mathematics

MDPI

Article

Resources Relocation Support Strategy Based on a Modified Genetic Algorithm for Bike-Sharing Systems

Horațiu Florian *, Camelia Avram, Mihai Pop, Dan Radu and Adina Aștilean

Department of Automation, Technical University of Cluj-Napoca, 40014 Cluj-Napoca, Romania; camelia.avram@aut.utcluj.ro (C.A.); mihai.ilie.pop@gmail.com (M.P.); dan.radu@aut.utcluj.ro (D.R.); adina.astilean@aut.utcluj.ro (A.A.)

* Correspondence: horaflor@yahoo.com

Abstract: In recent decades, special attention has been given to the adverse effects of traffic congestion. Bike-sharing systems, as a part of the broader category of shared transportation systems, are seen as viable solutions to these problems. Even if the quality of service in bike-sharing service systems were permanently improved, there would still be some issues that needed new and more efficient solutions. One of these refers to the rebalancing operations that follow the bike depletion phenomenon that affects most stations during shorter or longer time periods. Current work develops a two-step method to perform effective rebalancing operations in bike-sharing. The core elements of the method are a fuzzy logic-controlled genetic algorithm for bike station prioritization and an inference mechanism aiming to do the assignment between the stations and trucks. The solution was tested on traffic data collected from the Citi Bike New York bike-sharing system. The proposed method shows overall superior performance compared to other algorithms that are specific to capacitated vehicle routing problems: standard genetic algorithm, ant colony optimization, Tabu search algorithm, and improved performance compared to Harris Hawks optimization for some scenarios. Since the algorithm is independent of past traffic measurements, it applies to any other potential bike-sharing system.

Keywords: genetic algorithm; bike-sharing system; fuzzy-logic control; inference mechanism; capacitated vehicle routing problem

MSC: 90B20

Citation: Florian, H.; Avram, C.; Pop, M.; Radu, D.; Aștilean, A. Resources Relocation Support Strategy Based on a Modified Genetic Algorithm for Bike-Sharing Systems. *Mathematics* **2023**, *11*, 1816. https://doi.org/10.3390/math11081816

Academic Editor: Shih-Wei Lin

Received: 2 March 2023
Revised: 6 April 2023
Accepted: 10 April 2023
Published: 11 April 2023

1. Introduction

During the last few decades, cycling has been seen as an environment-friendly mobility solution [1] that can respond to the mobility needs of many categories of users [2]. Consequently, at this moment, there is an impressive number of bike-sharing programs, implying millions of bicycles [3].

Involving the activity of renting bikes from a determined set, from which the clients can pick up, ride, and park in different parts of the city, the bike-sharing system (BSS) has to face many challenges. The most important of these, related to capacity, transfer, reliability, and integration, are the same as those of other transportation modes [4].

One of the main problems that need to be solved by bike-sharing companies is that after a small number of usages, many bikes are parked outside of interest areas, where pickup demand is very high. Due to traffic polarization, bike scarcity hotspots will appear in some areas simultaneously with dock scarcity in others. In these circumstances, specific approaches are needed. These can be grouped into three categories: strategic (long-term), tactical (mid-term), and operational (short-term) management levels [5]. Strategic management deals with the capacities and locations of the stations. By processing the measured data on single trips, the tactical level could ensure the optimal load levels for every station, which have a periodic variation throughout the day. In contrast, operational planning

estimates levels based on typical user behavior [6]. Additionally, the system needs another resource for bikes and docks to rebalance agents [7].

Usually, trucks are used to pick up bikes from an overflowing station and transport them to a station that is too empty. The problems arising and the need for rebalancing operations are crucially affected by the non-deterministic nature of the bike depletion phenomenon. When many stations require rebalancing service simultaneously or very close in time, outnumbering the rebalancing agents, the vehicles routing for large systems becomes problematic [8–11]. Moreover, an inadequate, longer route means higher transportation costs. In addition to lowering company profits, inefficient routes increase road traffic, defeating the environment-friendliness objective of the BSS.

Even if many promising results have already been obtained, numerous published articles in this area highlight that the rebalancing issue in BSS is still an open research problem. Motivated by the necessity to find suitable and better-performing solutions, the paper presents a repositioning strategy for bikes in the BSS stations characterized by a pronounced dynamic nature. The core elements of this strategy are a fuzzy logic-controlled genetic algorithm (FLCGA) [11] and an inference mechanism, which are the main parts of a two-step method. In the first step, the FLCGA algorithm is used to select a cost-effective order for serving the bike stations that need rebalancing. To minimize the total transportation cost per transported bike, a method for providing the rebalancing agents for the bike stations is developed in the second step.

The proposed method was tested on traffic data from Citi Bike New York BSS and compared with the following algorithms: standard genetic algorithm (SGA), ant colony optimization (ACO) [12], Harris Hawks optimization (HHO) [13], Tabu search algorithm (TSA) [14]. The applicability of all methods is shown by performing numerical experiments using real historical traffic data on 1000 datasets, each corresponding to a cluster of stations in the system, and 17 scenarios, each corresponding to a unique combination of truck numbers and stations.

The next sections of the article are laid out as follows: Section 2 provides a literature review, Section 3 states the problem meant to be solved, Section 4 proposes the solution to the problem and its parameters, Section 5 presents the results, Section 6 contains the discussion on the results and methodology, and Section 7 states the conclusions.

2. Literature Review

In a dynamic regime, there are a few practical alternatives for providing the necessary number of bikes and accessible places on docks in a reasonable time interval. Some current solutions, defined as user-based repositioning, focus on finding modalities to determine the user's willingness to release the shared bikes at other destinations than those they consider to be more convenient [15,16].

Operator-based repositioning is another approach suitable to ensure available resources in due time, especially in urgent fleet relocation. For these cases, specific balancing modalities were proposed to minimize the number of situations where customers cannot find bikes or free places at dock stations [17,18]. There are also proposals to combine the two schemes [19].

The relocation operation, whether static or dynamic, implies finding solutions for routing the vehicles that extract or insert the required number of bikes into docks and is considered the most challenging operation of the BSS service. The difficulty in finding the best routes in the process of bike relocation is a direct consequence of demand uncertainty. Many strategies were proposed for this purpose, including exact, hybrid, heuristic, metaheuristic, and hyper-heuristic algorithms [20,21].

Moreover, the rebalancing operations are also influenced to a considerable extent by other factors, some of which are the number of vehicles used for repositioning operations, their capacities, and their corresponding allocations. Consequently, the proposed strategies and the chosen routing algorithms are directly determined by the prerequisites underlying the rebalancing operations. A multitude of solutions were presented in different contexts.

Bulhões et al. [22] developed a method based on an IP formulation and an iterated local search metaheuristic to solve the static bicycle relocation with multiple vehicles and visits problem.

Integer programming models are also used in [23] for optimal BSS planning from an integrated and long-term perspective and in [24] for inventory and routing optimization in BSS.

Another solution for the static rebalancing operations is proposed by Chemla et al. [25], where the routing activity is performed by a single vehicle.

Past information is analyzed in [26] to extract and use a critical pattern for dynamic rebalancing operations. Thereby, an advanced, planned rebalancing action is possible to offset future adverse effects.

The problem of efficient routing is tackled in [27] by developing approximation algorithms and hardness analyses for the BSS routing problem. Multiple simultaneous routing requests are analyzed, and solutions are proposed for allocating the limited resources and generating an optimization plan.

Problems encountered in the planning processes characteristic of bike-sharing services are described in [5], which gives an overview and classification of existing literature. Additionally, a study performed at [5] identifies research gaps, pointing out the importance of future research directions.

Other studies focus on solving the rebalancing problem by designing new frameworks. A consistency index of travel and a spatial-distribution learning method for static rebalancing, increasing the demand satisfaction, and decreasing the number of vehicles visiting stations, is presented in [28]. In [29], the authors develop a simulation framework that can be integrated with static and dynamic rebalancing optimization models, aiming to evaluate different rebalancing strategies.

A particular rebalancing problem in bike-sharing systems is the rebalancing with consideration for the collection of malfunctioning bikes, for which [30] proposes an integer linear programming model and [31] presents a greedy heuristic method to solve the problem. At the same time, tests on results are performed using actual data from Divvy BSS. In the end, a comprehensive repositioning strategy is used to quantify the benefits.

To find new efficient routing strategies for rebalancing BSS stations, some studies adapt and develop formulations typical for capacitated vehicle routing problem (CVRP). This is the basic and best-known variant of the vehicle route problem (VRP), considered the generalization of the traveling salesman problem (TSP), and has been intensively studied in the literature, considering deterministic and stochastic variants. Many applications, including different hypotheses and constraints, were defined in this context, with both deterministic and stochastic formulations and their corresponding solutions being proposed. A comprehensive literature review of the stochastic variants, namely the stochastic (capacitated) vehicle routing problem (SVRP or SCVRP), is presented by Oyola et al. in [10,32]. In [32], representative categories of solutions are analyzed. The first part of the review is focused on different types of problems and their approaches, starting with the CVRP with stochastic demand (CVRPSD) and continuing with the CVRP with stochastic travel and service times in different variants and combinations.

Considering classical solutions, exact methods are efficient only for small problem instances. Even if they do not guarantee optimal solutions, the heuristic and metaheuristic algorithms deliver feasible solutions for the NP-complete problem, with satisfactory results being obtained within a reasonable time interval [33].

During the latest decades, strategies belonging to the NP-optimization category of problems have enjoyed high importance for science and industry [34]. Hundreds of papers presenting a diversity of solutions based on metaheuristic techniques were proposed. These metaheuristics used for solving the VRP problem and its variants were analyzed in some comprehensive studies [35,36], having classification purposes and aiming to find new directions and topics of interest for future research. Recent contributions in this field are oriented not only toward the development of new, high-performance algorithms [13,37,38],

including the hybrid category, which combines different algorithmic concepts [39,40], but also toward the search for solutions in order to improve the performances of the existing ones [41].

A diversity of solutions based on metaheuristic techniques were proposed, considering the character of the searching process (trajectory-based or population-based search), memory or memoryless usage, and naturally and non-naturally inspired algorithms [42–44].

As a primary class of evolutionary algorithms, genetic algorithms are widely used to solve NP-complete combinatorial optimization problems [45] and in many logistics applications [46–49], including the generation of efficient rebalancing routes [50]. There are many studies aiming to improve the genetic algorithm's performance and extend its applicability [51].

Fuzzy-logic controllers are proposed to determine the most appropriate algorithm parameter values, having crossover and mutation rates as outputs of the fuzzy system [11,52–56]. A comprehensive review of research advances and interest directions for genetic algorithms is presented in [55].

3. Problem Description

In Figure 1, a map reconstruction of the bike stations from Citi Bike New York BSS is presented:

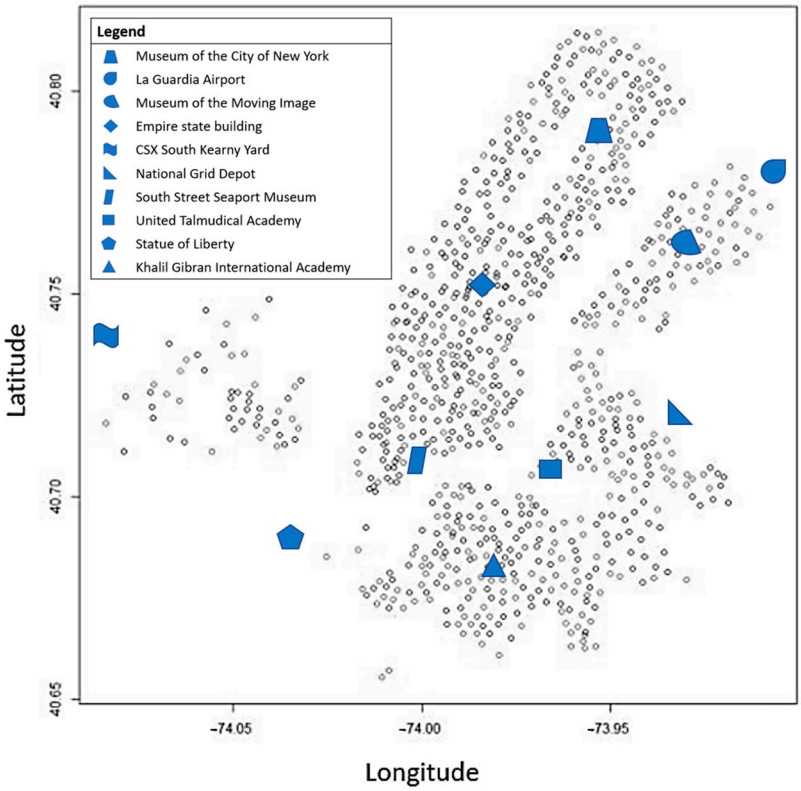

Figure 1. Citi Bike New York bike stations map reconstructed.

The data corresponding to this BSS bike station contains the following information: location, number of bikes, and capacity.

It has been observed that there are time intervals in which some of the bike stations tend to remain without bikes or available parking, therefore requiring rebalancing.

Due to the ever-increasing demand and inequality between stations, it is impossible to rebalance them once and for all. Therefore, rebalancing activities are everlasting, and the goal of using finite resources translates to resolving a scheduling problem.

The balancing activity of two stations is shown in Figure 2.

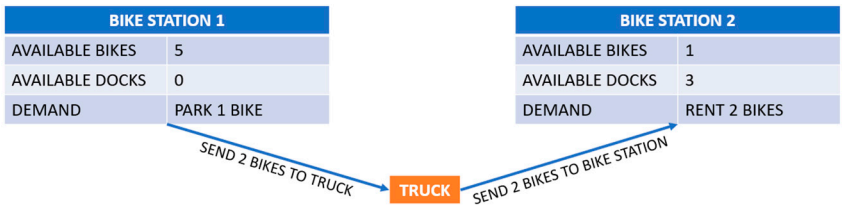

Figure 2. Rebalancing activity example.

Rebalancing several stations one after another assumes that the agent, in this case, the truck, is constantly moving between stations so that the trip's destination will become the origin of the journey, corresponding to the next rebalancing operation.

Another level of complexity is added by the fact that a system will contain more than one truck at a time. It leads to the general CVRP, expressed for our particular use case in Figure 3:

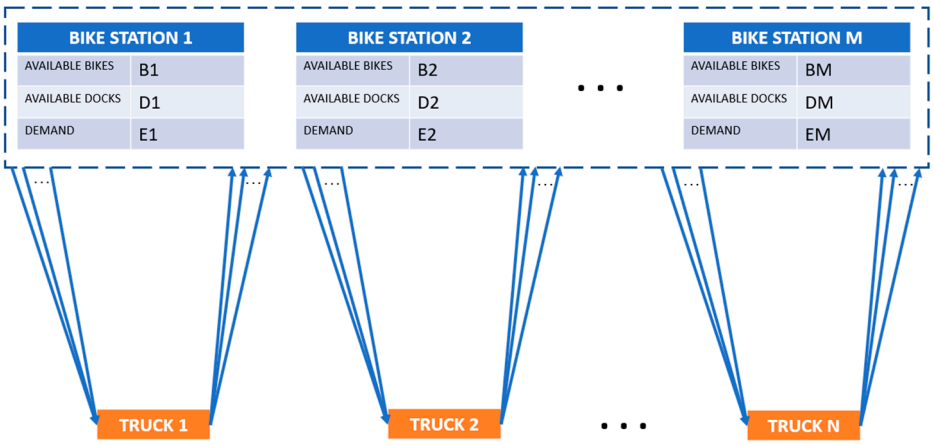

Figure 3. General rebalancing routing problem.

The set of essential attributes of the *BSS* is:

$$BSS = \{S, T, U\} \tag{1}$$

where the variables $S, T,$ and U are described in Table 1.

Table 1. Variables' descriptions.

Variable	Property	Definition
S	$S = \{s_1, s_2, \ldots, s_M\}$	The set of bike stations
M		Number of bike stations
s_i	$s_i = (Ls_i, Bs_i, Cs_i)$	Bike station (index i)
Ls_i		Geographical location
Bs_i		Number of available bikes
Cs_i		Total capacity (bikes)
T	$T = \{t_1, t_2, \ldots, t_N\}$	The set of trucks
N	$N < M$	Number of trucks
t_i	$t_i = (Lt_i, Bt_i, Ct_i, J_i)$	Truck (index i)
Lt_i		Geographical location
Bt_i		Number of available bikes
Ct_i		Total Capacity
Ji	$J_i = (s_k, Jp_i, Jb_i)$	Job assigned
s_k		Bike station destination
Jp_i	$Jp_i \in \{"Insert", "Extract"\}$	Job type
Jb_i		The number of bikes requested

Table 1. *Cont.*

Variable	Property	Definition
U	$U = \{U_1, U_2, \ldots, U_P\}$	The set of depots
P	$P < N$	Number of depots
U_i	$U_i = \left(S^i, T^i, Cu_i, \Lambda_i^\tau\right)$	Depot
S^i	$S^i \subset S$	Partition of bike stations set
T^i	$T^i \subset T$	Partition of trucks set
Cu_i		Capacity
Λ_i^τ	$\Lambda_i^\tau : T^i \to S^i$	Assignment function at the moment τ

$S = \{S1, S2, \ldots, SM\}$ corresponds to the set of M bike stations, with each station $Si = (SNi, SCi)$ having several available bikes, SN, and a capacity for a maximum number of bikes to be parked, SC. The number of available bikes inside a station is zero (as a minimum) or its total capacity (as a maximum):

$$0 \leq Bs_i \leq Cs_i. \tag{2}$$

We define station load as the ratio of present bikes over total capacity:

$$Ls_i = \frac{Bs_i}{Cs_i}, \tag{3}$$

This leads to the property:

$$0\% \leq Ls_i \leq 100\%. \tag{4}$$

Using this property, we can define both states of a bike station that are avoided by rebalancing operations:

$$Bike\ station\ s_i\ is \begin{cases} empty, & when\ Ls_i < 10\% \\ full, & when\ Ls_i > 90\% \end{cases}. \tag{5}$$

If a bike station receives rebalancing, the evolution of the number of available bikes is as follows:

$$\Delta Bs_i = Bs_i(\tau) - Bs_i(\tau - 1), \tag{6}$$

where ΔBs_i is the variation in the number of bikes and τ is the moment when the rebalancing operation occurs.

Based on the above relation, the following property is derived:

$$|\Delta Bs_i| \leq Cs_i. \tag{7}$$

$T = \{T1, T2, \ldots, TN\}$ corresponds to a set of N trucks, $N < M$. Each truck $Ti = (TNi, TCi, Ji)$ has a number of stored bikes (TN), a capacity (TC), and a job assigned to it: $Ji = (Si, JTi, JNi)$. This specifies what station to rebalance (Si), what rebalancing type to apply ($JTi \in \{\text{“}Insert\text{”}, \text{“}Extract\text{”}\}$), and the number of bikes involved in the rebalancing job. The number of available bikes inside a station is zero (as a minimum) or its total capacity (as a maximum):

$$0 \leq Bt_i \leq Ct_i. \tag{8}$$

We define a truck load as the ratio of the number of bikes over total capacity:

$$Lt_i = \frac{Bt_i}{Ct_i}, \tag{9}$$

This leads to the property:

$$0\% \leq Lt_i \leq 100\%. \tag{10}$$

If, at a moment in time τ, a truck t_j performs rebalancing at a station s_i, the evolution of the number of bikes is as follows:

$$\Delta Bt_j = Bt_j(\tau) - Bt_j(\tau - 1) = Bs_i(\tau - 1) - Bs_i(\tau) = -\Delta Bs_i. \tag{11}$$

The interaction between rebalancing type and internal variation of the number of bikes is:

$$\Delta Bt_i \begin{cases} > 0, if\ JTi = \text{“}Extract\text{”} \\ < 0, if\ JTi = \text{“}Insert\text{”} \end{cases}. \tag{12}$$

$U = \{U1, U2, \ldots, UP\}$ corresponds to a set of P depots, $P < N$, which provide the following amenities for trucks: parking and bike loading or unloading. Each depot $U_i = \left(S^i, T^i, Cu_i, \kappa_i^\tau\right)$, has a capacity Cu_i of stored bikes. The depot is assigned to a truck belonging to the partition $T^i \subset T$ and to a bike station to rebalance. This bike station belongs to the partition $S^i \subset S$, according to the assignment function $\kappa_i^\tau : T^i \to S^i$, at a given time τ.

The capacity of a depot is set using the following formula:

$$Cu_i = \sum_j Cs_j \cdot \sigma_{S^i}(j) \cdot \sigma_R(j), \tag{13}$$

where Cs_j was already defined as the capacity of the station s_j.

$\sigma_{S^i}(j)$ and $\sigma_R(j)$ are functions defined as follows:

$$\sigma_{S^i}(j) = \begin{cases} 1, & s_j \in \mathrm{S^i} \\ 0, & else \end{cases}, \tag{14}$$

and respectively,

$$\sigma_R(j) = \begin{cases} 1, & Ls_j < 10\%\ OR\ Ls_j > 90\% \\ 0, & else \end{cases}, \tag{15}$$

So, σ_{S^i} is used to select only stations that are part of the partition, including the assigned depot. In contrast $\sigma_R(j)$ are used to select only stations that needed rebalancing.

Since the depot size is not adjusted each time, a new rebalancing operation happens, eventually modifying the result of the formula for capacity. A worst-case size was determined based on the collected data. In this way, we eliminate the non-deterministic σ_R, by oversizing the depot. The worst-case size was defined as the maximum percentage of stations needing rebalancing, weighted by capacity. By applying the depot capacity formula to the worst-case scenario, we obtain:

$$Cu_i = 30\% \sum_j Cs_j \cdot \sigma_{S^i}(j). \tag{16}$$

In other words, the depot size is 30% of the cluster's total capacity of all bike stations. The depot's capacity was computed such that at the start it is half full, containing an equal number of bikes and parking lots.

The general assignment function, κ, represents a mapping between trucks and stations that varies in time. The mapping can always be formulated as follows:

$$\kappa = \begin{bmatrix} a_1^1 & \cdots & a_N^1 \\ \vdots & \ddots & \vdots \\ a_1^K & \cdots & a_N^K \end{bmatrix}, \tag{17}$$

where a_x^τ represents the station index which is assigned to a truck index x at the moment τ. N is the total number of trucks in the system, and K is the total amount of time samples collected. Each assignment function Λ_i^τ over time corresponds to the τ-th row of a partition (represented by the set of columns), defined as a cluster of Λ, visualized in Figure 4:

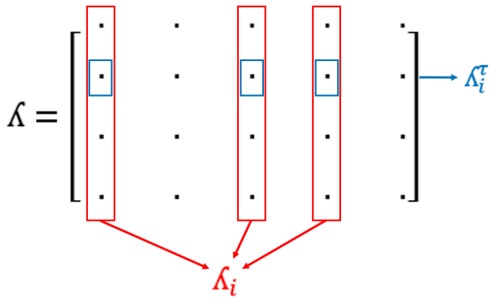

Figure 4. Truck to station assignment: general (Λ), cluster (Λ_i), and chain of instructions (Λ_i^τ).

We assume we have a fixed number of trucks that ensure the rebalancing, as described by the subset T. A rebalancing activity has two steps:

- transport to a station.
- extract or insert bikes into the station.

Similarly, each truck has a variable number of bikes and a fixed capacity. Furthermore, like the bike stations, a truck can be empty/full, in which case it loses the flexibility to balance stations for both types of demand; a full truck can only insert bikes at empty stations, and an empty truck can only extract bikes from full stations. This loss of flexibility translates to two possible adverse outcomes: some stations will experience long periods of being full/empty, threatening the promising prospects of the system being used; truck travel distance is increased because initial rebalancing opportunities are restricted.

To restore the truck flexibility, we have a fixed number of local depots, described by the subset U, each of which is always able to provide or extract bikes from trucks. Therefore, in this situation, a truck will go to the bike station through a depot. If this number of depots is high enough and they are correctly spatially distributed, it is more economically feasible for a truck to go through the depot instead of rejecting the order and requesting a new order for a different bike station.

The bike stations and trucks are clustered around the depots, as shown in Figure 5.

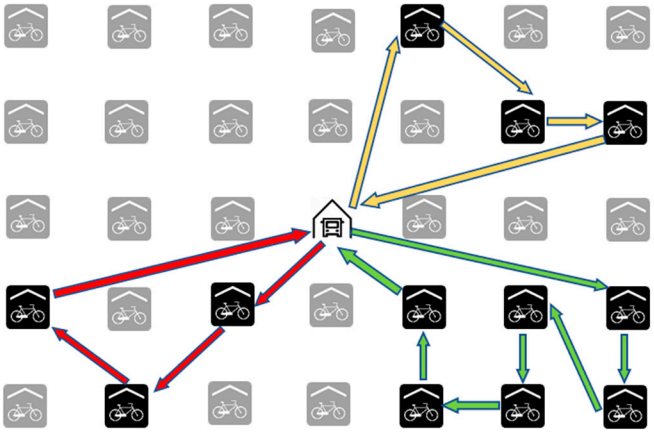

Figure 5. General representation of the routing scheme around a depot.

The bike stations are spatially clustered around the depots. Two steps of decision-making are associated with each cluster: first, a sorting operation is performed, resulting in a list of all bike stations that need rebalancing; then, each element of the sorted list is assigned to a truck.

The result of the decision process consist of a set of routes allocated to trucks:

If multiple trucks are considered, a minimum duration is required for each individual route (duration objective), while at the same time achieving a uniform allocation of tasks among trucks (uniformity objective).

The result of the decision process is summarized in the table presented below, in Table 2:

Table 2. The table for truck order assignment.

Truck Index	Station Origin Index	Station Destination Index	Number of Carried Units	Service Type
1	585	731	21	Extract
2	115	841	26	Extract
3	613	572	15	Insert
2	841	433	20	Insert
3	572	948	21	Extract
3	948	359	10	Extract
1	731	204	19	Insert

Therefore, the presented rebalancing strategy has the required characteristics that enable the method to be applied to BSS having dozens or hundreds of bike stations and is characterized by dynamic behavior.

4. Materials and Methods

4.1. The Proposed Solution

A genetic algorithm, which provides the ordered list, was developed to solve the bike station prioritization problem.

To obtain a better convergence to the final solution, a fuzzy logic control strategy, derived from the work presented in [11], was employed to adaptively modify the mutation and crossover probabilities during the optimization process.

Based on a list of rules, a truck-to-bike station assignment algorithm was conceived to solve the assignment component of the problem.

The bike station parameters were extracted from the Citi Bike New York BSS, and all the evaluations were performed in a real-world context. Moreover, due to the fact that the algorithm has all the selected parameters independent of BSS parameters, it has the potential to be applied to any other BSS.

More detailed aspects of the proposed solution are presented below.

4.2. The FLCGA Algorithm

This chapter presents the key components of the proposed method for bike rebalancing activities that has been developed to obtain the ordered list (described in Section 3).

Figure 6 is a schematic representation of the method in which a static rules list specifying rebalancing constraints, along with dynamic traffic data, constitute the input of a decision algorithm; the genetic algorithm is used to determine the order of the services. Truck assignment rules are then applied to obtain the Truck Scheduling List.

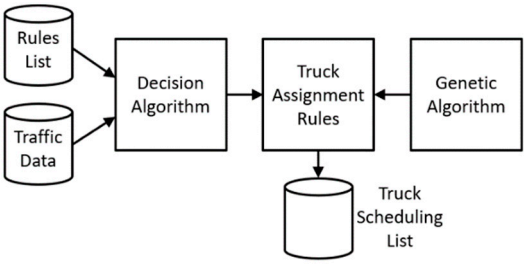

Figure 6. The schematic representation of the proposed method.

4.2.1. Rules List

The rules list gives all the details needed for the assignment of trucks to the subsequent stations, as follows in Figure 7:

```
1. For each station, the truck with the shortest planned path at any given
moment has priority for assignment.
2. If more than one truck has the shortest planned path, any truck is assigned
randomly.
3. If the assigned truck is unable to fulfill the bike delivery/pickup due to
capacity constraints, it will travel through depot. Transport cost computation
is adapted accordingly.
4. A station demands a quantity of bikes equal to half of its total capacity.
5. Trucks do not collide.
6. Trucks do not meet to make bike exchanges.
```

Figure 7. The rules list.

4.2.2. Traffic Data

The traffic data is composed of information regarding three types of elements:

1. bike-sharing stations,
2. trucks (the rebalancing agents),
3. depots.

Figure 8 provides a visual interpretation of an example of the input data configuration. The black numbers represent the station index, and the blue numbers represent the station capacity.

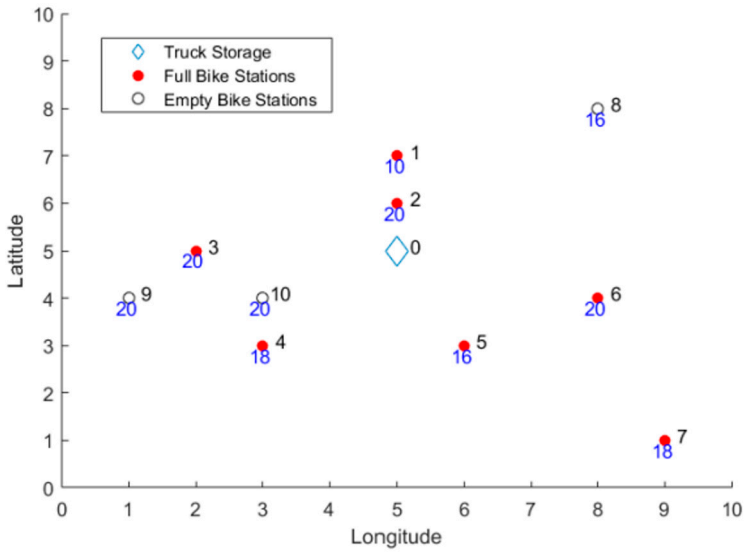

Figure 8. Geographical representation of a hypothetical subregion of a bike-sharing system.

4.2.3. Decision Algorithm

The decision algorithm for the truck assignment, which complies with the rules list, is expressed in Algorithm 1:

Algorithm 1. The decision algorithm for truck assignment.

```
for station in S do
  truck_assigned ← FALSE
  truck ← get_first_element(T)
  repeat
    journey ← get_journey_history(truck)
    if MINIMUM = length(journey) then
      pair(truck, station)
      if able_to_fulfil_demand(truck)
        route ← distance(truck, station)
      else
        route ← distance(truck, terminal) + distance(terminal, station)
      end
      journey ← journey + route
      update_journey_history(truck, journey)
      truck_assigned ← TRUE
    else
      truck ← get_next_element(T)
    end
  until truck_assigned = TRUE
end
```

The algorithm considers it a special event when the truck misses its capacity to directly serve the next bike station. The additional visits to the depot will lead to an increase in the total cost of the trip.

4.2.4. Genetic Algorithm

The block diagram of the FLGCA proposed for the bike-sharing rebalancing problem is given in Figure 9:

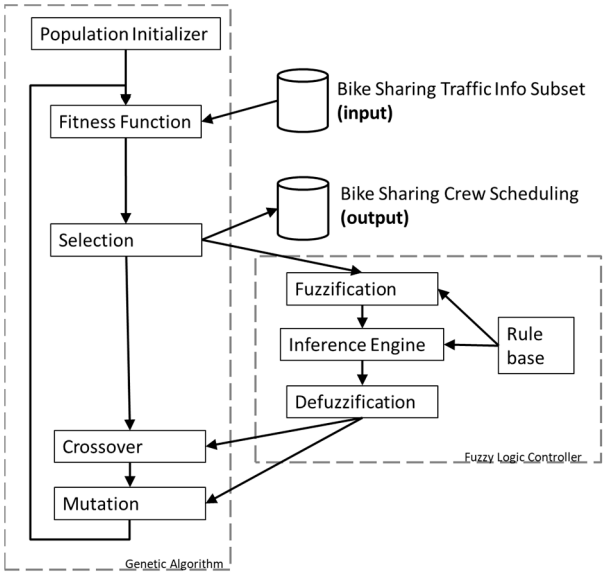

Figure 9. The block diagram of the FLCGA.

The diagram contains two main parts, corresponding to the genetic algorithm (GA) and the fuzzy-logic controller (FLC).

The controller dynamically adjusts the crossover and mutation probabilities for the next generation based on the statistics of the current candidate solution.

The result of the FLCGA is a cost-effective way to serve bike stations that need rebalancing.

Additional details about the components and their roles will be presented in the following chapters.

4.2.5. Crew Scheduling List

The assignment information attached to the execution order is obtained as a final result of the method. In these conditions, it must be mentioned that, at the output of FLCGA, it is not specified how the scheduling list will be partitioned to be distributed to the trucks.

Figure 10 presents a simplified interpretation of the output of the proposed solution.

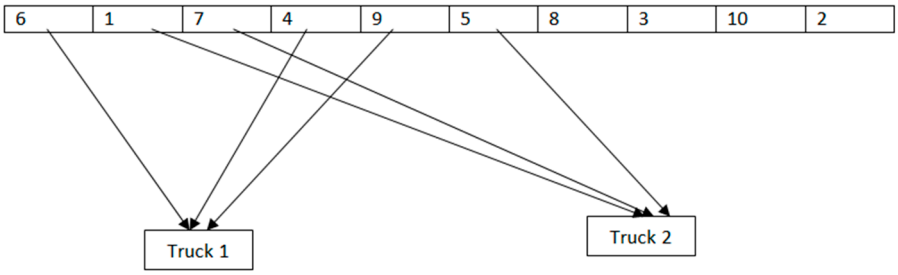

Figure 10. Simplified interpretation of the output.

The crew scheduling list is a numerical mapping of commands, where an individual element represents a rebalancing activity at a specific bike station.

4.3. GA Parametrization

This section presents details regarding the internal functionality of the genetic algorithm.

4.3.1. Population Initializer

The chromosome of the GA stores sequences of indexes and represent the order in which the stations need rebalancing. Therefore, each chromosome stores a sequence of instructions given to a set of trucks, decided which truck will visit which bike station, as exemplified in Figure 11.

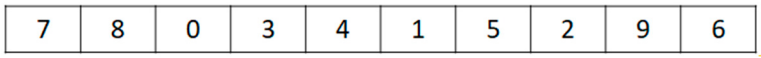

Figure 11. Graphical representation of a chromosome.

Further increasing the population size above 50 chromosomes did not improve the fittest candidate solution, so it has been decided to keep the population size at 50 individuals.

Initialization was performed by randomly generating permutations of station indexes.

4.3.2. Fitness Function

The fitness function model evolved from the initial idea to its final form as follows.

When the instruction chain is completed by a single truck, the efficiency score is increased at each instruction by the number of bicycles served and decreased by the distance traveled:

$$f(T,S) = \sum_{i=1}^{N} \frac{r_i}{d_{i,i-1}}, \tag{18}$$

where f is the function that describes the efficiency score; parameters T and S correspond to the vector of available trucks $T_1, \ldots, T_i, \ldots, T_M$, respectively, the vector of rebalanced stations $S_1, \ldots, S_i, \ldots, S_N$; N is the number of trips; i is the index of the current trip; r_i corresponds to the number of bikes transported; $d_{i,i-1}$ is the distance travelled between the previous station served and the current station served; $d_{1,0}$ is the distance travelled between the depot and the first station.

When more than one truck executes the chain of instructions, there is a difference between the previous instruction received by the truck ($k(i)$) and the last instruction of the general chain ($i-1$):

$$f(T,S) = \sum_{i=1}^{N} \frac{r_i}{d_{i,k(i)}}. \tag{19}$$

To calculate this efficiency score, all rebalancing instructions are ordered chronologically, regardless of which truck executed them. For each direct trip, the distance ($d_{i,k(i)}$) is between the bike station for which the previous instruction was given and the bike station corresponding to the current instruction (index i). The distance is stored at the same time as the number of bikes served (r_i).

In the case in which a truck is considered to have a reduced capacity, it will go through the depot to adjust its load. This is expressed as follows:

$$d(i,j) = \begin{cases} d_{i,j}, & \textit{truck has enough capacity} \\ d_{i,0} + d_{0,j}, & \textit{otherwise} \end{cases}. \tag{20}$$

Thus, the relation can be rewritten to take this into account:

$$f(T,S) = \sum_{i=1}^{N} \frac{r_i}{d(i,k(i))}. \tag{21}$$

However, if the time constraint is introduced, the expression of the fitness function will be accordingly modified:

$$f(T,S) = \min_{x} \sum_{i=1}^{N} \frac{r_i \cdot \sigma(x,i)}{d(i,k(i))}, \tag{22}$$

where σ is the assignment function for instructions to trucks:

$$\sigma(x,i) = \begin{cases} 1, & \textit{truck x executed instruction i} \\ 0, & \textit{else} \end{cases}. \tag{23}$$

The efficiency score has been calculated based on the concentration of services over a minimum distance.

To avoid the situation in which stations with low profitability will be prone to being ignored, a scaling factor is added based on the percentage of bikes whose orders have been fulfilled.

$$f(T,S) = \frac{\sum_{i=1}^{N} r_i}{R} \cdot \min_{x} \sum_{i=1}^{N} \frac{r_i \cdot \sigma(x,i)}{d(i,k(i))}, \tag{24}$$

where R is the total system demand, expressed as the number of bicycles to be relocated by all trucks.

In order to obtain immunity to noise or sudden variations in bike numbers, the fitness function becomes:

$$f(T,S) = \frac{\sum_{i=1}^{N} r_i}{\sum_{i=1}^{N} d_{i,i-1}}. \tag{25}$$

For several trucks, the formula will be accordingly modified:

$$f(T,S) = \frac{\sum_{i=1}^{N} r_i}{\sum_{i=1}^{N} d(i,k(i))}. \tag{26}$$

The objective of a uniform distribution of tasks for each truck is reflected as follows:

$$f(T,S) = \frac{\sum_{i=1}^{N} r_i}{\max\limits_{x}\left(\sum_{i=1}^{N} d(i,k(i))\cdot\sigma(x,i)\right)}. \tag{27}$$

To reflect the situation where there is unsatisfied rebalancing demand, the model is adapted accordingly:

$$f(T,S) = \frac{\left(\sum_{i=1}^{N} r_i\right)^2}{R\cdot\max\limits_{x}\left(\sum_{i=1}^{N} d(i,k(i))\cdot\sigma(x,i)\right)}. \tag{28}$$

The final version of the fitness function is formulated as follows:

$$f(T,S) = \frac{\sum_{i=1}^{N} r_i\cdot\sigma(i)}{\max\limits_{i} D_i}. \tag{29}$$

where T and S correspond to the vector of available trucks $T_1, \ldots, T_i, \ldots, T_M$, respectively, and the vector of stations to be rebalanced $S_1, \ldots, S_i, \ldots, S_N$. r_i denotes the number of bikes added or subtracted for S_i rebalancing. In contrast, D_i is the total sum of travelled distances of T_i.

Before setting this fitness function as final, other candidates were considered, most notably being:

$$f(T,S) = \min\limits_{i}\frac{\sum_{i=1}^{N} r_i\cdot\sigma(i)}{D_i}. \tag{30}$$

4.3.3. Selection

The binary tournament was the chosen selection method. It is less biased for individuals with higher fitness functions. Runtime and memory allocation efficiency are ensured by the avoidance of sorting the individuals. Furthermore, this method is usually preferred in many genetic algorithms for crew scheduling [11]. This selection method has the following steps: randomly select two individuals from the population and choose the fittest one between the two, who will represent the first parent; to obtain the second parent, the process is repeated.

4.3.4. Crossover and Mutation

Since a one- or two-point crossover is not guaranteed to avoid bike station index duplicates in the offspring, which translates to an invalid route, the Order Crossover (OX) method is used instead. The method can be described as follows. The alleles from a swath of parent 1 are reordered in the order they appear in parent 2. The alleles from parent 2, used for this ordering, are also reordered in the order they originally appeared in parent 1. Randomly selected chromosomes are paired for a crossover with an initial probability of 80%. This probability will later change according to the fuzzy logic controller's (FLC) output.

An interchange of two random genes represents a mutation. Randomly selected genes have an initial probability of being 30% mutated. This probability will later change according to the FLC output.

4.3.5. Termination Condition

The genetic algorithm has two alternative stopping conditions: a maximum number of iterations (100) is reached or the performance improvement hits a plateau. The plateau is reached when the number of iterations for which the fitness function of the best individual was not improved represents 35% or more of the total number of iterations. We call this ratio the stagnation ratio (SR), and it is expressed as a percentage.

Special attention has been given to the fact that the mutation and crossover probability values are controlled when the SR is set. If SR is set to a low level, the opportunity for FLC to react upon reaching a local solution is lost.

This kind of termination criteria, comprised of two conditions, one of a maximum number of iterations and another of a plateau, is typical for applications in routing problems [57].

4.4. The Fuzzy Logic Controller-Based Approach

Similar to the algorithm devised in [11], both the crossover and mutation rates are manipulated. Further fine-tuning was performed experimentally. Hence, a substantial level of diversity can be achieved. The applied technique is based on a modification of the algorithm proposed by [11] and is presented in more detail below.

Wang et al. [58] were among the pioneers who introduced FLC into GA to improve performances. The crossover and mutation operations handle solution space exploitation and exploration, respectively. A poor selection of parameters will reduce the diversity in the population, leading to either premature convergence or no convergence at all. Mutation rates that are too high will impede convergence, while those that are too low will lead to no convergence.

Due to the fact that current work is aimed at NP-optimization on large BSS featuring a high level of dynamicity, it is hard to extract crisp rules by observing the uncontrolled GA behavior. Due to the ever-changing nature of customer demands, any set of crisp rules would apply only to a limited number of stations or a limited time frame. Therefore, in our case, it is needed to manipulate the GA parameters during runtime and to do it in a manner that handles the uncertainty and imprecision, for which FLC is suited.

Running time and scalability of the algorithm are important due to real-time constraints and the size of the system, which can also grow in the future. Plerou et al. [59] mention the FLCGA as the most efficient in solving problems of scheduling compared to the standard GA. For multiple types of problems, the superiority of FLCGA over many other algorithms in running time and scalability is highlighted in [60].

Below we will define each statistic measure computed to provide input/output for the FLC.

4.4.1. FLC Inputs and Outputs

Related to the above description of the FLCGA, the inputs of FLC are obtained from the GA attributes. These attributes are derived from the fitness function results.

The FLC inputs are as follows: CF is the increase in the fitness function from one iteration to the next, and VF is the variance of the fitness function inside the population from the current iteration. At the same time, UF represents the number of iterations for which the fitness function was not improved.

CF and VF have the following formulas:

$$CF = \left(\frac{Cost_{best(t-1)}}{Cost_{best(t)}} - 1 \right) \cdot 100\%, \tag{31}$$

$$VF = \frac{Cost(t) - Cost_{best(t)}}{Cost_{best(t)}}. \tag{32}$$

where $Cost_{best(t)}$ represents the inverse of the fitness function of the best candidate of generation t. A candidate can be called the best when it has the highest fitness value compared to the rest of the same generation.

The FLC outputs, Δpm and Δpc, represent the amount of change in the mutation and crossover rates, respectively.

4.4.2. Membership Functions

The fuzzy rules applied are derived from those presented in [11]. Three linguistic variables {Low, Medium, and High} are used. The corresponding membership functions for the fuzzification of the input and output variables of the controller can be visualized in Figure 12. After computing the parameters and performing centroid defuzzification, the controller sends to the genetic algorithm the new mutation and crossover rates. These are applied to generate the latest iteration of the algorithm.

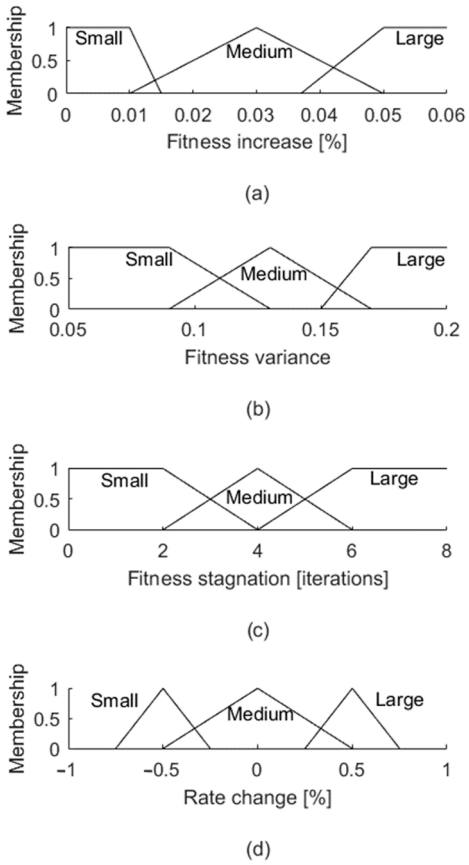

Figure 12. (**a**) CF membership functions. (**b**) VF Membership functions. (**c**) UF membership functions. (**d**) Δpm and Δpc membership functions.

The values of control parameters Δpm and Δpc are obtained by applying if-then fuzzy [11] rules, for example:

If CF = High and UF = Low → Δpm = Low and Δpc = High.

If UF = High and VF = Medium → Δpm = High and Δpc = Low.

5. Results

5.1. Quick Overview

An example of routes assigned to trucks resulting from running the application for a subsystem comprised of five trucks and twenty stations is given below, in Table 3:

Table 3. Routes and load factors.

Truck ID	Route	Route Length [km]	Load Factor [%]
1	$0 \rightarrow 7 \rightarrow 16 \rightarrow 4 \rightarrow 20 \rightarrow 0$	14.34	72
2	$0 \rightarrow 17 \rightarrow 23 \rightarrow 12 \rightarrow 0 \rightarrow 15 \rightarrow 9 \rightarrow 14 \rightarrow 0 \rightarrow 3 \rightarrow 2 \rightarrow 0$	13.80	86
3	$0 \rightarrow 6 \rightarrow 10 \rightarrow 8 \rightarrow 0$	16.96	53
4	$0 \rightarrow 11 \rightarrow 19 \rightarrow 13 \rightarrow 18 \rightarrow 0$	15.35	35
5	$0 \rightarrow 1 \rightarrow 0 \rightarrow 5 \rightarrow 0$	12.55	64

Figure 13 shows the performance of the applied method by simulating 1000 possible subsystems; the red line corresponds to the median value of the histogram. A subsystem (cluster) is comprised of randomly chosen stations from the Citi Bike system.

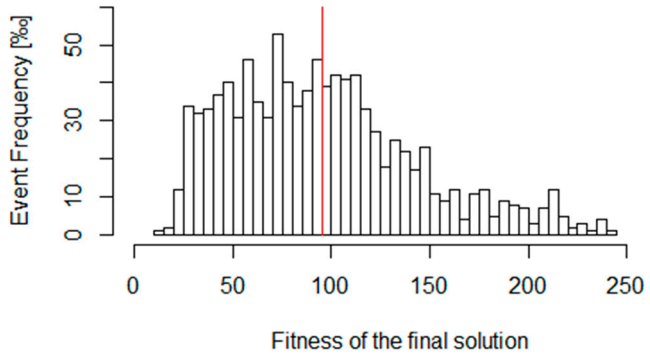

Figure 13. Histogram of the performance of the FLCGA approach.

Each value added to the histogram corresponds to the final fitness value obtained for each subsystem.

The convergence to the final solution for the median performance obtained can be observed in Figure 14.

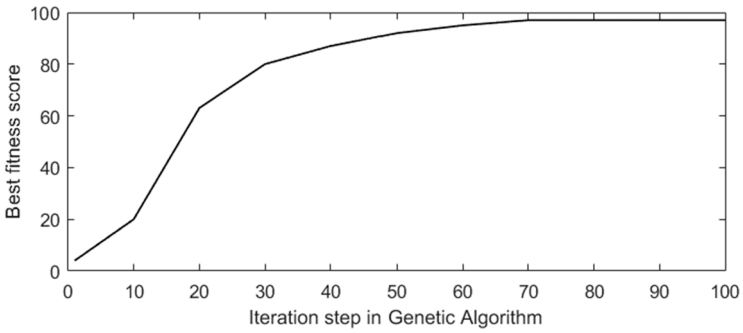

Figure 14. Best fitness score evolution throughout multiple generations.

Each subsystem was generated by employing a window search through the data of the geographical location (latitude and longitude) of bike stations. In this context, the window search represents the random positioning of a fixed-size window on the map of bike stations from the BSS. The window size has been chosen to capture a constant number of stations. If a window covers more stations than desired, a random selection of residual stations to be removed from the sample is performed. The procedure is exemplified in Figure 15.

Figure 15. Example of window search performed on bike stations of Citi Bike, New York.

5.2. Comparative Experimentation and Analysis of FLC

The experiments were repeated for different versions of the membership functions of the FLC, considered FLC instances. The repetition is aimed at discovering the optimal FLC instance, i.e., the one with the highest median fitness score.

Table 4 summarizes the results for the FLC instances.

Table 4. Results for the FLC instances.

Instance	Median Fitness Score	10th Percentile	90th Percentile	Standard Deviation
Inst0	88.32	10.36	107.22	17.09
Inst1	79.12	65.40	100.31	10.51
Inst2	12.44	2.33	20.14	5.11
Inst3	56.12	11.53	78.32	10.72
Inst4	90.32	30.78	189.64	52.18
Inst5	94.10	37.69	155.14	38.42
Inst6	92.13	37.65	155.27	35.60
Inst7	92.17	32.71	149.71	38.28
Inst8	93.29	33.14	169.31	55.16
Inst9	94.74	26.32	142.62	45.13
Inst10	94.85	21.16	158.34	46.45
Inst11	95.07	39.55	142.45	44.14
Inst12	95.22	44.08	168.97	47.31

By analyzing the results from Table 4, it can be observed that the fine-tuning of the parameters of the membership functions led to a trend of an overall increase in performance. There are two exceptions: instance Inst2 shows a massive drop in performance due to a too-high threshold for large UF parameter selection. Instance Inst6 also features a slight decrease in performance.

Since instance Inst4, a stable performance was reached (the median fitness score being 95% of the best median captured at instance Inst12).

Since instance Inst8, the choice of the parameters contributes to the reduction of the standard deviation.

5.3. Scalability Analysis

In order to analyze the scalability for different numbers of stations that need to be rebalanced, the average number of iterations of the genetic algorithm needed to reach stability was stored. Stability is defined as the state related to the moment when the performance score of the best individual remains greater than or equal to 97% of the final value. The number of trucks has no influence over the speed of the genetic algorithm because, under the presented method, the assignment of trucks is a subsequent operation based on the already obtained results of the genetic algorithm.

As it can be seen in Table 5, the increase in convergence time, measured by the number of generations, needs to scale up with the number of bike stations involved. For each additional five stations, two extra iterations are required on average, i.e., an additional

2.85% from the initial cluster size. Given that the FLCGA implementation consumes most of the runtime, we can conclude that a constant 2.85% runtime offset for each consecutive five stations provides good system scalability from this point of view. For example, if the cluster size is increased 90-fold, from 10 stations to 900 stations (roughly the size of Citi Bike New York), the runtime consumption will increase only 5-fold.

Table 5. The average speed of the genetic algorithm for different cluster sizes.

Number of Bike Stations	The Average Speed of the Genetic Algorithm [Number of Iterations]
10	70
15	72
20	77
25	78
30	82
35	81
40	83
45	85
50	86

5.4. Comparative Experiment and Analysis with Ant Colony Optimization

One of the classical approaches for this category of applications is Ant Colony Optimization (ACO) [61], which was chosen as the method to compare with. The same experiments performed before were reproduced using ACO.

The relevant advantages of ACO, which make it suited as a reference for our application, are its useability in dynamic applications, the fast discovery of a proper solution, its inherent parallelism, and its efficiency when dealing with routing problems [62].

ACO was preferred over other traditional routing methods, like Open Shortest Path First (OSPF) or Routing Information Protocol (RIP), because it uses less information storage, causes less overhead, and reacts faster to system changes [63]. Information storage and overhead are essential to be addressed because of the high number of stations involved. A quick reaction to changes in the system state is also required given that the stations' loads are ever-changing, potentially shifting the priority of rebalancing from one station to another.

Moreover, ACO has higher sensitivity and specificity, i.e., better performance, compared to other methods [64] for selection tasks.

ACO is a multi-agent probabilistic technique for obtaining a good path through graphs inspired by real ants' behavior [12]. Each agent, called an "ant", will generate multiple possible ways through several iterations. The best path obtained is selected at the end of all iterations.

At each step of path construction, the ant will decide where to go next based on previous decisions of the whole colony and a priori knowledge about the desirability of each possible next move. This desirability score increases along with the size of the bike set to be served and decreases for a longer distance to be covered:

$$\eta_{ij} = \frac{r_j}{d_{ij}}, \tag{33}$$

where η_{ij} is the desirability score of traveling from the current station (i) to the candidate station (j); d_{ij} is the distance in between them and r_j is the number of bikes to be served at the station j.

This a prior knowledge, combined with previous experience, gives the probability information that the ant k, currently at station i, will travel to station j; it is expressed in the following equation:

$$p_{ij}^{k}(t) = \frac{\left(\tau_{ij}^{\alpha}\right)\left(\eta_{ij}^{\beta}\right)}{\sum_{l \in S}\left(\tau_{il}^{\alpha}\right)\left(\eta_{il}^{\beta}\right)}, \tag{34}$$

where α and β specify the importance of the pheromone trail τ_{ij} against the heuristic information η_{ij}; S is the set of all station indexes. The critical parameters were set as follows, $\alpha = \beta = 1$.

Once all the ants have found their solution, the pheromone trail is reinforced according to the following formula:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k}^{m} \Delta\tau_{ij}^{k}, \tag{35}$$

where ρ is the pheromone evaporation coefficient, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the pheromone amount deposited by the kth ant:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{d_{ij}} \sum_{l \in S} r_l & \text{ant } k \text{ travelled from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}, \tag{36}$$

With r_l and d_{ij} defined for Equations (3) and (4). The number of ants was set to $m = 5$, the number of iterations was set to 100, and the evaporation coefficient was set to $\rho = 0.5$.

The method of validation of the proposed algorithm against the reference algorithm is as follows:

1. For the same input data, a classical ACO algorithm is used to obtain the output in the form of a sorted list of stations to be served in the same format;
2. The same fitness function presented in the previous chapters is used;
3. The performances are compared.

The obtained results prove that the FLCGA-based approach provides equal or superior results, as shown in Figure 16.

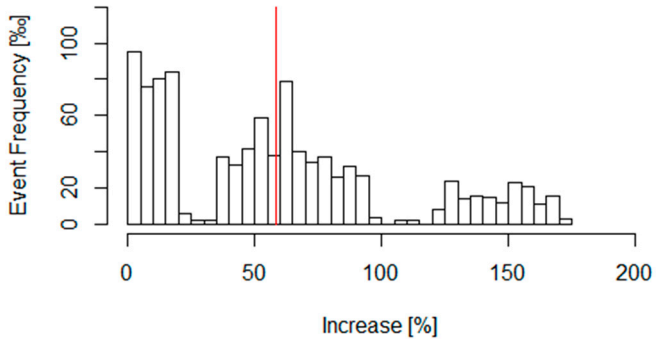

Figure 16. Histogram of relative performance (FLCGA compared to ACO).

It can be concluded that the average performance increase is 57%. At the same time, independently, ACO demonstrated a good performance.

The relative increase in performance was computed by using the following formula:

$$\Delta_{Perf}(i) = \frac{Perf_{FLCGA}(i) - Perf_{ACO}(i)}{Perf_{ACO}(i)}. \tag{37}$$

where i represents the index of the random subset of the BSS; $Perf_{FLCGA}$ and $Perf_{ACO}$ describe the performance of the FLCGA and the ACO implementation, respectively.

The detailed results for the whole set of experiments are summarized in Table 6a–d.

Table 6. (**a**) Performance of the FLCGA for different numbers of bike stations to rebalance. (**b**) Performance of the ACO for different numbers of bike stations to rebalance. (**c**) Performance of the FLCGA for different numbers of rebalancing trucks. (**d**) Performance of the ACO for different numbers of rebalancing trucks.

Number of Trucks	Number of Bike Stations	Median Fitness Score [Bikes/km]	Standard Deviation [Bikes/km]	10th Percentile Fitness Score [Bikes/km]	90th Percentile Fitness Score [Bikes/km]	Cumulative Distance of Rebalancing Routes [km]
(**a**)						
2	10	9.52	2.33	4.4	16.89	39.72
2	15	11.9	3.01	4.74	20.49	50.94
2	20	13.85	4.28	5.16	23.63	60.64
2	25	15.52	5.82	5.43	28.07	69.83
2	30	17.347	6.77	5.21	30.12	77.53
2	35	18.15	8.41	5.84	33.02	87.28
2	40	19.03	7.22	6.19	34.14	95.01
2	45	19.78	8.95	6.71	32.45	101.36
2	50	20.41	8.36	6.55	37.05	105.43
(**b**)						
2	10	5.91	1.06	4.18	10.17	52.27
2	15	7.38	1.59	4.21	12.36	78.21
2	20	9.79	1.67	4.71	12.78	78.39
2	25	10.29	3.99	3.78	15.74	104.47
2	30	12.58	3.35	4.57	16.91	109.5
2	35	13.07	3.69	4.58	17	134.73
2	40	12.69	3.22	5.24	17.83	114.38
2	45	11.96	2.83	5.2	18.66	139.9
2	50	14.34	4.09	5.82	18.91	134.86
(**c**)						
2	50	20.41	8.36	6.55	37.05	105.43
3	50	33.79	15.97	10.29	62.75	114.75
4	50	33.85	17.94	9.42	67.19	135.73
5	50	35.72	18.36	9.13	69.37	147.51
6	50	40.21	17.53	12.9	71.59	166.84
7	50	42.47	19.61	11.53	73.71	185.97
8	50	45.93	20.03	13.87	78.26	203.48
9	50	43.3	20.84	12.58	75.28	213.39
10	50	44.63	22.56	14.39	78.04	231.62
(**d**)						
2	50	14.34	4.09	5.82	18.91	134.86
3	50	25.3	14.86	8.26	49.98	144.22
4	50	28.74	16.11	7.93	51.52	163.18
5	50	31.3	15.52	7.49	63.74	177.76
6	50	34.34	13.7	11.19	53.98	213.99
7	50	38.08	18.38	8.96	58.22	251.56
8	50	35.21	20.71	13.03	65.58	253.67
9	50	39.8	19.75	10.1	63.77	251.59
10	50	38.68	18.29	12.9	66.23	292.42

By analyzing the presented results, it can be concluded that the performances of FLCGA remain consistently higher than those of ANT for each experiment. The standard deviation of ANT results is slightly better, but the median version of FLCGA is even more significant, compensating for the expected deviation loss.

By increasing the number of bike stations, we observe that the standard deviation of both FLCGA and ANT results is growing faster than the median performance, but not significantly.

An increase in median performance exists when increasing either the number of trucks or bike stations. By keeping the cluster area fixed, the density of resources is increased, i.e., the routes for rebalancing become shorter. Moreover, as expected, cumulative route distance and the median fitness score are growing.

5.5. Comparative Experiment and Analysis with Harris Hawks Optimization

HHO [13] was implemented and evaluated against the same dataset and scenarios as for the other algorithms included in this paper.

HHO was included in the proposed set of algorithms considered in our work because it has proven superiority over other large-scale applied metaheuristic algorithms considering accuracy and speed in the case of real-world optimization problems [65]. Moreover, HHO shows promising results for problems that feature a large number of dimensions [13], this algorithm being suited for applications where scalability is demanded.

Both phases of HHO, the exploration phase and the exploitation phase, are inspired by the real-world behavior of hawks hunting. The exploration phase corresponds to randomly searching for prey, while the exploitation phase corresponds to capturing the prey by exhaustion.

During the exploration phase, the model of the strategy of how hawks detect the prey by perching on a random tree (X_{rand}) is:

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)|, & q \geq 0.5 \\ \left(X_{prey}(t) - X_m(t)\right) - r_3(LB + r_4(UB - LB)) & q < 0.5' \end{cases} \quad (38)$$

where $X(t)$ and $X(t+1)$ are the positions of the hawks in the current and, respectively, next iteration; $X_{rand}(t)$ is the position of a randomly selected hawk; $X_{prey}(t)$ is the position of the prey; $X_m(t)$ is the average position of the hawks in the current iteration; r_1, r_2, r_3, r_4 and q are random numbers inside $(0, 1)$; LB and UB are the lower and upper bounds of the variables.

The algorithm will transition from the exploration phase to the exploitation phase based on the escaping energy of the prey, modelled as:

$$E = 2E_0\left(1 - \frac{t}{T}\right) \quad (39)$$

where E represents the escaping energy of the prey; E_0 corresponds to an initial state of energy randomly changing its value inside the interval $(-1, 1)$; t is the current iteration, and T is the maximum number of iterations.

Once the exploitation phase begins, different encircling strategies, so called besieges, are employed from the following set: soft besiege, hard besiege, soft besiege with progressive rapid dives, and hard besiege with progressive rapid dives. The selection of the type of besiege is conditioned by escaping energy, E, and the change of prey escaping, r, randomly changing values in the interval (0, 1).

In the context of HHO, two parameters of the method—the number of search agents (hawks) and the maximum number of iterations—were decided according to the needs of our application. Several deviations from the classical setting of parameters (30 hawks, 500 maximum number of iterations) were employed for the experiments before it was decided that for our application, 32 hawks and 562 maximum number of iterations lead to the best observed performance.

To compare the performances of FLCGA and HHO algorithms, an already defined set of experiments was performed. The obtained results are summarized in Table 7.

Table 7. (**a**) Performance of the FLCGA for different numbers of bike stations to rebalance. (**b**) Performance of the HHO for different numbers of bike stations to rebalance. (**c**) Performance of the FLCGA for different numbers of rebalancing trucks. (**d**) Performance of the HHO for different numbers of rebalancing trucks.

Number of Trucks	Number of Bike Stations	Median Fitness Score [Bikes/km]	Standard Deviation [Bikes/km]	10th Percentile Fitness Score [Bikes/km]	90th Percentile Fitness Score [Bikes/km]	Cumulative Distance of Rebalancing Routes [km]
(**a**)						
2	10	9.52	2.33	4.40	16.89	39.72
2	15	11.90	3.01	4.74	20.49	50.94
2	20	13.85	4.28	5.16	23.63	60.64
2	25	15.52	5.82	5.43	28.07	69.83
2	30	17.347	6.77	5.21	30.12	77.53
2	35	18.15	8.41	5.84	33.02	87.28
2	40	19.03	7.22	6.19	34.14	95.01
2	45	19.78	8.95	6.71	32.45	101.36
2	50	20.41	8.36	6.55	37.05	105.43
(**b**)						
2	10	7.36	2.01	3.52	12.40	45.14
2	15	10.30	5.37	4.11	18.66	55.28
2	20	14.52	5.21	4.95	26.84	56.06
2	25	14.98	5.53	5.89	27.38	74.72
2	30	15.31	5.77	5.72	26.29	82.61
2	35	17.20	8.37	6.49	31.44	90.93
2	40	18.57	8.88	6.06	33.26	96.90
2	45	20.58	9.31	6.29	33.56	98.34
2	50	22.82	9.96	8.17	39.81	100.38
(**c**)						
2	50	20.41	8.36	6.55	37.05	105.43
3	50	33.79	15.97	10.29	62.75	114.75
4	50	33.85	17.94	9.42	67.19	135.73
5	50	35.72	18.36	9.13	69.37	147.51
6	50	40.21	17.53	12.90	71.59	166.84
7	50	42.47	19.61	11.53	73.71	185.97
8	50	45.93	20.03	13.87	78.26	203.48
9	50	43.30	20.84	12.58	75.28	213.39
10	50	44.63	22.56	14.39	78.04	231.62
(**d**)						
2	50	22.82	9.96	8.17	39.81	100.38
3	50	32.35	14.22	10.43	60.72	119.68
4	50	35.61	18.90	10.05	69.47	132.09
5	50	38.17	17.50	11.86	68.38	143.28
6	50	40.93	18.00	13.36	70.99	165.97
7	50	44.32	20.26	12.94	73.55	180.54
8	50	46.83	21.42	14.79	80.21	200.69
9	50	47.05	23.23	17.41	78.71	205.80
10	50	48.89	23.17	16.48	80.30	220.66

HHO shows better scalability compared to FLCGA, considering the growth rate of the median fitness score when the number of bike stations increases. The data spread, described by the ratio of standard deviation over median score, is similar for HHO and FLCGA. HHO was always stable, while FLCGA showed only one instance of instability in the scenario of nine trucks and fifty bike stations; stability is defined here as the compliance to a consistent increase in the median fitness score when the number of trucks is increased.

Overall, FLCGA is performing better for scenarios featuring a low number of bike stations, while HHO demonstrated superior performances for scenarios with a high number of bike stations.

5.6. Comparative Experiment and Analysis with Tabu Search Algorithm

The same dataset, scenarios, and evaluation strategy as for the other algorithms included in this paper were used for the comparative evaluation of performance when TSA [14] was included in this study.

The demonstrated ability to provide good-quality solutions to CVRP in a feasible calculation time [66] is the reason for including TSA in the set of alternative algorithms considered. Other relevant advantages of TSA are adaptability, simplicity, robustness, and accuracy [42].

Moreover, by selecting TSA, we have included a deterministic algorithm in the set of algorithms selected for comparison purposes.

The TSA methodology and selected parameters can be briefly described as follows in Figures 17 and 18:

```
1.    Generate the initial solution (an ordered list of bike stations) using a
greedy algorithm.
2.    Generate all neighboring solutions by swapping two bike stations from the
ordered list.
3.    Use greedy procedure to select best neighboring solution, where evaluation
criterion is the fitness function from FLCGA implementation.
4.    A tabu list is used to promote searching the solution space without cycles.
Tabu Tenure is 7.
5.    Aspiration criterion allows to accept a better solution than best so far
even if it is produced by making a tabu move.
6.    The algorithm stopping condition is reaching the maximum number of
iterations; in our case 900, value decided after successive experiments.
```

Figure 17. The TSA steps.

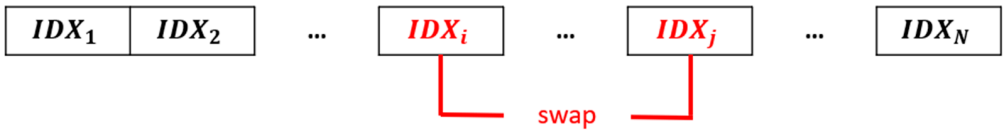

Figure 18. The generation of a neighboring solution.

Table 8a–d contains relevant results considering the usage of FLCGA and TSA under different scenarios.

Based on the above results, the data spread, as reflected by the ratio of standard deviation over median score, is higher for TSA than FLCGA, showing that TSA is less robust in this respect. According to the expectation that the median score increases when the resource number increases, the behavior of FLCGA and TSA is similar. Even though TSA proved to provide acceptable results, in almost all scenarios, FLCGA shows greater median performance values than TSA.

Table 8. (**a**) Performance of the FLCGA for different numbers of bike stations to rebalance. (**b**) Performance of the TSA for different numbers of bike stations to rebalance. (**c**) Performance of the FLCGA for different numbers of rebalancing trucks. (**d**) Performance of the TSA for different numbers of rebalancing trucks.

Number of Trucks	Number of Bike Stations	Median Fitness Score [Bikes/km]	Standard Deviation [Bikes/km]	10th Percentile Fitness Score [Bikes/km]	90th Percentile Fitness Score [Bikes/km]	Cumulative Distance of Rebalancing Routes [km]
(**a**)						
2	10	9.52	2.33	4.40	16.89	39.72
2	15	11.90	3.01	4.74	20.49	50.94
2	20	13.85	4.28	5.16	23.63	60.64
2	25	15.52	5.82	5.43	28.07	69.83
2	30	17.347	6.77	5.21	30.12	77.53
2	35	18.15	8.41	5.84	33.02	87.28
2	40	19.03	7.22	6.19	34.14	95.01
2	45	19.78	8.95	6.71	32.45	101.36
2	50	20.41	8.36	6.55	37.05	105.43
(**b**)						
2	10	6.05	2.07	2.88	10.32	50.14
2	15	7.90	5.12	3.57	13.42	74.41
2	20	10.22	6.11	3.83	18.99	76.06
2	25	10.08	6.35	2.57	16.38	108.44
2	30	11.43	5.93	4.20	18.30	112.47
2	35	14.14	6.91	5.66	20.41	130.81
2	40	15.19	7.88	4.56	23.62	140.64
2	45	17.21	8.77	6.03	27.54	149.51
2	50	18.94	9.85	7.00	29.16	156.28
(**c**)						
2	50	20.41	8.36	6.55	37.05	105.43
3	50	33.79	15.97	10.29	62.75	114.75
4	50	33.85	17.94	9.42	67.19	135.73
5	50	35.72	18.36	9.13	69.37	147.51
6	50	40.21	17.53	12.90	71.59	166.84
7	50	42.47	19.61	11.53	73.71	185.97
8	50	45.93	20.03	13.87	78.26	203.48
9	50	43.30	20.84	12.58	75.28	213.39
10	50	44.63	22.56	14.39	78.04	231.62
(**d**)						
2	50	18.94	9.85	7.00	29.16	156.28
3	50	27.31	13.22	8.42	38.25	142.51
4	50	29.46	16.13	9.46	54.60	160.56
5	50	32.81	16.45	10.86	64.46	175.78
6	50	36.52	17.04	11.43	68.33	208.22
7	50	40.40	19.11	10.44	72.28	243.59
8	50	42.82	19.41	12.83	74.47	262.45
9	50	43.25	21.31	14.41	73.56	253.42
10	50	45.89	24.18	16.08	72.32	290.88

5.7. Comparative Experiment and Analysis of Lost Customers and Unworking Time

For validation purposes, the system performances were compared considering three scenarios: a system without rebalancing, ACO-based scheduled rebalancing, and FLCGA-based scheduled rebalancing. As a performance indicator, the proportion of lost customers was considered. The performance criteria were computed, based on analyzed traffic data, for 25 stations in the system.

The proportion of lost customers was chosen based on a literature review for a standard metric for performance evaluation in the context of BSS rebalancing simulations with traffic prediction based on historical data [18,67–70]. Below we present the definition of the proportion of lost customers for a bike station:

$$\alpha_{lost} = \frac{n_{lost}}{n_{lost} + n_{util}}, \tag{40}$$

where α_{lost} is the proportion of lost customers, n_{lost} is the number of bikes that could not be rented because the station was empty or returned because the station was full, n_{util} is the number of bikes that were either rented or returned from the station.

In Figure 19, the system performances for the 3 scenarios are plotted.

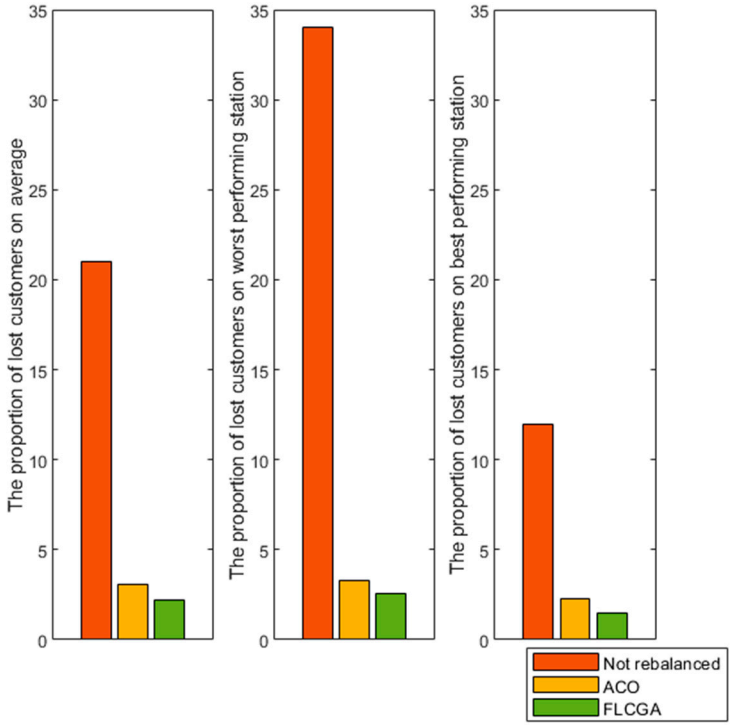

Figure 19. Performance comparison: the proportion of lost customers (%).

While it is visible that FLCGA performs consistently better than ACO, both performances are within acceptable limits. The proportion of lost customers is reduced tenfold for the worst-performing station.

The unworking time, which has a greater dependency on system configuration, is also helpful in evaluating the rebalancing performance [18]. The unworking time can be defined as the total number of time intervals in which a station is either full or empty during a rebalancing chain of orders. Figure 20 shows the unworking time for the same dataset under the considered scenarios.

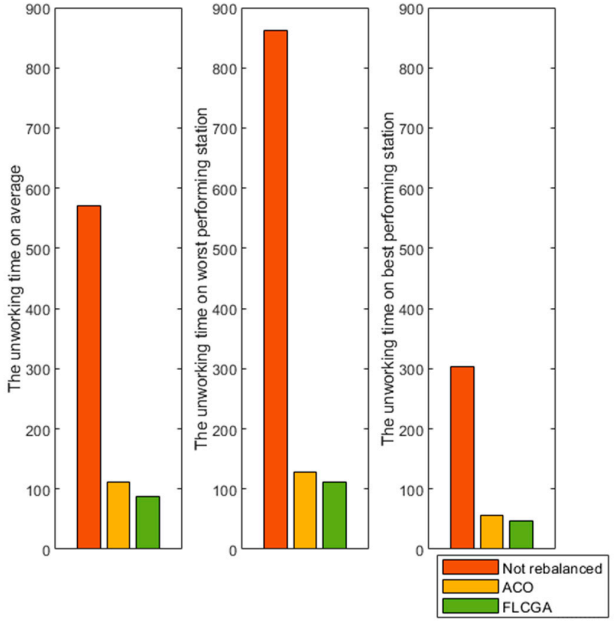

Figure 20. Performance comparison: unworking time (seconds).

According to customers' interpretation mentioned in published studies [18], the increase in performance is effective while the deviation is within acceptable terms.

5.8. Comparative Experiment and Analysis with a Standard Genetic Algorithm

To highlight the contribution of the inclusion of FLC in the presented method, SGA was also evaluated for the same dataset as presented. The values of SGA parameters were the same as for the FLCGA, except for the mutation and crossover probabilities and the termination condition. While FLCGA features adaptive mutation and crossover rates, SGA uses fixed rates. After several experiments, it was observed that the values of 90% for crossover rate and 3% for mutation rate yielded the best overall results for our application. This value pair is common in literature as a base for comparing modified GAs [71].

Due to the observed slower convergence, the maximum number of iterations was extended from 100 to 400 for the termination condition of SGA.

A side-by-side performance overview of FLCGA and SGA is presented in Table 9.

Table 9. Performance comparison of FLCGA and SGA.

Number of Trucks	Number of Bike Stations	Median Fitness Score [Bikes/km]		Standard Deviation [Bikes/km]		The Average Speed of the Genetic Algorithm [Number of Iterations]	
		FLCGA	SGA	FLCGA	SGA	FLCGA	SGA
2	10	9.52	3.28	2.33	1.36	70	192
2	15	11.90	4.10	3.01	1.54	72	210
2	20	13.85	3.14	4.28	1.32	77	232
2	25	15.52	6.83	5.82	2.78	78	259
2	30	17.347	9.27	6.77	3.93	82	288
2	35	18.15	8.52	8.41	2.59	81	316
2	40	19.03	9.04	7.22	4.14	83	305
2	45	19.78	8.30	8.95	3.95	85	360
2	50	20.41	9.63	8.36	3.66	86	347
3	50	33.79	9.41	15.97	3.44	86	331
4	50	33.85	11.37	17.94	3.70	86	376
5	50	35.72	16.42	18.36	5.11	86	365
6	50	40.21	17.51	17.53	5.54	86	381
7	50	42.47	16.02	19.61	4.86	86	351
8	50	45.93	18.88	20.03	6.30	86	357
9	50	43.30	19.59	20.84	6.17	86	382
10	50	44.63	22.83	22.56	6.94	86	349

FLCGA shows better scalability compared to SGA, considering the observed growth rate of average speed in given scenarios. A scenario refers to an individual pair of trucks and bike stations. FLCGA is converging significantly faster than SGA and consistently gives larger median fitness scores. The ratio of standard deviation over median score, which defines the data spread, is also lowered for SGA. Moreover, SGA shows an unstable growth in performance over the considered scenarios.

Overall, FLCGA provides superior results compared to SGA in any scenario.

6. Discussion

The strategy proposed was evaluated using real data collected from Citi Bike New York. The data corresponds to the following system properties about each bike station: location (latitude and longitude), capacity, and number of available bikes. The refinement of the algorithm parameters was performed via fine-tuning of the FLC.

Due to the large geographical area covered by the BSS and specific traffic limitations, it is impractical to employ a centralized strategy for rebalancing. Accordingly, a cluster-based approach was considered to evaluate the performances of the presented method, with a cluster being defined as a group of bike stations belonging to a delimited area. This also includes a depot, and it is served by a number of trucks. In order to have a robust evaluation, two aspects have been considered: cluster location and cluster dimension.

The cluster location is relevant because we have observed that the dynamic nature of the bike station load over time is heterogeneous, i.e., different neighborhoods have different degrees of traffic polarization. Moreover, the critical hotspots that need rebalancing do not have stability regarding their geographical distribution over time. Considering that the distribution is non-deterministic, in order to avoid bias in evaluation, a large number of experiments have been conducted by randomly selecting 1000 locations for evaluation purposes.

Regarding the cluster dimension, two characteristic values have been considered during the evaluation: the number of bike stations that need rebalancing and the number of trucks performing the rebalancing operation. Choosing a fixed dimension of the cluster is not a realistic solution because insight into the actual performance of the algorithm under different constraints could be lost. In order to prove the robustness of the proposed method, a diverse palette of resources was considered. Thus, the number of bike stations in a cluster ranged from 10 to 50, and the number of trucks ranged from 2 to 10.

Related to the above-discussed aspects, a large number of experiments were employed, and a histogram analysis of aggregated results was used for deriving performance indicators. Due to the already mentioned heterogeneity of rebalancing demands, median performance, i.e., the 50th percentile of the histogram of results, was used as a performance indicator. At the same time, a performance score as consistent as possible is desired; this is reflected in the low deviation values of the overall performance compared to the median. To capture performance consistency, different histogram properties were derived: standard deviation, 10th percentile score, and 90th percentile score.

Several comparative experiments and analyses were performed in order to prove that the results of the proposed method yield good results compared to already established algorithms for solving CVRP. During analysis, algorithm behavior was highlighted by discussing traits like convergence speed, data spread, score growth rate, and stable growth, showing that our proposed method is robust in this analysis framework.

The method's ability to maintain the expected performance regardless of how much the controlled BSS clusters increase in size, or in other words, its scalability in our case, is one of the most important performance indicators related to management and economic aspects. In this respect, FLCGA outperforms ACO, TSA, and SGA but underperforms HHO.

Our method shows high-performance results in terms of the total distance traveled by the trucks, given the median performance score and standard deviation observed during comparative experiments and analyses with other traditional methods. In most of the

scenarios, FLCGA was able to provide a shorter path of rebalancing than ACO, TSA, and SGA. When compared to HHO, FLCGA specializes in smaller clusters.

In terms of real-time constraints, the implemented application copes with the generation of solutions in a manner that does not introduce latency in the process of rebalancing, considering the trucks operating speed.

Following the conducted experiments, rebalancing with our method reduced the unworking time by more than 85%, and the average rate of lost customers was reduced from 21% to 2.5%, which translates to improved customer satisfaction.

The portability, i.e., that the method can be applied with success to any other BSS, is theoretically ensured because the algorithm refinement was based on tuning the FLC parameters instead of involving past real-world data collected for strategy refinement. The practical experimentation needed to verify the portability of our method is a topic that can be addressed by future research.

7. Conclusions

The work presents a method that has as its main purpose the rebalancing of the loads of the BSS stations under the conditions of an accentuated dynamic behavior. The imposed performances have tracked the essential aspects regarding the proper functioning of the system as follows.

Compliance with the imposed time constraints was ensured. The application of the method led to the minimization of the time of operation under the critical state of the bicycle stations and the minimization of the rate of lost clients. Another consequence was the achievement of a uniform allocation of tasks for each truck.

The effectiveness of the method has been tested in several case studies, using as a starting point real data extracted from the Citi Bike New York BSS database. The genetically modified algorithm presented in this method is one of the key elements that, together with the inference rules, determined the shortening of the travel routes for the rebalancing purpose.

The verification of the method involved the performance of some comparative studies. Several scenarios, which included a variable size of the clusters and their location in different areas, were considered for verification purposes and demonstrated the scalability of the method. In this regard, under similar conditions, the behavior of the system for the same case studies was evaluated, with the FLCGA-presented algorithm being replaced under the method with the algorithms SGA, ACO, TSA, and HHO. The study of comparative aspects was focused on two main directions: from the point of view of general algorithm performance indicators, FLCGA registers a higher convergence speed than SGA, and FCLGA provides the highest value of fitness function among the algorithms involved in the study for small clusters but is outperformed by HHO for large clusters. If the performances are evaluated from the point of view of an efficient logistic application, from the group of algorithms used for comparison, only HHO proved to have better performances for clusters with higher dimensions.

The overall results of comparative studies and the evaluation of performances recommend that the method be considered for other similar applications.

In future work, other logistic applications could be developed, integrating the FLCGA algorithm in their specific contexts.

Author Contributions: All authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available at https://ride.citibikenyc.com/system-data, accessed on 20 December 2022.

Acknowledgments: This paper was financially supported by the project "Entrepreneurial competencies and excellence research in doctoral and postdoctoral programs—ANTREDOC", a project co-funded by the European Social Fund financing agreement no. 56437/24.07.2019.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. John, P.; Ralph, B. Cycling towards a more sustainable transport future. *Transp. Rev.* **2017**, *37*, 689–694. [CrossRef]
2. Shaheen, S.A.; Nelson, C. *Mobility and the Sharing Economy: Potential to Overcome First- and Last-Mile Public Transit Connections*; Transportation Sustainability Research Center: Berkeley, CA, USA, 2016.
3. Kon, F.; Ferreira, C.; de Souza, H.A.; Duarte, F.; Santi, P.; Ratti, C. Abstracting mobility flows from bike-sharing systems. *Public Transp.* **2022**, *14*, 545–581. [CrossRef]
4. The Geography of Transport Systems. Available online: https://transportgeography.org/contents/chapter1/what-is-transport-geography/challenges-transport-systems/ (accessed on 6 November 2022).
5. Shui, C.S.; Szeto, W.Y. A review of bicycle-sharing service planning problems. *Transp. Res. Part C Emerg. Technol.* **2020**, *117*, 102648. [CrossRef]
6. Jan, B.; Marlin, U.; Dirk, M. Inventory Routing for Bike Sharing Systems. *Transp. Res. Procedia* **2016**, *19*, 316–327. [CrossRef]
7. Jie, W. Challenges and Opportunities in Algorithmic Solutions for Re-Balancing in Bike Sharing Systems. *Tsinghua Sci. Technol.* **2020**, *25*, 721–733. [CrossRef]
8. Schuijbroek, J.; Hampshire, R.C.; van Hoeve, W.-J. Inventory rebalancing and vehicle routing in bike sharing systems. *Eur. J. Oper. Res.* **2017**, *257*, 992–1004. [CrossRef]
9. Tsushima, H.; Matsuura, T.; Ikeguchi, T. Searching Strategies with Low Computational Costs for Multiple-Vehicle Bike Sharing System Routing Problem. *Appl. Sci.* **2022**, *12*, 2675. [CrossRef]
10. Oyola, J.; Arntzen, H.; Woodruff, D.L. The stochastic vehicle routing problem, a literature review, part I: Models. *EURO J Transp Logist* **2018**, *7*, 193–221. [CrossRef]
11. Khmeleva, E.; Hopgood, A.A.; Tipi, L.; Shahidan, M. Fuzzy-logic controlled genetic algorithm for the rail-freight crew-scheduling problem. *Künstliche Intell.* **2017**, *32*, 61–75. [CrossRef]
12. Dorigo, M.; Blum, C. Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **2005**, *344*, 243–278. [CrossRef]
13. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
14. Glover, F. Tabu search—Part I. *INFORMS J. Comput.* **1989**, *1*, 4–32. [CrossRef]
15. Guoxun, X.; Yanfeng, L.; Daxiang, J.; Jun, L. A user-based method for the static bike repositioning problem. *Syst. Eng. Theory Pract.* **2020**, *40*, 426–436. [CrossRef]
16. El Sibai, R.; Challita, K.; Bou Abdo, J.; Demerjian, J. A New User-Based Incentive Strategy for Improving Bike Sharing Systems' Performance. *Sustainability* **2021**, *13*, 2780. [CrossRef]
17. Benjamin, L. Dynamic repositioning strategy in a bike-sharing system; how to prioritise and how to rebalance a bike station. *Eur. J. Oper. Res.* **2019**, *272*, 740–753. [CrossRef]
18. Lin, Y.-C. A Demand-Centric Repositioning Strategy for Bike-Sharing Systems. *Sensors* **2022**, *22*, 5580. [CrossRef]
19. Svenja, R.; Klaus, B. A Relocation Strategy for Munich's Bike Sharing System: Combining an operator-based and a user-based Scheme. *Transp. Res. Procedia* **2017**, *22*, 105–114. [CrossRef]
20. Fan, Y.; Wang, G.; Lu, X.; Wang, G. Distributed forecasting and ant colony optimisation for the bike-sharing rebalancing problem with unserved demands. *PLoS ONE* **2019**, *14*, e0226204. [CrossRef] [PubMed]
21. Tang, Q.; Fu, Z.; Zhang, D.; Qiu, M.; Li, M. An Improved Iterated Local Search Algorithm for the Static Partial Repositioning Problem in Bike-Sharing System. *J. Adv. Transp.* **2020**, *2020*, 3040567. [CrossRef]
22. Bulhoes, T.; Subramanian, A.; Erdogan, G.; Laporte, G. The static bike relocation problem with multiple vehicles and visits. *Eur. J. Oper. Res.* **2018**, *264*, 508–523. [CrossRef]
23. Yuan, M.; Zhang, Q.; Wang, B.; Liang, Y.; Zhang, H. A mixed integer linear programming model for optimal planning of bicycle sharing systems: A case study in Beijing. *Sustain. Cities Soc.* **2019**, *47*, 101515. [CrossRef]
24. Possani, E.; Castillo, E. Optimizing the inventory and routing decisions in a bike-sharing system: A linear programming and stochastic approach. *Case Stud. Transp. Policy* **2021**, *9*, 1495–1502. [CrossRef]
25. Chemla, D.; Meunier, F.; Calvo, R.W. Bike sharing systems: Solving the static rebalancing problem. *Discreet. Optim.* **2013**, *10*, 120–146. [CrossRef]
26. Cipriano, M.; Colomba, L.; Garza, P. A Data-Driven Based Dynamic Rebalancing Methodology for Bike Sharing Systems. *Appl. Sci.* **2021**, *11*, 6967. [CrossRef]
27. Zheng, L.; Chen, L.; Shahabi, C. Centralized Routing for Bike-sharing Systems. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 154–166. [CrossRef]
28. Mahmoodian, V.; Zhang, Y.; Charkhgard, H. Hybrid rebalancing with dynamic hubbing for free-floating BSS. *Int. J. Transp. Sci. Technol.* **2021**, *11*, 636–652. [CrossRef]

29. Jin, Y.; Ruiz, C.; Liao, H. A simulation framework for optimising bike rebalancing and maintenance in large-scale bike-sharing systems. *Simul. Model. Pract. Theory* **2021**, *115*, 102422. [CrossRef]
30. Xue, B.; Ning, M.; Kwai, S.C. Hybrid Heuristic for the Multi-Depot Static Bike Rebalancing and Collection Problem. *Mathematics* **2022**, *10*, 4583. [CrossRef]
31. Du, M.; Cheng, L.; Li, X.; Tang, F. Static rebalancing optimisation with considering the collection of malfunctioning bikes in free-floating BSS. *Transp. Res. Part E Logist. Transp. Rev.* **2020**, *141*, 102012. [CrossRef]
32. Jorge, O.; Halvard, A.; David, L.W. The stochastic vehicle routing problem, a literature review, Part II: Solution methods. *EURO J. Transp. Logist.* **2017**, *6*, 349–388. [CrossRef]
33. Korayem, L.; Khorsid, M.; Kassem, S.S. Using Grey Wolf Algorithm to Solve the Capacitated Vehicle Routing Problem. *IOP Conf. Ser. Mater. Sci. Eng.* **2015**, *83*, 12014. [CrossRef]
34. Sajid, M.; Singh, J.; Haidri, R.A.; Prasad, M.; Varadarajan, V.; Kotecha, K.; Garg, D. A Novel Algorithm for Capacitated Vehicle Routing Problem for Smart Cities. *Symmetry* **2021**, *13*, 1923. [CrossRef]
35. Kris, B.; Katrien, R.; Inneke, V.N. The vehicle routing problem: State of the art classification and review. *Comput. Ind. Eng.* **2016**, *99*, 300–313. [CrossRef]
36. Raafat, E.; Hadeer, A. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Comput. Ind. Eng.* **2020**, *140*, 106242. [CrossRef]
37. Davoud, S.; Ellips, M.; Mostafa, S.; Hossein, A. GEPSO: A new generalized particle swarm optimization algorithm. *Math. Comput. Simul.* **2021**, *179*, 194–212. [CrossRef]
38. Kruszyna, M. NOAH as an Innovative Tool for Modeling the Use of Suburban Railways. *Sustainability* **2023**, *15*, 193. [CrossRef]
39. Lu, F.; Feng, W.; Gao, M.; Bi, H.; Wang, S. The fourth-party logistics routing problem using ant colony system-improved grey wolf optimization. *J. Adv. Transp.* **2020**, *2020*, 8831746. [CrossRef]
40. Lu, F.; Bi, H.; Huang, M.; Duan, S. Simulated annealing genetic algorithm based schedule risk management of IT outsourcing project. *Math. Probl. Eng.* **2017**, *2017*, 6916575. [CrossRef]
41. Vikram, K.K.; Ayani, N.; Ashutosh, B.; Shivani, S. An intensify Harris Hawks optimizer for numerical and engineering optimization problems. *Appl. Soft Comput.* **2020**, *89*, 106018. [CrossRef]
42. Ibrahim, O.O. Solving Capacitated Vehicle Routing Problem (CVRP) Using Tabu Search Algorithm (TSA). *Ibn AL-Haitham J. Pure Appl. Sci.* **2018**, *31*, 199–209. [CrossRef]
43. Vidal, T. Hybrid genetic search for the CVRP: Open-source implementation and swap neighbourhood. *Comput. Oper. Res.* **2020**, *140*, 105643. [CrossRef]
44. Zhang, X.; Chen, L.; Michel, G.; André, L. A branch-and-cut algorithm for the vehicle routing problem with two-dimensional loading constraints. *Eur. J. Oper. Res.* **2022**, *302*, 259–269. [CrossRef]
45. Moslem, S.; Amir, A.N.; Seyed, T.A.N. Statistical Design of Genetic Algorithms for Combinatorial Optimization Problems. *Math. Probl. Eng.* **2011**, *2011*, 872415. [CrossRef]
46. Ping, W. Application of Genetic Algorithm in Logistics Management and Distribution. In Proceedings of the Application of Intelligent Systems in Multi-Modal Information Analytics, Huhehaote, China, 23 April 2022. [CrossRef]
47. Xiao, J.; Lu, B. Application of Improved Genetic Algorithm in Logistics Transportation. In Proceedings of the Advances in Computer Science and Information Engineering, Berlin/Heidelberg, Germany, 13 July 2012. [CrossRef]
48. Xue, J.; Xu, Y. Application of Genetic Algorithm in Logistics Path Optimization. *Acad. J. Comput. Inf. Sci.* **2019**, *2*, 155–161. [CrossRef]
49. Ignaciuk, P.; Wieczorek, Ł. Continuous Genetic Algorithms in the Optimization of Logistic Networks: Applicability Assessment and Tuning. *Appl. Sci.* **2020**, *10*, 7851. [CrossRef]
50. Kroes, J.R.; Manikas, A.S.; Gattiker, T.F. Generating Efficient Rebalancing Routes for Bikeshare Programs Using a Genetic Algorithm. *J. Clean. Prod.* **2020**, *244*, 118880. [CrossRef]
51. Mohammed, G.A.; Sławomir, N.; Sepideh, P.; Peyman, S.M. Fast Genetic Algorithm for feature selection—A qualitative approximation approach. *Expert Syst. Appl.* **2023**, *211*, 118528. [CrossRef]
52. Mohammad, J.V.; Lai, S.L.; Mohd, R.A.B.; Wah, J.L. A Genetic Algorithm with Fuzzy Crossover Operator and Probability. *Hindawi Publ. Corp. Adv. Oper. Res.* **2012**, *2012*, 956498. [CrossRef]
53. Jalali, M.V. A genetic algorithm rooted in integer encoding and fuzzy controller. *Int. J. Robot. Autom.* **2019**, *8*, 113–124. [CrossRef]
54. Homayouni, S.; Tang, S. A fuzzy genetic algorithm for scheduling of handling/storage equipment in automated container terminals. *IJET* **2015**, *7*, 497–501. [CrossRef]
55. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
56. Marko, Č.; Marjan, L. Bus arrival time prediction based on a network model. *Procedia Comput. Sci.* **2017**, *113*, 138–145. [CrossRef]
57. Li, H.-C.; Lu, C.-C.; Eccarius, T.; Hsieh, M.-Y. Genetic algorithm with an event-based simulator for solving the fleet allocation problem in an electric vehicle sharing system. *Asian Transp. Stud.* **2022**, *8*, 100060. [CrossRef]
58. Wang, P.Y.; Wang, G.S.; Song, Y.H.; Johns, A.T. Fuzzy logic controlled genetic algorithms. In Proceedings of the IEEE International Conference FUZZ-IEEE, New Orleans, LA, USA., 11 September 1996. [CrossRef]
59. Pleurou, A.; Vlamou, E. Fuzzy Genetic Algorithms: Fuzzy Logic Controllers and Genetic Algorithms. *Glob. J. Res. Anal.* **2016**, *5*, 497–500. [CrossRef]

60. Salimi, N.; Rafe, V.; Tabrizchi, H.; Mosavi, A. Fuzzy Genetic Algorithm Approach for Verification of Reachability and Detection of Deadlock in Graph Transformation Systems. In Proceedings of the IEEE 3rd International Conference CANDO-EPE, Budapest, Hungary, 18–19 November 2020. [CrossRef]
61. Guo, N.; Qian, B.; Hu, R.; Jin, H.; Xiang, F. A Hybrid Ant Colony Optimization Algorithm for Multi-Compartment Vehicle Routing Problem. *Hindawi Complex.* **2020**, *2020*, 8839526. [CrossRef]
62. Selvi, V.; Umarani, R. Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *Int. J. Comput. Appl.* **2010**, *5*, 1–6. [CrossRef]
63. Sim, K.; Sun, W. Ant colony optimisation for routing and load-balancing: Survey and new directions. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2003**, *33*, 560–572. [CrossRef]
64. Venkatesh, S.; Jeyakarthic, M. Metaheuristic based Optimal Feature Subset Selection with Gradient Boosting Tree Model for IoT Assisted Customer Churn Prediction. *Seybold Rep.* **2020**, *15*, 334–351.
65. Yi, P.; Huang, F.; Peng, J. A Rebalancing Strategy for the Imbalance Problem in Bike-Sharing Systems. *Energies* **2019**, *12*, 2578. [CrossRef]
66. Alabool, H.; Al-Arabiat, D.; Abualigah, L.; Heidari, A.A. Harris hawks optimization: A comprehensive review of recent variants and applications. *Neural Comput. Appl.* **2021**, *33*, 8939–8980. [CrossRef]
67. Xia, Y.; Fu, Z.; Pan, L.; Duan, F. Tabu search algorithm for the distance-constrained vehicle routing problem with split deliveries by order. *PLoS ONE* **2018**, *13*, e0195457. [CrossRef] [PubMed]
68. Guanhua, M.; Bowen, Z.; Changjing, S.; Qiang, S. Rebalancing stochastic demands for bike-sharing networks with multi-scenario characteristics. *Inf. Sci.* **2021**, *554*, 177–197. [CrossRef]
69. Ban, S.; Hyun, K.H. Designing a User Participation-Based Bike Rebalancing Service. *Sustainability* **2019**, *11*, 2396. [CrossRef]
70. Angelelli, E.; Chiari, M.; Mor, A.; Speranza, M.G. A simulation framework for a station-based bike-sharing system. *Comput. Ind. Eng.* **2022**, *171*, 108489. [CrossRef]
71. Ahmad, H.; Khalid, A.; Esra'a, A.; Eman, A.; Awni, H.; Surya, V.B.P. Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach. *Information* **2019**, *10*, 390. [CrossRef]