Intelligent Automation And Soft Computing, 2018

Vol. 24,no. 2, 421–430

A clustering-based approach for balancing and scheduling bicycle-sharing systems

Imed Kacem, Ahmed Kadri, Pierre Laroche

LCOMS EA 7306, Université de Lorraine, Metz, FRANCE
Email: {imed.kacem; ahmed-abdelmoumene.kadri; pierre.laroche}@univ-lorraine.fr

ABSTRACT
This paper addresses an inventory regulation problem in bicycle sharing-systems. The problem is to balance a network consisting of a set of stations by using a single vehicle, with the aim of minimizing the weighted sum of the waiting times during which some stations remain imbalanced. Motivated by the complexity of this problem, we propose a two-stage procedure based on decomposition. First, the network is divided into multiple zones by using two different clustering strategies. Then, the balancing problem is solved in each zone. Finally, the order in which the zones must be visited is defined. To solve these problems, different algorithms based on approximate, greedy and exact methods are developed. The numerical results show the effectiveness of the proposed regulation methodology.

KEY WORDS: Combinatorial optimization, Scheduling problems, Clustering problems, Genetic algorithms, Greedy search algorithm, Branch-and-bound algorithm, K-Means algorithm.

1 INTRODUCTION

PUBLIC Bike Sharing Systems (BSS) have been recently implemented in many big cities around the world. They are one of the solutions to face some public transportation problems, including traffic congestion, air pollution, oil prices and global warming. Obviously, the exploitation and management of BSS imply crucial operational challenges, where the balancing of stations is the most important issue for their operational efficiency and economic viability. Thus, the system requires constant control to balance the network. The monitoring system dispatches motorized redistribution vehicles to balance bicycles between empty and fully overloaded stations. Raviv, et.al. (2013) notice that the balancing problems have similarities with other known routing problems in the literature such as the Pick-up and Delivery Problem (PDP). Indeed, the basic operations performed during balancing are pickup and delivery of identical items (bicycles). According to the literature, this operation can be performed in two different modes as noted in Callé, et.al. (2009):

- **Static mode**: In this mode, the balancing operation can be performed during the night when the utilization rate of the system is very low. The balancing operation is done according to the state of the stations just before starting the operation and taking into account the expected demand for the next day.
- **Dynamic mode**: The operation is performed during the day when the usage rate of the system is substantial. In this case, the bicycle balancing operation is carried out according to real-time states of stations as well as aggregate statistics indicating the station's usage patterns in order to forecast future demands.

The static balancing problem is addressed in several works in literature. Some of studies apply Mixed Integer Programming (MIP) techniques. In Chemla, et.al. (2013), authors assume that each station can be visited several times during an operation with the aim of minimizing the total travelled distance by the vehicle. They propose a branch-and-cut algorithm combined with a relaxed mixed integer linear programming (R-MIP) model and a Tabu search technique in order to obtain upper bounds. Benchimol, et.al. (2011) study the single vehicle routing problem and assume hard constraints for the balancing operation. They present in their study several lower bounds, approximation algorithms as well as a polynomial algorithm for special cases. In Raviv, et.al.

CONTACT Imed KACEM imed.kacem@univ-lorraine.fr

(2013), the balancing problem is addressed by considering multiple vehicles. Authors use mixed integer programming approaches with a convex penalty objective function. The objective is to maximize the users' satisfaction by neglecting the cost of the operation (i.e., the total distances travelled by vehicles as well as the number of loading operations). Similar problem settings are addressed in Di Gaspero, et.al. (2013), Rainer-Harbach, et.al. (2013) and Papazek, et.al. (2014). The main objectives are minimization of gaps between the target balancing levels and the states of stations after performing the balancing operation, as well as the total operational time and the number of operations.

Kadri, et.al. (2016) study the single vehicle routing problem in BSS by considering the static case with the aim of minimizing the weighted time during which some stations remain unbalanced. Many lower and upper bounds are developed and incorporated in a branch-and-bound algorithm. As an extension of this work, Kadri, et.al. (2016) addressed the balancing problem by considering multiple vehicles. They propose different strategies to solve the assignment problem, and they develop a memetic algorithm to solve the vehicle routing problem. Di Gaspero, et.al. (2016) address the balancing problem by means of constraint programming techniques and a large neighborhood search approach. Kloimüllner, et.al. (2015) propose an exact algorithm based on Benders decomposition where they first solve the assignment problem as the master problem, and therefore the vehicle routing problem as sub-problems. Their algorithm is able to solve instances with up to 60 stations.

In practice, the balancing operations are performed by districts, especially, when the network size is large as the Velib' BSS in Paris. From this point of view, Kacem, et.al. (2016) propose a two-step procedure for solving the vehicle routing problem. The idea is to decompose a network into multiple zones by considering a fixed length and width of zones. Thus, the vehicle routing problem is solved by using a genetic algorithm and a nearest neighbour search algorithm. The addressed work in this paper extends our previous work in Kacem, et.al. (2016). In addition to the presented decomposition strategy, we propose another efficient decomposition strategy based on the K-Means clustering algorithm. Moreover, we propose other algorithms to solve the vehicle routing problem including the greedy search algorithm, the combined method and the branch-and-bound algorithm.

In the remainder of the paper we use the abbreviation SZ-VRP to denote our problem. This paper is organized as follows. In Section 2, we describe the problem and the associated mathematical formulation. In Section 3, we present the proposed solving methods including the genetic algorithm, the nearest neighbour search algorithm, the greedy search method, the combined algorithm and the branch-and-bound algorithm. In Section 4, we test and compare performances of the proposed algorithms as well as the decomposition strategies on a large set of instances. Finally, we conclude the paper with some perspectives for future research.

2 PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

2.1 Problem description

THE static balancing problem in a BSS consists in finding a schedule that the balancing vehicle must follow such that one or multiple objectives are minimized or maximized, and under some constraints. A vehicle starts always a tour at a deposit, and visits the unbalanced stations in a specific order. In this work, the aim is to minimize the weighted sum of the waiting times during which the stations remain imbalanced until the arrival of the balancing vehicle. As shown by Figure 1, the imbalance is measured as a gap between the current state (E_i) of station i and its fixed threshold R_i (can be R_i^- *or* R_i^+).

This problem is strongly NP-hard (see Kadri, et.al. (2016)), and its complexity increases proportionally to the network size. In real situations, the balancing operations for large networks are performed by decomposing a network into multiple zones (clusters). Many works in literature deal with the travelling salesman problem (TSP) by using clustering techniques. Ding, et.al. (2007) address the large scale TSP, and propose a two-level genetic algorithm for the clustered TSP. In (Helsgaun, 2014), the author considers the Clustered Travelling Salesman Problem (CTSP) as an extension of the standard TSP, and solves the CTSP by using the Lin-Kernighan-Helsgaun algorithm. Karapetyan, et.al. (2011) address the generalized TSP and propose an adaptation of the Lin-Kernighan algorithm. The adapted algorithm is able to solve instances with large size and provides suboptimal solutions in a reasonable amount of time. With this motivation, we propose an approach based on the decomposition of the solution process into two steps (see Figure 2).

- **Step 1 (Zoning):** We determine a set of zones, where each zone is represented by a single point in the routing step of the algorithm. We propose two strategies for partitioning the network into multiple zones as follows:
  - **Strategy 1 (ST_1):** the network is decomposed into multiple zones by considering fixed length and width of zones. Each zone includes a virtual local deposit, where its coordinates represent the center of gravity calculated from the coordinates of stations present in that zone (see Figure 2.b and 2.c).
  - **Strategy 2 (ST_2):** the decomposition of the network is done by using the K-Means algorithm. This algorithm is one of the

simplest algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to divide a given set of stations into a number of clusters (m) fixed a priori. The algorithm starts by generating m random centroids representing the local deposits of zones, then forms (m) clusters by assigning a subset of stations to each centroid according to their positions (closest to centroid). When all stations are assigned, the positions of centroids are recalculated based on the assigned stations. This process is repeated until the coordinates of centroid become stable or a maximal number of iterations is reached (See Figure 2.e).

- **Step 2 (Routing)**: This step determines the sequence of the balancing vehicle in each zone starting from its local deposit. First, it calculates for each zone the sequence linking the stations belonging to (i.e., it constructs the intra-zones paths), then it calculates a sequence linking the zones (i.e., it constructs the inter-zones path). To solve the routing problem, we propose in Section 3 many algorithms based on exact and heuristic methods.

Figure 1. The min-max balancing levels in a station

2.2 *Mathematical formulation*

LET us consider a set N containing the network stations (0, ... , n). We consider the Euclidean case where all nodes including stations and deposits are characterized by their (x, y) coordinates. The vehicle starts a tour from the central depot at time $t_0 = 0$ and moves between zones according to a planned sequence. Each station must be visited only one time in a tour. The aim is to define the order of visits of the network stations such that their remaining time in an imbalanced state is minimized. To model this problem, we introduce the following notations, variables, and parameters:

N: A set of nodes including the stations indexed by i = 1,...,n, and the central depot ($i = 0$)

E_i: Current number of bicycles at station i before the repositioning operation starts

R_i^-: Minimal level of the required number of bicycles at station i

R_i^+: Maximal level of the required number of bicycles at station i

$d_{i,j}$: Travelling time from station i to station j

t_i: Arrival time of the vehicle to station i

w_i: Gap between E_i and the confidence level of R_i, where its value is equal to:

$$\begin{cases} R_i^- - E_i & \text{if } E_i < R_i^- \\ E_i - R_i^+ & \text{if } E_i > R_i^+ \\ 0 & \text{otherwise} \end{cases}$$

The model can be described as follows:

$$\text{Minimize} \sum_{i=1}^{n} t_i w_i \tag{1}$$

$$\sum_{j=1}^{n} x_{0,j} = 1 \tag{2}$$

$$t_0 = 0 \tag{3}$$

$$t_j \geq t_i + x_{i,j} d_{i,j} + (x_{i,j} - 1).M \quad \forall i \neq j,\ i,j \in N \tag{4}$$

$$\sum_{i=0}^{n} x_{i,j} = 1 \quad \forall j \in N \tag{5}$$

$$\sum_{i=0}^{n} x_{j,i} = 1 \quad \forall j \in N \tag{6}$$

$$x_{i,j} \in \{0,1\} \tag{7}$$

The objective function in Equation 1 minimizes the sum of times, weighted with the values of imbalances w_i. The value t_i-t_0 represents the time where a station i remains outside the confidence level until the arrival of the balancing vehicle. Constraint (3) forces the departure of the vehicle from the central depot. Constraint (2) represents the departure time of the vehicle from the central depot (initialized to 0), and the set of constraints (4) insure the necessary time required for the displacement of the vehicle from station i to station j. We associate to these constraints the *Big M method*. Constraints (5) and (6) are the vehicle conservation equations, i.e., a station can be visited only one time by a vehicle in a tour. Finally, the constraints (7) are the binary constraints, i.e., $x_{i,j}$=1 if the vehicle travels from station i to station j, and it equals 0 otherwise.

3 SOLVING METHODS

IN this section, we present the developed algorithms for solving the vehicle routing problem. These algorithms include heuristics and exact methods as detailed in the next subsections. The common factor between these algorithms is that they calculate first for each zone a sequence linking stations, then it links zones.

3.1 Genetic algorithm

Genetic algorithms are promising techniques for many optimization problems. They have been widely used and are well-known for giving good approximations to complex NP-hard problems (Kacem, 2013) as the SZ-VRP. A solution for the SZ-VRP must identify the order in which the zones and the stations of each of these zones must be visited. Hence, a suitable presentation is done by using two chromosomes. The first chromosome includes the solution linking zones, and the second one gives the order in which the stations must be visited in a given zone. Figure 3 illustrates the encoding of solutions.

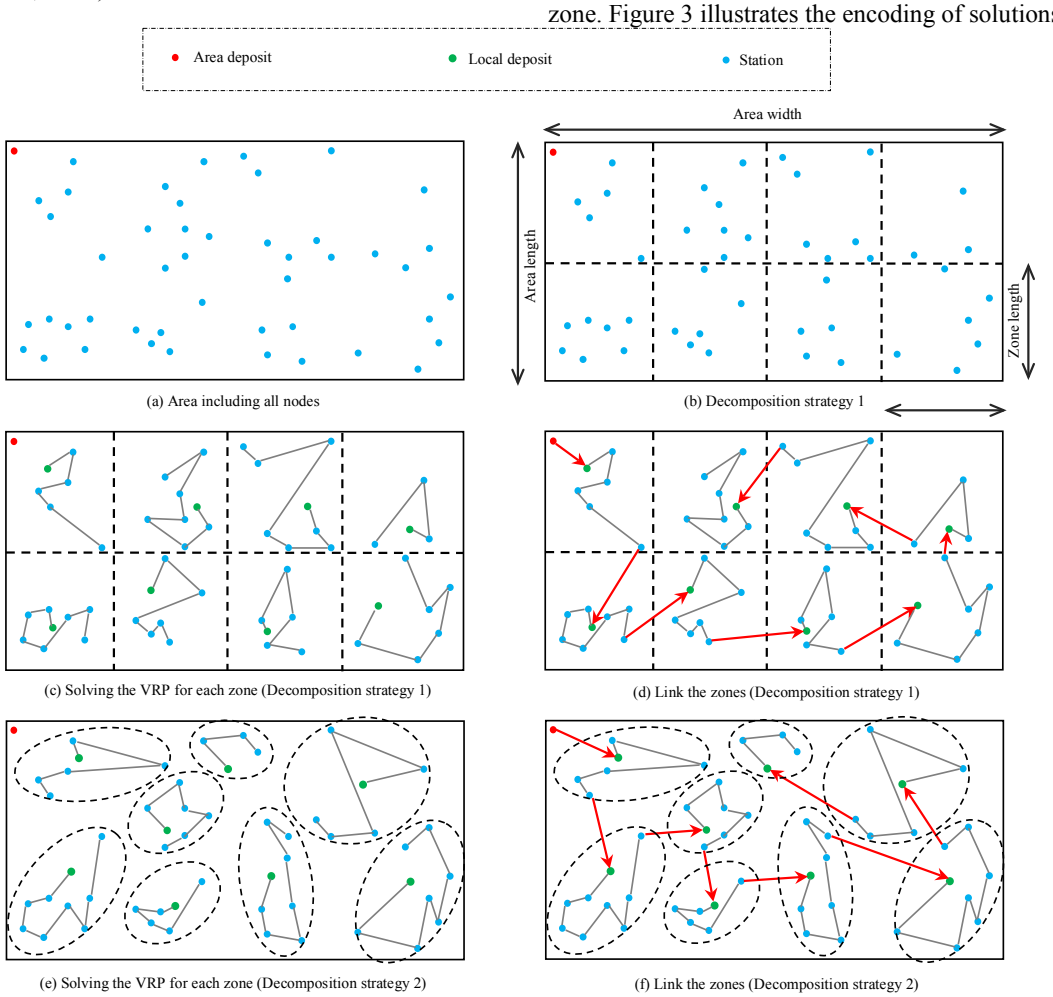


Figure 2. Solution process

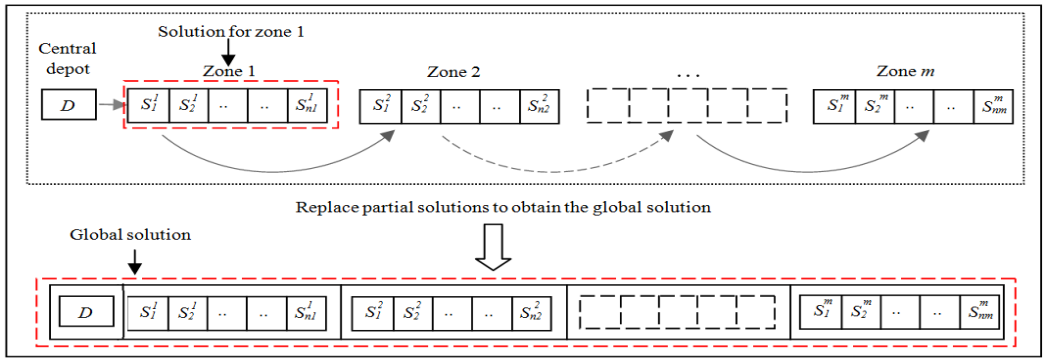


Figure 3. Solution representation

The Genetic Algorithm (*GA*) starts by computing the intra-zone sequence to link the stations in each zone. Then, it calculates the inter-zones sequence to link the zones. The representation of an inter-zones solution is done by means of chromosome, where each gene includes a zone number. Similarly, the representation of an intra-zone solution is done by means of another chromosome, where each gene includes a station number. Thus, the final inter-zones and intra-zone solutions are merged in one chromosome, such that each gene in the inter-zones chromosome is replaced by its corresponding intra-zone chromosome.

3.2 Nearest Neighbour Search algorithm

The Nearest Neighbour Search (*NNS*) is an optimization algorithm, which starts by an initial solution calculated by choosing at each step the point that minimizes the partial objective function. Once calculated, the initial solution is improved by performing some moves within the solution until reaching a limited number of iterations. As in the *GA*, the final solution is calculated in two steps: we calculate first intra-zone sequences to link the stations in each zone, and then we link the zones.

3.3 Greedy Search algorithm

Greedy programming techniques use some general principles or common sense knowledge to generate a sequence of sub-optimum that hopefully converges to an optimum value. Greedy algorithms look for simple, easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit (Bendall and Margot, 2006). The developed Greedy Search algorithm (*GS*) is based on an efficient Depth-First Strategy and a Branch-and-Bound algorithm. In this case, the Branch-and-bound does not require the initialization of the initial upper bound. At each level of the tree, the node giving the best lower bound is explored, and the remaining nodes are ignored and removed immediately. The search is done in depth until obtaining a leaf (feasible solution). The used branch-and-bound algorithm is described in the next sub-section. More details about the used lower bounds can be found in Kadri, et.al. (2016).

3.4 Branch-and-bound algorithm

The developed Branch-and-Bound algorithm (*BB*) uses a search tree. It consists initially of only the root node, which is developed in a dynamical way during the process. The value of the upper bound is initialized with the value of the best feasible solution from *GA*, *NNS*, *GS* and *CS* (here, CS denotes the Combined Strategy described in §3.5). The selection consists in choosing a node for exploration from the group of live nodes corresponding to unexplored feasible sub-problems by using a selection strategy. The selection is basically based on the value of the lower bound of the node, which represents a partial schedule of the vehicle. The branching scheme consists in assigning a new task after a partial schedule. The selected node is then explored, and the children of the node (unscheduled stations (resp. zones) in the sequence of stations (resp. zones)) are constructed. In this way the subspace is divided into other sub-spaces. Then, for each sub-space a lower bound is calculated. If the value of such a lower bound is equal or greater than the upper bound, then the sub-problem is eliminated, as any feasible solution of the sub-problem cannot be better than the best known solution. The search tree is explored by using Depth-First Search strategy (*DFS*), which allows us a fast update of the upper bound when possible, and therefore an elimination of nodes when the value of their lower bounds are equal or greater than the value of the best upper bound. More details about the used branch-and-bound algorithm and its associated lower bounds can be found in Kadri, et.al. (2016).

3.5 Combined strategy

We developed a combined method (*CS*), which uses the greedy search and the branch-and-bound algorithms. The choice of the optimization method depends on the size of problem (respectively sub-problems) i.e., the number of zones (respectively the number of stations in each zone). Indeed, the branch-and-bound is able to solve problems with up to 15 nodes in a short computation time. Therefore, in order to avoid long computation times, we limit the use of the branch-and-bound method only for problems with size up to 15 nodes. Otherwise, we use GS algorithm, which is able to solve problems with larger size in a very short computation time.

4 RESULTS

4.1 Test environment

IN this section, we outline the numerical experiments performed in order to evaluate our algorithms. All developed algorithms are implemented in the C++ Language and tested on an Intel Core i7 2.7 GHz processor with 8 GB of RAM, and under Windows 10 environment. Due to the unavailability of real BSS data, and in order to test our algorithms on a large set of instances by taking into account different combinations of parameters, we generate randomly our instances as follows. The network size n was chosen in 20, 40, 60 and 80, the grid coordinates (x,y) (resp. (x',y')) denoting the grid limits were fixed to (0, 0) (resp. (200, 200)). As we consider the symmetric Euclidean case, the coordinates (x, y) of stations and deposits were generated in the grid limits, and the distances between stations were calculated by using the Euclidean distance formula. The weights representing the imbalances of stations were chosen in the interval [1, 10]. For the decomposition strategy 1, we consider five combinations of parameters as

reported in Table 1. In order to compare the results provided by the decomposition strategies, we consider the same number of zones for ST_1 and ST_2. In order to obtain a reliable statistical average, we consider 8 groups for each combination of parameters n and m, where each group measures the average result of 10 instances.

Table 1. Combinations used for decomposition

Number of zones (m)	Grid cutting			
	L_{Grid}	H_{Grid}	L_{Zone}	H_{Zone}
1	200	200	200	200
2	200	200	200	100
4	200	200	100	100
8	200	200	100	50
16	200	200	50	50

For an easy analysis, we denote by N_nM_m the average result of the 8 groups, where each group measures the average results obtained from 10 instances when using n stations and m zones. The developed algorithms are tested on each instance by considering the two proposed decomposition strategies. We denote by A_{ST1} (resp. A_{ST2}) the algorithm A using the decomposition strategy ST_1 (resp. ST_2). We set the following parameters for our algorithms:

- ***GA*:** we perform one-point crossover and we fix the rates of crossovers (resp. mutations) to 0.9 (resp. 0.2). The number of iterations is limited to 200.
- ***NNS***: We limit the number of iterations to 200, otherwise, the algorithm stops if no improvement is recorded during 20 iterations.
- ***GS***: there is no specific requirement for this algorithm. We simply perform a *DFS* search until reaching a leaf node.
- ***BB***: the algorithm starts from the root node and explores all developed nodes by using the *DFS* strategy, until an optimal solution is found.
- ***CS***: this method combines the ***GS*** and ***BB*** algorithms. Due to the uncertainty of the computation time of the branch-and-bound algorithm, we limit the use of the *BB* algorithm for problems with ***n*** or ***m*** up to 15 stations. Otherwise, we use the ***GS*** algorithm.

4.2 Results analysis

WE analyze the obtained results according to two criteria: *i*) we test the impact of zoning on the quality of solutions including the number of zones and the chosen decomposition strategy, and *ii*) we compare the performances of the developed algorithms. Tables 2-5 provide the average results obtained, where the best solutions and their associated CPU times are highlighted in bold black, and the second solutions and their associated CPU times are reported in bold green.

4.3 Impact of decomposition

Figures 5 and 6 summarize the results reported in Table 2 and Table 4, and show that performing the balancing operations by splitting the network into multiple zones provides better results than when considering only a single area. However, from a given number of decompositions (m = 2 for n = 20, and m =16 for n = 40, 60 and 80), the solutions quality deteriorate. Indeed, the number of decompositions depends on the total number of stations and their locations. Moreover, Table 2 and Table 4 show that the zoning strategy has an impact on the quality of solutions. From Figure 6 (left side), we observe that *BB* and *CS* algorithms using the zoning strategies ST_1 and ST_2 provide very close results. In addition, the other algorithms using the zoning strategy 2, which is based on the K-Means algorithm, provide at the most of cases better results compared to algorithms that use ST_1. This is due to the assignment strategy used in ST_2, which assigns stations to the nearest local deposit. In contrast, the zoning strategy 1 forces stations to belong to a given zone even if some of these stations are close to a local deposit of an adjacent zone as illustrated by Figure 4.

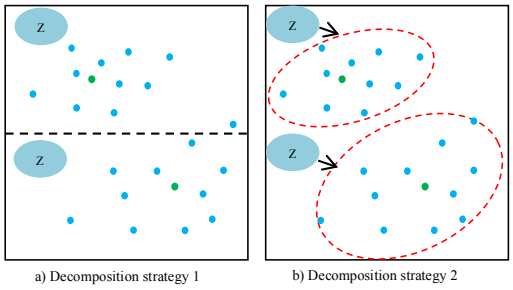


Figure 4. An example of assignment of stations with strategies ST1 and ST2

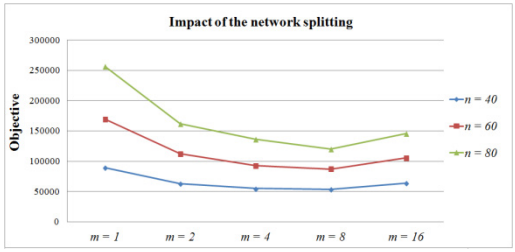


Figure 5. Impact of network decomposition (Average results from 8 groups of 10 instances)

4.4 Performances of algorithms

Figure 6 and Tables 2-3 show that *GA* is not able to provide good results for large size problems and sub-problems, and requires longer CPU times compared to GS and CS algorithms in particular when $m = 1$. Otherwise, *GA* provides close results compared to *NNS*, *GS* or *CS* algorithms when the problems' and sub-problems' sizes are small (i.e., when the size of m increases, the number of stations decreases per zone). This is due to the small size of the search space and to the efficiency of the genetic operators (crossovers, mutations).

Kadri, et.al. (2016) solved the Lagrangian relaxation of the presented mathematical formulation in order to obtain a valid lower bound. In the new model, the vehicle flow constraints are relaxed and included in the objective function. Despite that the relaxed problem is easier than the original one, CPLEX (version 12.6) was not able to solve it in 1 hour for problems with size of n equal to 20 nodes.

Tables 2 and 3 show that the branch-and-bound algorithm is able to find optimal solutions for problems with $n = 20$ stations in short computation time (166 sec in average), while considering the strong NP-hardness of the studied problem. Moreover, the gap between the optimal solution and the best heuristic method ($GAP_{H^*\text{-}OPT}$) gives an average of 8%.

For problems with larger values of n ($n > 20$), and in order to avoid long computation times, we perform experiments by using our developed approximate algorithms *GA, GS, NNS* and *CS*. The results of these experiments are given in the tables 4 and 5, and illustrated by Figure 6. As the *CS* algorithm uses the Branch-and-Bound method, we limit the use of *BB* algorithm within the *CS* algorithm in order to optimally solve only the problems and sub-problems with sizes up to 15 nodes, otherwise the greedy search is performed. From Table 2, Table 4 and Figure 6, we observe that the proposed algorithms give close values. However, the *CS* algorithm provides better results for most of cases (except for the case when $m = 1$). In this case, the *CS* algorithm performs only a greedy search without using the *BB* method as $m = 1$ and $n>15$. Beside to this, the *NNS* algorithm computes an initial greedy solution and then it improves it during a limited number of iterations, which gives better results compared to *CS* algorithm with a very small gap (1% in average).

Table 2. Performances of algorithms for n = 20 (Average results from 8 groups)

Groups	*n*	*m*	GA_{ST1}	GA_{ST2}	*NNS*	*NNS*	GS_{ST1}	GS_{ST2}	CS_{ST1}	CS_{ST2}	BB_{ST1}	BB_{ST2}
N20M1	2	1	**29843**	**30043**	30392	30424	31594	31594	31594	31594	**27724**	**27724**
N20M2	2	2	25352	25273	25886	25789	25278	25865	**23819**	**23812**	**23769**	**23805**
N20M4	2	4	24193	23987	25072	25007	24779	25094	**24057**	**23885**	**24051**	**23877**

Table 3. Computation times of algorithms for n = 20 (Average results from 8 groups)

CPU	*n*	*m*	GA_{ST1}	GA_{ST2}	*NNS*	*NNS*	GS_{ST1}	GS_{ST2}	CS_{ST1}	CS_{ST2}	BB_{ST1}	BB_{ST2}
N2	20	1	**3,4**	**3,4**	0,6	0,6	0,1	0,1	0,1	0,1	**159,4**	**165,9**
N2	20	2	0,6	0,6	0,3	0,3	0,2	0,2	**0,7**	**0,7**	**0,6**	**0,8**
N2	20	4	0,8	0,8	0,4	0,4	0,2	0,2	**0,5**	**0,6**	**0,4**	**0,6**

Table 4. Performances of algorithms for n = 40, 60 and 80 (Average results from 8 groups)

Groups	*n*	*m*	GA_{ST1}	GA_{ST2}	NNS_{ST1}	NNS_{ST2}	GS_{ST1}	GS_{ST2}	CS_{ST1}	CS_{ST2}
N40M1	40	1	106348	106552	**89242**	**89242**	89529	89529	89529	89529
N40M2	40	2	66361	63233	65393	64988	62896	63843	**62841**	**62979**
N40M4	40	4	55856	55335	60765	59753	58105	58017	**55560**	**54941**
N40M8	40	8	57775	56350	58739	57293	56380	56086	**53293**	**53281**
N40M16	40	16	74287	79263	69006	68799	64017	64074	**63892**	**63907**
N60M1	60	1	248321	239519	**167069**	**167069**	169554	169554	169554	169554
N60M2	60	2	143952	137538	114451	113815	110904	113622	**110904**	**112260**
N60M4	60	4	104311	102405	100214	97566	96178	96326	**94846**	**93027**
N60M8	60	8	94317	92958	97389	96582	93217	93131	**87494**	**87215**
N60M16	60	16	111595	114181	111368	110866	106885	106394	**106366**	**105747**
N80M1	80	1	377208	381911	**254713**	**254713**	256768	256768	256768	256768
N80M2	80	2	273303	276568	165078	164885	160488	164171	**160488**	**161895**
N80M4	80	4	151931	147757	141979	140577	136846	136959	**136592**	**136108**
N80M8	80	8	130775	128916	132060	130489	128732	127544	**121727**	**120383**
N80M16	80	16	161829	164549	155718	153559	147528	146833	**146516**	**145455**

Table 5. Average CPU times of algorithms for n = 40, 60 and 80 (Average results from 8 groups)

CPU time	n	m	GA_{ST1}	GA_{ST2}	NNS_{ST1}	NNS_{ST2}	GS_{ST1}	GS_{ST2}	CS_{ST1}	CS_{ST2}
N40M1	40	1	1,5	1,6	**6,4**	**6,2**	0,3	0,4	0,4	0,4
N40M2	40	2	0,3	0,3	0,6	0,6	0,4	0,4	**0,6**	**0,5**
N40M4	40	4	0,2	0,2	0,9	0,9	0,4	0,4	**1,3**	**1,4**
N40M8	40	8	0,2	0,2	1,0	1,0	0,5	0,5	**1,0**	**1,2**
N40M16	40	16	0,3	0,3	1,4	1,5	0,7	0,7	**1,4**	**1,9**
N60M1	60	1	21,5	20,2	**10,7**	**10,1**	0,9	0,9	0,9	0,9
N60M2	60	2	0,9	0,9	0,9	0,9	0,5	0,5	**0,5**	**0,5**
N60M4	60	4	1,0	1,1	1,1	1,2	0,5	0,5	**1,6**	**1,3**
N60M8	60	8	1,4	1,4	1,2	1,2	0,5	0,5	**1,4**	**1,5**
N60M16	60	16	2,5	2,7	1,8	1,9	0,7	0,7	**1,5**	**2,0**
N80M1	80	1	39,9	40,3	**18,1**	**17,8**	2,1	2,1	2,1	2,1
N80M2	80	2	1,4	1,5	1,1	1	0,7	0,7	**0,8**	**0,8**
N80M4	80	4	1,2	1,2	1,1	1,2	0,6	0,6	**0,7**	**0,9**
N80M8	80	8	1,5	1,5	1,2	1,3	0,6	0,6	**2,3**	**2,4**
N80M16	80	16	2,6	2,8	1,9	1,9	0,8	0,8	**1,8**	**2,2**

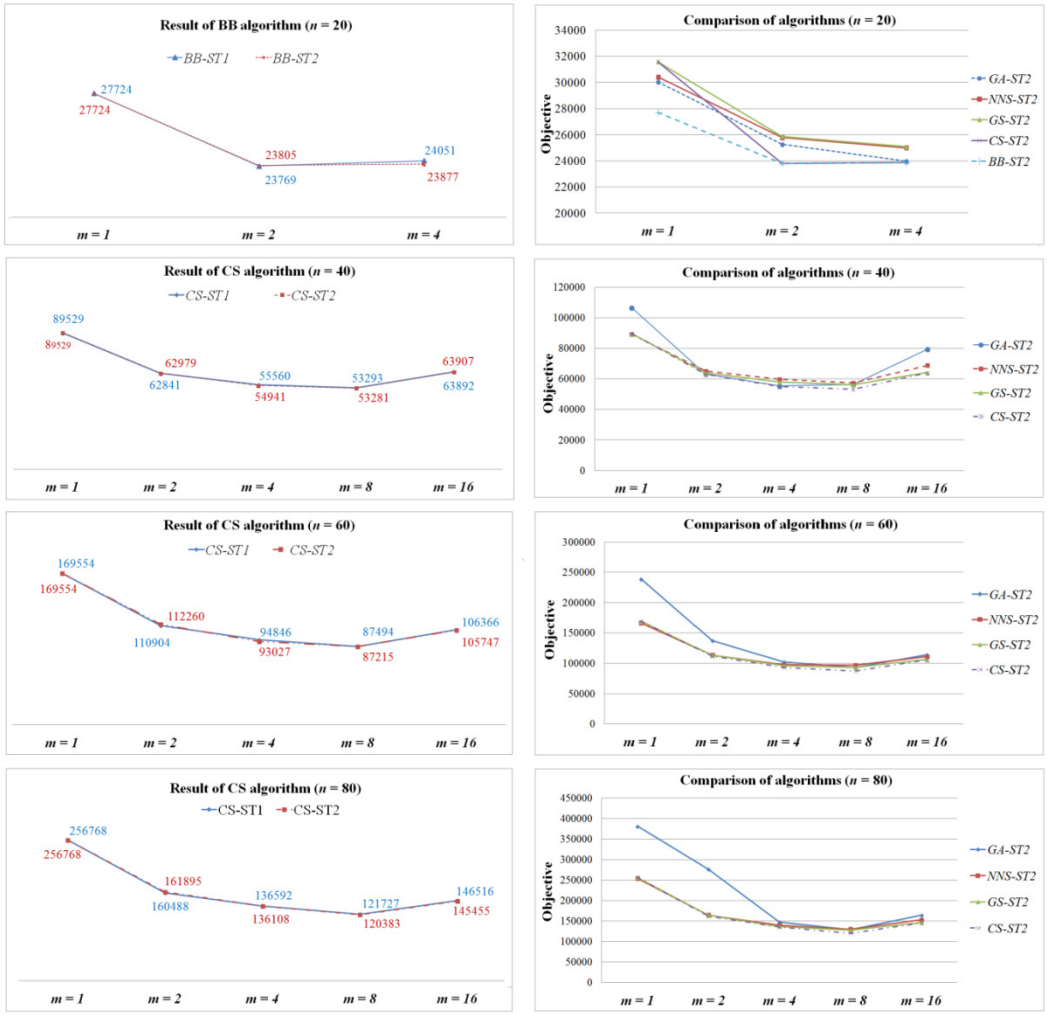


Figure 6. Performances of algorithms for n = 40, 60, 80 (Average results from 8 groups)

Indeed, the gap between the results provided by algorithms *GS* and *CS* increases by the increasing of the number of zones *m* as shown by the results reported in the tables 2 and 4. This is due to the decreasing of the number of stations per zone, and therefore the use of the *BB* method within the *CS* algorithm, which improves at the most of cases the results obtained by *GS* algorithm. Otherwise, the *CS* algorithm is able to solve problems with up to 80 stations in small computation time (2 seconds in average). Such a time is neglected in real life situations.

Finally, the proposed decomposition strategies and their associated algorithms provide a decision tool able to solve problems with different sizes in short computation time. Moreover, the use of the K-Means decomposition strategy and the combined algorithm seems to be a very interesting method to solve problems up to 80 stations, and provides solutions of good quality in short computation time.

5 CONCLUSION

THIS paper studied the balancing of bicycle-sharing systems considering the static case. The problem is similar to other known problems in the literature (e.g., the pick-up and delivery problem). The investigated problem is strongly NP-hard and requires the design of specific methods able to provide solutions of good quality in short computation time.

In this context, we proposed a 2-step approach for solving the balancing problem. First, the network is split into multiple zones by using two different clustering strategies including the K-Means algorithm. Once the network is split, two sub-problems are solved: the intra-zone vehicle routing problem aiming to find a sequence within each zone is solved. Thus, the inter-zones' vehicle routing problem linking the zones is solved.

Different resolution methods were developed and tested on a large set of instances. These methods integrate different kinds of algorithms as the genetic algorithm, the greedy search algorithm, the nearest neighbour search algorithm, the combined method and the branch-and-bound algorithms. The experiments were conducted on a large set of instances in order to evaluate and compare the developed algorithms as well as to show the impact of splitting the network on the quality of solutions. The obtained results showed the interest of balancing a network by considering multiple zones. It is emphasized that the number of zones and the choice of the decomposition strategy of network affect the quality of solutions.

As a conclusion, the proposed decomposition methods and their associated heuristic algorithms seem to be a very interesting method to solve the SZ-VRP, and to provide solutions of good quality in short computation time (at most 2 seconds when using the *CS* algorithm) despite the strong NP-hardness of the investigated problem. As a perspective, we aim to develop an approximation algorithm (with an approximation ratio) for the addressed problem.

6 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

7 REFERENCES

G. Bendall and F. Margot. (2006). Greedy Type Resistance of Combinatorial Problems. Discrete Optimization 3, 288-298.

M. Benchimol, P. Benchimol, B. Chappert, A.D.L. Taille, F. Laroche, F. Meunier, and L. Robinet. Balancing the stations of a self-service “Bike Hire” system. RAIRO Operations Research, 45, 37-61.

E. Call. (2009). Director of Operation in Vlib. April 2009, Personal communication.

D. Chemla, F. Meunier, and R. Wolfer-Calvo. (2013). Bike sharing systems: solving the static rebalancing problem. Discrete Optimization. 10(2), 120-146.

L. Di Gaspero, A. Rendl, and T. Urli. (2016). Balancing bike sharing systems with constraint programming. Constraints, 21(2), 318-348.

L. Di Gasperro, A. Rendl, and T. Urli. (2013). A hybrid ACO+CP for balancing bicycle sharing systems. In: Blesa, M.J., Blum, C., Festa, P., Roli, A., Sampels, M. (eds.) HM 2013. LNCS, vol. 7919, pp. 198-212. Springer, Heidelberg.

C. Ding, Ye Cheng, and M. He. (2007). Two-Level Genetic Algorithm for Clustered Travelling Salesman Problem with Application in Large-Scale TSPs. Tsinghua Science and Technology, 12(4), 459-465.

K. Helsgaun. (2014). Solving the Clisteres Traveling Salesman Problem Using the Lin-Kernighan-Helsgaun Algorithm. in Copmut. Sci. Rep, Roskilde University. ISSN 01099779.

I. Kacem. (2013). Genetic Algorithms for Solving Flexible Job Shop Scheduling Problems. In book: Metaheuristics for Production Scheduling, pp. 19-44.

I. Kacem, A.A. Kadri, and P. Laroche. (2016). A 2-Steps Procedure for Solving the Inventory Balancing Problem with Time Constraints. 46rd International Conference on Computer and Industrial Engineering (CIE46). October 29-31 2016, Tianjin (China). ISSN 2164-8689.

A.A. Kadri, I. Kacem, and K. Labadi. (2016). A branch-and-bound algorithm for solving the static rebalancing problem in bicycle sharing systems. Computers & Industrial Engineering journal (Elsevier). 95, 41-52.

A.A. Kadri, I. Kacem, and K. Labadi. (2016). A Memetic Algorithm for Solving the Multiple Vehicles Routing Problem in Bicycle-Sharing Systems. The 12th FLINS International

Conference. August 24-26 2016, Roubaix (France). ISBN 978-981-3146-96-9.

D. Karapetyan and G. Gutin. (2011). Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem. European Journal of Operational Re- search. 208(3), pp. 221-232.

C. Kloimllner, P. Papazek, B. Hu, and G.R. Raidl. A Cluster-First Route-Second Approach for Balancing Bicycle Sharing Systems. In International Conference on Computer Aided Systems Theory, pp. 439-446. Springer International Publishing.

P. Papazek, G.R. Raidl, M. Rainer-Harbach, and B. Hu. (2013). A PILOT/VND/GRASP hybrid for the static balancing of public bicycle sharing sys- tems. In: Moreno-Diaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST. LNCS, vol. 8111, pp. 372-379. Springer, Heidelberg.

M. Rainer-Harbach, P. Papazek, B. Hu, and G. R., Raidl. (2013). Balancing bicycle sharing systems: A variable neighborhood search approach. In: Middendorf, M., Blum, C. (eds.) EvoCOP 2013. LNCS, vol. 7832, pp. 121-132. Springer, Heidelberg.

T. Raviv, M. Tzur, and I. Forma. (2013). Repositioning in a Bike-Sharing System: Models and Solution Approaches. EURO Journal of Transport and Logistic, 2:187-229.

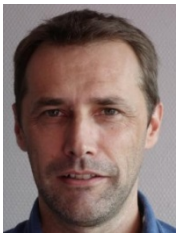
8 NOTES ON CONTRIBUTORS

Imed KACEM is Full Professor since 2009 at the Université de Lorraine, France, in Computer Science. He is the Founder and the Head of LCOMS Laboratory.

His scientific activity is mainly in the Operational Research (design of exact and approximate algorithms with a guaranteed performance for the NP-hard combinatorial problems).

Ahmed KADRI obtained his MSc degree and PhD degree in Computer Science from Université de Lorraine.

He is currently with the LCOMS laboratory and the Institut Universitaire de Technologie de Metz (Université de Lorraine). His research interests include operational research, combinatorial optimization, modeling, performance evaluation and optimization of stochastic systems.



Pierre LAROCHE is currently Assistant Professor in Computer Science, at the Université de Lorraine, France. His first research experiences were in artificial intelligence, and he is now a member of the « Decision and Optimisation » team of the LCOMS Laboratory, working in the field of operations research.