcal{S} \tag{3b}$$

$$t_i \leq M \sum_{j \in \mathcal{S}} \sum_{v \in \mathcal{V}} x_{jiv} \quad i \in \mathcal{S} \tag{3c}$$

Vehicle load. Constraints (4a) and (4b) ensure the vehicle load balance, while (4c) and (4d) put capacities on the vehicle loading and unloading, respectively.

$$l_{iv}^V + q_{iv}^U - q_{iv}^L - l_{jv}^V - M(1 - x_{ijv}) \leq 0 \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{4a}$$

$$l_{iv}^V + q_{iv}^U - q_{iv}^L - l_{jv}^V + M(1 - x_{ijv}) \geq 0 \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{4b}$$

$$q_{iv}^L \leq Q^V z_i - l_{iv}^V \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{4c}$$

$$q_{iv}^U \leq l_{iv}^V \leq Q^V z_i \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{4d}$$

Constraints (5a) and (5b) define the vehicle load at the planning horizon. To reduce the solution space, we enforce that if the vehicle load at the planning horizon is close to its lower or upper limit (within I units), then the final visit cannot be to a delivery or pickup station, respectively. This is imposed in constraints (5c) through (5f). The threshold value I is assumed to be the same for the upper and lower limit, but it can be defined differently.

$$\bar{l}_v^V \geq l_{iv}^V - Q^V (2 - x_{ijv} - x_{jdv}) \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{5a}$$

$$\bar{l}_v^V \leq l_{iv}^V + Q^V (2 - x_{ijv} - x_{jdv}) \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{5b}$$

$$\bar{l}_v^V \geq I(1 - y_v^L) \quad v \in \mathcal{V} \tag{5c}$$

$$\bar{l}_v^V \leq Q^V - I(1 - y_v^U) \quad v \in \mathcal{V} \tag{5d}$$

$$x_{idv} \leq (1 - y_v^L) \quad i \in \mathcal{S}^U \tag{5e}$$

$$x_{idv} \leq (1 - y_v^U) \quad i \in \mathcal{S}^L \tag{5f}$$

Station inventory, starvations and congestions. Turning to station inventory levels, vehicle (un)loading is also restricted by the available station capacity as given in constraints (6a). If a station is visited in the planning horizon, constraints (6b) put a capacity on the number of parked bicycles at the stations, while constraints (6c) calculate the station load and incurred starvations or congestions just before the rebalancing starts.

$$l_i + \sum_{v \in V} (q_{iv}^U - q_{iv}^L) \leq Q_i z_i \quad i \in \mathcal{S} \tag{6a}$$

$$l_i \leq Q_i z_i \quad i \in \mathcal{S} \tag{6b}$$

$$l_i = L_i^0 z_i + D_i t_i + s_i - c_i \quad i \in \mathcal{S} \tag{6c}$$

The station load and congestions or starvations at the planning horizon $\overline{T}$ are set in constraints (7a) through (7e). Constraints (7b) and (7c) state that if a station is visited during the planning horizon, then the station load at the planning horizon equals the load just after the visit plus subsequent changes due to customer demand and potential starvations or congestions. Constraints (7d) and (7e) do the same thing for the case when the station is not visited during the planning horizon.

$$\bar{l}_i \leq Q_i \quad i \in \mathcal{S} \tag{7a}$$

$$\bar{l}_i \leq \left(l_i + \sum_{v \in \mathcal{V}} q^L_{iv} - q^U_{iv}\right) + D_i\left(\overline{T} - t_i\right) - \bar{s}_i + \bar{c}_i + M(1 - z_i) \quad i \in \mathcal{S} \tag{7b}$$

$$\bar{l}_i \geq \left(l_i + \sum_{v \in \mathcal{V}} q^L_{iv} - q^U_{iv}\right) + D_i\left(\overline{T} - t_i\right) - \bar{s}_i + \bar{c}_i - M(1 - z_i) \quad i \in \mathcal{S} \tag{7c}$$

$$\bar{l}_i \leq L^0_i + D_i\overline{T} + \bar{s}_i - \bar{c}_i + Mz_i \quad i \in \mathcal{S} \tag{7d}$$

$$\bar{l}_i \geq L^0_i + D_i\overline{T} + \bar{s}_i - \bar{c}_i - Mz_i \quad i \in \mathcal{S} \tag{7e}$$

Deviations and rewards. Constraints (8a) and (8b) define the deviation at station i at time $\overline{T}$ to equal the absolute difference between the target inventory level and actual level. The reward is defined in (8c) and (8d), which can equal the deviation at station i. Finally, the extra driving time after the end of the planning horizon is defined in (8e) and (8f).

$$\bar{d}_i \geq \bar{L}_i - \bar{l}_i \quad i \in \mathcal{S} \tag{8a}$$

$$\bar{d}_i \geq \bar{l}_i - \bar{L}_i \quad i \in \mathcal{S} \tag{8b}$$

$$r_i \leq \bar{d}_i \quad i \in \mathcal{S} \tag{8c}$$

$$r_i \leq Q_i \sum_{v \in \mathcal{V}} x_{idv} \quad i \in \mathcal{S} \tag{8d}$$

$$t^R_v \leq t_i - \overline{T} + M(1 - x_{idv}) \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{8e}$$

$$t^R_v \geq t_i - \overline{T} - M(1 - x_{idv}) \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{8f}$$

Domains. Eqs. (9a) –(9c) declare the domains of the variables:

$$x_{ijv}, z_i, y^L_v, y^U_v \in \{0, 1\} \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{9a}$$

$$q^U_{iv}, q^L_{iv} \in \mathbb{Z}^+_0 \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{9b}$$

$$t_i, t^R_v, l_i, \bar{l}_i, \bar{l}^V_{iv}, c_i, \bar{c}_i, s_i, \bar{s}_i, \bar{d}_i, r_i \in \mathbb{R}^+_0 \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{9c}$$

4. Column generation heuristic

As solutions to the DDBRS have to be generated *online*, the solution approach must be able to provide high-quality results within mere seconds. We propose to use a column generation heuristic (CGH), which has proved to provide good quality solutions within a short amount of time. In particular, CGH is a promising approach when routes are short, which is the case since the subproblem has a relatively short planning horizon.

Fig. 3 presents a conceptual overview of the CGH. Columns include feasible routes for service vehicles and potentially also information about the number of bicycles loaded/unloaded along each route. In Section 4.2 we describe how to generate the initial set of columns using a route extension algorithm. Given this set of initial columns the CGH iterates between a master problem (MP), presented in Section 4.1, and a scoring problem (SP), described in Section 4.3, where the MP finds the optimal combination of columns (solution) and the SP adds new columns that potentially can improve the current solution.

We test two variants of the CGH, varying in what information the columns include:

1. *Route-based*: columns represent routes only, i.e., the sequence of stations to visit. Hence, all other variables, i.e., related to loading quantities, arrival times, inventory levels and violations, are determined endogenously in the MP.
2. *Pattern-based*: columns also include loading/unloading patterns, which means that all information about loading quantities, arrival times, inventory levels and violations is given. Hence, the master problem reduces to a set packing problem.

4.1. Master problems

We now present the MPs for the *Route-based* and *Pattern-based* variants of the CGH.

4.1.1. Route-based

In this variant, columns consist only of the geographical routes for the service vehicles. For each service vehicle v we define a set of feasible columns $\mathcal{R}_v$. The binary parameter A_{ijvr} indicates whether service vehicle v drives directly from station i to station j in route r. The binary variable λ_{vr} takes a value of 1 if route r is allocated to service vehicle v, and 0 otherwise. The MP for the *Route-based* variant is obtained by taking the mathematical program as defined by (1)–(9) in Section 3 and adding the following constraints:

$$x_{ijv} = \sum_{r \in \mathcal{R}_v} A_{ijvr}\lambda_{vr} \quad i, j \in \mathcal{S}, v \in \mathcal{V} \tag{10a}$$

$$\sum_{r \in \mathcal{R}_v} \lambda_{vr} = 1 \quad v \in \mathcal{V} \tag{10b}$$

$$\lambda_{vr} \in \{0, 1\} \quad v \in \mathcal{V}, r \in \mathcal{R}_v \tag{10c}$$

Constraints (10a) link the routing and flow variables, while constraints (10b) allow exactly one route per service vehicle, and (10c) are the binary restrictions. We note that in model (1)–(9) the flow variables x_{ijv} are completely replaced by the interior representation $\sum_{r \in R_v} A_{ijvr}\lambda_{vr}$.

4.1.2. Pattern-based

The pattern-based master problem considers columns which include both information about geographical routes as well as loading/unloading patterns. To formulate the corresponding model, we have to define the following new notation. Let the binary variable

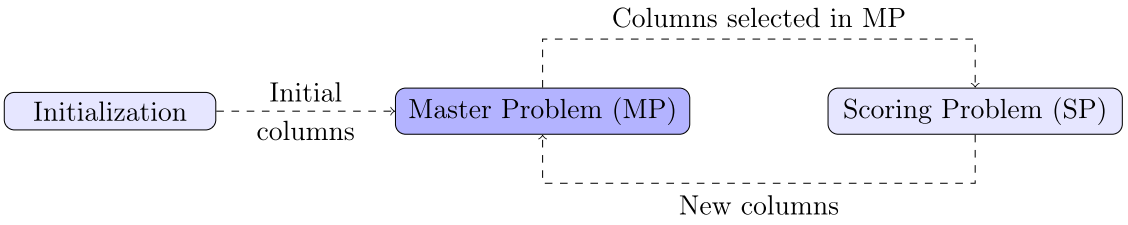


Fig. 3. Conceptual overview of the column generation heuristic.

λ_{vr} equal 1 if service vehicle v is allocated route r, and 0 otherwise. Let the binary parameter A_{ivr} equal 1 if vehicle v visits station i in route r, and 0 otherwise. The parameters V and D represent the total number of violations and deviations when no rebalancing actions are performed. Since stations only can be visited by a single vehicle during the planning horizon, the routes are independent of each other. As such, V_{vr} and D_{vr} define the number of prevented violations and deviations, respectively, when vehicle v performs the rebalancing actions in route r. The parameter R_{vr} represents the reward that service vehicle v obtains when visiting the last station in route r after the planning horizon. The MP is now given by:

$$\min \quad \omega^V \left(V - \sum_{v \in V} \sum_{r \in R_v} V_{vr} \lambda_{vr} \right) + \omega^D \left(D - \sum_{v \in V} \sum_{r \in R_v} D_{vr} \lambda_{vr} \right) - \omega^R \left(\sum_{v \in V} \sum_{r \in R_v} R_{vr} \lambda_{vr} \right) \tag{11a}$$

$$\sum_{r \in \mathcal{R}_v} \lambda_{vr} = 1 \quad v \in \mathcal{V} \tag{11b}$$

$$\sum_{v \in \mathcal{V}} \sum_{r \in \mathcal{R}_v} A_{ivr} \lambda_{vr} \leq 1 \qquad i \in \mathcal{S} \tag{11c}$$

$$\lambda_{vr} \in \{0, 1\} \quad v \in \mathcal{V}, r \in \mathcal{R}_v \tag{11d}$$

The objective is defined in (11a), consisting of the weighted sum of violations, deviations and rewards for visiting stations after the planning horizon. This objective function value is equivalent to the value obtained from objective (1) if the geographical routes, loads and arrival times are identical. Constraints (11b) state that each service vehicle must drive exactly one route, constraints (11c) restrict the stations from having more than one visit each during the planning horizon, while (11d) are the binary restrictions.

4.2. Initialization - route extension algorithm

As the set of columns can be very large, the aim of the initialization step is to generate a subset of good initial columns. To generate these columns we propose a route extension algorithm that considers specific problem characteristics. The main elements of this algorithm are: (1) the estimation of *loading quantities*; (2) the *filtering* of stations that can be added to a route and (3) the calculation of a *station criticality*, with each element having a dedicated subsection. Routes can be visualized using a tree, where each path in the tree represents one geographical route, while each node represents a station visit. Fig. 4 shows a small example of a route extension tree for a service vehicle starting at station 1. In this example, four different routes are generated, i.e., the routes (1,10,2), (1,10,4), (1,12,5) and (1,12,8).

The route extension algorithm is summarized by Algorithm 1, which is being used for both variants of the CGH. New routes are created by extending routes that are under construction with the most promising stations. The number of extensions depends on a branching constant B, which is defined as the maximum number of branches created from each node. In the example in Fig. 4, $B = 2$. When the duration of a route exceeds the planning horizon, we add the route to the set of finished routes and remove it from the set of partial routes.

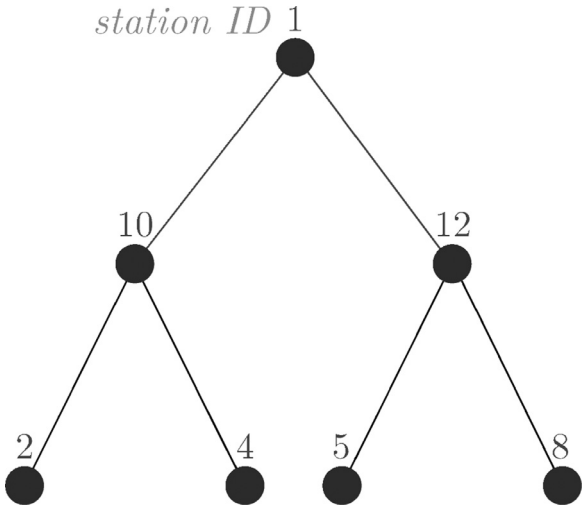


Fig. 4. A route extension tree.

Data: S := First station visit; $\overline{T}$:= planning horizon;
Result: F: Set of finished routes
1 *R: List with routes under construction initialized with S*
2 **while** *size of* $R > 0$ **do**
3 **for** *each route r in R* **do**
4 Estimate loading quantities and arrival times
5 **if** *duration of* $r < \overline{T}$ **then**
6 Determine the subset S^R of stations that can be added to route r
7 Calculate the station criticality for each station in subset S^R
8 Create up to B new routes by extending r with the stations with highest station criticality and insert these into R
9 **else**
10 Add r to F
11 **end**
12 Remove r from R
13 **end**
14 **end**

Algorithm 1: Route extension algorithm.

4.2.1. Loading quantity

The *Pattern-based* variant of the MP requires the loading quantity patterns to be predefined in the columns. We use a greedy heuristic to estimate the loading quantities at each station visit in

a route. Given a (partial) route r for vehicle v, we iterate through the station visits in chronological order. If vehicle v goes from station i to j in route r we set the (un)loading quantities as follows:

$$q^L_{iv} = \min\left\{Q^V - l^V_{iv}, l_i, Q^{0.5}_{ij}\right\} \quad \text{if } i \in \mathcal{S}^L,$$
$$q^U_{iv} = \min\left\{Q_i - l_i, l^V_{iv}, Q^{0.5}_{ij}\right\} \quad \text{if } i \in \mathcal{S}^U,$$

where $Q^{0.5}_{ij}$ equals half the capacity of the service vehicle if station visits i and j are of the same type (pickup/delivery), and the full capacity (Q^V) otherwise. So, we (un)load as much as we can, unless the subsequent station visit is of the same type. When visiting two stations of the same type consecutively, it might happen that the service vehicle is not able to (un)load as much as possible at the second visit, while having extra capacity during the first. In this case we could have (un)loaded more during the first visit. In these situations, the algorithm performs a *regret*. The regret function takes the loading algorithm two steps back, and the loading quantities at the two last stations are re-estimated. This time, $Q^{0.5}_{ij}$ is set to half the service vehicle's capacity plus the remaining service vehicle load or available slots. If a node in the route extension tree is part of several routes, different loading quantities are set for each route. All loading quantities are updated each time a route is extended. So, the loading quantities are set chronologically for a route, starting with the first station visit.

4.2.2. Filtering

We filter the set of stations that can be added to a route based on domain knowledge, leading to a subset $\mathcal{S}^R$. First, we consider the case when a partial route consists of only one station visit. If the first visit was a delivery and the service vehicle's initial load is lower than some threshold value, then we infer that the second visit must be a pickup station. A similar logic is applied in the case where the first station being a pickup station and the service vehicle's load being larger than some threshold value. Second, if a partial route contains at least two station visits, we do filtering based on the last two visits in the route. As the service vehicles have a limited capacity, we assume that it is inefficient to visit more than two pickup or two delivery stations consecutively. Thus, if the two last station visits are pickup stations, all pickup stations are filtered out of $\mathcal{S}^R$ for the next visit, and the other way around. Visiting multiple stations of the same type typically leads to increased costs due to, e.g., extra driving and handling time. Even though allowing for more than two consecutive visits to pickup or delivery stations can be incorporated in the approach, we believe it leads to a more complicated modelling approach with little potential to improve the solutions.

4.2.3. Station criticality

Before the branching algorithm picks a new station to add to an existing route, each station in subset S^R is given a station criticality γ_i that expresses the importance of visiting the station. The station criticality is meant to capture the benefit of visiting that station and is defined as the weighted sum of (1) time to violation (t^V); (2) net demand during the planning horizon (D_i); (3) driving time from the previous station j (T^D_{ji}); and (4) deviation from the target state, given that the station is not visited during the planning horizon ($\overline{d}^*_i$):

$$\gamma_i = -\omega^1 t^V + \omega^2 D_i - \omega^3 T^D_{ji} + \omega^4 \overline{d}^*_i,$$

where:

$$t^V = \begin{cases} (Q_i - l_i)/D_i & i \in \mathcal{S}^L \\ -l_i/D_i & i \in \mathcal{S}^U. \end{cases}$$

4.3. Scoring problem

The aim of the heuristic scoring problem (SP) is to explore more of the search space and identify promising columns that were omitted in the initialization. After having solved the MP for a subset of columns, the SP identifies stations that (1) are not visited in the current solution of the master problem, and (2) contribute the most to reducing violations and deviations. The SP then creates new columns where these stations have a greater chance of being visited. These new columns are constructed based on information from the most recent MP solution (violations and deviations of stations that are not included in the MP solution), but they are not based on dual prices from the MP.

The SP algorithm is summarized in Algorithm 2, where N^S is the number of scoring iterations, σ_i is a score for violations and deviations at station i, ω^S is the corresponding weight, B^S is a branching constant, and β is a Bernoulli random variable equaling one with probability p^S. In short, we extract the violations and deviations from the stations that are not visited in the solution of the MP. These violations and deviations are scaled to obtain a score. We then generate new columns using the route extension algorithm, branching constant B^S and an updated station criticality. We add the score to the existing station criticality with a probability of $p^S \cdot 100\%$. This probabilistic approach is used to prevent columns from only containing stations that were not visited before. When new columns have been generated for all service vehicles, the MP is re-solved.

```
F := list with new columns
Number of iterations = 0
while Number of iterations < N^S do
    Read results from MP
    σ_i := 0, for all stations i
    S^S := Set of stations not included in MP solution
    for each station i ∈ S^S do
        Calculate the scores as a weighted sum of the
        violations and deviations: σ_i = ω^S(c̄_i + s̄_i + d̄_i)
    end
    for each service vehicle do
        F ← Generate new columns through Algorithm 1 with
        branching constant B^S and station criticality
        γ̃_i = γ_i + βσ_i
    end
    Number of iterations ++
    Execute MP
end
```

Algorithm 2: Scoring problem algorithm.

5. Simulation environment

As the DDBRS assumes known demand, real-world uncertainties are not explicitly taken into account. Hence, a discrete-event simulation framework is developed to test how our modelling and solution approach performs in a realistic setting. In this section, we describe how the simulation framework works.

The simulation framework iterates between a simulator and subproblem in a rolling horizon fashion, as illustrated in Fig. 5. When solving the subproblem, we obtain rebalancing decisions for a certain time period that includes information about the routes, arrival times and loading quantities for the different service vehicles. The simulator takes this as input and tracks the evolvement of the system until the next decision epoch, which can be whenever substantial new information becomes available. It is typically done when a service vehicle reaches a new station or after a given time

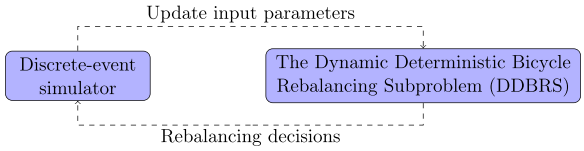


Fig. 5. Simulation framework: iterative process between simulator and the DDBRS.

interval. The simulator then provides updated information about the state of the system, which again gives new input parameters to the subproblem, and the problem is solved to obtain new rebalancing decisions.

Fig. 6 illustrates three iterations within the simulation framework for a case with one service vehicle and two stations. In line a), we visualize a complete customer arrival *scenario*, which consists of all customers arrivals (both lock and bicycle requests) during a particular planning horizon for the simulation. This information can be sampled before the actual simulation. Then, the information describing the system state at the start of the planning horizon is sent to the DDBRS. In line b), the subproblem is solved for a planning horizon of one hour, and the decisions about service vehicles' routes, loading quantities and arrival times are passed back to the simulator. The system is now simulated until route resimulation is triggered by the event that the service vehicle visits a new station (which happens at 8:45am). The information describing the system state at 8:45am is sent to the subproblem and a new route is generated in line c). Again, we proceed the simulation, using the arrival scenario and the current route, until the final route-generation is triggered that is visualized in line d). The simulation is terminated when we pass the stopping time for the scenario.

While the subproblem uses a linear approximation of customer demand, the simulator aims to model the customer demand as realistic as possible through the above mentioned demand scenarios. These customer demand scenarios are generated in the following way. For each hour and station, the simulator draws a random number of requests for bicycles and locks. We use log-normal distributions for these demands, and round the obtained value to the nearest integer. We calculate the mean and standard deviation of the demand for locks and bicycles based on historical customer demand data for each hour and station. When stations are congested or starved, extrapolation is used to estimate the true demand. Given a number of demand requests per hour, we then randomly distribute these requests within the considered time period.

In Fig. 6, we simulate a customer arrivals scenario from 8:00am to 10:00am for a system with two stations. For the first hour, the sampled total number of customers requesting a bicycle and a lock equals three and one respectively. In the second hour, three requests for locks are sampled, and zero for bicycles. The arrival times are then sampled randomly.

6. Computational study

This section presents computational results for the CGH. We start in Section 6.1 by defining the case study and the data used in the testing. Next, we test the performance of the CGH on the DDBRS in Section 6.2, before we test its performance on the DBRP described in Section 6.3, which is the actual problem of interest. The latter is done by using the CGH in a rolling horizon setting within the simulation framework described in Section 5. Finally, we do a number of tests to obtain some managerial insights in Section 6.4.

All tests are performed on a PC with an Intel Core i7-6700 CPU @ 3.40 gigahertz processor with 32 GB RAM, running Windows 10. The master problems of the CGH were implemented in Mosel 4.6.0 and solved with Xpress Optimizer 8.3. The initialization heuristic, scoring problem and simulation framework have been programmed with Java in the IntelliJ programming development environment.

6.1. Case study

We test the developed framework on real-life data from the BSS in Oslo, Norway. The data is provided by the company Urban Sharing AS, who operate the system. The BSS in Oslo currently utilizes up to five service vehicles, each with a capacity of 23 bicycles. Moreover, at the time of the study, there were 158 stations and around 1790 bicycles in the system, which corresponds to approximately half the number of locks in the system. We assume a linear relation between the handling time and number of bicycles handled. Based on the actual performed rebalancing operations in November 2017 we estimate the handling time per bicycle to 0.25 minutes, while the parking time is set to 2 minutes. Driving times are based on Google Maps' open API using the station coordinates. The demand for bicycles and locks at each station are both assumed to follow a log-normal distribution and are estimated based on historical hourly demand patterns for bicycles and locks during weekdays from July 2017 to September 2017. As the historical data only contains information about the customers that actually picked up or delivered a bicycle, we perform extrapolation to estimate the real demand. While this gives us the total number of requests during an hour, we use a uniform distribution to determine when during the hour these requests take place. This assumption is supported by the observation that for the BSS in Oslo the demand does not vary a lot within specific hours, although it changes throughout the day. The target state for each station and time period is defined to be the point where the probability of congestion equals the probability of starvation, which is in line with the approach used by our case company.

6.2. Computational results for the DDBRS

To test the performance of the two versions of the CGH, we compare their performance in Table 1 with using a commercial MIP-solver (Xpress) that solves the full model presented in Section 3. The presented results are for single runs of the subproblem with a planning horizon of 20 minutes (no simulation). The tests are performed on eight different instances, varying in size from eight to 158 stations. Note that the considered time periods represent rush hours, either in the morning or in the afternoon, so that for example instance "3_50_5pm" represents instance number 3, having 50 stations for a time period ranging from 5:00pm to 5:20pm. We perform the analyses for two and five service vehicles, maintaining the same demand. As the DDBRS has to be solved online in practice, we require its solution time to be at most 10 seconds. While the solutions obtained for the two column generation heuristics satisfy this requirement, solving the subproblem exactly using a commercial solver takes a lot more time. The presented objective function values for the commercial MIP-solver (*Exact*) are obtained after running the model for 200 seconds.

To obtain the results in Table 1 we have performed extensive parameter tuning on the main parameters such as the branching constants and the objective-, scoring- and criticality-weights, while complying to the 10 seconds solution time requirement. An overview of the tuned parameters is given in Appendix B. We observed that the key parameter was the branching constant in the route extension algorithm. A higher value allows for a more extensive solution space but also leads to increased solution times. In the final configuration, the route extension algorithm in the initialization procedure managed to generate good columns. We

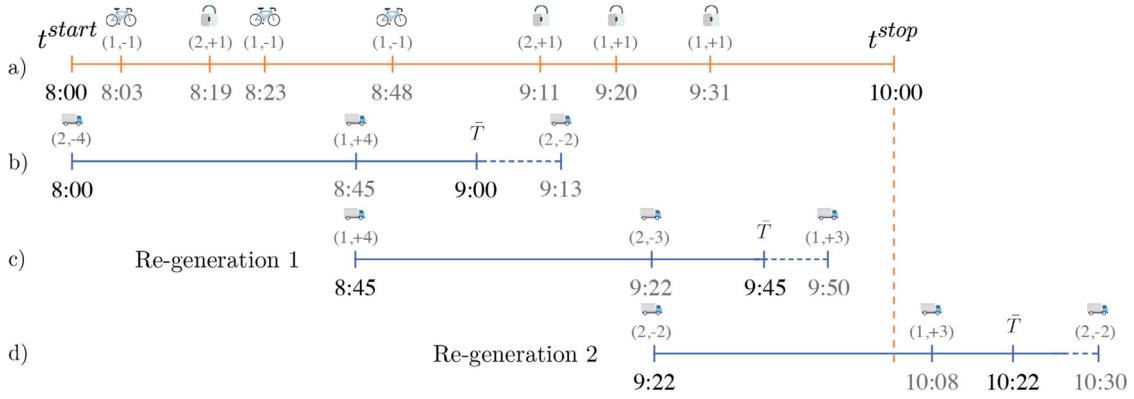


Fig. 6. Simulation framework: flow-chart for a case with two stations and one vehicle. The capacity of both stations and the vehicle is five bicycles. The pairs denote (station ID, bicycle/lock demand) in line a) and (station ID, number of bicycles loaded) in line b) through d).

Table 1
Computational results on the DDBRS for the *Route-based (RB)* and *Pattern-based (PB)* heuristic approach and the commercial MIP-solver (*Exact*).

	Objective function value (Optimality gap in brackets.)					
	$V = 2$			$V = 5$		
Instance	*Exact*	*RB*	*PB*	*Exact*	*RB*	*PB*
1_8_7am	55.67 (0.00%)	55.67	56.34	39.01 (0.00%)	44.75	50.93
2_8_5pm	18.01 (0.00%)	18.01	27.15	11.68 (0.00%)	12.35	20.66
3_50_7am	220.76 (86.14%)	213.31	214.74	No solution	195.97	193.38
4_50_5pm	169.95 (89.09%)	158.81	167.83	153.13 (85.48%)	141.98	144.41
5_100_7am	No solution	389.36	391.03	No solution	372.14	370.97
6_100_5pm	No solution	239.25	240.11	No solution	224.95	224.08
7_158_7am	No solution	536.90	535.56	No solution	521.47	515.48
8_158_5pm	No solution	362.77	369.39	No solution	341.61	345.45

therefore only use one or two scoring iterations to generate new columns.

Turning to the results in Table 1, we observe that for the two smallest instances, the MIP-solver (*Exact*) finds the optimal solutions. Nevertheless, the *Route-based* CGH variant obtains solutions that are optimal or close to optimal within a much shorter amount of time, while *Pattern-based* performs somewhat worse.

For the larger instances, we observe that the commercial MIP-solver is not able to find any feasible solutions within the time limit, while *Route-based* and *Pattern-based* have similar performance.

6.3. Simulated results for the DBRP

Even though the results in Section 6.2 show that the CGH has a good performance on the DDBRS, it is not obvious that this also means that that it performs well on the DBRP, which is the problem we are really interested in solving. So, to test the performance on the DBRP, we simulate the largest instance with 158 stations and five service vehicles (unless stated differently), which corresponds to the real BSS in Oslo. We simulate over the four hour time period of 7am – 11am. At the beginning of the simulation period at 7am, the locations of the bicycles are initialized with historic data, while the vehicles are randomly distributed in the system. The planning horizon of the subproblem (i.e., the DDBRS) is fixed to 20 minutes. In the following analyses we present average numbers based on simulations of 10 different demand scenarios over the four-hour time period.

We start by comparing the performance of *Route-based* and *Pattern-based* in the simulation framework. Table 2 compares the average number of violations for both versions of the CGH.

Table 2
Average number of violations over 10 demand scenarios for the two heuristic approaches. The last column presents the *p*-value for a *t*-test comparing the two averages.

	Route-based	*Pattern-based*	*p*-value
$V = 2$	1562.4	1547.3	0.035
$V = 5$	1316.5	1129.0	2×10^{-6}

Assuming a significance level of $\alpha = 0.05$, we conclude that *Pattern-based* outperforms *Route-based* when considering two or five service vehicles. This contrasts some of the results presented in Table 1, where *Route-based* and *Pattern-based* had fairly similar performance for the larger instances.

By examining the solutions generated with *Route-based* and *Pattern-based*, we observe that within the four hour time period each service vehicle visits on average 22.1 and 25.5 stations, respectively, whereas the average loading quantities are 11.9 and 11.4, respectively. It appears that *Pattern-based* seems to generate better geographical routes as more station visits are completed within the time period, which might be a result from the higher branching constant.

Fig. 7 illustrates the generated routes for the first hour of the simulation for a service vehicle starting at station 35 when routes are re-generated (i.e., the DDBRS is solved) every time the vehile arrives at a station. We are mainly interested in the loading quantity at the current station, as well as the planned station to visit next, as these are only the decisions that will be executed with certainty. These decisions are visualized in the red boxes. Moving ahead, and focusing on the loading quantity at the first planned station visits, we observe that this quantity remains unchanged

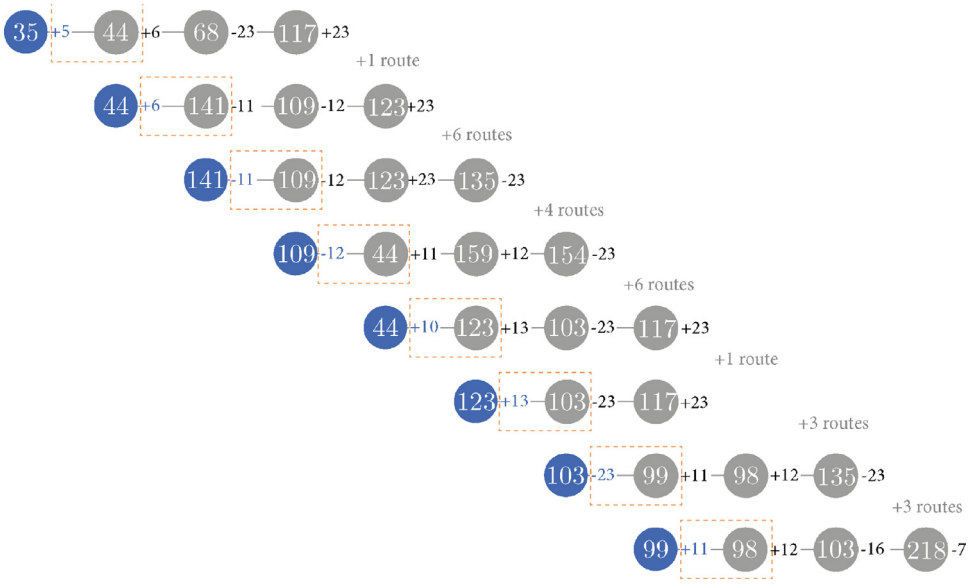


Fig. 7. Sequence of generated routes for route re-generation at the station visits. Completed station visits are marked in blue, loading quantities are to the right of the station ID number and total number of additional generated routes for the other vehicles are given in grey. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Average number of violations for different combinations of the subproblems planning horizon and route re-generation point (first station visit, third station visit, fixed 10 minute time interval). The last column presents the computational time in seconds of running a single subproblem.

$\overline{T}$	Route re-generation			Comp. time (seconds)
	first	third	10 minutes	
10	1261	1299	1370	0.28
20	1129	1155	1225	3
30	1104	1096	1122	11.1

compared to the actual execution in all re-generated routes, except on the fourth line. However, the second planned station visit changes in four out of seven cases, indicating that frequent re-planning is beneficial.

Table 3 presents the average number of simulated violations for three different planning horizons (i.e., 10, 20 and 30 minutes) of the subproblem (DDBRS) and three different route re-generation strategies (re-planning every time a service vehicle arrives at a station, every third time a vehicle arrives at a station, and with a fixed time interval of 10 minutes).

We observe that frequent re-planning leads to a reduction in the number of violations, except when the planning horizon for the subproblem is 30 minutes and re-planning occurs every third time a service vehicle arrives at a station. The number of violations also decreases when the planning horizon of the DDBRS is prolonged, indicating that having short planning horizon can be too myopic. However, this goes at the cost of the computational performance. For a planning horizon spanning thirty minutes, the computational time for each subproblem exceeds the 10 seconds requirement. Henceforth, we will in the following tests use a planning horizon of 20 minutes in the subproblem and perform route re-generation every time a vehicle visits a station.

6.4. Managerial insights

In this section we present some managerial insights obtained from a number of additional tests.

6.4.1. Comparison with current planning practice

To further test the performance of the CGH, we compare it with the current planning practice of the case company. The current rebalancing strategy does not utilize any analytic program to determine routes for the service vehicles. However, the service vehicle operators are equipped with tablets showing a real-time overview of the distribution of bicycles in the system and they make decisions based on some simple decision rules. The overarching strategy behind these decision rules is that each service vehicle is assigned two zones (which can vary over a day) that can be categorized as a delivery zone and a pickup zone. To gain experience with the current rebalancing strategy, we participated in the daily operations and observed that, amongst others, loading quantities tend to correspond to either half the service vehicle's capacity or the entire vehicle's capacity; operators strive to balance several stations partially, rather than balancing a few stations entirely; operators visit at most two delivery stations or two pickup stations consecutively; and whether a pickup station or delivery station is visited depends on the service vehicle's load after the current visit.

We have translated the observed rebalancing decision rules into a heuristic procedure. To make loading decisions on the current station, this heuristic uses the loading quantity algorithm from Section 4.2.1, though without the regret function as we experienced that future loading decisions were not considered. To determine the next station, we use the following logic. If there are many bicycles on the vehicle after the current visit, then we visit either one or two delivery stations. If the vehicle load is almost empty, then we visit one or two pick-up stations. For all other vehicle load levels, the next station can either be a pickup or delivery station. The exact stations to be visited are then based on the station criticality as described in Section 4.2.3 as well as the presence of stations in the pre-defined zones. The weights in the station criticality reflect the information that the operator has available. In particular, the operator puts less weight on the driving time and deviation. In real-life, operators do not calculate these criticalities and therefore do not always select the most critical station. Thus, the implementation of the current rebalancing method chooses a station randomly among the five stations with the highest station

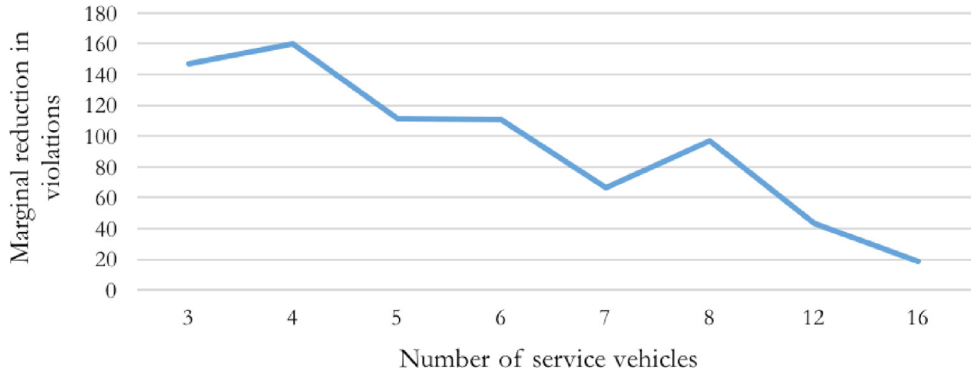


Fig. 8. Marginal reduction in violations as a function of number of service vehicles.

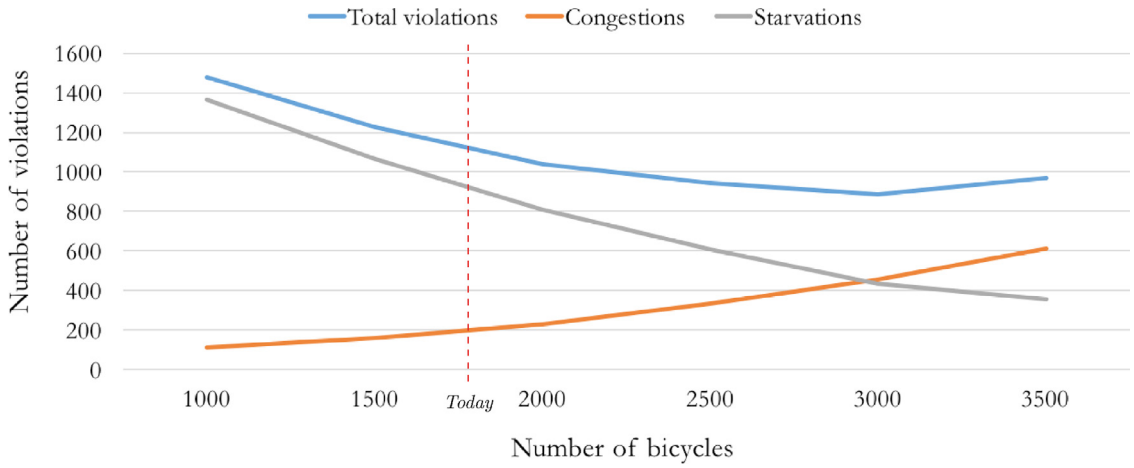


Fig. 9. Violations as a function of number of bicycles in the system.

Table 4
Average number of violations for different rebalancing strategies.

CGH Pattern-based	Current method	No rebalancing	Num requests
1129	1482	1916	8476

criticality. Here we note that the chosen station must be in the zone of the service vehicle.

Table 4 now compares the CGH with the current planning strategy, as well as with the strategy of not providing any rebalancing at all.

We observe that there is a significant difference between the performance of the CGH and the two other strategies. Compared with current planning practice, the average number of violations during a four hour period can be reduced by 24% with the same number of service vehicles and employees. When comparing with the no rebalancing case, the reduction equals 41%.

6.4.2. The value of service vehicles, bicycles and geo-fencing

The use of additional service vehicles and bicycles can lead to a reduction in the number of violations in the system. This means that the BSS operator should aim to strike a balance between the number of violations and the cost of operating the system. As such, Fig. 8 presents the marginal reduction of the average number of violations for a varying number of service vehicles. The marginal reduction is defined as the number of violations that can be avoided compared with having one less service vehicle. We observe, as expected, a downward trend, which implies that the added value of additional vehicles reduces for larger fleets.

In addition, Fig. 9 presents simulation results for a varying number of bicycles in the system. The system under consideration contains 1790 bicycles and 3580 locks. The placement of bicycles in the system is done by scaling the initial distribution at 7am. We observe that the curve for the total number of violations as a function of the number of bicycles has a convex shape. At first, the number of violations decreases when adding bicycles, but as the number of bicycles gets closer to the total capacity (number of locks), the violations start to increase. Moreover, the lowest number of violations is achieved when the number of congestions and starvations are equal, which corresponds with the definition of the target state used by our case partner. While the current number of bicycles in the system does not lead to the lowest number of violations, there might be a trade-off with having a lower number of congestions. However, when station capacities can be increased (e.g., by means of geo-fencing) it might be beneficial with extra bicycles.

A geo-fenced area is an area around a station in which bicycles can be locked even if there are no physical locks available. This allows more bicycles to be parked and can increase the fixed station capacities. In turn, geo-fencing can mitigate the issue of congestion at stations, i.e., when there are no available locks. Fig. 10 presents the results of a simulation where the stations' capacities are multiplied by different capacity factors. Factors larger than 1 indicate an increase in capacity due to geo-fencing compared with the current system. We see that the number of congestions converges to zero

Fig. 10. The value of geo-fencing and different capacity factors.

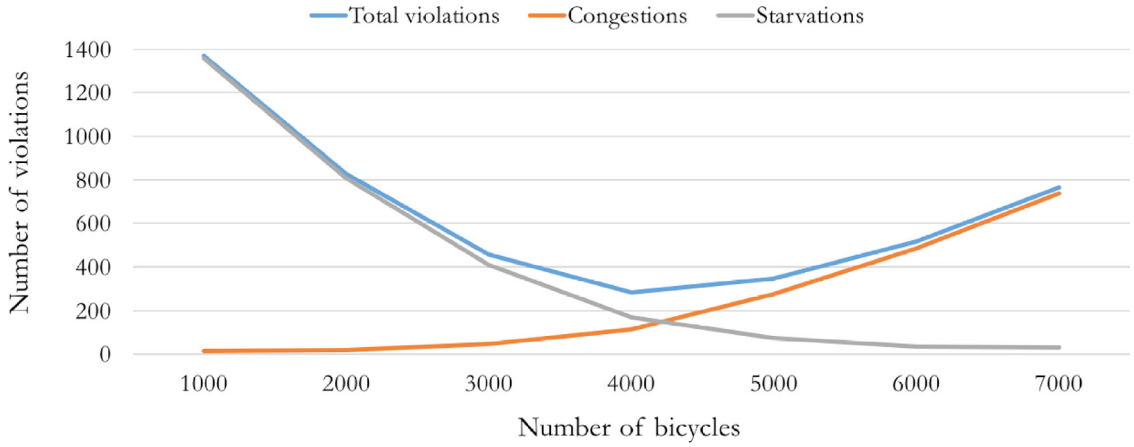


Fig. 11. Violations as a function of number of bicycles in the system when geo-fencing is enabled.

when increasing the capacity. For a capacity factor of 4, there are no more congestions. However, the lowest number of violations is achieved for a capacity factor equaling 2. This is because too large station capacities might result in more imbalances in the system with more starvations.

Finally, we investigate the case where we allow for geo-fencing with a capacity factor of 2 and vary the number of bicycles. Fig. 11 illustrates the corresponding results. We observe that the combination of geo-fencing and a larger number of bicycles in the system can be highly efficient in terms of the average number of violations. This set-up can achieve approximately 280 violations for 4000 bicycles, while today's system leads to 1129 violations with 1790 bicycles. These benefits should be compared with the added cost of introducing additional bicycles into the BSS.

7. Conclusions

In this paper we have considered the dynamic bicycle rebalancing problem (DBRP). To be able to solve instances of realistic size, we approximate the DBRP through a set of smaller subproblems with known customer demand that are solved in a rolling horizon fashion. The subproblems have a shorter planning horizon than the original problem and are referred to as dynamic deterministic bicycle rebalancing subproblems (DDBRS). To prevent myopic decisions, the DDBRS penalizes deviations from the target state and forces vehicles to start on trips that exceed the planning horizon. Within the considered planning horizon of the subproblem, a vehicle can visit a fair number of stations. This leads to a too large number of routes to be solved efficiently. Therefore, to solve instances of realistic size, we develop a column generation heuristic (CGH). The CGH starts with an initialization procedure that generates initial routes using a route extension algorithm. It then iterates between a master problem that picks the optimal routes for each vehicle; and a scoring problem that generates new improving routes. We investigate two versions of the CGH that differ in the definition of columns, which either only include information about the geographical route (*Route-based*), or additionally also information on loading patterns (*Pattern-based*). It is critical that solution methods to the subproblem can be executed in an online fashion. Our proposed CGH provides good solutions with an upper bound on the solution time of 10 seconds. A balance between solution quality and computational performance can be struck by correctly setting the length of the planning horizon as well as the branching constant.

To test the performance of the CGH in a realistic setting with real world uncertainties, we developed a discrete-event simulator and constructed a case study that uses real-life data from the BSS in Oslo, Norway, provided by our industry partner. We think that it is crucial to perform the computational tests on such a simulator, since the DDBRS is only part of the actual problem of interest, i.e., the DBRP. For the large instances, we observed that *Pattern-based* outperformed *Route-based* in the simulation framework, while this was not visible when testing on an isolated subproblem. Our explanation is that with *Pattern-based* a higher branching constant can

be set, leading to good geographical routes. *Route-based* requires a lower branching constant to adhere to the 10 seconds solution time requirement. The improvement in the objective function due to potentially better loading and unloading decisions cannot offset the disadvantage of considering fewer routes.

We use the developed simulation framework to obtain additional operational insights. Focusing on the rolling horizon framework, we observe that frequent route-regeneration leads to the lowest number of expected violations. Extending the planning horizon of the subproblems also leads to a decrease in the number of expected violations, but increases the computational time. An operator should therefore aim to strike a balance between the route re-generation frequency, planning horizon length and computational time. Comparing the CGH approach with not doing any rebalancing at all, we find that the total number of expected violations during a four hour period can be reduced by 41%. When comparing the performance of the CGH with the current planning practice, we find that the expected violations can be reduced by 24%, using the same number of service vehicles. This is one of the reasons that the case company now is making preparations to implement an optimization-based decision support system based on the CGH proposed in this paper. Harvesting these potential gains by performing rebalancing using the proposed CGH can be crucial for obtaining good functioning BSSs, as customer satisfaction is key. Finally, we find that the combination of a high number of bicycles in the system, together with a modest increase in station capacity (possibly due to geo-fencing) can lead to significant violation reductions.

CRediT authorship contribution statement

Marte D. Gleditsch: Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft. **Kristine Hagen:** Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft. **Henrik Andersson:** Conceptualization, Methodology, Supervision, Funding acquisition, Writing – original draft. **Steffen J. Bakker:** Conceptualization, Methodology, Visualization, Writing – original draft. **Kjetil Fagerholt:** Conceptualization, Methodology, Supervision, Funding acquisition, Writing – original draft.

Acknowledgements

This paper was prepared as a part of the FOMO - Future of Micromobility - project, funded by the Research Council of Norway through the Transport 2025 program (p-nr: 309654). We are also grateful to Urban Sharing, a company that is a partner in FOMO and operates BSSs in a number of European cities, for providing data and insight about the problem considered in this paper.

Appendix A. Nomenclature

Table A.5 presents the notation used in this paper.

Table A5
Notation used for modelling the DDBRS .

Sets	
$\mathcal{S}$	Set of stations
$\mathcal{S}^L, \mathcal{S}^U \subseteq \mathcal{S}$	Sets of pick-up (loading) and delivery (unloading) stations
$\mathcal{V}$	Set of service vehicles
Indices	
i, j	Station $i, j \in \mathcal{S}$
o, d	artificial origin and destination nodes
v	Service vehicle $v \in \mathcal{V}$

(continued on next column)

Table A5
(continued)

Parameters	
D_i	Net customer demand at station i
I	Loading constant used to assign vehicles to pickup or delivery stations
L_v^V	Initial load of bicycles at service vehicle v
L_i^0	Initial load of bicycles at station i
$\overline{L}_i$	Target state / inventory level at time $\overline{T}$
Q^V	Storage capacity at service vehicle v
Q_i	Capacity at station i
$\overline{T}$	Planning horizon
T_{ij}^D	Driving time between station i and j
T^H	Unit handling time used for picking up or delivering bicycles
Binary variables	
x_{ijv}	1 if service vehicle v drives from station i to j, 0 otherwise
y_v^L, y_v^U	1 if the load at vehicle v is lower than (or above) a certain threshold value at time $\overline{T}$
z_i	1 if station i gets a visit before $\overline{T}$
Integer variables	
q_{iv}^L	Number of bicycles loaded from station i by service vehicle v
q_{iv}^U	Number of bicycles unloaded to a station i by service vehicle v
Continuous variables	
c_i	Congestion when visiting station i
$\overline{c}_i$	Congestion for station i at time $\overline{T}$
d_i	Deviation at station i at time $\overline{T}$
l_i	Inventory level (Bicycle load) at station i at *start* of visit
$\overline{l}_i$	Inventory level at station i at time $\overline{T}$
l_{iv}^V	Bicycle load of service vehicle v just *after* visit i
$\overline{l}_v^V$	Inventory level of vehicle v at time $\overline{T}$
r_i	Reward for visiting station i after the planning horizon
s_i	Starvation when visiting station i
$\overline{s}_i$	Starvation for station i at time $\overline{T}$
t_i	Time station visit i begins
t_v^R	Extra driving time after the planning horizon

Appendix B. Parameter tuning

To determine the values of the different parameters and weights in the model, we tested the performance of different configurations in the developed simulation framework. For the weights, we considered values from 0 to 1, with 0.1 increments. The other parameters were tested with unit increments. The final configuration of the parameters is given in Table B.6.

Table B6
Final configuration of model parameters for the *Route-based* and *Pattern-based* CGH.

Parameter description			Values	
			Route	*Pattern*
Subproblem	Planning horizon (minutes)	$\overline{T}$	20	
Objective weights	Violations	ω^V	0.6	
	Deviations	ω^D	0.3	
	Reward	ω^R	0.1	
	Reward, prevented violations	ω^{R+}	0.06	
	Reward, extra driving	ω^{R-}	0.04	
Route extension	Branching Constant	B	2	20
Scoring	Branching Constant	B^S	3	15
	Num scoring iterations	N^S	1	2
	Scoring weight	ω^S	4	4
	Scoring Probability	p^S	0.4	0.4
Criticality weights	Time to violation		0.1	0.1
	Net demand		0.7	0.5
	Driving time		0	0
	Deviation		0.2	0.4

References

Brinkmann, J., Ulmer, M. W., & Mattfeld, D. C. (2019). Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems. *Computers and Operations Research, 106*, 260–279. https://doi.org/10.1016/j.cor.2018.06.004.

Brinkmann, J., Ulmer, M. W., & Mattfeld, D. C. (2020). The multi-vehicle stochastic-dynamic inventory routing problem for bike sharing systems. *Business Research, 13*(1), 69–92. https://doi.org/10.1007/s40685-019-0100-z.

Bulhões, T., Subramanian, A., Erdoğan, G., & Laporte, G. (2018). The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research, 264*(2), 508–523. https://doi.org/10.1016/j.ejor.2017.06.028.

Chemla, D., Meunier, F., & Wolfler Calvo, R. (2013). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization, 10*(2), 120–146. https://doi.org/10.1016/j.disopt.2012.11.005.

Contardo, C., Morency, C., & Rousseau, L.-M. (2012). Balancing a dynamic public bike-sharing system. *Technical report*. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf

Dell'Amico, M., Iori, M., Novellani, S., & Stützle, T. (2016). A destroy and repair algorithm for the bike sharing rebalancing problem. *Computers and Operations Research, 71*, 149–162. https://doi.org/10.1016/j.cor.2016.01.011.

Erdogan, G., Battarra, M., & Wolfler Calvo, R. (2015). An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research, 245*(3), 667–679. https://doi.org/10.1016/j.ejor.2015.03.043.

Erdoğan, G., Laporte, G., & Wolfler Calvo, R. (2014). The static bicycle relocation problem with demand intervals. *European Journal of Operational Research, 238*(2), 451–457. https://doi.org/10.1016/j.ejor.2014.04.013.

Espegren, H. M., Kristianslund, J., Andersson, H., & Fagerholt, K. (2016). The static bicycle repositioning problem - literature survey and new formulation. In A. Paias, M. Ruthmair, & S. Voß (Eds.), *Computational logistics. ICCL 2016*. In *Lecture notes in computer science: vol. 9855* (pp. V–VI). Cham: Springer. https://doi.org/10.1007/978-3-319-44896-1.

Fu, C., Zhu, N., Ma, S., & Liu, R. (2021). A two-stage robust approach to integrated station location and rebalancing vehicle service design in bike-sharing systems. *European Journal of Operational Research, 298*(3). https://doi.org/10.1016/j.ejor.2021.06.014.

Gammelli, D., Wang, Y., Prak, D., Rodrigues, F., Minner, S., & Pereira, F. C. (2022). Predictive and prescriptive performance of bike-sharing demand forecasts for inventory management. *Transportation Research Part C: Emerging Technologies, 138*(March), 103571. https://doi.org/10.1016/j.trc.2022.103571.

Ghosh, S., Varakantham, P., Adulyasak, Y., & Jaillet, P. (2017). Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research, 58*, 387–430. https://doi.org/10.1613/jair.5308.

Ho, S. C., & Szeto, W. Y. (2017). A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. *Transportation Research Part B: Methodological, 95*, 340–363. https://doi.org/10.1016/J.TRB.2016.11.003.

Kaspi, M., Raviv, T., & Ulmer, M. W. (2022). Directions for future research on urban mobility and city logistics. *Networks, 79*(3), 253–263. https://doi.org/10.1002/net.22092.

Laporte, G., Meunier, F., & Wolfler Calvo, R. (2018). Shared mobility systems: An updated survey. *Annals of Operations Research, 271*(1), 105–126. https://doi.org/10.1007/s10479-018-3076-8.

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. https://doi.org/10.1016/j.ejor.2018.06.051.

Maggioni, F., Cagnolari, M., Bertazzi, L., & Wallace, S. W. (2019). Stochastic optimization models for a bike-sharing problem with transshipment. *European Journal of Operational Research, 276*(1), 272–283. https://doi.org/10.1016/j.ejor.2018.12.031.

Nair, R., Miller-Hooks, E., Hampshire, R. C., & Bušić, A. (2012). Large-scale vehicle sharing systems: Analysis of Vélib'. *International Journal of Sustainable Transportation, 7*(1), 85–106. https://doi.org/10.1080/15568318.2012.660115.

Pfrommer, J., Warrington, J., Schildbach, G., & Morari, M. (2014). Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems, 15*(4), 1567–1578. https://doi.org/10.1109/TITS.2014.2303986.

Powell, W. B. (2007). Approximate dynamic programming: Solving the curses of dimensionality. *Wiley Series in Probability and Statistics* (2nd ed.). USA: Wiley-Interscience. https://doi.org/10.1002/9780470182963.

Rainer-Harbach, M., Papazek, P., Raidl, G. R., Hu, B., & Kloimüllner, C. (2015). PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization, 63*(3), 597–629. https://doi.org/10.1007/s10898-014-0147-5.

Raviv, T., Tzur, M., & Forma, I. A. (2013). Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics, 2*(3), 187–229. https://doi.org/10.1007/s13676-012-0017-6.

Regue, R., & Recker, W. (2014). Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review, 72*, 192–209. https://doi.org/10.1016/j.tre.2014.10.005.

Schuijbroek, J., Hampshire, R. C., & van Hoeve, W. J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research, 257*(3), 992–1004. https://doi.org/10.1016/j.ejor.2016.08.029.

Shu, J., Chou, M. C., Liu, Q., Teo, C. P., & Wang, I. L. (2013). Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems. *Operations Research, 61*(6), 1346–1359. https://doi.org/10.1287/opre.2013.1215.

Shui, C. S., & Szeto, W. Y. (2018). Dynamic green bike repositioning problem—A hybrid rolling horizon artificial bee colony algorithm approach. *Transportation Research Part D: Transport and Environment, 60*, 119–136. https://doi.org/10.1016/j.trd.2017.06.023.

Shui, C. S., & Szeto, W. Y. (2020). A review of bicycle-sharing service planning problems. *Transportation Research Part C: Emerging Technologies, 117*(June), 102648. https://doi.org/10.1016/j.trc.2020.102648.

The Meddin Bike-sharing World Map (2021). Mid-2021 report. *Technical report*. https://bikesharingworldmap.com/reports/bswm_mid2021report.pdf

Vadseth, S. T., Andersson, H., & Stålhane, M. (2021). An iterative matheuristic for the inventory routing problem. *Computers and Operations Research, 131*(October 2020), 105262. https://doi.org/10.1016/j.cor.2021.105262.

Wang, Y., & Szeto, W. Y. (2021). The dynamic bike repositioning problem with battery electric vehicles and multiple charging technologies. *Transportation Research Part C: Emerging Technologies, 131*. https://doi.org/10.1016/j.trc.2021.103327.

Yuan, Y., Cattaruzza, D., Ogier, M., Semet, F., & Vigo, D. (2021). A column generation based heuristic for the generalized vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review, 152*, 102391. https://doi.org/10.1016/J.TRE.2021.102391.

Zhang, D., Yu, C., Desai, J., Lau, H. Y., & Srivathsan, S. (2017). A time-space network flow approach to dynamic repositioning in bicycle sharing systems. *Transportation Research Part B: Methodological, 103*, 188–207. https://doi.org/10.1016/j.trb.2016.12.006.