Politecnico di Torino

Operational Research: theory and applications

Project report
Bike Sharing Repositioning problem

Professors:

Emilio Leonardi
Edoardo Fadda

Students Group 20:

Akgol Can 274948
Braccio Jacopo 273999
Conforte Francesco 277683
De Santis Paolo 280398
Minardi Andrea 267646
Polidori Brendan David 274990

Academic year 2020/2021

Contents

Abstract

The aim of this work is to study the problem of bike repositioning (BRP) that bike-sharing service providers have to face with. The purpose is to help the management of a fleet of bikes over a set of bike-stations, each with a fixed capacity and stochastic demand. Being the problem very complex, the scope of the project is restricted by the following assumptions: 1. Users pick up a bike at a station and drop them at a different one. Trips happen all at once. 2. Transshipment of bikes among stations is performed at the end of the day, so as to be ready for the next day trips.

The proposed solutions for these problems in this paper are: a two-stage stochastic optimization model based on GUROBI Exact Solver [1] and a heuristic model based on the Progressive Hedging (PH) algorithm [2]. The focus of our work was to find a better solution for bike sharing systems with bigger dimensions than the one shown in the work done by Maggioni et al. in [3].

The main strength of our PH heuristic method lays in the ability to find solutions with hypothetical real scenarios having a large number of bike stations considered in the computation. This, indeed, turned out to be not feasible with the used exact solver.

1 Introduction

In recent years, new paradigms of transports are emerging and contributing to solve many transport related problems that cities face at the moment, such as traffic, accidents, pollution, and high costs associated to the ownership of a private vehicle. One of the new trend in urban mobility is called *micromobility* defined by the International Transport Forum (2020) as "[...] the use of micro-vehicles: vehicles with a mass of no more than 350 kg (771 lb) and a design speed no higher than 45 km/h. This definition limits the vehicle's kinetic energy to 27 kJ, which is one hundred times less than the kinetic energy reached by a compact car at top speed." According to this definition, two different types of micromobility can be defined [4]: firstly, human powered micro-vehicles, such as bicycles and scooter; secondly, electronically powered micro-vehicles such as e-scooter and micro-cars. Being sustainable, and healthy, bicycling regained popularity, going from 17 sharing programs worldwide in 2005 to over 2900 in 2019 [5] and still flourishing in smart cities. Although green and inexpensive, bikesharing systems must be carefully designed: even though they reduce fuel consumption by preventing people from using cars, they require bicycle relocation trucks which also consume fuels and money [6]. This makes the problem of the rebalancing docking stations (relocating bikes from overcrowded station to emptier ones) interesting for studies and widely treated. Relocation can be either dynamic or static. In the dynamic case, rebalancing is performed while the bikesharing systems is being used, minimizing the chance of a user encountering an empty station, having better performance than the static case [7] in which rebalancing is performed when the traffic is low, for example at night. In this paper, we study the latter scenario, in particular a bike-sharing service provider that needs to manage a fleet of bikes over a set of bike-stations, given a fixed capacity, in order to serve the stochastical demand over space and time and minimizing the faced costs.

1.1 Problem Description

As it was considered by Maggioni et al. in [3], a unit procurement cost is paid for each bike assigned to each station at the beginning of the service, by considering one day of operational time frame. The delivery of bikes to bike-stations is assumed to be instantaneous and a unit stock-out cost is paid if the demand exceeds the number of bikes assigned to the docking station. When rebalancing of bike station is performed, a unit transshipment cost is paid at the end of the rentals. When a user arrives at one station, two situations may happen: he could be looking for a bike in an empty station, by facing a shortage of bikes which causes a cost increase, or he may want to park a bike in a full station, by facing an overflow which causes a time waste for the user and then a cost. Hence, our goal is to determine the number of bikes to assign to each bike station at the beginning of the service in order to minimize the total cost, given by the sum of the procurement costs, the expected stock-out costs for unmet demand, the expected time-waste costs for overflow and the expected transshipment costs for repositioning bikes at the end of the service. A more precise description of the problem and how we modelled it can be found in Section 3.

2 Literature Review

Bike sharing systems rapidly gained popularity during the last decade, first in West Europe and East Asia and more recently in North America. DeMaio and Meddin [8] began to document the growth of bike sharing systems since 2007, by maintaining a geographical database that describes the location and the characteristic of all public bike-sharing systems around the world.

The push towards the popularity is given overall by the fact that bike-sharing systems contribute to get sustainability, as highlighted by Macioszek et al. in [9].

Nevertheless, the design and practical operations of bike sharing systems raise a set of new interesting problems to be addressed, like the bike repositioning problem (BRP). According to the classification of Berbeglia et al. [10], the BRP is essentially a many-to-many pickup and delivery problem, and it is first proven to be an NP-hard problem by Benchimol et al. [11].

The BRP has been widely studied, therefore it is possible to find multiple papers where this problem is treated and solved by using many different approaches. First of all, it is important to distinguish the static case, where the state of a station doesn't change while a bike is travelling, from the dynamic case, in which the state of the stations can change at any time. Most literature is limited at the study of the former case, because the latter is clearly more difficult to face.

Focusing only on the static case, there are still many variations of the bike-sharing relocation problem; for example Alvarez-Valdes et al. [12] face the problem by dividing it in 2 steps: the estimate of the unsatisfied demand at each station for a given time period in the future and the prediction of the optimal redistribution of bikes, using a heuristic algorithm. Although limited, their approach, which was tested in Palma de Mallorca (Spain), showed the usefulness of this type of analysis in a real-world scenario. Pal and Zhang [13] presented a hybrid nested large neighborhood search with a variable neighborhood descent algorithm to face with a static BRP. Cruz et al. [14] as well as Bulhões et al. [15] used an iterated local search algorithm to solve, respectively, the single-vehicle BRP and the multi-vehicle BRP. Always by facing with static BRP, Dell'Amico et al. [16] presented mixed–integer linear programming models for the static rebalancing problem. They also developed a destroy and repair metaheuristic algorithm, which makes use of effective constructive heuristic and several local search procedures together for solving the BRP [17].

Another distinction can be done between deterministic BRP and BRP with stochastic demands. Several models and solution algorithms for the BRP have been discussed over the last decades to solve the deterministic BRP and models for the BRP with stochastic demands, by including heuristic and metaheuristic solution methods as well as exact methods.

A tentative of solving the deterministic BRP with a large number of stations, was made by Forma et al. [18]. They proposed a 3-step mathematical programming based heuristic.

Other exact, heuristic and metaheuristic algorithms for this type of problem are proposed by Ho and Szeto [19], Erdogan et al. [20] and Datner et al. [21].

For what regards BRP with stochastic demands, according to Tang et al. [22] who faced with this type of problem, Dell'Amico et al. [23] appear to be the pioneer that proposed a scenario-based two-stage stochastic model for bike repositioning problem, considering redistribution demands of stations as random variables. They, indeed, tried to face with the BRP with Stochastic Demands, which is a variant of the one-commodity many-to-many pickup and delivery vehicle routing problem where demands at each station are represented by random variables, with associated probability distributions, that depend on stochastic scenarios. It is exactly the problem addressed in our work. They used different approaches, in particular, the L-Shaped and branch-and-cut. Moreover, they also proposed heuristic algorithms based on an innovative use of positive and negative correlations among stations' stochastic demands, as well as an efficient strategy for checking feasibility.

Tang et al. [22] always modelled the problem with a two-stage stochastic programming model and proposed a simulated annealing algorithm to solve it. They conducted numerical experiments on a set of instances from 20 to 90 stations to demonstrate the effectiveness of the solution algorithm and the accuracy of the proposed two-stage stochastic model.

In this paper, we exploit the two-stage model proposed by Maggioni et al. [3] and we propose two solutions: one based on GUROBI Exact solver [1] and one by using the Progressive Hedging algorithm as Heuristic model, with the aim to determine the number of bikes to assign to each bike-station at the beginning of the service, in order to minimize the expected total costs, given by the sum of the procurement costs, the expected stock-out costs for unmet demand, the expected time-waste costs for overflow and the expected transshipment costs for repositioning at the end of the service.

3 Mathematical Model

In this section, we present a two-stage stochastic optimization formulation for the problem described in the previous section. The following notation for the two stage stochastic optimization model was used in the mathematical model:

3.1 Sets

1. $\mathcal{B}$: set of bike-stations, $\mathcal{B}$ = {1,...,B};
2. $\mathcal{S}$: set of scenarios, $\mathcal{S}$ = {1,...,S} or finite set of possible realizations of the uncertainty.

3.2 Deterministic Parameters

- procurement cost per bike at each bike-station at the beginning of the service: $c = 2€, \quad c \in \mathbb{R}^+$;
- out of stock cost per bike at bike-station $i \in \mathcal{B}$: $v_i = 4€, \quad v_i \in \mathbb{R}^+$;
- time-waste cost per bike due to overflow at bike station $i \in \mathcal{B}$: $w_i = 8€, \quad w_i \in \mathbb{R}^+$;
- unit transshipment cost per bike transshipped from bike-station i to bike-station j: $t_{ij} = 1€$, $i, j \in \mathcal{B}$, $t_{ij} \in \mathbb{R}^+$;
- capacity of bike-station $i \in \mathcal{B}$: $k_i = 30$, $k_i \in \mathbb{Z}^+$.

3.3 Stochastic Parameters

The problem is a two-stage stochastic problem. This means that the objective of the first stage is to assign bikes to each station and the second stage needs to enact the stochastic demand for those bikes.

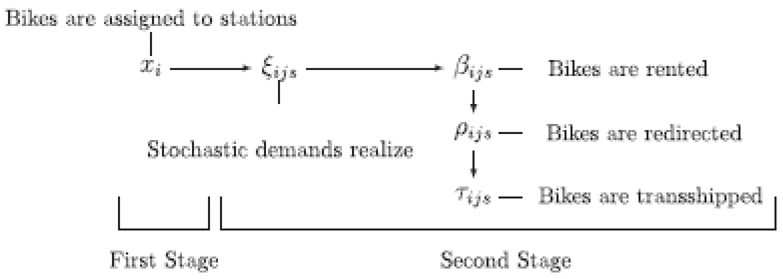
Bikes are assigned to stations
x_i
ξ_{ijs}
β_{ijs} — Bikes are rented
ρ_{ijs} — Bikes are redirected
τ_{ijs} — Bikes are transshipped
Stochastic demands realize
First Stage
Second Stage

Figure 1: Sequence of operations for each scenario s

The **first stage** variable is:

- x_i: number of bikes to assign to bike-station i at the beginning of the service. The optimal number of bikes to assign to a station is the solution of the first stage problem.

After finding the solution of the first stage, then the stochastic demands ξ_{ijs} are added to each bike station. Once the bikes demand is known (ξ_{ijs} being the rental demand from bike-station i to bike-station j in the scenario s), the surplus and shortage can be computed at each bike-station.

The **second stage** decision variables are the following ($\forall i, j \in \mathcal{B}$, $s \in \mathcal{S}$):

- β_{ijs}: Number of rented bikes from bike-station i to bike-station j in scenario s;
- I^+_{is} : Realized surplus of bikes at bike-station i in scenario s. Note that the surplus does not involve any cost for the provider;
- I^-_{ijs} : Realized shortage of bikes at origin-destination pair i, j in scenario s;
- ρ_{ijs} : Number of redirected bikes from bike-station i to bike-station j in scenario s;
- O^+_{is} : Residual capacity at bike-station i in scenario s;

- O^-_{is} : Overflow at bike-station i in scenario s;
- τ_{ijs} : Number of transshipped bikes from bike-station i to bike-station j in scenario s;
- T^+_{is} : Excess of bikes at bike-station i in scenario s;
- T^-_{is} : Lack of bikes at bike-station i in scenario s.

Figure 1 shows the sequence of operations for any scenario s. The problem can be formulated as the following two-stage integer stochastic program:

$$\min c \sum_{i=1}^{\mathcal{B}} x_i + \sum_{s=1}^{\mathcal{S}} p_s \sum_{i=1}^{\mathcal{B}} \left[v_i \sum_{j=1}^{\mathcal{B}} I^-_{ijs} + w_i O^-_{is} + \sum_{j=1}^{\mathcal{B}} t_{ij} \tau_{ijs} \right] \tag{1}$$

$$x_i \le k_i \qquad \forall i \in \mathcal{B} \tag{2}$$

$$\beta_{ijs} = \xi_{ijs} - I^-_{ijs} \qquad \forall i, j \in \mathcal{B}, \forall s \in \mathcal{S} \tag{3}$$

$$I^+_{is} - \sum_{j=1}^{B} I^-_{ijs} = x_i - \sum_{j=1}^{B} \xi_{ijs} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{4}$$

$$O^+_{is} - O^-_{is} = k_i - x_i + \sum_{j=1}^{B} \beta_{ijs} - \sum_{j=1}^{B} \beta_{jis} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{5}$$

$$\sum_{j=1}^{B} \rho_{ijs} = O^-_{is} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{6}$$

$$\sum_{j=1}^{B} \rho_{jis} \le O^+_{is} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{7}$$

$$T^+_{is} - T^-_{is} = k_i - O^+_{is} + \sum_{j=1}^{B} \rho_{jis} - x_i \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{8}$$

$$\sum_{j=1}^{B} \tau_{jis} = T^+_{is} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{9}$$

$$\sum_{j=1}^{B} \tau_{jis} = T^-_{is} \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{10}$$

$$x_i, I^+_{is}, O^+_{is}, O^-_{is}, T^+_{is}, T^-_{is}, \in \mathbb{Z}^+ \qquad \forall i \in \mathcal{B}, \forall s \in \mathcal{S} \tag{11}$$

$$\tau_{ijs}, \beta_{ijs}, \rho_{ijs}, I^-_{ijs}, \in \mathbb{Z}^+ \qquad \forall i, j \in \mathcal{B}, \forall s \in \mathcal{S} \tag{12}$$

The objective function (1) aims to minimize the expected total cost for the system operation, given by the sum of the procurement cost needed to assign the bikes to each station, the expected cost for a shortage of bicycle stock at each given station, the expected time cost associated with a user having no space to park the bike at the chosen station resulting in an overflow and the expected transshipment cost to relocate bikes at the end of the service. Moving on to the constraints:

- (2) enforces that the assigned bikes to each station shall not exceed their capacity,
- (3) defines the number of bikes that are rented by computing the difference between the stochastic demand and the shortage of bikes for each station,
- (4) ensures the balance between surplus and shortage, and is imposed by assuring that the surplus minus the shortage is equal to the number of bikes located at each station minus the stochastic demand for each station
- (5) ensures the balance between the residual parking availability and the overflow, given by the result of the capacity of each bike station minus number of bikes initially placed and accounting for the bikes that depart and arrive at that station,
- (6) defines the sum of all the redirected bikes from a bike station equal to its overflow, since when there is no available parking a user is redirected to another station with an available spot,

- (7) guarantees that the sum of all the redirected bikes to a bike station cannot exceed its residual capacity,
- (8) ensures the balance between exceeding and failure
- (9) defines the sum of all the transshipped bikes from a bike station equal to its excess,
- (10) guarantees that the sum of all the transshipped bikes to a bike station is equal to its failure,
- (11) and (12) define the integrity and non-negativity of first-stage and second-stage variables.

4 Modelling of the demand

4.1 Scenario Generation

A very crucial step of the model building is the generation of the demand for bikes, by considering the couple origin-destination of each trip. In a real-world problem, the demand matrix could be extracted from stored historical data but, due to the lack of real data about possible trips among stations scattered around a city, for the realization of our work, a realistic scenario was invented. It was assumed to there be 22 stations (afterwards the number of stations was increased; go to Section 6 to see numerical results), each with a fixed capacity of 30 bikes slots. It has been necessary to simulate the demand and to generate many different demand matrices, each representing a possible scenario. In order to have a more realistic simulation, the distance between origin and destination was used as a probability coefficient in a way that shorter trips would have a higher probability of occurring, but each scenario was considered with equal probability of happening.

Before generating OD matrices, a minimum m_{ij} and a maximum M_{ij} number of trips between each pair of bike-stations i and j was assumed; in particular $m_{ij} \in [0, 3]$ and $M_{ij} \in [3, 10]$. Then, two uniformly distributed square matrices, m and M, each having as many rows and columns as the number of bike-stations were generated: each entry of matrix m was a number $m_{ij} \in [0, 3]$ and each entry of matrix M was a number $M_{ij} \in [3, 10]$. In this way, m represents the lower bound of the number of trips between each pair of bike station *i,j* that could occur and M represents the upper bound. After that, a number of matrices equal to the optimal number of scenarios (500, chosen according to what is explained in Section 4.2) were randomly generated according to a Uniform Distribution having as lower and as upper bound respectively m and M. Finally, the mean matrix μ among all the generated matrices and the standard deviation matrix σ among all the different scenarios were computed and used afterwards to generate the used scenarios.

Furthermore, as mentioned before, the distance between stations was used to generate a probability matrix P, where each cell contains a number between 0 and 1. The method used to generate this matrix, whose formula is shown in (13), assumes that the sum of the indexes of rows (i) and columns (j), i.e. origins (O) and destinations (D), is the distance between the two stations. The probability of having a particular trip is inversely proportional to the distance between its origin and destination (it is assumed that a trip starting and ending in the same station has zero probability).

$$P_{i,j} = 1 - \frac{i + j}{(n_{stations} - 1) \times 2} \qquad \forall i \in O, \forall j \in D \tag{13}$$

The last step of the scenario generation process is to use *Monte Carlo* sampling to obtain the demand matrices. In particular, scenarios are randomly sampled between three different distributions: the Gaussian Normal distribution $\mathcal{N}(\mu, \sigma)$, the Exponential distribution $\mathcal{E}(\mu)$ and the Uniform Distribution $\mathcal{U}(\mathrm{m}, \mathrm{M})$. Once a matrix is sampled, it is then multiplied by the probability matrix P and it is added to the list of possible scenarios. Equation (14) explains the formula used to generate each scenario ξ_s.

$$\xi_s = P \times S_s \quad \forall s \in \mathcal{S} \tag{14}$$

where S_s is the sampled matrix for the scenario s generated according one of the three aforementioned distributions, P is the probability matrix and ξ_s is the final OD matrix for scenario s.

4.2 Choice of number of scenarios

The number of scenarios used for the analysis was found by performing an in-sample analysis, as suggested by Maggioni & Co. [3]. The analysis is carried out by optimally solving the two-stage stochastic program with scenario trees of increasing size $|\mathcal{S}| = 100$, 200, 300,...,1000 and by generating each scenario according to each of the aforementioned distributions. Results illustrated in Figure 2 show that the objective function value is quite stable over the three probability distributions for an increasing number of scenarios. Therefore, $|\mathcal{S}|$=**500** was chosen as optimum cardinality for the scenario trees, as reported in Figure 3.

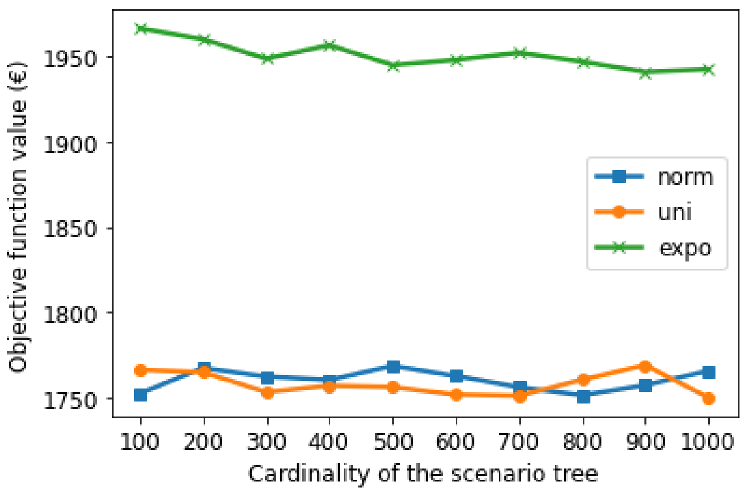

Figure 2: In-sample stability for the expected total cost under the Exponential distribution, Uniform and Normal distributions of the demand.

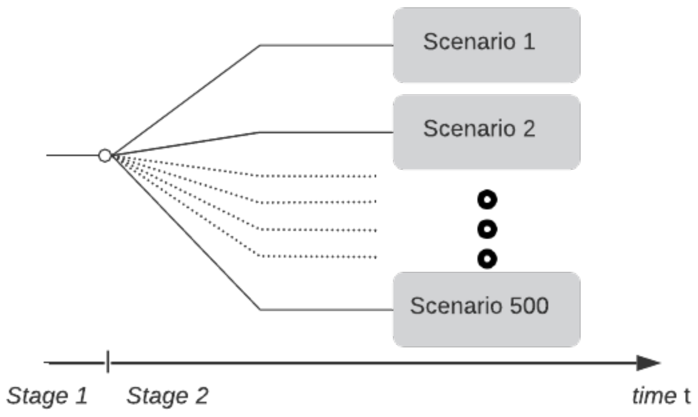

Figure 3: Scenario Tree two-stages.

5 Heuristics: Progressive Hedging Algorithm

In this paper, we use the *Progressive Hedging* algorithm of Rockafellar and Wets [2] as a heuristic method to solve the stochastic problem. According to this algorithm, the deterministic problem is reformulated by making explicit, through non-anticipativity constraints, the fact that each scenario shares first-stage decisions x_s with all the other scenarios: $x_s = \bar{x} \quad \forall s \in \mathcal{S}$. This constraint is also know as **Global Consensus**. The non-anticipativity constraints cause the major obstacle to decomposition. For this reason, the augmented Lagrangian method helps in overcoming this difficulty by moving the constraints in the objective function. The problem can be then rewritten as:

$$\min_{x} \mathbb{E}_{\boldsymbol{\xi}} \Big[f(x_s, \xi_s) + \lambda_s \big(x_s - \hat{x} \big) + \frac{\rho}{2} (x_s - \hat{x})^2 \Big] \tag{15}$$

where $\boldsymbol{\xi}$ is the set of all the scenarios, λ_s is the Lagrange multipliers vector and ρ is the penalty parameter and $f(x_s, \xi_s)$ is the deterministic problem of each single scenario:

$$c \sum_{i=1}^{\mathcal{B}} x_i + \sum_{i=1}^{\mathcal{B}} \big[v_i \sum_{j=1}^{\mathcal{B}} I_{ij}^{-} + w_i O_i^{-} + \sum_{j=1}^{\mathcal{B}} t_{ij} \tau_{ij} \big] \tag{16}$$

This relaxation creates a number of similar and independent sub-problems equal to the number of scenarios $|S|$ (mono-scenario problems), which must be solved optimally. The global consensus on the first-stage decisions is evaluated and iteratively achieved by updating the Lagrangean multipliers according to Equation (17).

$$\lambda_s^{(k)} = \lambda_s^{(k-1)} + \rho^{(k-1)} (x_s^{(k)} - \bar{x}^{(k)}) \quad \forall s \in \mathcal{S} \tag{17}$$

with parameters $\rho^k = \alpha \rho^{(k-1)}$ being the penalty value and α being a positive fixed step size, chosen according to Section 5.1.

At each iteration, an estimation of the solution is computed as the expectation over the current solution of the scenario sub-problems, weighted by the probability of happening of that scenario $p_s = \frac{1}{|\mathcal{S}|}$. It is called *Temporary Global Solution* (TGS).

$$\bar{x} = \sum_{s \in \mathcal{S}} p_s x_s \tag{18}$$

The objective function then becomes a weighted sum of the deterministic sub problems over each generated scenario.

We now focus on the situation where the decision maker considers a set of representative scenarios $\mathcal{S}$, each of them represented by a matrix s. The scenarios are generated by sampling from continuous distributions as explained in Section 4.1.

Algorithm 1 Progressive Hedging (PH)

```
input  : A scenario tensor $\mathcal{S}$ of size $stations \times stations \times scenarios$
output: An array $1 \times station$ with the max number of poles needed per each station
1 Set the iteration counter $k \leftarrow 0$, set the step size parameter $\alpha$ and initialize the penalty parameter $\rho^{(0)}$
2 Decompose the main Problem in $|\mathcal{S}|$ mono-scenario problems and solve the problem of Equation (19) by
    finding first stage solution $x_s^{(0)}$ for each scenario $s \in \mathcal{S}$.
      $$\min c\sum_{i=1}^{\mathcal{B}} x_i + \sum_{i=1}^{\mathcal{B}} \left[v_i \sum_{j=1}^{\mathcal{B}} I_{ij}^- + w_i O_i^- + \sum_{j=1}^{\mathcal{B}} t_{ij}\tau_{ij}\right] \qquad (19)$$
3 Compute the initial Temporary Global Solution:
      $$\bar{x}^{(0)} = \sum_{s\in\mathcal{S}} p_s x_s^{(0)} \qquad (20)$$
    and the initial multiplier
      $$\lambda_s^{(0)} = \rho^{(0)}\left(x_s^{(0)} - \bar{x}^{(0)}\right) \qquad (21)$$
4 for $k \leftarrow 1$ to 100 do
5     if $\left\|x_s^{(k)} - \bar{x}^{(k)}\right\| == 0$ then break ;
6     foreach mono-scenario problem do
      $$x_s^{(k)} = \min c\sum_{i=1}^{\mathcal{B}} x_i + \sum_{i=1}^{\mathcal{B}} \left[v_i \sum_{j=1}^{\mathcal{B}} I_{ij}^- + w_i O_i^- + \sum_{j=1}^{\mathcal{B}} t_{ij}\tau_{ij}\right] + \left(\lambda_s^{(k)}\right)^T\left(x_s - \bar{x}^{(k)}\right) + \frac{\rho^{(k)}}{2}(x_s - \bar{x}^{(k)})^2 \qquad (22)$$
      ;
7     Update the solution
      $$\bar{x}^{(k)} = \sum_{s\in S} p_s x_s^{(k)} \qquad (23)$$
8     Update multipliers $\lambda_s^{(k)} = \lambda_s^{(k-1)} + \rho^{(k-1)}(x_s^{(k)} - \bar{x}^{(k)})$ where $\rho^{(k)} = \alpha\rho^{(k-1)}$
9 end
```

5.1 Input Parameter Choice: penalty ρ and step size α

A non trivial problem for a correct implementation of the Progressive Hedging algorithm is to set the correct value of the Lagrangian multipliers that relax the problem enough to let all the first stage solutions for each scenario to converge to a unique optimal value. The performance of the *penalty method*, used to find the correct value of λ, is strictly sensitive to the choice of the initialization of the penalty parameter ρ. This parameter scales the penalty term in the Augmented Lagrangian relaxation (Equation (22)) and it also affects the step length in going towards the minimum of the objective function, that is the meaning of the lagrangean multiplier (see Equation (17)). There are no theoretical good values for the selection of the penalty parameter; indeed, the choice is typically based on empirical testing.

Shohre Zehtabian & Co. listed in [24] a number of possible update strategies for the penalty parameter. The one we opted for in this project is the incremental one with a fixed parameter $\alpha > 1$, given as input to the algorithm.

In order to see which of the combination of values for ρ and α is best for our use case, we designed a grid search testing all possible combinations between $\rho \in [10, 40, 70, 100]$ and $\alpha \in [1.1, 10, 100]$, as shown in Table 1. Moreover, we manually ran the Heuristics method for value pairs: $[\rho = 9, \alpha = 100]$, $[\rho = 2, \alpha = 100]$ and $[\rho = 70, \alpha = 200]$ and $[\rho = 70, \alpha = 1000]$ just to check that results were not better outer the used ranges of grid search.

Method	**OF value(€)**	**Time (s)**	ρ	α	**It.s**	**Number of bikes per stations**
exact	1817.528	48.25	-	-	-	30 30 30 30 30 30 30 30 29 30 30 30 26 25 23 22 21 20 15 17 15 10
heu	1826.910	150.58	9	100	3	30 30 30 30 30 30 30 30 27 30 30 30 24 22 22 21 19 16 12 17 14 8
heu	1829.132	139.69	2	100	3	30 30 30 30 30 30 30 30 27 30 29 30 24 24 22 20 19 18 12 17 14 8
heu	1831.790	162.73	70	200	3	30 30 30 30 30 30 30 30 27 30 30 30 24 22 22 20 19 16 12 17 14 8
heu	1825.390	156.40	70	1000	3	30 30 30 30 30 30 30 30 27 30 30 30 24 22 22 21 19 17 13 17 14 8
heu	1783.908	233.28	10	1.1	3	30 30 30 30 30 30 30 30 26 30 30 30 23 22 21 20 18 16 12 16 13 7
heu	1823.608	161.3	10	10	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1823.608	140.6	10	100	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1635.408	139.3	40	1.1	3	30 30 30 30 30 30 30 30 26 30 30 30 23 22 21 20 18 16 12 16 13 7
heu	1823.608	155.7	40	10	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1823.608	140.6	40	100	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1486.908	138.6	70	1.1	3	30 30 30 30 30 30 30 30 26 30 30 30 23 22 21 20 18 16 12 16 13 7
heu	1823.608	144.0	70	10	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1823.608	140.5	70	100	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1338.408	138.1	100	1.1	3	30 30 30 30 30 30 30 30 26 30 30 30 23 22 21 20 18 16 12 16 13 7
heu	1823.608	148.2	100	10	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8
heu	1823.608	141.3	100	100	3	30 30 30 30 30 30 30 30 27 30 30 30 24 23 22 21 19 17 13 17 14 8

Table 1: Results of Progressive Hedging method with different pairs values of ρ and α.

The choice of the two parameters values was taken according to criteria:

- proximity to exact solution
- lowest speed of the heuristics
- lowest number of iterations needed

As reported in Table 1, the two selected values were $\rho = 70$ and $\alpha = 100$, because they were the ones giving value of the Objective Function closest to the one given by the Exact Method in the shortest amount of time. We also tried with $\rho = 2$ or $\rho = 9$ and $\alpha = 200$ or $\alpha = 1000$ to see whether by exiting out of the range we could get better results, but it was not the case. Therefore, we can say that the chosen ranges were good.

6 Results

6.1 Exact vs Heuristic Comparison

The focus of our work was to find a better solution for bike sharing systems with bigger dimensions than the one shown in the work done by Maggioni et al. in [3]. They focused their work on a small city such as Bergamo (Italy), having 22 bike stations. In our case, we repeated the study twice: for a hypothetical city having 22 bike stations and for a bigger hypothetical city with more stations. In Table 2, we can see that an optimal solution for the two stage stochastic problem proposed by them can be obtained by using an exact solver (in our case GUROBI) but only if the size of the bike sharing network is small enough, like the case of 22 stations. When the number of stations increases and overcomes 70, the exact solver crashes. Indeed, optimal solution is not found. Our proposed Progressive Hedging-based heuristic approach, instead, finds a close to optimal solution for any dimension of the bike-station set, but as shown in Table 2, the time needed to find it is bigger with respect to the one found through the not decomposed model. Therefore, the main strength of our method lays in the ability to find solutions with more bike stations considered in the computation. Anyway, we think that the time expended by the PH heuristic method could be dramatically reduced by a parallelization of the computation of the solution for each scenario at each iteration of the algorithm. We actually performed an attempt at parallelizing the work but, probably due to Python's task scheduler, the time needed for the initialization of the tasks was bigger than the time saved by using multiprocessing.

Method	**N.Stations**	**Time (s)**	**OF value (€)**	**iterations**
heu	10	19.92	346.18	3
exact	10	4.50	343.12	None
heu	20	68.18	1511.80	3
exact	20	68.15	1495.51	None
heu	22	119.09	1750.09	3
exact	22	97.33	1725.95	None
heu	30	169.59	3802.81	3
exact	30	28.05	3793.40	None
heu	40	329.89	7347.18	3
exact	40	67.04	7333.79	None
heu	50	526.03	12293.65	3
exact	50	117.28	12288.32	None
heu	60	798.13	17826.11	3
exact	60	123.13	17824.67	None
heu	70	228.52	25804.10	1
exact	70	-	-	-

Table 2: Results of Progressive Hedging method with numbers of bike stations.

However, in our opinion, when the number of stations becomes high, similar to the numbers easily found in cities making an extensive use of bike sharing systems (think at american cities like New York or Washington), the exact solver is limited by the computational capabilities of the used machine (For this computations the machine used had: • CPU: 6 cores (12 threads) Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2200 MHz • Grafics Card: nVidia GP106M [GeForce GTX 1060 Mobile])

Where the exact solver fails, the developed heuristic approach catches up and starts performing better than before, still maintaining reasonable results with respect to the one that one would expect from the exact solution for the same scenario realizations (see the penultimate line of Table 2). This makes our solution fitting for bigger city planning projects and in line with the scope of our work.

6.2 Stability Analysis

A minimal requirement that a scenario-generation method must satisfy, since the method involve some randomness, is *stability*: if we generate several scenario trees (with the same input) and solve the optimization problem with these trees, we should get the same optimal value of the objective function.

6.2.1 In-sample stability

The first type of stability we checked was the *in-sample stability.* We generated K = 100 scenario trees $\hat{\xi}_k, k = 1, ..., K$, we solved the stochastic programming problem with each one of them and we obtained optimal solutions $\hat{x}_k^*, k = 1, ..., K$. We say that we have *in-sample stability* if whichever scenario tree we choose, the optimal value of the objective function is (approximately) the same:

$$F(\hat{x}_k^*; \hat{\xi}_k) \approx F(\hat{x}_l^*; \hat{\xi}_l) \quad k, l \in \{1, \ldots, K\} \quad k \neq l \tag{24}$$

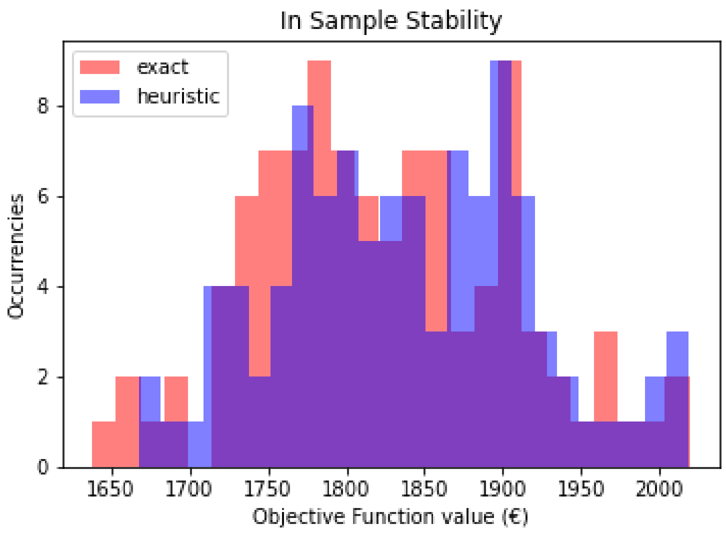

Figure 4: Histogram showing the in-sample stability

Figure 4 shows the number of times that the two methods gave same value of the objective function in euro.

By observing it, it is clear that the scenario-generation method we used, explained in Section 4.1, appears to be very stable during the in sample analysis. Indeed, the computed standard deviations over 100 different scenario trees for the exact method σ_e and for the Progressive Hedging heuristic method σ_h were respectively: $\sigma_e = 81.30$€ and $\sigma_h = 79.76$€, taking into account that the mean values of the objective function were 1823.44€ for the exact method and 1837.92€ for the heuristics method.

6.2.2 Out-of-Sample Stability

The second type of stability we checked was the *out-of samples stability.*

On the contrary of the in-sample stability, where we need only to solve the scenario-based optimization problem, for the out-of-sample stability we have to be able to evaluate the "true" objective function $F(\hat{x}_k^*; \hat{\xi})$, because we say that we have the *out-of-sample stability* when:

$$F(\hat{x}_k^*; \hat{\xi}) \approx F(\hat{x}_l^*; \hat{\xi}) \quad k, l \in \{1, \ldots, K\} \quad k \neq l \tag{25}$$

Equation (25) says that: having out-of-sample stability means that the real performance of the solution $\hat{x}_k^*, k = 1, ..., K$ is stable, i.e. it does not depend on which scenario tree $\hat{\xi}$ we choose.

As suggested by Kaute & Wallace in [25], there are several possible ways to do the out-of-sample testing, i.e. the *evaluation* of the objective function $F(\hat{x}_k^*; \hat{\xi})$ for a given first stage solution $\hat{x}_k^*, k = 1, ..., K$: if we use historical data in the scenario generation, back-testing may be an appropriate option. Or, if we have another scenario generation method we believe to be stable, we may use it to create a reference scenario tree and evaluate the first stage solution $\hat{x}_k^*$ for each scenario tree $k = 1, ..., K$.

Due to the lack of real historical data and since the scenario generation method of Section 4.1 proved to be stable, as demonstrated in Section 6.2.1, we chose it to create a reference scenario tree to be used as a representation of the "real world" and evaluate the first stage solutions for each scenario tree, with a given seed being set for the randomness.

In practice, we firstly generated $K = 100$ scenario trees each with cardinality of 500 scenarios by setting a "seed" (consider those scenarios as reference scenarios); then, for each one of the scenario tree, we solved the optimization problem, by obtaining optimal first stage solutions $\hat{x}_k^*, k = 1, \dots, 100$ and we used the $\hat{x}_k^*$ solutions to solve only the second stage problem over new scenarios generated with the method of Section 6.2.1 but with a new seed for the randomness. In this way, we can consider to test the out-of-sample stability, because by changing seed we can assume to sample the new scenarios in a new "benchmark scenario set", therefore *out* of the reference "benchmark scenario set".

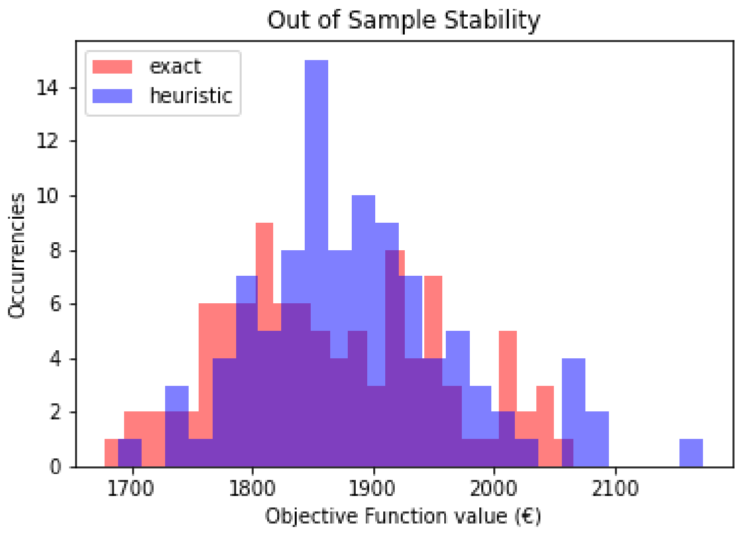

Figure 5: Histogram showing the out-of-sample stability

Figure 5 shows the number of times that the two methods gave same value of the objective function in euro.

By observing it, it is clear that the scenario-generation method we used, explained in Section 4.1, appears to be very stable during the out-of sample analysis. Indeed, the computed standard deviations over 100 different scenario trees for the exact method σ_e and for the Progressive Hedging heuristic method σ_h were respectively: $\sigma_e = 90.84$€ and $\sigma_h = 85.32$€, taking into account that the most returned objective function values were 1823.17 € for the exact method (output 9 times) and 1863.23 € for the heuristics method (output 15 times).

6.3 Stochastic Programming Model evaluation

There are several ways to evaluate the quality of the Stochastic Programming (SP) model. We chose to compute two indexes: the *Expected Value of Perfect Information* (EVPI) and the *Value of the Stochastic Solution* (VSS).

6.3.1 Expected Value of Perfect Information

The EVPI is computed according to Equation (26), where RP is the Recourse Problem (RP) solution and WS is the Wait-and-See (WS) solution.

This quantity represents the maximum amount one would pay in return for complete information about the future. In other words, how much we are going to loose due to the presence of stochasticity, because it is the difference between when we consider the stochasticity (RP) and when we assume to have certainty on the future (WS).

$$EVPI = RP - WS \tag{26}$$

The two stages RP problem consists of solving the first stage by obtaining the *here-and-now* solution x^* and then solving the second stage for each scenario of vector $\boldsymbol{\xi}$ by considering always the same here-and-now solution of the first stage. The RP solution is then given by Equation (27), i.e. it is the expected value of all the objective function values for all the scenarios of vector $\boldsymbol{\xi}$ (see Figure 3).

$$RP = \mathbb{E}_{\boldsymbol{\xi}}\big[z(x^*, \boldsymbol{\xi})\big] \tag{27}$$

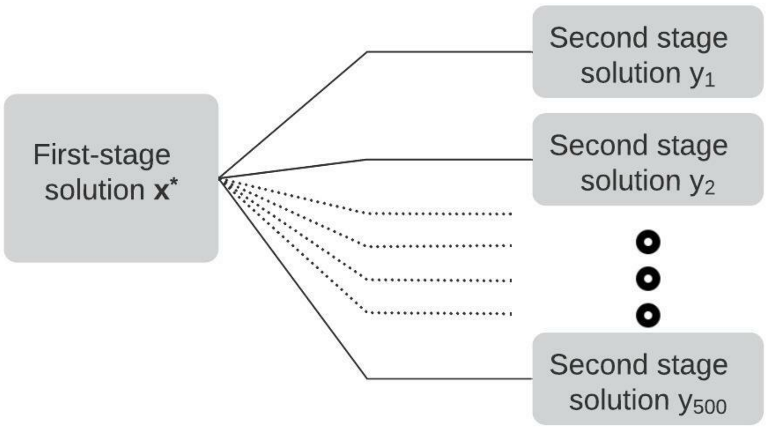

Figure 6: Recourse Problem graph representation

The WS solution consists instead of solving for each scenario of vector $\boldsymbol{\xi}$ the first and second stage problems. In this way, it is as we were able to see the future by knowing what is the future demand (and therefore the scenario). We choose the optimal solution $\bar{x}(\xi)$ for any realization ξ (scenario) of vector $\boldsymbol{\xi}$ and their related optimal values of the objective functions $z\big(\bar{x}(\xi), \xi\big)$. The WS solution will be given by Equation (28), i.e. the expected value of the solutions of each scenario (see Figure 7).

$$WS = \mathbb{E}_{\boldsymbol{\xi}}\big[z\big(\bar{x}(\xi), \boldsymbol{\xi}\big)\big] \tag{28}$$

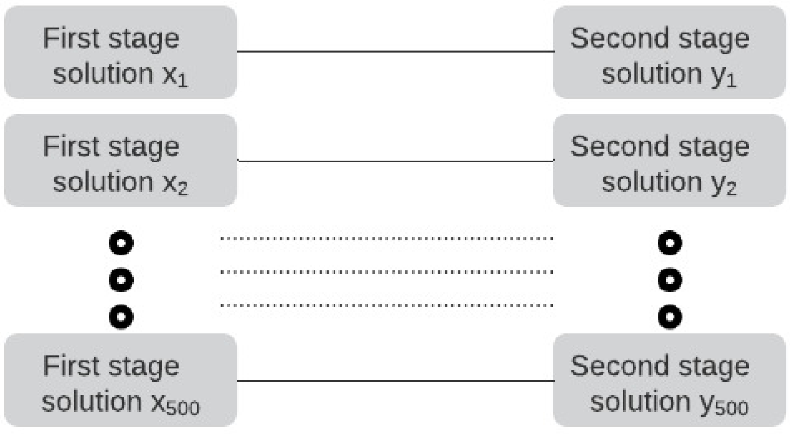

Figure 7: Wait and See solution graph representation

The obtained result was an EVPI = 85€. It means that studying the stochasticity makes us loose only 85€.

6.3.2 Value of the Stochastic Solution

The VSS (see Birge & Louveaux [26]) is computed according to Equation (29), where EEV is the Expected objective function value by considering the solution of the Expected Value problem (EV) and RP is the Recourse Problem.

$$VSS = EEV - RP \tag{29}$$

It measures the expected increase in value from solving the stochastic version of a model rather than the simpler deterministic one. In other words, how much we can gain if we consider the stochasticity with respect to considering only the Expected Value of random variables.

The EV problem is the problem in which the decision maker replaces all random variables by their expected values and solves a deterministic program:

$$EV = \min_{x} z\big(x, \bar{\xi}\big) \tag{30}$$

where $\bar{\xi} = \mathbb{E}(\boldsymbol{\xi})$, i.e. the deterministic problem is solved over a scenario that is the average of all the scenarios we considered (500). It yields an optimal solution noted as $\bar{x}(\bar{\xi})$ called *expected value solution.*

We use this solution as the first stage solution for the stochastic program and we compute the expected value of the objective function over several scenarios of vector $\boldsymbol{\xi}$ by using the EV first stage solution:

$$EEV = \mathbb{E}_{\boldsymbol{\xi}}\big[z(\bar{x}(\bar{\xi}), \boldsymbol{\xi})\big] \tag{31}$$

The computed VSS was 222€, that turns out to be a very good result. Indeed, as also stressed by Maggioni & Wallace [27], having a low VSS would have indicated that for the model builders the uncertainty was important, when in fact it wasn't.

For this reason, studying the stochastic programming model was necessary despite the efforts involved.

7 Conclusions

This work presents a new heuristic approach for the bike sharing repositioning problem solution based on Progressive Hedging. The two stage stochastic problems under consideration tackle one of the key points of bike sharing systems which is to find the optimal number of bikes for each stations in order to minimize the management cost and improve users Quality of Life. This work followed the one done in [3] for a small city like Bergamo, Italy. Our work tried to expand it, by scaling it up in the prospect of a more intensive use of the bike as a mean of transport in the future. With ≈ 30 bike stations, our novel approach finds a solution that is very close to the exact one but in more time. Whereas our solution is strong when the computation loads are very high (≈ 200 bike stations). Our heuristic, indeed, finds the solution in a reasonable amount of time, whereas the exact solution cannot be found by the exact solver, by crashing after a while.

We also performed an analysis of the Scenario Generation stability, in particular out-of-sample and in-sample stability. It turned to be stable, with low values of standard deviations.

Finally, we performed the model evaluation by calculating the Expected Value of Perfect Information and the Value of Stochastic Solution.

An important step to be done in the future might be to use real data in order to improve the Scenario Generation method of Section 4.1 and fit the solution to particular cities for tailored solutions. Moreover, a parallelization of the computation of the solution for each scenario at each iteration of the PH algorithm could be implemented to dramatically reduce the time expended by the PH heuristic method to output a solution. Also other heuristics like "genetic algorithms" or "particle swarm optimization" might be developed for our solution as a comparison.

References

[1] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. `https://www.gurobi.com`.

[2] R. Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[3] Francesca Maggioni, Matteo Cagnolari, Luca Bertazzi, and Stein W. Wallace. Stochastic optimization models for a bike-sharing problem with transshipment. *European Journal of Operational Research*, 276(1):272–283, 2019. `https://www.sciencedirect.com/science/article/pii/S0377221718311044`.

[4] Karlo Heijnen. Mobility as a service. solution to urban mobility problems? *Individual Project*, January 2021.

[5] Nikolaos-Fivos Galatoulas, Konstantinos N. Genikomsakis, and Christos S. Ioakimidis. Spatio-temporal trends of e-bike sharing system deployment: A review in europe, north america and asia. *Sustainability*, 12(11), 2020.

[6] Feiyang Sun, Peng Chen, and Junfeng Jiao. Promoting public bike-sharing: A lesson from the unsuccessful pronto system. *Transportation Research Part D: Transport and Environment*, 63:533–547, 2018.

[7] Federico Chiariotti, Chiara Pielli, Andrea Zanella, and Michele Zorzi. A dynamic approach to rebalancing bike-sharing systems. *Sensors*, 18, feb 2018.

[8] P. DeMaio and R Meddin. The bike sharing blog, 2021. `http://bike-sharing.blogspot.com/`.

[9] Elżbieta Macioszek, Paulina Świerk, and Agata Kurek. The bike-sharing system as an element of enhancing sustainable mobility—a case study based on a city in poland. *Sustainability*, 12:3285, 04 2020.

[10] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 15:1–31, 02 2007.

[11] Mike Benchimol, Pascal Benchimol, Benoit Chappert, Arnaud Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. Balancing the stations of a self service "bike hire" system. *RAIRO - Operations Research*, 45:37–61, 05 2011.

[12] Ramon Alvarez-Valdes, Jose M. Belenguer, Enrique Benavent, Jose D. Bermudez, Facundo Muñoz, Enriqueta Vercher, and Francisco Verdejo. Optimizing the level of service quality of a bike-sharing system. *Omega*, 62:163–175, 2016.

[13] Aritra Pal and Yu Zhang. Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. *Transportation Research Part C: Emerging Technologies*, 80:92–116, 2017.

[14] Fábio Cruz, Anand Subramanian, Bruno P. Bruck, and Manuel Iori. A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. *Computers & Operations Research*, 79:19–33, 2017.

[15] Teobaldo Bulhões, Anand Subramanian, Güneş Erdoğan, and Gilbert Laporte. The static bike relocation problem with multiple vehicles and visits. *European Journal of Operational Research*, 264(2):508–523, 2018.

[16] Mauro Dell'Amico, Eleni Hadjicostantinou, Manuel Iori, and Stefano Novellani. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, 2014.

[17] Mauro Dell'Amico, Manuel Iori, Stefano Novellani, and Thomas Stützle. A destroy and repair algorithm for the bike sharing rebalancing problem. *Computers & Operations Research*, 71:149–162, 2016.

[18] Iris A. Forma, Tal Raviv, and Michal Tzur. A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological*, 71:230–247, 2015.

[19] Sin C. Ho and W.Y. Szeto. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180–198, 2014.

[20] Güneş Erdoğan, Maria Battarra, and Roberto Wolfler Calvo. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245(3):667–679, 2015.

[21] Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. Setting inventory levels in a bike sharing network. *Transportation Science*, 53(1):62–76, 2019.

[22] Qiong Tang, Zhuo Fu, Dezhi Zhang, Hao Guo, and Minyi Li. Addressing the bike repositioning problem in bike sharing system: A two-stage stochastic programming model. *Scientific Programming*, 2020:8868892, May 2020.

[23] Mauro Dell'Amico, Manuel Iori, Stefano Novellani, and Anand Subramanian. The bike sharing rebalancing problem with stochastic demands. *Transportation Research Part B: Methodological*, 118:362–380, 2018.

[24] Shohre Zehtabian and Fabian Bastin. *Penalty parameter update strategies in progressive hedging algorithm.* Cirrelt, 2016.

[25] Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, page 271.

[26] J. R. Birge and F. Louveaux. *Introduction to stochastic programming.* Springer Science & Business Media, 2011. `https://books.google.it/books?id=Vp0Bp8kjPxUC&lpg=PR1&ots=q6AK40hcBv&lr&hl=it&pg=PP1#v=onepage&q&f=false`.

[27] Francesca Maggioni and Stein Wallace. Analyzing the quality of the expected value solution in stochastic programming. *Annals of Operations Research*, 200:37–54, 08 2013.