CIRRELT

CIRRELT-2024-26

An Integer L-Shaped Method for the Static Stochastic Bicycle Repositioning Problem

Lucas Parada
Jean-François Côté
Michel Gendreau

July 2024

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2024-003

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

An Integer L-Shaped Method for the Static Stochastic Bicycle Repositioning Problem

Lucas Parada[1,2], Jean-François Côté[1,3,*], Michel Gendreau[1,2]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
2. Department of Mathematical and Industrial Engineering, Polytechnique Montréal
3. Department of Operations and Decision Systems, Université Laval, Québec, Canada

Abstract. This work deals with the static stochastic bicycle repositioning problem by proposing new formulations, an exact solution method, new valid inequalities, and new lower bounds. The problem is NP-hard, and it arises in the context of bicycle-sharing systems that need to ensure quality of service. The latter is understood as the availability of bicycles and docks to park them in a network of stations. Each station has a random request that can be in the form of either pickups or deliveries of bicycles. The static nature of the problem is because none of the input parameters change during the solution process, and stochasticity is given exclusively in the form of random variables for the station requests. The problem is formulated as a variant of the stochastic vehicle routing problem, and we propose an implementation of the integer L-shaped method to solve it. To speed up the resolution, we create customized lower-bounding functionals based on a complexity result we developed for the recourse problem. The results of computational experiments show that our implementation is superior to those in the literature, which leads us to generate and propose a new, challenging benchmark set of instances.

Keywords: Bicycle sharing systems, bicycle rebalancing, stochastic programming, integer L-shaped method, vehicle routing problem.

Acknowledgements. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) [grants 2021-04037 and 2019-00094]. This support is greatly acknowledged. We thank the Digital Research Alliance of Canada for providing high-performance parallel computing facilities.

---

* Corresponding author: jean-francois.cote@fsa.ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2024

1 Introduction

The static stochastic bicycle repositioning problem (SBRP) arises in bicycle sharing systems (BSS), a particular case of shared mobility systems when the means of transport is a set of bicycles (Laporte et al., 2018). A BSS typically comprises a network of stations with docks (to park the bicycles), requests for bicycles at each station, and a capacitated vehicle fleet to reposition them by picking them up or delivering them. These requests are not known in advance, and they are revealed when a vehicle arrives at a station. There is also a centralized depot from which bicycles can be picked up or delivered. To be an effective means of transportation, the BSS must ensure the quality of service, which is achieved by making available both bicycles and docks (Datner et al., 2019). The problem is to determine a set of repositioning decisions for the fleet of vehicles to follow such that requests are exactly or partially satisfied, incurring a non-negative penalty in the latter case. Following the definitions of Pillac et al. (2013) for static and dynamic stochastic processes, we characterize this repositioning problem as static because it involves data with partially known values represented as random variables whose realizations are only revealed during the visit to the stations.

The SBRP, which has received little attention from the literature on exact methods, is an NP-hard problem, as it can be seen to generalize stochastic vehicle routing problems with recourse by imposing the conditions that each station be visited exactly once, and all vehicle routes must start and end at the depot. The few exact methods that have been implemented for the SBRP have relied on decomposing the problem into simpler sub-problems by mathematical programming techniques. However, these same methods have not taken into account successful techniques that speed up convergence to the optimal solution, like the lower bounding functionals (LBFs) of Birge and Wets (1986). Without such techniques, numerical results show that considerable computational times are required to obtain good feasible solutions for the SBRP, and few optimal solutions have been reported.

Recently, the disaggregated integer L-shaped (DL-shaped) method was proposed by Parada et al. (2024) as an efficient method to solve stochastic vehicle routing problems, and this paper proposes an implementation of the method to solve the SBRP. The method decomposes solutions into a finite number of disjoint components and then builds on the integer L-shaped method of Laporte and Louveaux (1993) to introduce different and more efficient LBFs and optimality cuts. These new LBFs are linear inequalities built from lower bounds of the recourse value of a component, which help to bind the recourse for a large number of solutions (Laporte et al., 2002; Jabali et al., 2014). Adding such inequalities in the search process has been shown to reduce the number of enumerated solutions (Côté et al., 2020). The suitability of the DL-shaped method for the SBRP arises from how the latter is formulated as a stochastic vehicle routing problem, with the a priori paradigm of Bertsimas et al. (1990). This formulation minimizes the vehicle travel cost plus the expected recourse value. The recourse value is a linear program that minimizes penalties for partially

unsatisfied requests, subject to a set of bicycle flow constraints in a capacitated network.

The contributions of this paper are as follows. First, we propose a new dynamic programming (DP) formulation to compute the recourse value. This new formulation has been shown to be very efficient. Next, we propose an implementation of the DL-shaped method for the SBRP, which uses new valid inequalities and new recourse lower bounds. These bounds were inspired by the 1-bicycle repositioning problem, a variant of the recourse problem that does not require a set of a priori routes. We demonstrate that this variant is NP-hard to solve. Lastly, we propose a new set of challenging instances that require several vehicles, unlike the existing benchmark set where the majority of instances need a single vehicle.

The rest of the paper is structured as follows. A literature review is presented in Section 2. The problem is mathematically formulated in Section 3. Section 4 provides new lower bounds for the fleet size and for the recourse. Section 5 provides the theoretical and practical aspects behind the DL-shaped method. Section 6 describes valid inequalities from the literature and new ones that are used by the DL-shaped method, and Section 7 details the implementation of the method. Section 8 presents the computational results of our study. Lastly, we present our conclusions in Section 9.

2 Literature Review

This section conducts a literature review on the SBRP, structured as follows. First, a historical retrospective of BSS is presented. Second, the diverse combinatorial optimization problems that are typically encountered and typically manifest in BSS settings are shown. Third, the particular BSS addressed in this article is characterized by the reference to where it was proposed. Finally, this review highlights the contributions of the cited reference and expands on the superior performance achieved by our proposed approach.

Bike-sharing systems (BSS) have a rich history of nearly 50 years, embodying the pursuit of convenient and low-emission transportation options through a blend of public and private policies (DeMaio, 2009). Over the past few decades, these systems have experienced remarkable growth, with some expanding to encompass thousands of bicycles and hundreds of stations, such as the BSS in Montréal, Canada (Bixi-Montreal, 2024). Furthermore, recent trends suggest that the worldwide market size for BSS will continue to grow and nearly triple its market valuation by the end of the 2020s (Yahoo-Finance, 2023). However, managing the quality of service in these large-scale systems presents a complex challenge (Alvarez-Valdes et al., 2016). The significance of maintaining adequate BSS performance extends beyond the systems themselves, as they often operate as integral components within broader multi-modal and integrated transportation networks (Kuo et al., 2023). For a comprehensive review of contemporary BSS studies, as well as broader mobility systems, interested readers are encouraged to explore the works of Shui and Szeto (2020) and Todd et al.

(2021).

Diverse combinatorial optimization problems arise in BSS due to significant variations in their designs and features, tailored to meet the specific needs of users and cities. For instance, some BSS adopt free-floating bicycles, allowing users the convenience of picking up and dropping off bikes anywhere in the service area for spontaneous trips. Such free-floating BSS gives rise to problems involving Markov decision processes (Luo et al., 2022). In contrast, station-based BSS have fixed docking stations where bikes must be returned after use, giving rise to inventory routing problems and vehicle routing problem variants (Brinkmann et al., 2016; Erdoğan et al., 2014). Multi-modal integration with public transit networks further enhances user experiences, enabling a seamless combination of bikes with other transportation modes and resulting in multi-layer network design problems for the different modes (Kuo et al., 2023; Crainic et al., 2022). Similar issues arise when integrating electric scooters into BSS, as their usage differs from bicycles, often serving for short trips in high-employment areas (Caspi et al., 2020). Another category of problems occurs when BSS design prohibits the repositioning vehicle fleet from using stations for temporary bicycle storage and transshipment (Bruck et al., 2019) or when a limit is imposed on the number of times that a bicycle can be loaded and unloaded from a repositioning vehicle (Chemla et al., 2013). In all these problems, the objective depends on whether a repositioning time is considered, with goals typically centered on minimizing the total time for bicycle repositioning or routing costs of the repositioning fleet. Additional objectives often seek to minimize penalties for unsatisfied or partially satisfied demand at stations (Raviv et al., 2013). Furthermore, BSS features may include mobile apps, user accounts, and memberships, providing exclusive benefits and discounted rates to encourage desired user behaviors, such as bicycle returns to specific locations or adherence to designated routes. Such features give rise to pricing problems in BSS, where the objective aims to maximize utility or profit (Haider et al., 2018).

This article deals with a station-based BSS without additional features and a vehicle routing problem with stochastic pickups and deliveries that arises in order to manage and reposition bicycles in the many stations. The problem was proposed in Dell'Amico et al. (2018), and the objective is to minimize the repositioning time while allowing transshipment between stations. Furthermore, a two-stage program is formulated where each feasible first-stage solution is a set of a priori routes, and the second stage sees the realization of a finite number of scenarios where, in each one, the unsatisfied demands of the stations in the a priori routes must be minimized. To solve the problem, the authors begin by computing an initial upper bound from a heuristic, which is based on the nearest neighbor and constructive search of Toth and Vigo (2014). A novel aspect is introduced by developing insertion and deletion operators that take into account the correlations between pairs of stations with different types of requests (one pickup and the other delivery, or vice versa). Using the initial upper bound, the authors implement five different exact algorithms, including separation algorithms for subtour inequalities, Bender's feasibility, and optimality cuts. This methodology is implemented for a set of 22 instances whose optimal solutions

require one vehicle for the majority of the cases.

The SBRP, as introduced by Dell'Amico et al. (2018), extends the problem initially proposed by Hernández-Pérez and Salazar-González (2004), known as the one-commodity pickup and delivery traveling salesman problem (1-PDTSP). Alternatively, one can view the SBRP as closely related to the single-commodity vehicle routing problem with split pickups and deliveries (SPDVRP) outlined by Li et al. (2023). The main difference with the latter two is in the fleet size and number of visits allowed to the stations. For the 1-PDTSP, exactly one vehicle needs to be used to visit each station exactly once while in the SPDVRP, multiple station visits are allowed by a fleet of vehicles. In the SBRP, a fleet of vehicles is available while maintaining the constraint that each station be visited exactly once. By considering different features besides fleet size and number of station visits, many other closely related problems to the SBRP arise. We refer the interested reader to Table 1 in Li et al. (2023) for further details on these closely related problems.

The primary contribution of Dell'Amico et al. (2018) lies in their design and implementation of the exact approaches for the problem. In this article, we extend their work by developing enhanced exact approaches. Notably, we compare the authors' most efficient exact approach, the so-called Multicut algorithm, with our own methods and demonstrate the superior performance of the latter. We show that our proposed approaches achieve more optimal solutions, exhibit better average gaps, and accomplish these results in a fraction of the computational time required by the Multicut algorithm.

3 Mathematical formulation

This section presents the notation that will be used throughout this paper, as well as a mathematical formulation of the problem. Section 3.1 provides the definition of the recourse, and Section 3.2 presents a novel DP formulation to compute the recourse. Section 3.3 proposes a generalization of the recourse and shows the complexity of this new problem.

The SBRP is defined as follows. Let $G = (V, A)$ be a directed graph where $V = \{0, 1, .., n\}$ is the set of nodes with node 0 being the depot. $N = \{1, ..., n\}$ are the station nodes, and $A = \{(i, j) \in V \times V : i \neq j\}$ is the set of arcs. Each arc $(i, j) \in A$ has an associated travel time c_{ij}. Each station $i \in N$ has a maximum capacity of H_i bicycles as well as a desired balanced level d_i (with $H_i \geq d_i$). The random station requests are modeled via a vector (q_i^ξ) for each station i and scenario ξ in the finite set of all scenarios Ξ. q_i^ξ can take positive or negative values to denote pickups or deliveries respectively. Each scenario ξ occurs with a given probability p^ξ such that $\sum_{\xi \in \Xi} p^\xi = 1$. The scenario request needs to be exactly or partially satisfied, in the latter case incurring a penalty cost of Δ for each unsatisfied bicycle. These partially unsatisfied bicycles cannot exceed the quantities $w_i^+ = \min\{\lceil \delta H_i \rceil, H_i - d_i\}$ and $w_i^- = \min\{\lceil \delta H_i \rceil, d_i\}$, with δ given as an input parameter, and w_i^+, w_i^- denoting the maximum unsatisfied pickups and deliveries at station i.

Let x_{ij} be a binary variable that indicates whether arc $(i,j) \in A$ is traversed or not, and z be a strictly positive integer variable denoting the number of vehicles to be used. Let $\mathcal{Q}(x)$ denote the total expected recourse cost of the solution x. The model follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \mathcal{Q}(x) \tag{1}$$

$$\text{s. to} \sum_{i\in V\setminus\{0\}} x_{0i} = z, \tag{2}$$

$$\sum_{i\in V} x_{ij} = 1 \qquad \forall j \in N, \tag{3}$$

$$\sum_{i\in V} x_{ji} = 1 \qquad \forall j \in N, \tag{4}$$

$$x(S) \leq |S| - 1 \qquad S \subseteq N, |S| \geq 2, \tag{5}$$

$$x(P) \leq |P| - 1 \qquad P \in P^{inf}, \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in A, \tag{7}$$

$$z \in \mathbb{Z}^+. \tag{8}$$

The objective in (1) is to minimize total travel time plus the total expected recourse cost. Constraints (2)–(4) are connectivity constraints. Constraints (5) are subtour elimination constraints where $x(S) = \sum_{i,j\in S} x_{ij}$. For computational efficiency, 2-cycles are directly added to the model, whereas 3-cycles and above are dynamically added. Constraints (6) are infeasible path constraints that are dynamically added in the resolution, where P denotes a sequence of stations (a path) of the form $P = (i_1, \ldots, i_j, \ldots, i_t),\ i_j \in N,\ t \leq n$, $x(P) = \sum_{j=1}^{t-1} x_{j(j+1)}$, and P^{inf} is the set of all infeasible paths. Lastly, equations (7)–(8) define the domain and the nature of the variables.

3.1 Recourse Calculation

The recourse can be calculated as follows. For a given solution x^ν, let $\mathcal{R}^\nu$ be the set of routes, where each one starts and ends at the depot. Then, the expected recourse cost is defined as:

$$\mathcal{Q}(x^\nu) = \sum_{r\in\mathcal{R}^\nu} \mathcal{Q}(r), \tag{9}$$

where $\mathcal{Q}(r)$ is the cost of the expected number of unsatisfied bicycles in route $r = (0, i_1, \ldots, i_t, 0)$ and is defined as follows:

$$\mathcal{Q}(r) = \Delta \sum_{\xi\in\Xi} p^\xi \mathcal{Q}(\xi, r). \tag{10}$$

$\mathcal{Q}(\xi, r)$ is the number of unsatisfied bicycles for route r and scenario ξ and can be calculated by solving the following linear program proposed by Dell'Amico et al. (2018). Let $N(r)$ be the set of stations of route r, and $\pi^+(i)$ be the successor of station i in route r. The number of bicycles inside the vehicle is represented by the variable f_i^ξ, while the bicycles that were not picked up are counted by the variable $w_i^{\xi+}$, and those that were not delivered are counted in $w_i^{\xi-}$. The linear program follows:

$$\mathcal{Q}(\xi, r) = \min \sum_{i \in N(r)} (w_i^{\xi+} + w_i^{\xi-}) \tag{11}$$

$$\text{s. to } f_i^\xi = f_{\pi^+(i)}^\xi + q_i^\xi + w_i^{\xi-} - w_i^{\xi+} \qquad i \in N(r), \tag{12}$$

$$0 \leq w_i^{\xi+} \leq w_i^+ \qquad i \in N(r), \tag{13}$$

$$0 \leq w_i^{\xi-} \leq w_i^- \qquad i \in N(r), \tag{14}$$

$$0 \leq f_i^\xi \leq Q \qquad i \in N(r). \tag{15}$$

The objective (11) minimizes the recourse cost of route r by summing all $w_i^{\xi+}$ and $w_i^{\xi-}$ variables. These excess and shortage variables are bounded in constraints (13) and (14) with $w_i^+ = \min\{\lceil \delta H_i \rceil, H_i - d_i\}$ and $w_i^- = \min\{\lceil \delta H_i \rceil, d_i\}$. Constraints (12) and (15) are flow conservation of bicycles at each station. Throughout this paper, we adopt the convention that the expected recourse cost of a route of stations $\mathcal{Q}(r) = (0, P, 0)$ will be equivalently denoted as $\mathcal{Q}(P)$, with P being a path. Lastly, whenever the program in (11)–(15) is infeasible for a given scenario, an infeasible path inequality (6) is added to the model.

3.2 A New Dynamic Programming Formulation for the Recourse

This section proposes a DP formulation that can solve the recourse problem in pseudo-polynomial time, which turns out to be very efficient in practice.

The recourse problem of route $r = \{0, i_1, \ldots, i_t, 0\}$ and scenario ξ is divided into $t+1$ stages where each stage $k = 1, ..., t+1$ corresponds to a visit to a station following the sequence in r. The requests at the stations are denoted by q_k^ξ and can be positive or negative in the case of a pickup or a delivery. Define the variable $x_k^\xi \in [0, Q]$ as the vehicle load after repositioning operations at stage k, and set $x_{t+1}^\xi = 0$. Additionally, for $k = 1, ..., t$, define the variables $0 \leq w_k^{\xi+} \leq w_k^+$ and $0 \leq w_k^{\xi-} \leq w_k^-$, for the missed pickups and missed deliveries, similar to equations (13) and (14). Let $H_k^\xi(x_k^\xi)$ be the cost function. The DP algorithm follows:

$$H_k^\xi(x_k^\xi) = \min_{\substack{w_k^{\xi-} \in [0, w_k^-] \\ w_k^{\xi+} \in [0, w_k^+] \\ 0 \leq x_k^\xi + w_k^{\xi-} - w_k^{\xi+} \leq Q - q_k^\xi}} \left\{ w_k^{\xi+} + w_k^{\xi-} + H_{k+1}^\xi(q_k^\xi + x_k^\xi + w_k^{\xi-} - w_k^{\xi+}) \right\} \tag{16}$$

The DP sums a each stage the minimum missed pickups or deliveries plus the cost of going forward with residual vehicle capacity $q_k^\xi + x_k^\xi + w_k^{\xi-} - w_k^{\xi+}$. The optimal cost is obtained by $\min_{0 \leq x_0 \leq Q}\{H_0^\xi(x_0)\}$.

Solving this DP algorithm has a pseudo-polynomial time complexity of $O(nQw^*)$, where $w^* = \max_{k=1,\ldots,t}\{w_k^+, w_k^-\}$.

3.3 Complexity

In Parada et al. (2024), the authors demonstrated that the DL-shaped method can be enhanced by incorporating LBFs to gradually bound the second stage cost from below. They also proposed a new type of LBFs that are based on sets of nodes, which are used in conjunction with their optimality cut. The LBF of a given set of nodes requires calculating a lower bound on the recourse. The problem of finding the highest lower bound can be formulated as finding the route that leads to the lowest recourse. Parada et al. (2024) had an easy way to find such a route given the properties of their problem. In this section, we show that finding such a route for the SBRP is an NP-hard problem.

To demonstrate our results, we first introduce a modification of the recourse model from Section 3.1. Suppose we want to determine if a specific set of stations can be satisfied by a single vehicle, without considering the routing cost. This setting resembles the recourse model, with the exception that no routes are provided as input. We refer to this problem as the 1-bicycle repositioning problem (1-BRP).

Consider the following 1-BRP instance. Define $N = \{i_1, \ldots, i_{13}\}$ stations with one scenario and bicycle requests of $\{3, 6, 6, 7, 7, 7, 8, 8, 8, 9, 22, -22, -22\}$, and a vehicle capacity of $Q = 22$. Assume, without loss of generality, that only stations i_1 and i_{11} allow missed bicycles equaling their requested values of 3 and 22. Figure 1 shows a solution to the instance as a sequence of stations starting and ending in the depot. Different colors (grey and red) indicate pickup and delivery requests without missed bicycles, while white denotes stations with missed bicycles. The number of unsatisfied requests is shown by the corresponding w_1^+, w_{11}^+ values. The residual capacity of the vehicle is also displayed above the arcs.

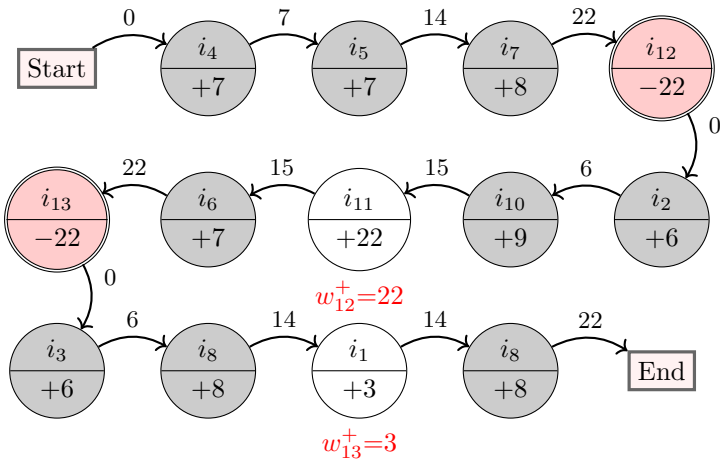

Figure 1: A Solution to the 1-Bicycle Repositioning Problem.

A further examination of Figure 1 shows that the sequence of stations is not unique. Stations with the same number of bicycle requests, such as i_4, i_5, or i_7, can be swapped, creating different sequences. Notably, stations i_1 and i_{11} can be moved anywhere in the sequence, and the resulting number of missed bicycles will remain the same. However, the solution structure remains unique, consisting of groups of three stations with no unmet bicycle requests, totaling 22 requests, followed by a station with a negative request. This unique triplet arrangement can be used to prove the NP-completeness of the 1-BRP by using a transformation from the following problem.

Definition 1. *3-Partition. Let B be a finite set of $3s$ non-negative integers $\{b_1, b_2, ..., b_{3s}\}$ and Q a non-negative integer such that the following two constraints $\sum_{i=1}^{3s} b_i = sQ$ and $\frac{Q}{4} < b_i < \frac{Q}{2}$, for all $1 \leq i \leq 3s$ are satisfied. The goal is to partition B into disjoint sets of triplets, such that the sum of each triplet is Q.*

Proposition 1. *Finding a feasible solution for the 1-BRP with a recourse smaller than or equal to k, where $k \geq 0$ and is an integer, is strongly NP-complete.*

Proof. For a given 3-Partition instance, we can create an instance of the 1-BRP problem in polynomial time using the following transformation. From the $3s$ integers, we generate a set of $4s - 1 + \lceil k/Q \rceil$ stations, where station requests associated with the $3s$ integers are assigned in a way that satisfies the two constraints of 3-Partition. Next, there are $s - 1$ stations with a request of $-Q$. The remaining $\lceil k/Q \rceil$ stations have a request of $k (\mathrm{mod} Q)$ for the first one and Q for the rest. These $\lceil k/Q \rceil$ stations allow maximum missed requests equaling their demand. All the other stations do not allow missed requests. The vehicle capacity is set to the value of Q.

In a feasible 1-BRP solution created from 3-Partition, the stations are arranged so that there are groups of stations consisting of a 3-Partition triplet plus a negative request station. The remaining stations that allow missed requests can be sequenced anywhere, as they will always have maximum missed requests regardless of their position. Therefore, the 1-BRP has a feasible solution if and only if 3-Partition has one, and since 3-Partition is strongly NP-complete the same holds true for the 1-BRP. □

Figure 1 is an example of a recourse of $k = 25$ missed bicycles and a 3-Partition instance $\{6, 6, 7, 7, 7, 8, 8, 8, 9\}$ with $s = 3$. This solution has an equivalent 3-Partition solution given by the three triplets of gray stations.

In Proposition 1, we encounter a decision problem related to the 1-BRP. However, our focus is on finding the minimum recourse sequence, as this would establish a lower bound for the SBRP. The latter is the 1-BRP optimization problem. In other words, while we have a certificate of NP-completeness for the decision problem, we are concerned with the optimization problem's complexity. It's evident that a polynomial algorithm for the optimization problem would also address the decision version, constituting a Turing reduction between the two problems Garey and Johnson (1979). This, coupled with the NP-completeness certificate, serves as proof that the 1-BRP optimization problem is NP-hard.

4 Lower Bounds

This section proposes lower bounds for the fleet size and the expected recourse cost that are used to develop new valid inequalities for the SBRP. Section 4.1 presents the lower bounds for the fleet size, and Section 4.2 presents the lower bounds for the recourse.

4.1 Lower Bounds for the Fleet Size

This section proposes two lower bounds on the fleet size required to serve a set of stations $S \subseteq N$. Section 3.3 demonstrates that finding the least-recourse feasible route to serve a given set S of stations is an NP-hard problem. We thus turn our attention to proposing conditions that determine whether S can be feasibly served or not. These conditions are denoted as feasibility conditions, and the first bound proposed in this section uses them by enumerating fleet sizes starting from one vehicle. The second bound embeds the conditions in a set covering formulation. We begin by proposing the feasibility conditions.

The feasibility conditions build on the following idea. For one scenario, a set of pickup request stations can be paired together with a set of delivery request stations to reduce the unsatisfied request of the fleet that visits those two sets. Assume that these two sets of stations are visited by a fleet of l vehicles, and consequently, the total fleet capacity is lQ. Then, for the set of pickup requests, we determine if the sum of the maximum unsatisfied pickup quantities, the sum of delivery requests, and the fleet capacity is less than the sum of the

pickup requests. If this condition does not hold, it implies that the set S is infeasible. Similarly, for the delivery requests, we evaluate whether the sum of the maximum unsatisfied delivery quantities, the sum of pickup requests, and the fleet capacity is less than the sum of the delivery requests. If this condition is not satisfied, the set S is also deemed infeasible. These two relations for pickups and deliveries are consequently referred to as feasibility conditions, and they are as follows:

$$\sum_{\substack{i \in S \\ q_i^\xi > 0}} q_i^\xi \leq lQ + \sum_{i \in S} w_i^+ + \sum_{\substack{i \in S \\ q_i^\xi < 0}} |q_i^\xi|, \tag{17}$$

$$\sum_{\substack{i \in S \\ q_i^\xi < 0}} |q_i^\xi| \leq lQ + \sum_{i \in S} w_i^- + \sum_{\substack{i \in S \\ q_i^\xi > 0}} q_i^\xi, \tag{18}$$

where equation (17) is the pickup feasibility condition and (18) the delivery feasibility condition. Both conditions need to be satisfied for all scenarios.

In cases where the set S is found to be infeasible based on either the pickup or delivery condition, we increase l until both conditions are satisfied.

The conditions can be used to find a lower bound of the fleet size to satisfy the requests for the entire set of stations N. Let l_N^+, l_N^- be such that:

$$l_N^+ = \min_{l \geq 1} \left\{ l : \sum_{\substack{i \in N \\ q_i^\xi > 0}} q_i^\xi \leq lQ + \sum_{i \in N} w_i^+ + \sum_{\substack{i \in N \\ q_i^\xi < 0}} |q_i^\xi|, \ \forall \xi \in \Xi \right\}, \tag{19}$$

$$l_N^- = \min_{l \geq 1} \left\{ l : \sum_{\substack{i \in N \\ q_i^\xi < 0}} |q_i^\xi| \leq lQ + \sum_{i \in N} w_i^- + \sum_{\substack{i \in N \\ q_i^\xi > 0}} q_i^\xi, \ \forall \xi \in \Xi \right\}. \tag{20}$$

After computing equations (19), (20) for all scenarios, the following lower bound K_0 on the fleet size arises:

$$K_0 = \max\{l_N^+, l_N^-\}. \tag{21}$$

The second lower bound on the fleet size proposed in this section is given by the following set covering formulation. Let $\mathcal{K}$ be a collection of sets of stations that satisfy conditions (17) and (18) for one vehicle:

$$\mathcal{K} = \left\{ S \subseteq N \mid \sum_{\substack{i \in S \\ q_i^\xi < 0}} |q_i^\xi| \leq Q + \sum_{i \in S} w_i^- + \sum_{\substack{i \in S \\ q_i^\xi > 0}} q_i^\xi, \text{ and} \right.$$

$$\sum_{\substack{i\in S\\ q_i^\xi>0}} q_i^\xi \leq Q + \sum_{i\in S} w_i^+ + \sum_{\substack{i\in S\\ q_i^\xi<0}} |q_i^\xi|, \xi \in \Xi \Bigg\}.$$

Let ζ_k be a binary variable that takes a value of 1 if set $k \in \mathcal{K}$ is taken and 0 if it is not, and b_{ik} be a parameter taking a value of 1 if station i is in set k or 0 if it is not. The set covering formulation follows:

$$\min \sum_{k\in\mathcal{K}} \zeta_k \tag{22}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} b_{ik}\zeta_k \geq 1 \qquad \forall i \in N, \tag{23}$$

$$0 \leq \zeta_k \leq 1,\ k \in \mathcal{K}. \tag{24}$$

The lower bound on the fleet size is given by K_1, computed as the ceiling of the optimal solution to the problem in (22)–(24).

The size of $\mathcal{K}$ is exponential, and solving (22)–(24) to optimality is impractical. The following column generation (CG) approach is applied. Let α_i be the dual variable associated with the covering constraints (23) and z_i a variable taking a value of 1 whenever station i is in subset k. The pricing problem is as follows:

$$\max \sum_{i\in N} \alpha_i z_i \tag{25}$$

$$\text{s.t.} \sum_{\substack{i\in N\\ q_i^\xi<0}} |q_i^\xi| z_i - \sum_{i\in N} w_i^- z_i - \sum_{\substack{i\in N\\ q_i^\xi>0}} q_i^\xi z_i \leq Q \qquad \xi \in \Xi, \tag{26}$$

$$\sum_{\substack{i\in N\\ q_i^\xi>0}} q_i^\xi z_i - \sum_{i\in N} w_i^+ z_i - \sum_{\substack{i\in N\\ q_i^\xi<0}} |q_i^\xi| z_i \leq Q \qquad \xi \in \Xi, \tag{27}$$

$$z_i \in \{0,1\} \qquad i \in N. \tag{28}$$

The objective (25) maximizes the value of the reduced cost of the entering variable z_i to the set covering problem. Equations (26)–(27) are the feasibility conditions for one vehicle. Equations (28) are variable definitions.

Lastly, we can compare both lower bounds $K_0,\ K_1$ in the following expression:

$$K = \max\{K_0, K_1\}. \tag{29}$$

We set K as the lower bound on the fleet size.

4.2 Lower Bounds for the Recourse

This section proposes two recourse lower bounds for the recourse of the SBRP. The bounds use the results of Section 4.1, namely the fleet size required for a set of stations $S \subseteq N$.

Let $l_S = \max\{l_S^-, l_S^+\}$ be the fleet size required to serve S, given by the feasibility conditions (17) and (18). Then, for a given scenario ξ, a lower bound of the recourse of S is given as follows:

$$L_0(\xi, S, l_S) = \Delta \max\{0, |\sum_{i \in S} q_i^\xi| - l_S Q\}, \tag{30}$$

where the quantity $|\sum_{i \in S} q_i^\xi| - l_S Q$ is the maximum unsatisfied requests in S given that l_S vehicles are minimally required. Next, by setting $S = N$ in equation (30), and considering the fleet size lower bound K, the following lower bound on the recourse arises if K vehicles are used:

$$L_1(K) = \sum_{\xi \in \Xi} p_\xi L_0(\xi, N, K), \tag{31}$$

5 The DL-shaped Method

This section presents the theoretical and practical aspects of the method. First, the recourse function $\mathcal{Q}(x)$ is relaxed and bounded from below by the sum of a set of continuous non-negative variables θ_i defined for each station i. The authors refer to the inclusion of these variables in the model as a way to disaggregate the recourse. A master program is then constructed with the following objective function:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{i \in N} \theta_i. \tag{32}$$

The value of θ_i variables can be viewed as their contribution to the cost of the second stage, and this value is optimized by dynamically adding optimality cuts from Parada et al. (2024), denoted as path cuts (P-cuts). A P-cut is added for each feasible path of stations found during the resolution, and the cut binds the x_{ij} variables of the path to the θ_i variables of the stations in that path. Consequently, a P-cut will be active whenever the corresponding path is active in a solution. The P-cut follows:

$$\sum_{i \in N(P)} \theta_i \geq \mathcal{Q}(P) \left(x(P) - |P| + 1\right), \tag{33}$$

where $N(P)$ is the set of stations in P. In Parada et al. (2024), it was proved that if the recourse function $\mathcal{Q}(x)$ respects the *monotonicity condition*, then the method will yield an optimal solution, if one exists in a finite number of iterations of the method. This property requires that the expected recourse cost of a path P must be no less than that of any path P', where P' is a

subpath of P obtained by the removal of stations at the start or at the end of P. Consequently, P' is denoted as a subpath of P and its definition follows:

Definition 2. *Let $P = (i_1, \ldots, i_t)$ be a path of stations that yields a feasible solution to the recourse problem with respect to equations (11)–(15) for all scenarios. P' is a subpath of P if it can be written as $P' = (i_a, \ldots, i_b)$ for some $1 \leq a \leq b \leq t$, and it also yields a feasible solution to the recourse problem in equations (11)–(15) for all scenarios.*

Next, denote the set of subpaths of path P as $\mathcal{P}$. As Dell'Amico et al. (2018) state, removing the first and/or last stations of a feasible path yields a feasible subpath, and this means that all the subpaths in $\mathcal{P}$ are feasible. We extend the definition of the monotonicity condition of Parada et al. (2024) to the SBRP as follows:

Definition 3. *An instance of the SBRP satisfies the monotonicity condition if, for any feasible path $P \subseteq N$, the following inequality holds:*

$$\mathcal{Q}(P) \geq \mathcal{Q}(P'),\ P' \in \mathcal{P}, \tag{34}$$

Proposition 2 proves that the recourse function of the SBRP respects the monotonicity condition.

Proposition 2. *The SBRP respects the monotonicity condition.*

Proof. The expected recourse cost for both $\mathcal{Q}(P)$ and $\mathcal{Q}(P')$ is computed using the program in equations (11)–(15). As P has at least one additional station compared to P', the program $\mathcal{Q}(P)$ will include one extra flow variable and two penalty variables, all of which are non-negative. Furthermore, the objective in equation (11) is monotonically increasing with respect to the number of $w_i^{\xi+}$ and $w_i^{\xi-}$ variables. This implies that the additional penalty variables in $\mathcal{Q}(P)$ cannot reduce the objective's value compared to the program $\mathcal{Q}(P')$. It follows that the cost $\mathcal{Q}(P)$ is necessarily greater than or equal to $\mathcal{Q}(P')$. □

6 Valid Inequalities

This section builds on the theoretical properties described in Section 4 and on the disaggregation of the recourse in Section 5 to develop new valid inequalities for the SBRP.

6.1 Recourse Lower Bound Inequalities

The DL-shaped method relies on a new type of LBF called set cuts (S-cuts) (Parada et al., 2024) to improve the convergence of the method. In the SBRP, these S-cuts take a set S of stations as input, the minimum number of vehicles required to serve S as $l_S = \max\{l_S^-, l_S^+\}$, and a lower bound on the expected recourse of this set, which we denote as $L(S) = \sum_{\xi \in \Xi} p^\xi L_0(\xi, S, l_S)$. The result

is a cut that bounds the set whenever a path that consecutively visits the stations of S is active in a given solution.

In Section 5, the LBFs were mentioned as a means to accelerate the convergence of the DL-shaped method to the optimal solution. Then, for a set $S \subseteq N$, inequality:

$$\sum_{i \in S} \theta_i \geq L(S) \left(x(S) - |S| + l_S + 1 \right), \tag{35}$$

is a S-cut that is valid for the SBRP. In the inequality, l_S is the minimum number of vehicles required to serve S. We refer to Parada et al. (2024) for proof that the S-cuts are valid inequalities.

Next, the following inequality:

$$\sum_{i \in N} \theta_i \geq L_1(K) - \Delta Q(z - K) \tag{36}$$

is valid for the SBRP. Validity of this inequality arises from the fact that $L_1(K)$ was proposed in Section 4 as a lower bound on the recourse when K vehicles are used to satisfy all the stations in N. If $z > K$, this means that there are at least $Q(z - K)$ additional bicycle requests that can be satisfied. The recourse can be penalized by a factor of Δ times the additional bicycle requests served.

From equation (30), another valid inequality can be developed if all arcs in G are individually considered. Let $\sigma_{ij}^{\xi} = \max\{0, |q_i^{\xi} + q_j^{\xi}| - Q\}$ be the recourse cost associated with traversing arc (i, j) in scenario ξ. Then, the inequality:

$$\sum_{i \in N} \theta_i \geq \Delta \sum_{(i,j) \in A} \sum_{\xi \in \Xi} p^{\xi} \sigma_{ij}^{\xi} x_{ij}, \tag{37}$$

is valid for the SBRP. The validity of this inequality is given by first defining the set $S = \{i, j\}$ for each arc $(i, j) \in A$. In the SBRP, at most one vehicle can traverse each arc, so equation (30) for S reduces to $L_0(\xi, S, 1) = \max\{0, |q_i^{\xi} + q_j^{\xi}| - Q\}$. It follows that any solution using a given set of arcs cannot have a smaller expected recourse cost than the sum of the lower bounds on each arc.

Equations (36) and (37) are added at the root node of the DL-shaped method.

6.2 Infeasible Path Inequalities

The infeasible path inequality (6) is a valid inequality that is dynamically added whenever there is a path P in the resolution that is infeasible for the recourse problem in (11)–(15), for at least one scenario. In this case, we search for the smallest infeasible subpath and add an inequality (6) for it.

6.3 Infeasible Set Inequalities

The conditions defined by equations (17) and (18) can be extended to find sets of stations that do not satisfy either condition. First, suppose that an infeasible set of stations $S \subset N$ is given. From equations (19) and (20), compute $l_S = \max\{l_S^-, l_S^+\}$ as the minimum number of vehicles required to satisfy S. The infeasible set inequality follows:

$$x(S) \leq |S| - l_S, \tag{38}$$

The validity of this inequality is due to the fact that l_S is a lower bound of the fleet required to visit the stations in S. Then, equation (38) takes the form of the classical rounded capacity constraints, which are valid inequalities for variants of the capacitated vehicle routing problem and, in particular, for the SBRP.

There are two ways in which we identify infeasible sets of stations. First, starting from a given route $r = (0, i_1, \ldots, i_t, 0)$, denote set $S = \{i_1, \ldots, i_t\}$ and compute l_S. If l_S is strictly greater than one, the route requires at least one more vehicle, and the set S is infeasible. Second, any set of stations violating a subtour elimination constraint is also an infeasible set.

6.4 Infeasible Arc Inequalities

The infeasible arc inequalities were proposed in Dell'Amico et al. (2018) as a direct consequence of properties taken from the traveling salesman problem with pickups and deliveries, as proposed by Hernández-Pérez and Salazar-González (2004). These properties define a set of infeasible arcs A^{inf}, and the inequalities are as follows:

$$x_{ij} + \sum_{h \in N} x_{jh} \leq 1, \ (i,j) \in A^{inf}, \tag{39}$$

$$\sum_{h \in N} x_{hi} + x_{ij} \leq 1, \ (i,j) \in A^{inf}. \tag{40}$$

We include these inequalities at the root node of our branch and cut framework.

7 Implementations of the DL-shaped Method

To solve the SBRP, we implemented three different variants of the DL-shaped method proposed by Parada et al. (2024). The variants differ based on the choice of one of the three types of optimality cut separation routines that we propose.

The DL-shaped method solves the problem like a branch-and-cut algorithm. A model is created in a mixed-integer solver that takes care of solving the linear relaxations and performing branching to find integer solutions. At each

node of the branch-and-bound tree, the solver calls user-defined methods to identify violated inequalities. In those methods, we search for violated subtour elimination, S-cuts, infeasible sets, infeasible paths, and optimality cuts. The solver initialization includes the following two steps:

a) Compute the lower bounds K_0 and K_1 on the fleet size and compute the expected recourse cost as $L_1(K)$.

b) Build the model with objective (32) and constraints (2)–(8), (36), (37), (39), (40). Provide an upper bound value to the problem if one is given.

For step (b), we provide an upper bound computed by means of the Adaptive Large Neighborhood Search (ALNS) heuristic of Ropke and Pisinger (2006).

Next, upon an incumbent solution x^ν found by the solver, the following four steps are executed:

c) Build a graph $G(x^\nu)$ with all the non-zero valued arcs in x^ν and enumerate all connected components of $G(x^\nu)$.

d) For each connect component S of $G(x^\nu)$, check if it violates subtour inequality (5). If so, compute the minimum number of vehicles required to satisfy S, namely $l_S = \max\{l_S^-, l_S^+\}$ from equations (19) and (20), and add an infeasible set inequality (38) to the model.

e) For each connect component S with a violated subtour or infeasible set inequality, compute recourse lower bound given by (30). If its S-cut (35) is violated, add it to the model.

f) If no violated inequality has been found in steps (c)-(e) and the solution x^ν is integral, check the feasibility of each path of the solution by computing the end load feasibility window of Dell'Amico et al. (2018). Add an infeasible path inequality (6) for each infeasible path.

If no inequality is added in steps (a)-(f) and the solution is integral, then solution x^ν is feasible, and we proceed with one of three optimality cut separation routines:

1. *P&S cuts:* For each route in solution x^ν, we enumerate all subpaths of contiguous stations of length two or more. For each subpath P, check if the set S that is composed of the stations in P violates an S-cut, and if so, add it to the model. Also, compute the recourse cost of subpath P using (16). Check if P violates a P-cut, and if so, add it to the model. We only add the five most violated cuts of each kind.

2. *Benders cuts:* Find a violated Benders according to the equation (18) of Dell'Amico et al. (2018). To do so, we formulate and solve the linear relaxation of the dual of the subproblem given by equations (11)–(15). The cut is provided by the optimal value of the dual variables.

3. *Hybrid cuts:* Find a mixture of violated P&S cuts and Benders cuts. To do so, we invoke the Benders cuts separation routine whenever the solution uses one vehicle, and the P&S cuts separation routine otherwise. This approach was inspired by initial computational experiments that showed the effectiveness of the Benders cut whenever the solution uses exactly one vehicle and the P&S cuts for solutions using multiple vehicles.

The procedure stops when all the nodes from the branch-and-bound tree have been explored. In this case, the incumbent solution is the optimal one.

8 Computational Results

This section presents the results obtained for our implementations of the DL-shaped method in the set of instances of Dell'Amico et al. (2018) and a newly generated set of challenging instances. The DL-shaped method was implemented in C++ with Cplex 12.10, and experiments were run in a CPU with the following characteristics: Intel E5-2683 v4 Broadwell @ 2.1Ghz. A maximum run time of 1 hour was set for each instance.

Section 8.1 describes the existing instances for the SBRP and introduces the new set of instances. In Section 8.2, the results of our DL-shaped implementations are compared with the best results of Dell'Amico et al. (2018). Section 8.3 shows the effectiveness of our new valid inequalities. Section 8.4 presents the results of our vehicle lower bounds K_0 and K_1. Finally, Section 8.5 presents the results for the new set of instances. These latter results are compared with our implementation of the Multicut algorithm of Dell'Amico et al. (2018), the best algorithm of the authors.

We adopt the following formulas to present our results. For an individual instance, the gap percentage is computed as $(UB - LB)/LB \times 100$, where UB and LB represent the upper and lower bounds. When calculating the average gaps for instances within the same class, we use the non-optimal solutions as the denominator. Dell'Amico et al. (2018) use different conventions, so to ensure comparability with their results, we re-calculated their numerical values according to our adopted formulas

8.1 Characterization of the Instances

Dell'Amico et al. (2018) proposed 22 instances that were developed from real-world data gathered from BSS of mostly American cities. Each instance varies from 20 to 100 stations and has $|\Xi| = \{30, 91\}$ scenario realizations of the requests. The authors use $\Delta = \lceil \epsilon c_{min} \rceil$ where ϵ is a parameter and $c_{min} = \min_{(i,j)\in A}\{c_{ij}\}$. The parameters ϵ, δ are each considered to have the following values: $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. With the various parameter combinations, a total of 242 instances are defined across 11 distinct ϵ, δ configurations. The majority of these 242 instances require one vehicle to be solved, and the scenarios for a given instance are generally similar to one another in terms of the station requests.

Our numerical results demonstrate that instances with similar scenario requests and requiring only one vehicle, such as those from Dell'Amico et al. (2018), are easier to solve compared to those with very different scenario requests and multiple vehicles. The main reason why is that the resolution of the latter encounters a higher number of infeasible solutions, necessitating efficient feasibility checks and recourse bounding procedures by the solution algorithm. Inspired by this fact, we generated 100 new instances with different scenario requests and required multiple vehicles.

Specifically, to generate the new set of 100 instances, we placed a given number of $|V|$ stations (including the depot) in a [0,100]×[0,100] area. To obtain a challenging set, station requests were randomly defined in the interval $[-Q, Q]$ within a fixed number of $|\Xi| = 20$ scenarios. Lastly, we fixed $\epsilon = \delta = 0.2$ for this new set.

8.2 Results for Values of ϵ and δ

Results for pairs ϵ, δ are presented in Table 1. The rows show each of the ϵ, δ configurations that were tested (11 in total). Each configuration shows the values of the Multicut algorithm and our values that were obtained by solving with the three types of optimality cuts detailed in Section 5. The first column gives the values of 'δ' or 'ϵ' according to the respective parameter value. When 'δ' varies, $\epsilon = 0.2$ is set, and conversely, when 'ϵ' varies, $\delta = 0.2$ is set. 'Ins' denotes the number of instances for each pair ϵ, δ. 'Opt' denotes the number of optimal solutions found, 'Gap%' denotes the average percentage gap of the non-optimal instances, and 'Time' denotes the average time in seconds to reach either the optimal solution or the best solution found.

		Multicut			P&S cuts			Benders cuts			Hybrid cuts		
δ	Ins	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time
0.0	22	2	203.9	3287.5	7	38.2	2427.3	7	38.2	2521.7	7	38.1	2490.1
0.2	22	11	34.2	1847.9	13	22.3	1436.5	13	22.2	1519.0	13	22.2	1542.4
0.4	22	17	2.8	872.2	18	1.9	939.4	20	3.0	761.9	18	1.5	796.6
0.6	22	20	0.3	476.0	17	0.9	1072.7	19	1.1	922.3	20	1.6	820.1
0.8	22	20	0.3	473.3	16	0.9	1265.2	19	1.1	830.0	19	1.1	921.0
1.0	22	20	0.3	461.3	16	0.9	1303.7	20	1.5	952.2	19	1.1	917.2
Total/Avg.	132	90	40.3	1236.4	87	10.8	1407.5	98	11.2	1251.2	96	10.9	1247.9
ϵ		Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time
0.0	22	13	38.5	1690.1	14	23.9	1249.5	14	23.9	1306.6	14	24.0	1292.3
0.2	22	11	34.2	1847.9	13	22.3	1436.5	13	22.2	1519.0	13	22.2	1542.4
0.4	22	10	32.9	2003.3	12	20.1	1688.1	13	23.4	1609.2	12	20.3	1577.2
0.6	22	10	35.4	2063.2	9	16.9	2080.8	11	19.8	1771.6	10	18.5	1914.6
0.8	22	10	37.2	2220.8	9	18.1	2074.6	10	19.0	2044.4	9	18.5	2032.9
1.0	22	8	33.0	2303.5	9	18.4	2123.6	10	19.7	2094.8	9	18.6	2054.3
Total/Avg.	132	62	35.2	2021.5	66	20.0	1775.5	71	21.3	1724.3	67	20.3	1735.6
Global Tot.	264	152	37.8	1628.9	153	15.4	1591.5	169	16.3	1487.7	163	15.6	1491.8

Table 1: Results for the instances of Dell'Amico et al. (2018)

Our methods are superior when compared to the Multicut algorithm. For

example, for Benders cuts, with respect to the 11 possible parameter pairs, our implementation of the DL-shaped method improves results from the Multicut algorithm in 7 of these parameter pairs. Specifically, the improved parameter pairs are $\epsilon = 0.2,\ \delta = \{0.0, 0.2, 0.4\}$ and $\epsilon = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\},\ \delta = 0.2$.

When we compare the performance of our three types of optimality cuts, the Benders cut performs best, although this is partly owing to our available instance set, which consists mainly of instances requiring one vehicle to solve. To show this, Table 2 presents the results of the DL-shaped method and Multicut algorithm only for the instances whose known solution requires more than one vehicle.

		Multicut			P&S cuts			Benders cuts			Hybrid cuts		
δ	Ins	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time
0.0	6	1	64.6	3054.1	6	0.0	305.1	6	0.0	276.4	6	0.0	264.1
0.2	3	3	0.0	258.8	3	0.0	121.0	3	0.0	37.2	3	0.0	119.2
0.4	1	0	3.3	3600.0	1	0.0	3207.4	1	0.0	3411.3	0	0.0	3590.0
0.6	-	-	-	-	-	-	-	-	-	-	-	-	-
0.8	-	-	-	-	-	-	-	-	-	-	-	-	-
1.0	-	-	-	-	-	-	-	-	-	-	-	-	-
	10	4	22.6	2304.3	10	0.0	1211.2	10	0.0	1241.6	9	0.0	1324.5
ϵ	Ins	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time	Opt	Gap%	Time
0.0	3	3	0.0	27.1	3	0.0	0.4	3	0.0	0.4	3	0.0	0.5
0.2	3	3	0.0	258.8	3	0.0	121.0	3	0.0	37.2	3	0.0	119.2
0.4	3	2	0.5	1275.0	2	1.5	1242.9	3	0.0	576.6	2	1.5	1202.7
0.6	3	3	0.0	220.7	2	2.4	865.9	3	0.0	308.3	2	2.3	1207.3
0.8	3	3	0.0	920.1	2	4.7	900.3	2	1.4	1200.1	2	4.7	1209.5
1.0	2	2	0.0	57.1	2	0.0	268.2	2	0.0	34.9	2	0.0	184.4
Total/Avg.	17	16	0.1	459.8	14	1.4	566.4	16	0.2	359.6	14	1.4	653.9
Global Tot.	27	20	7.6	1074.6	24	1.0	781.3	26	0.2	653.6	23	1.0	877.4

Table 2: Results for only the multi-vehicle instances

The column headers in Table 2 are the same as Table 1, with the exception of a different meaning for the columns 'Ins' and 'Opt'. In Table 2, these columns count the number of instances whose known optimal solution uses more than one vehicle and the number of these latter instances that are optimal for each type of optimality cut or the Multicut algorithm. A value of '-' indicates that no multi-vehicle solution is known for the given ϵ, δ pair.

These multi-vehicle instances are few in comparison to the total number of solved instances (10 and 17 for the different $\epsilon,\ \delta$ pairs in contrast to the 98 and 71 that the Benders cut solve in Table 1), but they show that our three optimality cuts perform similarly, as all three of them achieve a similar number of optimal solutions. Additionally, the superiority of our methods compared to the Multicut algorithm is preserved, as all three optimality cuts achieve significantly more optimal solutions for the different values of δ.

8.3 Effectiveness of the Valid Inequalities

This section analyzes the effectiveness of our proposed valid inequalities. We created eight different configurations of the DL-shaped method to test. They are divided into two groups of 4 configurations each, where the first group uses the Benders cuts and the second one uses the P-cuts. Each configuration can be differentiated by the use or not of the S-cut (35) and by the use or not of the Infeasible Set inequality (38). Results are presented in Table 3 on the 242 instances of Dell'Amico et al. (2018). Each instance was run for a maximum of 5 minutes. Column 'Opt' indicates the number of optimal solutions found by the configuration, 'Gap%' denotes the average percentage gap of the non-optimal instances, 'Time' represents the average time to either the optimal solution or the best one found within the time limit of 5 minutes, and 'Inf.Set', 'Inf.Path', 'S-cuts', 'Benders' and 'P-cut' are the average number of cuts of type of equations (38), (6), (35), Benders optimality and (33).

Optimality Cut: Benders								
S-Cut	Inf. Set	Opt	Gap	Time	Inf. Sets	Inf. Paths	S-Cuts	Benders
Yes	Yes	125	26.0	197.7	321.9	0.0	422.5	35.7
No	Yes	118	28.5	211.7	470.3	0.0	0.0	63.3
Yes	No	116	25.1	215.7	0.0	382.4	356.2	31.3
No	No	111	29.0	220.4	0.0	464.3	0.0	49.5
Optimality Cut: P-cuts								
S-Cut	Inf. Set	Opt	Gap	Time	Inf. Sets	Inf. Paths	S-Cuts	P-Cuts
Yes	Yes	119	23.4	168.6	365.3	0.0	549.7	481.1
No	Yes	105	22.1	184.3	775.6	0.0	0.0	691.0
Yes	No	110	23.6	181.0	0.0	305.0	431.2	363.2
No	No	102	21.5	186.4	0.0	509.3	0.0	186.4

Table 3: Convergence with and without the S-cuts and Infeasible Sets.

As shown in Table 3, the use of S-cuts and infeasible sets accelerates the convergence of the DL-shaped method. Specifically, when employing the Benders optimality cuts, the utilization of S-cuts alongside infeasible sets resulted in a notable improvement in convergence. Instances where both S-cuts and infeasible sets were used showcased a reduced average gap percentage and a higher number of optimal solutions within the 5 minutes compared to runs where these cuts were not utilized. Similarly, in the context of P-cuts, the incorporation of S-cuts and infeasible sets also demonstrated a similar trend. The average gap percentage decreased, and the number of optimal solutions increased notably in instances where these cuts were employed in conjunction.

Lastly, we note that when a route is infeasible, it always seems there is an infeasible set of stations that can be found. This is an exciting result because infeasible set inequalities are stronger than infeasible path inequalities. We also noted this behavior in our other results.

8.4 Results for Lower Bounds on the Fleet Size

This section analyzes the effectiveness of the two new lower bounds K_0 and K_1. To do so, we first present results without these two bounds and we then present the values of K_0 and K_1 for the instances with $\delta = 0.0$, because these are the instances that require the largest fleet sizes in the set of Dell'Amico et al. (2018).

The results with and without the lower bound K are given in Table 4 for the 242 instances of Dell'Amico et al. (2018) with the following columns. Column 'K' indicates the lower bound used, and the columns 'Ins' and 'Opt' count the number of instances and optimal solutions in the set. The column 'Gap%' is the average percentage gap of the non-optimal instances, and the column 'Time' is the average time to either the optimal solution or the best one found within the time limit of 5 minutes.

K	Opt	Gap%	Time
$\max\{K_0, K_1\}$	125	26.0	197.7
1	117	34.6	213.2

Table 4: Effectiveness of the vehicle lower bound K in a 5 minute-run.

Results from Table 4 show that using a good lower bound on the number of vehicles helps solve more instances, reducing computing times and reducing the optimality gap.

We also present results for the case of $\delta = 0.0$ in Table 5. Column 'Opt' indicates with a value of 1 that the instance was solved to optimality and 0 otherwise. The columns 'K_0', and 'K_1' denote the number of vehicles resulting from both lower bounds on fleet size. A value of '-' for the 'K_1' column indicates that the linear relaxation could not be solved during the time limit for the column generation algorithm. The column '#veh.' denotes the number of vehicles used by the best solution found at the end of the method. For the cases where the best solution is non-optimal, the value in the column '#veh.' is an upper bound on the required fleet size. Solutions might exist with fewer vehicles, but these may be more costly.

	Multicut		DL-Shaped			
Instance	#veh.	Opt	K_0	K_1	#veh.	Opt
Reggio Emilia	1	1	1	1	1	1
Washington (20)1	3	0	2	2	2	0
Washington (20)2	6	0	5	8	8	1
Chicago (20)1	8	0	6	6	8	1
Chicago (20)2	7	0	3	5	6	1
Washington (30)1	3	0	3	3	3	1
Washington (30)2	3	1	3	3	3	1
Chicago (30)1	13	0	7	8	13	1
Chicago (30)2	5	0	4	4	4	0
Washington (40)1	7	0	4	4	4	0
Washington (40)2	12	0	7	10	10	0
Chicago (40)1	10	0	4	4	7	0
Chicago (40)2	8	0	3	3	6	0
Washington (50)1	12	0	7	8	8	0
Washington (50)2	17	0	6	11	13	0
Chicago (50)	13	0	6	6	8	0
Washington (66)	4	0	3	3	3	0
Chicago (66)	22	0	10	10	13	0
Washington(80)1	6	0	3	3	4	0
Washington(80)2	33	0	11	-	30	0
Washington(90)	36	0	12	-	33	0
Washington(100)	14	0	7	-	7	0
Total	243	2			194	7

Table 5: Lower Bound Values on Fleet Size for $\epsilon = 0.2$ and $\delta = 0.0$.

The total number of optimal solutions, presented in the last row of the table, is the smallest compared to any other ϵ, δ configuration, making it the most challenging to solve. However, thanks to K_0, K_1, the DL-Shaped algorithm achieves many more optimal solutions than Multicut. Moreover, for many instances, the values of K_0, K_1 are close to the number of vehicles used by the best solution found. Table 5 also indicates that further research is needed in the SBRP for cases where the lower bounds of the fleet size deviate significantly from the best solution found.

8.5 Results for the new instances

This section presents the results for the new set of instances and compares them to our implementation of the Multicut algorithm of Dell'Amico et al. (2018). All of these instances were solved with the pair $\epsilon = \delta = 0.2$, and the results are presented in Tables 6 to 9. Additionally, we also analyze the effectiveness of the valid inequalities and vehicle lower bounds in Tables 8 and 9. To implement the Multicut algorithm, we followed Section 4.3 in the authors' work.

Tables 6 and 7 present the comparison between our implementation of the Multicut algorithm and the variant of our DL-shaped method with P&S cuts as follows. Column 'n' denotes the number of stations while the column '#veh.' denotes the average number of vehicles used by optimal solutions or the best solutions found. Column 'Opt' gives the number of optimal solutions, while column 'Gap%' provides the percentage with the gap of the non-optimal instances.

Column 'Time' shows the average time to either the optimal solution or the best one found within the time limit of 3600 seconds. For the results with the Multicut algorithm, the columns 'Benders Feas.' and 'Benders Opt' denote the average number of Benders feasibility and optimality cuts. For the results with the DL-shaped, the columns 'Inf.Set', 'S-cuts' and 'P-Cuts' denote the average number of cuts of type of equations (38), (35) and (33). A '-' under the 'Opt' header indicates that no optimal solution was found.

Multicut							
n	Ins	#veh.	Opt	Gap%	Time	Bender Feas.	Benders Opt
30	20	2.1	12	20.1	1413.8	1142.2	451.0
40	20	2.1	5	42.8	2834.4	1969.8	422.0
50	20	1.9	3	51.8	3225.8	1342.3	485.0
60	20	1.5	-	71.1	3600.0	1087.8	340.0
70	20	1.3	-	54.4	3600.0	883.9	162.0
Total/Average	100	1.8	20	48.0	2934.8	1285.2	372.0

Table 6: Results for Multicut algorithm in the new instances.

DL-Shaped								
n	Ins	#veh.	Opt	Gap%	Time	Inf. Set	S-Cuts	P-Cuts
30	20	2.6	20	0.0	66.0	100.6	211.8	41.6
40	20	2.5	18	3.3	729.0	279.1	657.2	58.7
50	20	2.4	12	6.8	1759.4	578.6	1330.6	300.4
60	20	2.8	10	8.7	2621.4	632.7	1898.7	377.2
70	20	1.8	2	2.5	3442.2	770.3	1705.5	680.2
Total/Average	100	2.4	62	4.3	1723.6	472.2	1160.8	291.6

Table 7: Results for the new instances.

The DL-shaped method consistently outperformed the Multicut algorithm throughout this new instance set by achieving three times more optimal solutions in, on average, less computation time. Our approach yielded average gaps of approximately 4%, significantly smaller than the Multicut algorithm's average gaps, which were more than ten times larger. Notably, our algorithm was able to solve instances with up to 70 stations, while the Multicut can only solve up to one instance with 50 stations.

This superior performance is attributed to our efficient algorithms for handling infeasible set inequalities and S-cuts. Unlike Multicut, which potentially generates multiple Benders feasibility cuts for each infeasible solution, our method efficiently returns a single cut for the smallest infeasible component found, preventing rapid problem size growth. Moreover, our DL-shaped method's ability to generate S-cuts for both feasible and infeasible solutions enables more effective bounding of the recourse than solely binding it in feasible solutions via Benders cuts.

Next, to analyze the effectiveness of our new valid inequalities and vehicle

lower bounds in the new set of 100 instances we present in Tables 8 and 9 the results of a 5-minute run with each of the new valid inequalities as well as the vehicle lower bound K.

Configuration		Optimality Cut: P-cuts						
S-Cut	Inf. Set	Opt	Gap	Time	Inf. Sets	Inf. Paths	S-Cuts	Benders
Yes	Yes	36	11.3	214.1	572.1	0.0	789.8	163.6
No	Yes	34	7.8	218.3	736.1	0.0	0.0	377.6
Yes	No	27	12.2	236.9	0.0	628.4	737.0	131.4
No	No	31	7.9	227.7	0.0	931.0	0.0	318.4

Table 8: Convergence with and without the S-cuts and Infeasible Sets for the new instances.

K	Opt	Gap%	Time
$\max\{K_0, K_1\}$	36	11.3	214.1
1	32	11.7	225.8

Table 9: Effectiveness of the vehicle lower bound K on the new instances.

Table 8 demonstrates the impact of utilizing the S-cuts and the infeasible sets on convergence. When both are employed, we observe a significant improvement in convergence rates compared to when either one or none are utilized. This is particularly evident in the higher number of optimal solutions achieved, along with reduced average gaps and computation times.

Similarly, Table 9 showcases the effectiveness of the vehicle lower bounds. By incorporating both K_0 and K_1 into the resolution instead of setting $K = 1$, we attain more optimal solutions at lesser average gaps and computation times.

In summary, within this new benchmark set, the utilization of both S-cuts and infeasible sets, along with the incorporation of the vehicle lower bounds, leads to the most favorable outcomes in terms of convergence to optimal solutions, reduced average gaps, and computation times.

9 Conclusions

In this article, we have presented a new formulation, valid inequalities, and lower bounds for the stochastic bicycle repositioning problem. We implemented the disaggregated integer L-shaped method to solve the problem exactly. The idea of this method is to disaggregate the recourse into components and bind each of these components in a dynamic fashion using optimality cuts and lower bounding functionals. A novel aspect of our implementation is that we successfully included a mixture of more than one type of optimality cut. This was the case when we included both path cuts and Benders cuts to produce to so-called Hybrid cuts. Additionally, a key aspect of the method is that it develops

and includes lower bounding functionals to accelerate convergence to the optimal solution. We built our lower bounding functionals using novel recourse lower bounds that we proposed. These bounds were inspired by the 1-bicycle repositioning problem, which has been proven to be NP-hard.

In terms of our findings, we achieve state-of-the-art results for solving a benchmark set, and this leads us to propose a challenging set of instances. Among the factors that contribute to our success are the new dynamic programming formulation for the recourse, the infeasible set inequalities, and the lower bounds on fleet size. On this latter topic, future research should focus on deriving tighter bounds for the fleet size.

10 Acknowledgments

Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grant 2021-04037. We thank Digital Research Alliance of Canada for providing high-performance computing facilities.

References

Alvarez-Valdes, R., Belenguer, J., Benavent, E., Bermudez, J., Muñoz, F., Vercher, E., and Verdejo, F. (2016). Optimizing the level of service quality of a bike-sharing system. *Omega*, 62:163–175.

Bertsimas, D., Jaillet, P., and Odoni, A. (1990). A priori optimization. *Operations Research*, 38(6):1019–1033.

Birge, J. and Wets, R. (1986). *Designing approximation schemes for stochastic optimization problems, in particular for stochastic programs with recourse*, pages 54–102. Springer Berlin Heidelberg.

Bixi-Montreal (2024). BIXI Montréal system size. `https://bixi.com/en/who-we-are`. Accessed: April 8, 2024.

Brinkmann, J., Ulmer, M., and Mattfeld, D. (2016). Inventory routing for bike sharing systems. *Transportation research procedia*, 19:316–327.

Bruck, B. P., Cruz, F., Iori, M., and Subramanian, A. (2019). The static bike sharing rebalancing problem with forbidden temporary operations. *Transportation Science*, 53(3):882–896.

Caspi, O., Smart, M. J., and Noland, R. B. (2020). Spatial associations of dockless shared e-scooter usage. *Transportation Research Part D: Transport and Environment*, 86:102396.

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

Côté, J.-F., Gendreau, M., and Potvin, J.-Y. (2020). The vehicle routing problem with stochastic two-dimensional items. *Transportation Science*, 54(2):453–469.

Crainic, T. G., Gendron, B., and Akhavan Kazemzadeh, M. R. (2022). A taxonomy of multilayer network design and a survey of transportation and telecommunication applications. *European Journal of Operational Research*, 303(1):1–13.

Datner, S., Raviv, T., Tzur, M., and Chemla, D. (2019). Setting inventory levels in a bike sharing network. *Transportation Science*, 53(1):62–76.

Dell'Amico, M., Iori, M., Novellani, S., and Subramanian, A. (2018). The bike sharing rebalancing problem with stochastic demands. *Transportation research part B: methodological*, 118:362–380.

DeMaio, P. (2009). Bike-sharing: History, impacts, models of provision, and future. *Journal of public transportation*, 12(4):3.

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

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability*, volume 174. freeman San Francisco.

Haider, Z., Nikolaev, A., Kang, J. E., and Kwon, C. (2018). Inventory rebalancing through pricing in public bike sharing systems. *European Journal of Operational Research*, 270(1):103–117.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126–139.

Jabali, O., Rei, W., Gendreau, M., and Laporte, G. (2014). Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. *Discrete Applied Mathematics*, 177:121–136.

Kuo, Y.-H., Leung, J. M., and Yan, Y. (2023). Public transport for smart cities: Recent innovations and future challenges. *European Journal of Operational Research*, 306(3):1001–1026.

Laporte, G. and Louveaux, F. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142.

Laporte, G., Louveaux, F., and Van Hamme, L. (2002). An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423.

Laporte, G., Meunier, F., and Calvo, R. W. (2018). Shared mobility systems: an updated survey. *Annals of Operations Research*, 271(1):105–126.

Li, J., Luo, Z., Baldacci, R., Qin, H., and Xu, Z. (2023). A new exact algorithm for single-commodity vehicle routing with split pickups and deliveries. *INFORMS Journal on Computing*, 35(1):31–49.

Luo, X., Li, L., Zhao, L., and Lin, J. (2022). Dynamic intra-cell repositioning in free-floating bike-sharing systems using approximate dynamic programming. *Transportation Science*, 56(4):799–826.

Parada, L., Legault, R., Côté, J.-F., and Gendreau, M. (2024). A disaggregated integer l-shaped method for stochastic vehicle routing problems with monotonic recourse. *European Journal of Operational Research*, 318(2):530–533.

Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

Raviv, T., Tzur, M., and 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.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Shui, C. and Szeto, W. (2020). A review of bicycle-sharing service planning problems. *Transportation Research Part C: Emerging Technologies*, 117:102648.

Todd, J., O'Brien, O., and Cheshire, J. (2021). A global comparison of bicycle sharing systems. *Journal of Transport Geography*, 94:103119.

Toth, P. and Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Yahoo-Finance (2023). Bike sharing market size is projected to reach usd 9.96 billion by 2030. `https://finance.yahoo.com/news/bike-sharing-market-size-projected-144000935.html`. Accessed: April 8, 2024.