

Stochastic agent-based models optimization applied to the problem of rebalancing bike-share systems

Daniel Soto Forero¹, Yony Ceballos²

¹FEMTO-ST DISC Institute, University of Franche-Comté, Besançon, France

²Faculty of Engineering, University of Antioquia, Medellín, Colombia

Article Info

Article history:

Received Nov 18, 2022

Revised Jun 16, 2024

Accepted Jul 7, 2024

Keywords:

Agent based models

Bike-share systems

Complex systems

Stochastic optimization

Uncertainty optimization

ABSTRACT

This paper presents an agent-based model employing a stochastic optimization search that attempts to find an optimal solution to the online bicycle rebalancing problem for general bike-sharing systems. The algorithm receives the initial and final global state configuration of the system. The main objective of the study is to find the minimum cost path from the initial to the final state. Each agent of the model has four behavioral options that search the optimal configuration; at each iteration, it selects one of these options based on random thresholds and shares the temporary solution found with neighboring agents to improve their search process. The algorithm presents a high exploratory behavior of the search space, which helps to find an approximation away from the local optimal configuration. Additionally, the exchanges between agents allow a consensus on the solutions found. The algorithm has been tested with two different generated configurations using as a basis a real dataset extracted from a functional bike-sharing system collected in 2019. The results show a positive evolution originating from the emerging effect of stochastic selection and interaction between agents.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Yony Ceballos

Faculty of Engineering, University of Antioquia

Calle 67 No. 53-108, Medellín, Colombia

Email: yony.cebillos@udea.edu.co

1. INTRODUCTION

The bike-share systems (BSS) are an individual service that offer the use of bikes in urban areas [1]. BSS is composed of stations that are distributed throughout the city [2]. Generally, the system is integrated with others public transport systems [1]. Typically, the BSS are divided in two categories: BSS with docking and BSS without docking [3]. The BSS helps to get more durable mobility, the automobile traffic decrease and to reduce the pollution generated by the motorized means of transportation [4]–[6]. Research has tried to identify the challenges, environmental, economic impacts, users motivations and security, as well as the creation, extension and balance [7]–[10]. The first system has been installed in Amsterdam in 1965; throughout the years, it has become quite popular and increased its quantity in several cities. Nowadays, there are near of 1,000 of them in use and 300 more in development [11].

Many problems can be studied to improve the system such as inventory level, fleet size design or station design [8]. Other difficulties are the socio-demographic characteristics, perceptions and motivations for use the BSS [6], or the prediction of demand [12]. One of the issues that is very associated with the system is it repositioning [13]–[15]. There are two types of repositioning: static (offline) and dynamic (online). The former is made when the system is not used, and the initial and final states are fixed and known. Its objective is to find the optimal state transitions by going from initial to final states. On the other hand, the dynamic takes place when the system is used, and the initial and final states are variable [16].

This paper is organized as follows: section 2 presents the most recent related works found in literature. Section 3 explains the proposed algorithm for optimizing the rebalancing of BSS for the specific problem previously discussed. Section 4 describes and analyzes the experimental results obtained with the tested scenarios and the predetermined parameters. Finally, section 5 presents the conclusions and future work of the study.

2. LITERATURE REVIEW

The bike-share systems with docks provide the users with a certain number of bikes that can be used in a specific slot of time; some docks must be empty in order to receive bikes of users who arrive at the station. Given that the system is constantly used throughout the day, it must ensure that, at a specific time, there are x number of bikes and y number of free docks to be able to operate correctly and avoid congestion to provide the service to the users. The problem of estimating and guaranteeing a certain number of bikes that should be available per day and at each slit of time, and docks that should be free in each station is called the rebalancing problem for the bike-share systems [11].

In order to propose an algorithm for the rebalancing problem, it is necessary to identify the type of stations and try to predict their individual dynamic behavior. There are several alternatives to do so, as explained in [5], where is proposed the detection of station having a significantly different behavior from their neighboring stations using the spatial correlation with the Moran scatterplot method. Consequently, the proposed method enhances significantly the resources. Another method for estimate the demand is proposed in [15] where a game theory algorithm is used and in which each player has an objective function. Later for the rebalancing, a complete tour is constructed by using the 2-opt algorithm. The focuses change in [3], where the center of research for the estimation is the user's behavior, the authors present a simulation for evaluating the loss demand and use it as a measurement of user dissatisfaction. With statistical parametrical models [12], try to identify patterns of demand and get a 5% as margin of error.

Most of the studies found in the literature consider static rebalancing as show in study [17], whose main objective is to find a minimum cost route. There, a vehicle performs a series of pickup and delivery operations while satisfying the demand and capacity constraints. Furthermore, a vehicle can visit stations multiple times but cannot use them to store it temporally or to provide bikes. Therefore, the focus of the research is to compare the computational complexity and worst-case analysis of three exact algorithms.

The contributions from Raviv *et al.* [18] are the formulation of static rebalancing problem as a mixed-integer linear problem (MILP) and the proposal of two other mixed integer linear programs where the computational performances are compared. Such a process results in obtaining very effective high-quality solutions for real life instances. Another MILP formulation appears in study [11], which presents a comparison between four models of mixed linear programming in integers. The works cited above show that the formulation of the problem as MILP is effective, but it also has limitations when more complex constraints are considered.

Similarly, the heuristics algorithms has been taken into consideration as shown in [4], where the authors divide the problem by clustering all the stations geographically; after that, they calculate the optimal local path with a mixed integer-linear model; finally, all the local paths are integrated to get the global response. Another formulation is found in study [19], where they suggest a hybrid algorithm between a large neighborhood search and the tabu search with good results. In study [3], an approximate solution is obtained with a version of the random search metaheuristic. The work of [2] develops two constraint programming approach incorporating the large neighborhood search. The results are positive even within a short timeout.

Inspired in the artificial intelligent (AI), deep sequential learning, Chen *et al.* [14] uses the knowledge from the past problem instances for predicting and calculating a solution, the limitation is that the authors consider only one truck. First, they try to identify the best time slot for executing the rebalance, and then, they use the real data for training the proposed algorithm. Finally, they apply the trained model to predict and present a particular solution. This type of approximation works very well with problems that are quite similar to each other because if a problem with additional constraints or new parameters appears, the trained model does not work with the same accuracy.

Several variants of the static rebalancing problem have been studied as well: particularly, Li *et al.* [13] study the case in which many types of bicycles are considered. The work of [20] considers only a single available vehicle to make the rebalancing. Repositioning bikes involving multiple vehicles is analyzed in [21]. In study [22], the main focus is to investigate the case with only full loads for the vehicles allowed between car rental stations. The paper written by Szeto and Shui [23] investigates a bike repositioning problem that determines repositioning vehicle routes and loading/unloading quantities at each bike station. Its main objective is to minimize the average gap between total demand dissatisfaction tolerance and service time. Meanwhile, Wang and Szeto [16] formulate a mixed linear programming model to calculate the

repositioning of good or broken bikes by minimizing total CO₂. Most of the literature published on bike sharing systems does not take into consideration the random phenomena related to this type of problem such as weather-related hazards, social movements, inventory management, bike and station maintenance. To consider these random phenomena, stochastic estimation and stochastic optimization is one of the most appropriate approaches.

3. PROPOSED ALGORITHM

The proposed algorithm seeks to find an approximate optimal solution S to the on-line rebalancing problem. The algorithm is planned as follows: $S = \{(id_{st1}, n_{bike}); (id_{st2}, n_{bike}); \dots; (id_{stn}, n_{bike})\}$; where S is a structured tuple vector. id_{st} is the number identifier of a specific station, and n_{bike} is the number of bikes to take (positive number) from id_{st} or leave (negative number) in id_{st} station. The potential of solution is calculated as formally shown in (1) by using the potential of the agent-based models to explore and exploit the research space as the method proposed in [24]. In such an approach, every agent minimizes its own objective function while locally exchanging information with other agents in the network over a time-varying topology. The strengths of the algorithm are nonparametric statistical inference, the retroaction effects, the individual stochastic behavior, diversity of actions, the auto-organization, and emergent effects during execution. Table 1 describes the parameters of the algorithm. The structure of agents and stations are specified in Tables 2 and 3 respectively.

$$\text{Minimize } \sum_{i=0}^{|S|-2} d(S_i, S_{i+1}) \quad (1)$$

where $|S|$ is the length of S and $d(a, b)$ is the distance between a and b

Table 1. Table of algorithm parameters

Id	Parameter	Type
C	Truck loading capacity (homogeneous)	\mathbb{N}
L	Historic data	List of Table 3
N	Initial state of the stations	List of Table 4
F	Final state of the stations	List of Table 4
V	Distances between stations	$V_{ L \times L }(\mathbb{R})$
I	Number of agents in the population	\mathbb{N}
T	Number of iterations	\mathbb{N}

Table 2. Table of agent properties

Id	Property	Type
x_a	Discrete probability for action	$x_a(\mathbb{R})$
S_a	The solution of the problem	List of Table 4
f_a	The fitness for the proposed solution	\mathbb{R}
p_r	Threshold of group exchange	\mathbb{R}
p_b	Threshold of acceptance of a poorer fitness	\mathbb{R}
p_g	Threshold of neighbor exchange	\mathbb{R}

Table 3. Table of state properties

Id	Property	Type
id	Identifier of station	\mathbb{N}
O	Name of station	String
$N_{b,t}$	Number of bikes in the time t	\mathbb{N}
$N_{s,t}$	Number of disponible spaces in the time t	\mathbb{N}

To initiate the stochastic optimization algorithm, it is necessary to know the initial and final states of the general system whose structure is shown in Table 4. Since in this case neither of the two states is known with certainty, the first phase of the algorithm is to divide the day in time slots (in this case are 48 slots where each one of them corresponds to 30 minutes). Moreover, for each slot, a probability distribution is generated from the historical data obtained. With this probability distribution, it is possible to generate the most probably states: a random initial state and a random final state for two consecutive given time slots, with which the stochastic optimization algorithm can be executed. Once the two necessary states have been generated, the second phase of the algorithm is executed. The flowchart of second phase is shown in Figure 1. Each agent generates its own response by considering one of four possible behaviors for each

iteration. The action is selected randomly based on the agent probability discrete distribution for the four defined actions like formally shown in the (2) where 0-does not change the solution, 1-generates a new solution, 2-exchanges between groups and 3-modifies the solution depending on the information obtained from the neighborhood.

$$D \sim x_a(0, 3) \quad (2)$$

Table 4. Table of station properties

Id	Property	Type
id	Identifier of station	\mathbb{N}
c	Number of bikes. Positive (surplus) and Negative (missing)	\mathbb{Z}

From two generated states, two sets are created, the set A contains the stations with an excess of bikes and whose structure is the id of stations and c is the number of excess of bikes and the set B contains the stations with lack of bikes, whose structure is the id of stations and c the number of lack of bikes. For the first iteration all the agents set $D = 1$ and $S = \emptyset$. The construction of solution S is explained in the (3), where the solution S is transformed by the corresponding function associated to the action.

$$S = \begin{cases} S & \text{if } D = 0 \\ G(S) & \text{if } D = 1 \\ C(S) & \text{if } D = 2 \\ R(S) & \text{if } D = 3 \end{cases} \quad (3)$$

If the selected action is 1, then the generation of the solution is calculated as shown in the recursive (4), (5), (6) and (7). The equation (4) verifies that all the elements of sets A and B are inside the vector solution S .

$$G(S) = \begin{cases} S & \text{if } (\sum_{S, S_c < 0} |S_c| \leq \sum_A A_c \wedge \sum_{S, S_c > 0} S_c = \sum_B |B_c|) \vee \\ & (\sum_{S, S_c < 0} |S_c| = \sum_A A_c \wedge \sum_{S, S_c > 0} S_c \leq \sum_B |B_c|) \\ G(G_a(S)) & \text{otherwise} \end{cases} \quad (4)$$

The solution S must be completed by adding stations and bikes to carry in the truck as shown in the (5). If the truck has zero or a positive number of bikes lower than the capacity C , then, the threshold p chooses a station from set A considering a random value by using (6). Nevertheless, if the truck has full capacity C , it is necessary to select a station from set B by using (7).

$$G_{a(S)} = \begin{cases} G_b(S) & \text{if } ((\sum_S S_c) = 0) \vee (|S| = 0) \vee (rand() > p \wedge (\sum_S S_c) < C) \\ G_c(S) & \text{if } ((\sum_S S_c) = C) \vee (|S| > 0 \vee (\sum_S S_c) > 0) \end{cases} \quad (5)$$

If the chosen option is “select a station from set A ”, then a random station is selected, however, it is necessary to check the truck capacity C . If the truck is fully loaded and there are surplus bikes, the number on the station is updated as shown in the (6).

$$\begin{aligned} G_b(S) &= \{S^\cap \langle X, i(X, S) \rangle : X \sim U(A)\} \\ \text{where} \\ i(X, S) &= \begin{cases} X_c, A = A \setminus \langle X \rangle & \text{if } (\sum_S S_c) + X_c \leq C \\ C - (\sum_S S_c), A(X)_c = A(X)_c + (\sum_S S_c) - C & \text{otherwise} \end{cases} \end{aligned} \quad (6)$$

If the chosen option is “select a station from set B ”, a random station is selected, but the number of bikes that can be left at the station must be less or equal to the amount carried in C by the truck, as shown in the (7).

$$\begin{aligned} G_c(S) &= \{S^\cap \langle Y, j(Y, S) \rangle : Y \sim U(B)\} \\ \text{where} \\ j(Y, S) &= \begin{cases} Y_c, B = B \setminus \langle Y \rangle & \text{if } |Y_c| \leq (\sum_S S_c) \\ (\sum_S S_c), B(Y)_c = B(Y)_c + (\sum_S S_c) & \text{otherwise} \end{cases} \end{aligned} \quad (7)$$

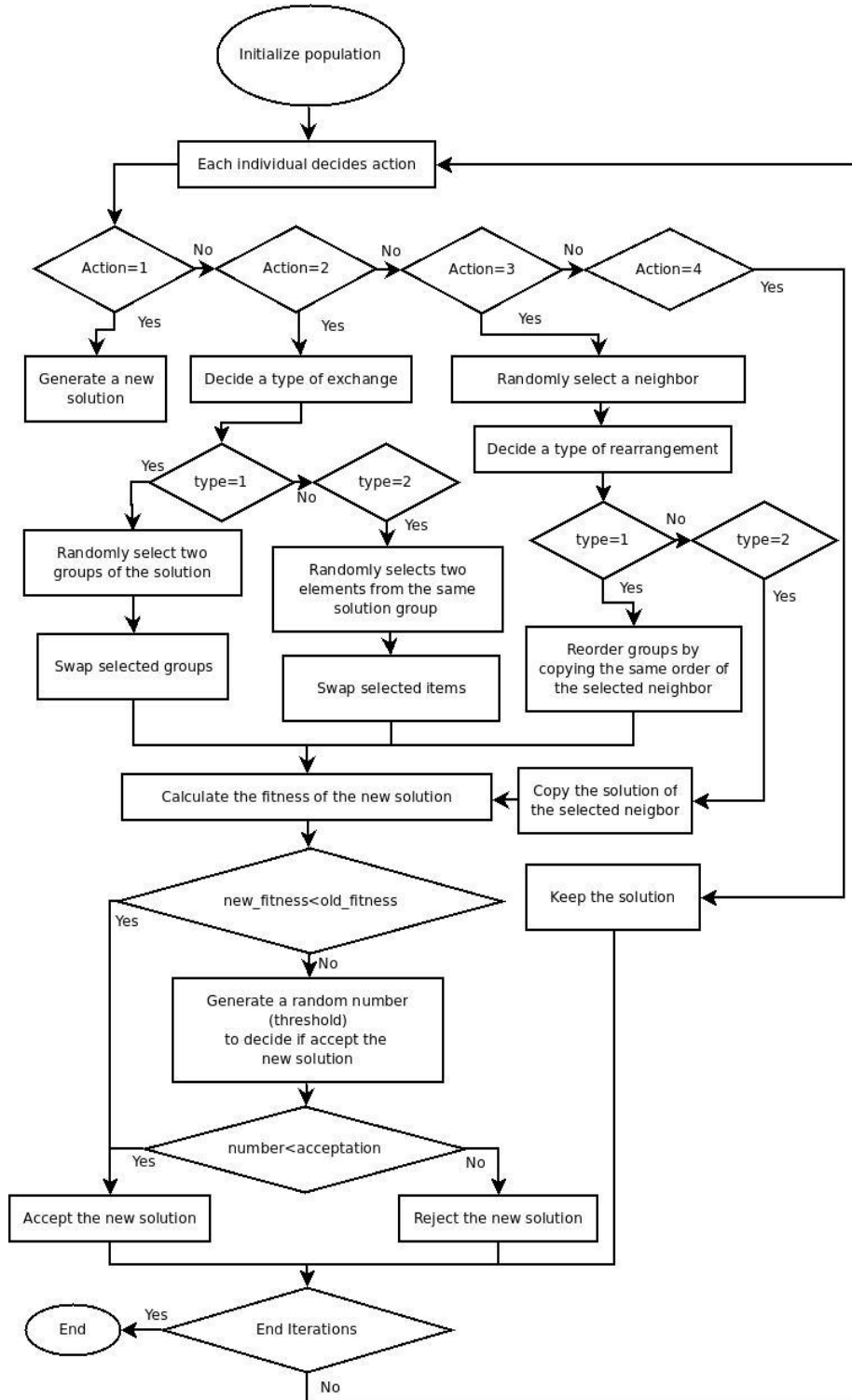


Figure 1. Flowchart representing the second phase algorithm

When the action is 2, the exchange between groups is calculated by using the (8), (9), (10) and (11). The function G_r , generates multiple vectors from the vector S by using the function M , which splits a vector with the constraint received as a parameter. In this case, each generated sub-vector corresponds to a group of the solution.

$$Gr(S) = M((\sum_{i=0}^{|S|-1} S_c) = 0 \tag{8}$$

The threshold p_r allows to decide the exchange between two groups with the function τ_g or the elements of same group with the function τ_e as shown in (9).

$$C(S) = \begin{cases} C_a(S) & \text{if } \text{rand}() < p_r \\ C_b(S) & \text{Otherwise} \end{cases} \quad (9)$$

The groups g_1 and g_2 are generated with the (8). The “permutation” function τ_g exchanges the vectors of indexed groups as g_1 and g_2 , then concatenates them in the solution vector S , as shown in (10).

$$C_a(S) = \begin{cases} \tau_g(S, g_1, g_2) & \text{if } \text{rand}() < p_b \vee \text{fit}(\tau_g(S, g_1, g_2)) \leq \text{fit}(S) \\ S & \text{otherwise} \end{cases} \quad (10)$$

The “permutation” function τ_e exchanges the elements e_{g1} and e_{g2} that belong to the same group vector g . Later, it concatenates all group vectors in the solution vector S as described in the (11).

$$C_b(S) = \begin{cases} \tau_e(S, e_{g1}, e_{g2}) & \text{if } \text{rand}() < p_b \vee \text{fit}(\tau_e(S, e_{g1}, e_{g2})) \leq \text{fit}(S) \\ S & \text{otherwise} \end{cases} \quad (11)$$

The performance of the solution S as shown in the (12) is calculated as a sum of the distances between all the stations inside vector S .

$$\text{fit}(S) = \sum_{i=0}^{|S|-2} d(S_{id,i}, S_{id,i+1}) \quad (12)$$

where $d(a, b)$ is the distance between a and b

If the action is 3, then the solution rearrangement is produced with the neighborhood information. In this case, two options are possible: the first is to copy the randomly selected neighbor’s solution K function; second one is to reorder the groups of the existing solution based on the selected neighbor’s order E function, as shown in the (13), where S_v is the solution of the randomly selected V neighbor. The E function searches groups that are like S and S_v based only on the initial Station ID. Subsequently, it reorders S by placing the similar groups in the same done position as S_v .

$$R(S) = \begin{cases} K(S, S_v) & \text{if } \text{rand}() < p_g \\ E(S, S_v) & \text{otherwise} \end{cases} \quad (12)$$

The generated solution is a list that has a structure, the identifier of the station id and the number of bikes to take c (positive number) or leave (negative number).

4. EXPERIMENTAL RESULTS

The results were obtained within 6 weeks of data collected in 2019 from the Dk-Velo system. The system entered service on August 31, 2013. Nowadays, it is composed of 46 automatic stations and 350 bicycles as shown in Figure 2. It is automatic and available 24 hours a day, 7 days a week. The data has been collected automatically, because the web page allows knowing in the state of global system in real-time, the state of each of the stations and the number of bikes in service [25].

For phase 1, the probability distribution has been constructed as a histogram with data obtained for each station in the specific time slot, as shown in Figure 3, for the station “01 Gare SNCF” in the time slot “00:00-00:30”. By applying the constructed histograms, a complete test scenario is generated to demonstrate the algorithm evolution. Such scenario is shown in Table 5 where the identification of the station, the number of bikes in the initial state and the number of bikes in the final state appear. The missing stations in the table have the same number of bikes in the initial and final states.

Tables 6 and 7 present the numerical results obtained for the generated scenario with a change of the number of agents I , iterations T and truck capacity C . The results have been generated with 100 runs. The data in Tables 6, 7 and Figures 4, 5 show that the algorithm works better with more iterations than with more agents. Furthermore, there is no direct relationship between variance, number of agents and number of iterations. Boxplot representing the results comparison between the six algorithm configurations can be seen in Figure 4(a) and Figure 5(a). Boxplot representing the numerical results for the I50C5T100 configuration can be seen in Figure 4(b) and Figure 5(b).



Figure 2. Geographical distribution of DK-velo system stations (adapted from [25])

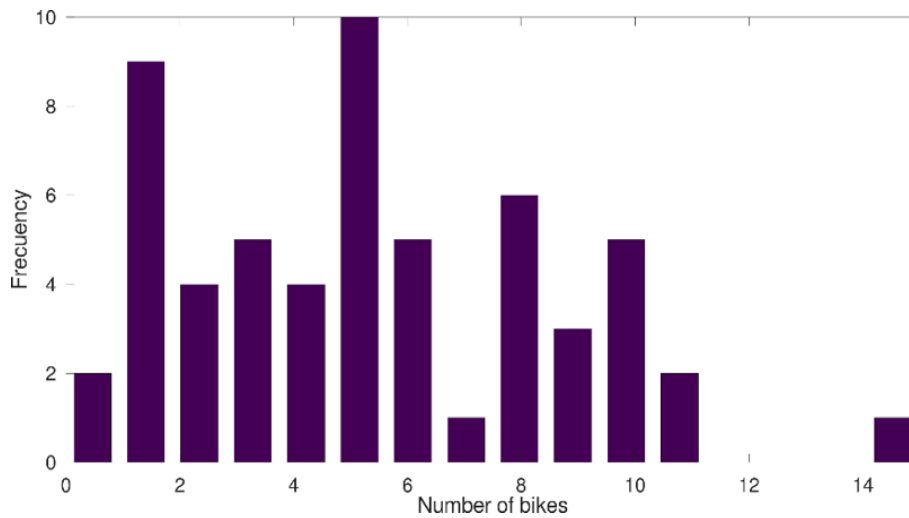


Figure 3. Histogram generated for the station “Gare SNCF” in the time slot 00:00-00:30

Table 5. Data generated for the test scenario with change in 40 stations

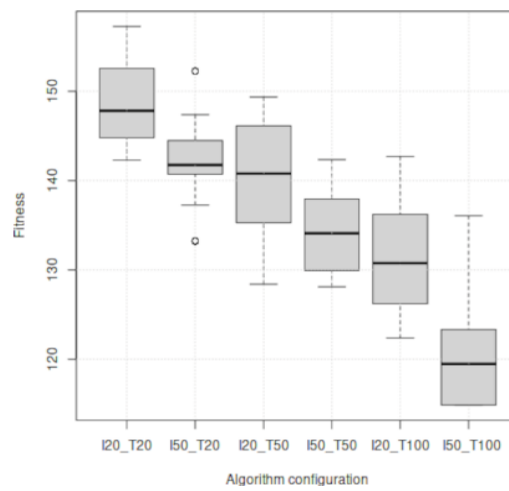
Id Station	1	3	4	5	6	7	8	9	10	11	12	13	14	15
Initial State	7	0	6	6	2	4	4	10	6	4	8	0	5	9
Final State	8	4	2	8	4	0	3	9	10	9	10	5	3	2
Id Station	16	17	18	19	20	21	23	24	25	26	29	31	32	33
Initial State	9	6	11	4	8	9	10	7	4	7	9	7	11	11
Final State	4	7	8	1	9	10	12	10	2	4	6	12	1	14
Id Station	34	35	36	37	38	40	41	42	43	44	45	46		
Initial State	6	4	9	9	7	7	7	5	1	2	6	5		
Final State	7	5	5	10	11	10	1	9	2	3	2	10		

Table 6. Numerical results for configurations of algorithm with C=5

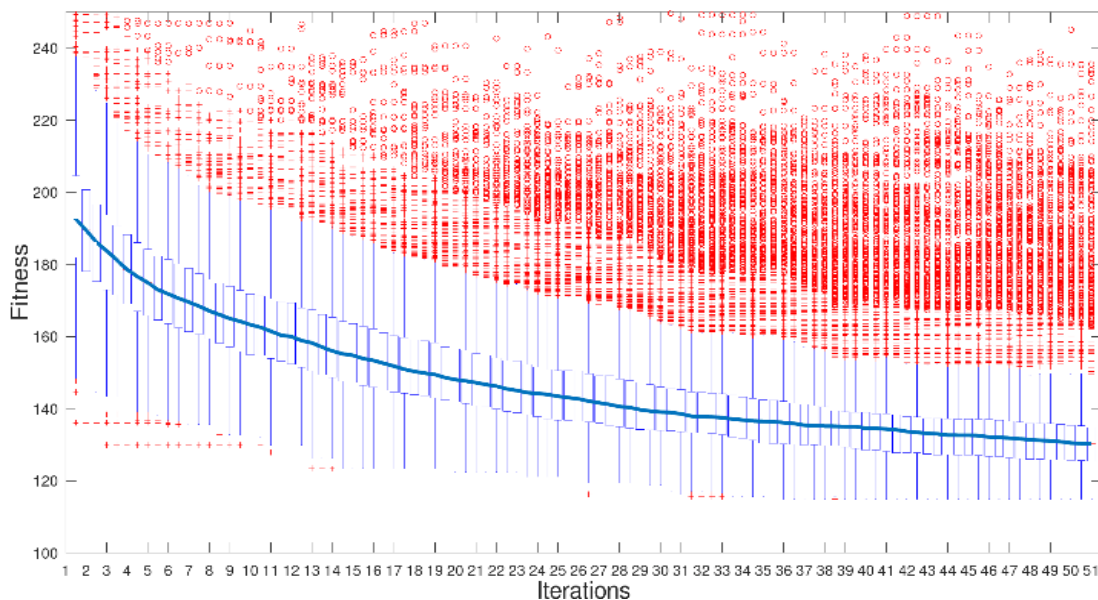
Parameters	Best	Worst	Mean	Variance
I20C5T20	142.28	157.25	148.7	20.748
I50C5T20	133.24	152.25	142.14	16.371
I20C5T50	128.41	149.35	139.88	41.5
I50C5T50	128.12	142.35	134.46	20.53
I20C5T100	122.37	142.71	131.16	30.496
I50C5T100	114.92	136.52	126.43	29.991

Table 7. Numerical results for configurations of algorithm with C=10

Parameters	Best	Worst	Mean	Variance
I20C10T20	105.22	137.06	129.32	81.161
I50C10T20	111.30	134.27	128	36.228
I20C10T50	113.21	130.74	122.7	27.552
I50C10T50	109.13	126.69	119	24.641
I20C10T100	111.08	128.16	119.24	26.962
I50C10T100	102.36	122.07	114.25	27.96



(a)



(b)

Figure 4. Boxplots obtained in the simulation with $C = 5$; (a) Boxplot representing the results comparison between the six algorithm configurations; and (b) Boxplot representing the numerical results for the I50C5T100 configuration

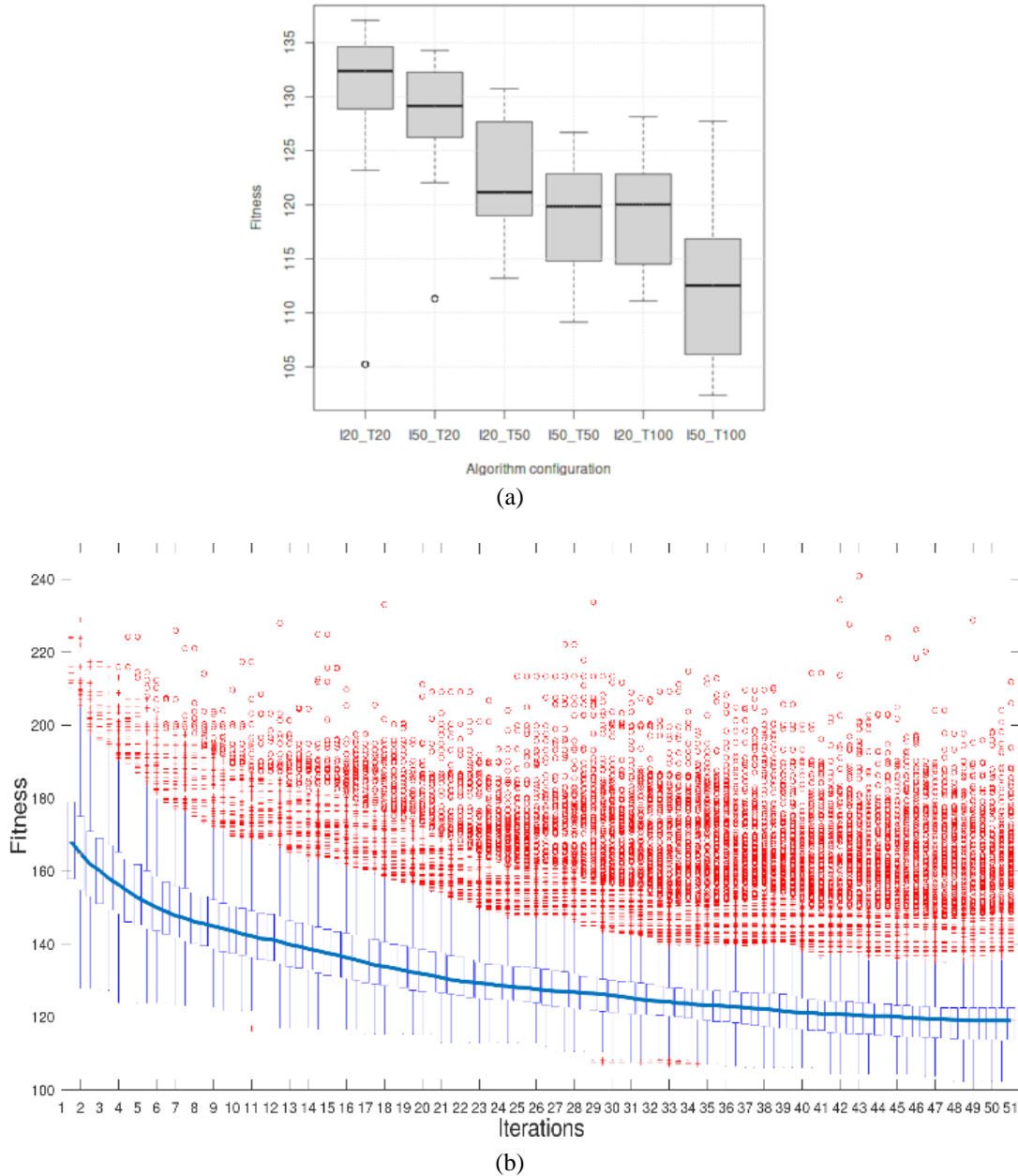


Figure 5. Boxplots obtained in the simulation with $C = 10$; (a) Boxplot representing the results comparison between the six algorithm configurations, and (b) Boxplot representing the numerical results for the I50C10T100 configuration

The boxplot charts of the Figures 4(b) and 5(b) show that in the tested cases the algorithm is highly exploratory in the search space and generates many outliers' solutions, which allows the algorithm not to stay in local solutions. The best solution obtained with generated test scenario and parameters $I = 50, C = 5, T = 100$ is $S = (36\ 4)\ (20\ -1)\ (5\ -2)\ (4\ 4)\ (17\ -1)\ (19\ 1)\ (42\ -4)\ (33\ -1)\ (32\ 5)\ (37\ -1)\ (19\ 1)\ (38\ -4)\ (32\ 4)\ (13\ -5)\ (19\ 1)\ (8\ 1)\ (15\ 2)\ (9\ 1)\ (46\ -5)\ (45\ 4)\ (3\ -4)\ (7\ 4)\ (10\ -4)\ (15\ 5)\ (11\ -5)\ (32\ 1)\ (1\ -1)\ (41\ 5)\ (31\ -5)\ (29\ 3)\ (24\ -3)\ (26\ 3)\ (23\ -2)\ (16\ 4)\ (21\ -1)\ (16\ 1)\ (12\ -2)\ (35\ -1)\ (33\ -2)\ (14\ 2)\ (40\ -2)\ (41\ 1)\ (25\ 2)\ (43\ -1)\ (44\ -1)\ (40\ -1)\ (18\ 3)\ (34\ -1)\ (6\ -2)$ with fitness equal to 114.920286.

5. CONCLUSION




This paper proposes a stochastic algorithm using an agent-based model as a possible alternative to find a solution for the problem of rebalancing bike-share systems; their behavior is stochastic as well as the space exploration. The algorithm finds an approximated minimum solution for the suggested scenario,

resulting in exploring several possibilities. With the tested configurations, it can be concluded that the algorithm behavior works better with more iterations and fewer agents to get a good approximation. The results obtained confirm the power of collective intelligence, the strength of stochastic search, and the importance of interactions and information exchanges. The algorithm can be useful to solve one of the main problems of bike-sharing systems and to increase their use as they have proven to be beneficial for cities. As future research can be considered the BSS without docking and geo-referential positions, the city traffic, and the state of streets for the same time slots of the algorithm and calculate the fitness with that penalization, convert the algorithm for multi-objective optimization. Additionally, it is possible to add constraints like truck damage, the itinerary change, or CO₂ emissions.




REFERENCES

- [1] P. Leclaire and F. Couffin, "Method for static rebalancing of a bike sharing system," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1561–1566, 2018, doi: 10.1016/j.ifacol.2018.08.274.
- [2] L. Di Gaspero, A. Rendl, and T. Uri, "Balancing bike sharing systems with constraint programming," *Constraints*, vol. 21, no. 2, pp. 318–348, Feb. 2015, doi: 10.1007/s10601-015-9182-1.
- [3] R. Costa Affonso, F. Couffin, and P. Leclaire, "Modelling of user behaviour for static rebalancing of bike sharing system: Transfer of demand from bike-shortage stations to neighbouring stations," *Journal of Advanced Transportation*, vol. 2021, pp. 1–15, Jan. 2021, doi: 10.1155/2021/8825521.
- [4] I. A. Forma, T. Raviv, and M. Tzur, "A 3-step math heuristic for the static repositioning problem in bike-sharing systems," *Transportation Research Part B: Methodological*, vol. 71, pp. 230–247, Jan. 2015, doi: 10.1016/j.trb.2014.10.003.
- [5] C. Fricker, R. El Sibai, and Y. Chabchoub, "Bike sharing systems: a new incentive rebalancing method based on spatial outliers detection," *International Journal of Space-Based and Situated Computing*, vol. 9, no. 2, p. 99, 2019, doi: 10.1504/ijssc.2019.10025978.
- [6] X. Ma, Y. Yuan, N. Van Oort, and S. Hoogendoorn, "Bike-sharing systems' impact on modal shift: A case study in Delft, the Netherlands," *Journal of Cleaner Production*, vol. 259, p. 120846, Jun. 2020, doi: 10.1016/j.jclepro.2020.120846.
- [7] E. Fishman, S. Washington, N. Haworth, and A. Watson, "Factors influencing bike share membership: An analysis of Melbourne and Brisbane," *Transportation Research Part A: Policy and Practice*, vol. 71, pp. 17–30, Jan. 2015, doi: 10.1016/j.tra.2014.10.021.
- [8] C. S. Shui and W. Y. Szeto, "A review of bicycle-sharing service planning problems," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102648, Aug. 2020, doi: 10.1016/j.trc.2020.102648.
- [9] E. Macioszek, P. Świerk, and A. Kurek, "The bike-sharing system as an element of enhancing sustainable mobility—a case study based on a City in Poland," *Sustainability*, vol. 12, no. 8, p. 3285, Apr. 2020, doi: 10.3390/su12083285.
- [10] M. Straub *et al.*, "Semi-automated location planning for urban bike-sharing systems," *Proceedings of 7th Transport Research Arena TRA 2018*, Vienna, Austria 2018, 2019, doi: 10.5281/zenodo.1483822.
- [11] M. Dell'Amico, E. Hadjiconstantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," *Omega*, vol. 45, pp. 7–19, Jun. 2014, doi: 10.1016/j.omega.2013.12.001.
- [12] S. Sohrobi, R. Paleti, L. Balan, and M. Cetin, "Real-time prediction of public bike sharing system demand using generalized extreme value count model," *Transportation Research Part A: Policy and Practice*, vol. 133, pp. 325–336, Mar. 2020, doi: 10.1016/j.tra.2020.02.001.
- [13] Y. Li, W. Y. Szeto, J. Long, and C. S. Shui, "A multiple type bike repositioning problem," *Transportation Research Part B: Methodological*, vol. 90, pp. 263–278, Aug. 2016, doi: 10.1016/j.trb.2016.05.010.
- [14] J. Chen, Z. Yang, P. Cheng, and Y. Shu, "Rebalancing bike-sharing system with deep sequential learning," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 4, pp. 92–98, 2021, doi: 10.1109/mts.2019.2926252.
- [15] M. Elhenawy and H. Rakha, "A heuristic for rebalancing bike sharing systems based on a deferred acceptance algorithm," *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Naples, Italy, 2017, pp. 188–193, doi: 10.1109/mts.2017.8005663.
- [16] Y. Wang and W. Y. Szeto, "Static green repositioning in bike sharing systems with broken bikes," *Transportation Research Part D: Transport and Environment*, vol. 65, pp. 438–457, Dec. 2018, doi: 10.1016/j.trd.2018.09.016.
- [17] B. P. Bruck, F. Cruz, M. Iori, and A. Subramanian, "The static bike sharing rebalancing problem with forbidden temporary operations," *Transportation Science*, vol. 53, no. 3, pp. 882–896, May 2019, doi: 10.1287/trsc.2018.0859.
- [18] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, Aug. 2013, doi: 10.1007/s13676-012-0017-6.
- [19] S. 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*, vol. 69, pp. 180–198, Sep. 2014, doi: 10.1016/j.tre.2014.05.017.
- [20] F. Cruz, A. Subramanian, B. P. Bruck, and M. Iori, "A heuristic algorithm for a single vehicle static bike sharing rebalancing problem," *Computers & Operations Research*, vol. 79, pp. 19–33, Mar. 2017, doi: 10.1016/j.cor.2016.09.025.
- [21] S. C. Ho and W. Y. Szeto, "A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem," *Transportation Research Part B: Methodological*, vol. 95, pp. 340–363, Jan. 2017, doi: 10.1016/j.trb.2016.11.003.
- [22] C. Kloimüller and G. R. Raidl, "Full-load route planning for balancing bike sharing systems by logic-based benders decomposition," *Networks*, vol. 69, no. 3, pp. 270–289, Mar. 2017, doi: 10.1002/net.21736.
- [23] W. Y. Szeto and C. S. Shui, "A static multi-vehicle bike repositioning problem: exact loading and unloading strategies and an enhanced artificial bee colony algorithm," *INFORMS*, Mar. 2018.
- [24] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009, doi: 10.1109/tac.2008.2009515.
- [25] Dk'Bus, "Self-service bicycle system of the city of dunkirk," *dk-velo.fr*, <http://www.dk-velo.fr> (accessed May 30, 2019).

BIOGRAPHIES OF AUTHORS

Daniel Soto Forero    received the M.S. degree in computer science from the University of Antioquia. He has worked at the same university as an assistant professor teaching courses in mathematical modeling, machine learning and optimization. His research interests focus on optimization, agent-based, simulation. He is currently a PhD student and teaches courses in basic programming and web programming. He can be contacted at email: daniel.soto_forero@univ-fcomte.fr.



Yony Ceballos    received his Ph.D. degree in computer science from the National University of Colombia and is currently a tenured professor in the Engineering Faculty at the University of Antioquia. He teaches courses in computer simulation, numerical methods, data analytics, and programming logic. His research interests focus on agent-based, discrete event, and continuous simulation. He is also a member of the System Dynamics Society. He can be contacted at email: yony.ceballos@udea.edu.co.