Planning Nightly Rebalancing in Bike Sharing Systems - GIRA case study

Miguel Chastre Rodrigues
miguel.chastre.rodrigues@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa – Portugal
July 2019

Abstract. Over the last few decades, environmental awareness and the international political landscape have induced each country to seek a more sustainable growth, so shared mobility and low emission transportation initiatives were encouraged. Both coincide in shared bicycle networks, whose diffusion rate has shoot up in recent years. Nowadays, almost all major cities have some variant of these systems. However, asymmetric usage flows leave the network unbalanced: some stations have an excess of bikes, preventing the parking of incoming users, while others do not have enough to meet the demand. Typically, one or more vans reposition bikes so that the network returns to a steady state. It is in optimizing the route of these vans that consists of the so-called “Bike Rebalancing Problem” (BRP). This work evaluated some of the existing formulations to solve the BRP and selected the one considered the best fitting for the Lisbon case study, the GIRA network. The chosen MILP formulation has been generalized, in order to take into account the different types of vehicles and bicycles, the autonomy of each vehicle, the duration of its route and the stations that do not require repositioning but inspection actions. Besides, the possibility of temporarily storing bicycles at an intermediate route station has been introduced in the model, as to relax the restrictions imposed by the vehicles’ capacity. After studying the incorporation of some simple acceleration procedures, the model, solved in a Branch-and-Cut scheme, recalculated under an established time-stretch the routes of five case study instances. Comparing the results with the routes that were actually performed, a 30% to 75% decrease in the total length has been verified for every instance.

Keywords: Bike Sharing Systems; Bike Rebalancing Problem; Repositioning; Optimization; MILP; Branch-and-Cut

1. Introduction

In recent decades, widespread ambitions for green urbanizations have made Shared Mobility a hot topic. Sustained by technological advances, the previously infamous Bike Sharing Systems (BSS) now glint as a reliable yet economic solution, and its fast-paced global adoption foresees a long-lasting stay on the metropolitan panorama.

Nonetheless, as users pick up a bike at a station and park it elsewhere, bicycles tend to concentrate on particular sectors of the city. The bike sharing network enters a state of imbalance that impairs its proper functioning and undermines user satisfaction, threatening the system’s successful establishment. Operators must, then, reinstate the system’s harmony, and the decision for the best routes to be taken is an assignment titled the Bike Rebalancing Problem (BRP).

The present work contemplates the nightly BRP at the recently introduced Lisbon BSS, GIRA. Lisbon’s peculiar topography and social dynamics result in an easily imbalanced system that, due to contractual duties, is to be levelled back under one of the most challenging time stretches encountered. Nightly rebalancing takes place when the system is inactive and arranges it for the critical morning period. Existing models have been selected, validated and improved according to identified fieldwork restrictions. Using operator’s data, real BRP demands have been worked into input for the model, and the results were then compared and pored over for better understanding of the model’s limitations by means of a sensitivity analysis.

The Bike Rebalancing Problem

As Raviv et al. [1] point out, “persistent unavailability of bicycles and/or lockers engenders distrust among the system’s users and could eventually lead them to abandon it.”. Unlike other inventory dilemmas, increasing the total number of bikes on the network would not solve the imbalance situation, as the number of free docks would decrease by the same figures the bike inventory would increase. The most common solution is the employment of a fleet of vehicles to relocate the surplus of bicycles from one station to another one in need.

The BRP falls under the scope of a VRP class: the extensive Pickup and Delivery Vehicle Routing Problems (PDVRPs). Usually, in these problems, the network consists of a depot and customers that either pickup nodes or delivery nodes and the goal is to minimize the total route length while satisfying every customer’s demand, by repositioning the commodities in question between them, which is an NP-complete problem. It is implicit that each vehicle can never carry a greater load than its respective capacity.

The formulations used for the different VRPs are either path-based or arc-based. Arc-based formulations associate a binary variable to each arc,
to indicate whether it is travelled in the final solution or not. A commodity flow arc-based formulation associates, as well, flow variables that relate to the load of the vehicle travelling the arc. Additionally, arc-based formulations are divided according to their approach to subtours. While some formulations present a polynomial number of subtour elimination constraints, others eliminate subtours with a set of constraints that, if no separation procedure is considered, grows exponentially with network size. Separation procedures determine which sets of arcs constitute a subtour, consequently violating the respective subtour breaking constraints, and iteratively introduce these into the model.

Data analysis of a real-world case
GIRA is the municipal BSS acting in Lisbon. It is more than an alternative transportation choice and its performance in the view of the citizens, in which a competent network rebalancing poses an important role, is crucial for its acceptance in this relatively early life stage. Data relative to the months of January and February of 2019 has been provided by the operator and is analysed throughout this document.

The system presently counts 74 stations within a 10 km radius from the city centre but is planned to hold up to 140 in a close future. The network is divided into four different zones. Zone 1 (Pq. das Nações), a plain area located northeast of the city centre, appears to be almost isolated from the rest of the network. Zone 2 (downtown) does not seem to be as regularly used as the remaining areas, probably because of its topographic unevenness. On the other hand, central Zones 3 and 4 contain the main avenues prepared for bicycle dislocations, on which journeys concentrate, justifying the previous investment in these infrastructures. In general, stations located near relevant transit hub spots, such as 481 (Campo Grande) or 446 (Entre-Campos), and the ones across Av. Duque de Ávila, Saldanha, are the ones with the most affluence. Train and metro station Oriente also appears as a main base point of the network.

Complementing this information, Figure 1 displays each station’s net flow: the difference between the number of arrivals and departures. This is a form of evaluating the effects of network imbalance that is to be solved in this work. Dark green circles indicate a net inflow, and light green circles a net outflow.

Zone 1 and the periphery of Zone 4 seem to be almost self-sufficient in rebalancing terms, whereas, when it comes to trips that are not performed inside the same zone, evening the net flow, it appears that most users travel in a downhill direction from Zone 4 to Zones 2 and 3. Even though relatively few of such trips are performed, Figure 1 suggests that this is the main cause of bike inventory imbalance in the network.

Indeed, it is interesting to note that a station with relatively big numbers of absolute outflow and inflow does not necessarily tend to become unbalanced (net in/outflow) at the end of the day, consequence of daily routines. The referred station 105 appears to be a station at which users arrive at and depart from in the same day, such that the BSS is probably used most as a “last-mile” solution to get from the train station to the respective destination and vice-versa.

Lisbon’s topographic characteristics are no invitation to cycling, unlike many other European capitals. Electric bicycles are, therefore, an important asset in this BSS. In fact, although about a half of the total 550 bikes currently available are conventional, a mere 25% of the average 4500 daily bike trips are performed by the latter.

The preference for e-bikes is relevant when aiming to competently rebalance the system and, for that reason, such distinction has been implemented into the developed model. Also, it is immediately noticeable that there are periods of the day at which the system finds itself more burdened. Morning demand, starting around 07:30 and peaking around 09:00, has the most intense rise and fall. Repositioning the system during such peak-hours is both ineffective and expensive, stressing the importance of overnight rebalancing for smooth system operation.

Repositioning vehicles are electric, by contractual demand. Since different trails are attached to them, to enhance vehicle capacity, vehicle distinction has also been included in the model. No previous studies considering electric vehicle autonomy in BRP solutions have been found.

Since the distance matrix that has been provided does not violate the triangular inequality, nodes with no demand are not required to be visited, hoping to lower the total route distance. Nonetheless, a feature to comprehend the frequent inspection visits is included.

Finally, to alleviate vehicle capacity constraints, preemption was allowed at one vertex – the buffer vertex. Allowing preemption implies that vehicles can temporarily drop bikes at intermediate locations for future pickup, a feature that the GIRA optimizer, Siemens, believes to benefit the model.

2. Related Literature Review
Bektas, in his overview of formulations and solution procedures for the multiple salesmen TSP (a particular case of the VRP) [2] states that most exact solution methods are branch-and-bound derivations but refers how heuristic techniques are the most common, with special relevance to Neural-Network-based procedures. Still, heuristics do not provide an optimality gap, and therefore cannot assure a found solution is optimal. Besides, if no solution is found, it is unknown whether the cause is the failure of the
heuristics applied on the given problem or the infeasibility of the problem itself. A common strategy is the use of heuristics inside an exact method like Branch-and-Bound, whose bounds are commonly tightened using specific knowledge of the problem at hands. Such is the approach followed throughout the present study. Few works that consider multiple types of bikes have been found. Li et al. [3] formulated a MIP problem, solved by a genetic algorithm and a greedy heuristic, that takes different bikes into account, allowing the possibility of some of these types to take the place of others in demand. This work also provides loading and unloading instructions at each station. On the other hand, Zhang et al. [4], in 2018, claimed to be the first to formulate a BRP that considers the need to collect bikes in disrepair and its vehicle capacity consequences, which is solved by a discrete particle swarm optimization algorithm that incorporates a reduced variable neighbourhood search. The possibility to avoid visiting every node has also not been extensively studied in the literature. Ting and Liao [5] presented a variation of the classical routing problems called the selective pickup-and-delivery problem, that relaxes the constraints that imposes all pickup nodes to be visited. The study proves this variation of the problem to be NP-hard and proposes a memetic approach, based on genetic and local search algorithms. In 2016, Ho and Szeto [6] improved Ting and Liao’s solutions using a hybrid heuristic consisting of a greedy randomized adaptive search procedure, combined with a path relinking heuristic. In 2017, Schuijbroek et al. [7] have combined the estimation of minimum and maximum service level requirements for each station with a routing problem. If a station’s inventory is found to be self-sufficient for the station’s predicted needs, the corresponding node may not be visited. This formulation of Schuijbroek et al. was inspired by the one developed by Raviv et al. [1] in 2013, which allows preemption. Raviv et al. developed an arc indexed formulation that limits the total number of visits a station is allowed to have, in order to lower the computational time necessary to solve the problem to optimality, and a time-indexed formulation that does not require such limitations and is addressed further on. Both are solved by heuristic methods. Chemla et al.’s paper [8], also from 2013, allows multiple visits and preemption, as well, and relies on theoretically understanding this variation of the problem and possible solving methods. A Branch-and-Cut approach for the case where the buffer vertexes confirmed that initially relaxing the exponentially-many integrality

Figure 1: Net flow of bicycles in each station for the period of January and February 2019.
constraints provides good lower bounds for the problem. On the other hand, a Tabu search algorithm was successfully used to determine upper bounds. Table 1 summarizes the main characteristics of the referred works and compares them to the present one.

Table 1: Summary of the mentioned previous works.

<table>
<thead>
<tr>
<th>Name</th>
<th>Objective:</th>
<th>Multiple Vehicles</th>
<th>Multiple commodities</th>
<th>Visit every node</th>
<th>Preemption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chehda et al., 2013 [52]</td>
<td>Minimize</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Raviv et al., 2013 [11]</td>
<td>Total route length</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Ting and Liao, 2013 [60]</td>
<td>Total route length</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dell’Amico et al, 2014 [17]</td>
<td>Total route length</td>
<td>Yes, uniform fleet</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Li et al., 2016 [55]</td>
<td>Operational costs and penalties</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Ho and Szeto, 2016 [61]</td>
<td>Total route length</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Schrijbroek et al., 2017 [44]</td>
<td>Total route time</td>
<td>Yes, uniform fleet</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Zhang et al., 2018 [58]</td>
<td>Total route length</td>
<td>Yes, uniform fleet</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Present work</td>
<td>Total route length</td>
<td>Yes, non-uniform fleet</td>
<td>Yes</td>
<td>No</td>
<td>Yes, preemption allowed in 1 vertex</td>
</tr>
</tbody>
</table>

3. Construction of a MILP model

This model is a commodity flow arc-based formulation with an exponential number of subtour elimination constraints. It was based on Dell’ Amico et al.’s [9] work, which consisted on developing and testing four different MILP formulations to solve the nightly (stationary) BRP with a fleet of vehicles of uniform, limited capacity. Dell’ Amico et al. concluded that their third formulation, F3, bested the remaining in respect to the total number of instances solved to optimality, while still presenting the lowest average gap for the non-optimally solved ones and the lowest total and average computational times required.

The cause of this superior performance is the fact that F3 solves the problem for a single set of binary decision variables, $x_{ij}$, while all the remaining formulations present at least another one, and an integer one. Recalling the Branch-and-Bound method for solving integer constrained programs, it is noticeable how an additional variable, especially if left unbounded, would mean supplementary branching and decision variable combinations. Yet, after filtering through the results, it has been concluded that the implicit design of the formulation makes it ill-suited for the case study that is to be presented ahead. Albeit the fastest formulation per iteration, F3’s numerous calls for an extra separation procedure, motivated by the many possible combinations that a large network offers, would apparently outweigh this advantage. With a larger grid and more evident capacity restrictions, F3 is likely to run the separation procedure more often than it has when confronted with most of the paper’s instances and tendency shows that it would definitely lag behind the second formulation, F2.

The greater reliability of F2’s optimality gap estimation is also to be considered, because greater networks pose a bigger risk of not reaching an optimal solution under an established time stretch and optimality gap thresholds are a useful stop criterion for those situations. F3’s struggle to identify the infeasibility of an instance has also been decisive. Finally, an arc flow variable is very compatible with the additional constraints that are to be introduced into the model, which may explain the popularity of arc flow formulations in the literature (see, e.g., [7]). For these reasons, F2 was the chosen formulation to adapt and test the case study data.

The objective function is the total distance travelled ($c_{i,j}$ indicates the cost of travelling from node $i$ to $j$), and the goal is to minimize it while fulfilling the repositioning needs imposed by each station $j$’s demand for each bike type $b$, $q_{b,j}$, using as many capacitated vehicles (i.e., each vehicle $v$ is constrained by a maximum capacity, $Q_v$) routes departing from the depot as necessary. Binary decision variable $x_{i,j,v}$ indicates whether vehicle $v$ travels an arc $(i,j)$ of the network in the final solution or not. The feasibility of the solution with respect to the capacity constraints and rebalancing requirements guaranteed by making use of an additional decision variable, $f_{i,j,b,v}$, that represents the flow of a given type of bicycles, $b$, over arc $(i,j)$, travelled by $v$. Indexes $v$ and $b$ have been added to the original formulation to include the distinction between the different vehicles and conventional, electric and in-need-of-repair bikes, which are all treated differently in the field. Evidently, each index range results in a proportional increase of the number of variables.

The complete formulation is implemented in the solver Xpress-MP, which uses the Mosel programming language, and solved via Branch-and-Cut.

\[ Z = \sum_{i \notin V} \sum_{j \notin V} c_{i,j} x_{i,j,v} \quad \text{(1)} \]

Constraint (1) minimizes objective function, $Z$, in this case coming in the form of the total distance travelled.

\[ \sum_{i \notin V} x_{i,j} \leq 1, \quad j \in V \setminus \{\text{depot, buffer}\} \quad \text{(2)} \]
\[ \sum_{i \in \mathcal{V}} x_{ji} \leq 1, \quad j \in \mathcal{V} \setminus \{\text{depot}, \text{buffer}\} \quad (3) \]

Constraints (2) and (3) have been adapted and now ensure that each node, apart from the depot and the buffer, is visited once at most, which allows for vertexes with no demand not to be visited, except for those who require an inspection visit.

\[ \sum_{i \in \mathcal{V} \setminus \{\text{depot}\}} x_{\text{depot,}j} \leq m \quad (4) \]

\[ \sum_{j \in \mathcal{V} \setminus \{\text{depot}\}} x_{\text{depot,}j} = \sum_{j \in \mathcal{V} \setminus \{\text{depot}\}} x_{j,\text{depot}} \quad (5) \]

Constraint (4) imposes that no more than \( m \) vehicles leave the depot, whereas constraint (5) imposes that each vehicle leaving the depot returns to the depot.

\[ x_{i,j,v} \in \{\text{depot, 1}\}, \quad i,j \in \mathcal{V}, \quad v \in \mathcal{M} \quad (6) \]

\[ f_{i,j,v,b} \in \mathbb{N}, \quad i,j \in \mathcal{V}, \quad b \in B, \quad v \in \mathcal{M} \quad (7) \]

Constraints (6) and (7) impose \( x_{i,j,v} \) to be binary and \( f_{i,j,v,b} \) to be a non-negative integer.

\[ \sum_{i,j \in \mathcal{V}} f_{i,j,b,v} - \sum_{j \in \mathcal{V}} f_{i,j,b,v} = q_{b,j} \quad (8) \]

Constraints (8) guarantee each node is rebalanced through the flow difference of the arcs entering and leaving it.

\[ \max\{\text{depot}, q_i - q_j\} x_{ij} \leq f_{i,j,b,v} \quad (9) \]

\( f_{i,j,b,v} \leq \min\{Q_v, Q_v + q_i, Q_v - q_j\} x_{ij}, \quad (i,j) \in A, \quad i,j \neq \text{buffer}, \quad b \in B, \quad v \in \mathcal{M} \)

The bounds on flow variables are tightened thanks to constraints (9), which impose a minimum and maximum flow value on each node, based on the demands of \( i \) and \( j \), as well as the vehicle’s maximum capacity. The flow on each arc must be such that, if the arrival node is lacking bikes, enough bikes travel the arc. On the contrary, if the arrival station has an excess of bikes, the vehicle must have enough free space to collect them.

\[ \sum_{b \in B} f_{i,j,b,v} \leq Q_v, \quad (i,j) \in A, \quad v \in \mathcal{M} \quad (10) \]

Constraints (10) also contribute to bound tightening by ensuring that the total number of bikes in the vehicle is never greater than its capacity. In order to fit the case study parameters into the formulation, the latter had to be brought to a broader, generalized form. In this section, the adaptations performed are carefully detailed. All these adaptations consist, one way or another, in generalizations that introduce further combinatorial complexity. The branch-and-bound tree is expected to grow wider and deeper, and total solving time is therefore likely to increase. After presenting each adaptation, its impact on computational time requirements is tested and quantified. Based on the results, the adaptation may or may not be deemed a valuable investment for the final formulation used on the case study instances.

\[ \sum_{i \in \mathcal{V}} x_{i,j,v} = \sum_{k \in \mathcal{K}} x_{j,k,v}, \quad j \in \mathcal{V}, \quad v \in \mathcal{M} \quad (11) \]

Constraints (11) impose that the total number of arcs leaving a node be the same that entered. In other words, these constraints prevent the generation of arcs leaving from an unvisited node and impose that an arc entering a node must be followed by another one leaving it, both performed by the same vehicle.

\[ \sum_{i \in \mathcal{V}} x_{i,j,v} \times \left( \frac{c_{i,j}}{vel} + \text{stop \_time} \right) \leq \text{shift}, \quad v \in \mathcal{M} \quad (12) \]

Since each shift lasts for eight hours, constraints (12) are introduced to guarantee that the time necessary to travel the route and serve all stations in it does not exceed this time stretch by limiting the travelling time (ratio of arc distance and average vehicle speed, \( vel, \) of 30 km/h) and the average stoppage time at each station, \( \text{stop \_time}, \) set at 15 minutes.

\[ \sum_{i \in \mathcal{V}} x_{i,j,v} \times G_{i,j} \leq AUT(v), \quad v \in \mathcal{M} \quad (13) \]

The autonomy of the electric vehicles is given by the array \( AUT(v) \) and considered in constraints (13).

\[ \sum_{j \in \mathcal{V}} f_{\text{buffer},j,b,v} - \sum_{i \in \mathcal{V}} f_{\text{buffer},b,m} \leq q_{\text{buffer}, b}, \quad b \in B \quad (14) \]

\[ \max\{\text{depot,} - q_{b,j}\} x_{\text{buffer},j,v} \leq f_{\text{buffer},i,j,v} \leq \min\{Q, Q + q_{b,j}\} x_{\text{buffer},i,j,v}, \quad j \in \mathcal{V} \setminus \{\text{buffer}\}, \quad b \in B, \quad v \in \mathcal{M} \quad (15) \]

\[ f_{\text{buffer},b,v} \leq \min\{Q, Q + q_{b,j}\} x_{\text{buffer},i,j,v}, \quad i \in \mathcal{V} \setminus \{\text{buffer}\}, \quad b \in B, \quad v \in \mathcal{M} \quad (16) \]

Constraints (14) state that the demand of the buffer vertex is necessarily fulfilled. Also, because the buffer’s demand can be fulfilled in more than one trip, a particular form of constraints (9) is devised to fit the buffer vertex, resulting in constraints (15) and (16).

\[ \sum_{i \in \mathcal{V}} f_{\text{buffer},i,v} \leq \text{INV}_{\text{buffer}}, \quad j \in \mathcal{V} \quad (17) \]

\[ \sum_{i \in \mathcal{V}} f_{\text{buffer},i,v} \leq \text{CAP}_{\text{buffer}} - \text{INV}_{\text{buffer}}, \quad i \in \mathcal{V} \quad (18) \]
Constraints (17) and (18), ensure, together with the arrays of station inventory, INV, and station capacity, SCAP, that bicycles are not shipped from the buffer before they are brought to it, or delivered before space is available.

\[
\sum_{i \in V} \sum_{j \in V} x_{i,j,v} \leq |S| - 1, \quad (19)
\]

\[S \subseteq V \setminus \{\text{depot}\}, \quad S \neq \emptyset, \quad v \in M\]

Constraints (19) are the classical DFJ subtour elimination constraints introduced by Dantzig et al. [10]. The number of these grows exponentially with problem size, so a separation procedure is necessary to decide the set S these should be applied to. To separate these constraints, this work has followed the purely-integer scheme suggested by Pferschy and Staněk in 2017 [11]. Noticing how modern ILP solvers trivially find the violated constraints (in this case, using the Mosel language), fractional solutions are not required in this approach, thus avoiding the search for an efficient, elaborate algorithm.

Empirical knowledge obtained by isolating each new feature of the model and comparing it to the original formulation says that running times generally grow accordingly to the number of variables and constraints, which is intuitive.

On the other hand, it has been verified that setting too permissive constraints, as a virtually unlimited vehicle capacity, may stop efficient node fathoming at the BB and result in an increase of the computational time required until an optimal solution is found.

4. Acceleration procedures

Reportedly, Siemens intelligence solves a rebalancing problem in under 30 minutes, while also accounting for demand prediction. Although the present solver still cannot aspire to compete with such software, the 30-minute mark was set as an appropriate computational time limit.

Since the basic formulation does not usually deliver optimal results under such time stretch, a set of procedures was then introduced as a way of lowering the program’s running time.

**General procedures**

It has been noticed how the model firstly presents a number of solutions that contain subtours before eventually reaching optimality.

A strategy towards lowering total running time might pass by reducing the number of possible subtours, which hopefully lowers the number of iterations necessary until a final feasible solution is found. Such ideas motivated the introduction of the sets of constraints that follow, already suggested by Dell’Amico et al. [9].

Constraints (20) eliminate subtours that contain a single vertex:

\[
\sum_{v \in M} x_{i,i,v} = 0, \quad i \in V \quad (20)
\]

Constraints (21) have the same effect on subtours within two vertexes:

\[
\sum_{v \in M} x_{i,j,v} + x_{j,i,v} \leq 1, \quad i,j \in V \setminus \{\text{depot, buffer}\} \quad (21)
\]

Pferschy and Staněk [11] also experimented with the addition of SECs to sets of subtours of size 3 and small length, but such approach turned out unfruitful. It is thus not followed in the present work.

In the presented formulation, subtours may arise from a vehicle with no arc leaving or entering the depot, which is unrealistic. Constraints (22) are added to prevent this situation.

\[
x_{i,j,v} \leq \sum_{k \in V \setminus \{\text{depot}\}} x_{\text{depot},k,v}, \quad i,j \in V, \quad v \in M \quad (22)
\]

Constraints (21), (20) and (22) represent no loss of generality, but its introduction in the model corresponds to a significant increase in the total number of constraints. Their effectiveness on the results is expressed in Figure 2, presented after the next section.

It has been mentioned that Guére et al.’s subtour elimination procedure [12] eliminates a single subtour per iteration. Trying to take advantage of the subtours found, an option where the procedure has been modified to eliminate all subtours of the found solution is also tested.

**Particularizations**

The introduction of subtour elimination constraints is limited by the exponentially large number of sets that exist. However, if one previously knows of an existing set, a SEC regarding it may be introduced in the program. Observing the GIRA map, it is noticeable how the network is divided into four different areas. If all areas have at least one vertex in need of rebalancing, then (23) may be applied to each area A. Let set \(A\) be defined as \(A = V \setminus \{A \cup \{\text{depot}\}\}\), for every A. Then, (23) comes as:

\[
\sum_{j \in A, k \in A, v \in M} x_{j,k,v} \geq 1 \quad (23)
\]

These constraints impose that at least an arc outside \(A\) enters this set of arcs. Paired with the fact that each regular node can only withstand one arrival arc, this forces at least one arc to leave \(A\), thus preventing a subtour containing all vertexes inside it. Even though they are equivalent to the SECs used until now, these would not be as effective in this case, since the introduced generalizations allow some nodes inside each \(S\) not to be visited by any arc.

Since lower bounds represent the minimum distance required to be travelled, these constraints can also
contribute to bound tightening, as imposing connectivity between each area forces a “minimum length” for the solution route. The influence of constraints (23) on bound tightening is then likely to grow if areas distanced further apart are chosen.

Determining other sets of nodes upon which constraints (23) could be introduced is not a complex task. A straightforward analysis of the statistical appearances of each set of arcs associated with subtours would provide useful information for accelerating lower bounds and preventing subtour formation through SEC introduction in key sets of arcs. Pferschy and Staněk’s [11] “cluster” search for quality subtours identified isolated sets of nodes and established them as a sub-network to be ran by the model in search for its subtours.

On the other hand, it would be a definitive particularization of the model, which is not the aim of this work. Such endeavor was therefore not pursued and constraints (23) have been limited to the areas originally designated by the BSS designers and its different possible conjunctions. Still, finding other key subtours would certainly contribute significantly to raise lower bounds and improve the model’s performance. It is an advised practice for those who seek the best practical results in this or other networks.

It was immediately evident that the model finds most of the solutions at the beginning of its execution, before locally stagnating at a best solution. On the other hand, because lower bounds are computed slowly for big network instances, reduction of the optimality gap proceeds at a rate too low for optimally solving the problem under 30 minutes. Therefore, after the first sets of best solutions are found, small reductions in the objective function become very time consuming. Besides, even if an optimal solution is found under acceptable time, there is no guarantee that a subtour is not present, restarting the process with a new iteration.

A compromise between total running time and best objective function value must be found.

Establishing a maximum running time might make sense for formulations that do not pose the threat of finding solutions that include subtours, but that is not the present case and this approach is hence disregarded. Solving until a solution is within an accepted optimality gap tolerance presents itself a more suitable compromise because there is an upper limit on the error involved.

Recalling the definition of optimality gap:

\[
\text{Optimality gap} = \frac{\text{best solution found} - \text{upper bound}}{\text{best solution found}}
\]  

(24)

Introducing a tolerance on the optimality gap not only conducts to a faster conclusion of each iteration but also serves to update the lower bound between iterations, whose low evolution rate conditions the speed at which the optimality gap develops.

It is known that, after a subtour is found in a first solution, a cut is introduced specifically on that set of nodes. As a result, the iterations that follow cannot produce a better optimal solution than the one that would have been found if the first iteration had been solved to optimality. Consequently, if the first solution is within an established tolerance for the first iteration, it is also at least as good as any solution that would satisfy the tolerance gap for the iterations that follow.

It is then safe to introduce constraint (25) at the end of each iteration:

\[
Z \geq \text{solution found at the last iteration} \times (1 - \text{tolerance})
\]  

(25)

By tightening bounds, these constraints save computational effort that would have been spent raising the lower bound.

Since, as section 5.3 of the dissertation showed, computational time tends to increase with the number of vehicles used (in the context of the new formulation), the effect of limiting the considered vehicles to the ones available has also been tested.

5. Results and discussion

In this chapter, the results of the GIRA case study are examined after selecting some appropriate acceleration procedures. All experiments were carried out on a computer equipped with Windows 10, an Intel(R) Core(TM) i7-3630 QM CPU @ 2.40 GHz, and a 8.00GB of RAM.

In order to test the model on the GIRA bike sharing system, the “cleanest” of all static rebalancing data instances were selected from amongst the ones regarding the months of January and February of 2019. These instances have a negligible number of unpredictable actions, a high percentage of rebalancing actions and are coherent with the net flow of departures and arrivals shown in Figure 1. Also, aiming towards a fair comparison to the performed routes, actions performed by teams ending the respective shifts have been left out of the analysis, thus avoiding an additional station-depot dislocation.

The calculation of inter-vertex distances has been conducted with the help of a cost matrix supplied by EMEL, the municipal entity in charge of the Lisbon BSS, containing the estimated shortest distances between stations, for a cyclist. Knowing the respective GPS coordinates, the distance from the depot to each station has been calculated using Vincenty’s method [13], which is the most successful at approximating the effects of Earth’s curvature, assumed symmetric and multiplied by Schuijbroek et al.’s [7] detour factor: 1.4.

Firstly, the results obtained after implementing the abovementioned procedures are displayed and analysed. The maximum computational time allowed was extended to 3600 s, in order to better understand the influence of each. Several individual
tests were conducted on each date’s data and the results are graphically represented in Figure 2. Tolerance was set at 5% for the applicable tests. This figure presents, for each procedure considered and specified on the horizontal axis, the respective test’s percentual difference to the optimal solution – both quantified within the left vertical axis – and the necessary computational time (up to 3600 s). A colour code is used to tell the rebalancing data instances referent to each different date apart.

After analysing the figure, several conclusions arise. It is evident that instances are not equally influenced. Nonetheless, procedures aiming at reducing the number of possible subtours have proved the most effective, with special relevance to (20), (21) and, above all, (23). This last procedure is dependent on the network at cause, meaning that calculations need only be performed once for instances on the same BSS, and is the most powerful one encountered, almost eliminating the difference to the optimal solution in all instances but one.

When all procedures but “Tolerance”, which defines the acceptable optimality gap and (25), are present, the best results are found: only the most intractable instance, 4\textsuperscript{th} of February, did not reach its optimal result. Also, 24\textsuperscript{th} of February failed to annul the optimality gap within one hour of computational time. It can be observed that both instances ran a single iteration during this time, indicating a slow lower bound evolution. In these cases, introducing an optimality gap tolerance is useful to reduce the necessary time for each iteration – notice how instances ran with “Tolerance” tend to present a higher number of iterations – and hopefully prevent the model from running indefinitely.

Tolerance, however, is not beneficial to all instances. The results express that this procedure is useful only in the cases described above. When models are computed fast, optimality gaps are already low, and introducing tolerance makes a relatively poor solution acceptable, immediately stopping a cost-effective search for a better one. Curiously, 11\textsuperscript{th} January takes the longest time to compute and the highest number of iterations when this procedure is involved, which indicates that this particular instance has several solutions with subtours within a 5% optimality gap. In cases like this, relaxing the exigence on problem solutions actually increases the total running time.

On the other hand, 5% of tolerance is not enough to terminate a solution search for the case of the 4\textsuperscript{th} of February. This is not a surprising result. In the literature, problems of such size frequently encounter such differences between the best result found and the optimal solution. Even though Bulhões et al.’s work [14] allows every station to be visited a limited amount of times and the present formulation allows a single station to be visited as many times as needed, the study’s differences to the optimal solution are used as a comparative example. These frequently (13 out of 60 possible cases) get above the 10% mark on randomly generated instances of size 30, with 2 to 3 vehicles available and 2 to 3 station visits allowed. Also, Dell’Amico et al.’s [9] real instances have been tested by this formulation and only 5 out the 19 instances of size 20 to 50 that have not been solved to optimality within one hour of CPU time presented better difference values.

Having determined the generally best set of procedures (all but “Tolerance”), Table 2 presents the final results obtained. The respective number of stations, repositioned bikes, zero-demand vertexes are detailed in the referred table, as well as the assigned buffer vertex and the capacities of the vehicles available in the early hours of the specified days. As a performance reference, the distances
travelled by the operating vehicles during the considered shifts have also been estimated and included in Table 2. Finally, the best solutions obtained by the model under the amount of time considered acceptable (1800 seconds), as well as its percentual difference to the best solution found for each instance and the computational time required, are presented.

Table 2: Case study instances selected, relevant data and final results.

<table>
<thead>
<tr>
<th>Selected instances</th>
<th>10/01</th>
<th>11/01</th>
<th>04/02</th>
<th>18/02</th>
<th>24/02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nr. of stations</td>
<td>31</td>
<td>22</td>
<td>34</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Nr. of repositioned bikes</td>
<td>102</td>
<td>80</td>
<td>41</td>
<td>56</td>
<td>90</td>
</tr>
<tr>
<td>Nr. of 0_demand stations</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Buffer vertex</td>
<td>417</td>
<td>462</td>
<td>484</td>
<td>446</td>
<td>225</td>
</tr>
<tr>
<td>Initial cost (km)</td>
<td>153.1</td>
<td>59.9</td>
<td>97.9</td>
<td>164.9</td>
<td>130.2</td>
</tr>
<tr>
<td>Optimal result (km)</td>
<td>51.3</td>
<td>41.8</td>
<td>40.3</td>
<td>40.2</td>
<td>68</td>
</tr>
<tr>
<td>Best result (km)</td>
<td>51.3</td>
<td>41.8</td>
<td>41</td>
<td>40.2</td>
<td>68</td>
</tr>
<tr>
<td>Difference (%)</td>
<td>0.00</td>
<td>0.00</td>
<td>1.66</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1800 s Time (s)</td>
<td>175.1</td>
<td>15.8</td>
<td>1800</td>
<td>262.1</td>
<td>1800</td>
</tr>
<tr>
<td>Iterations</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Nr. of vehicles</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Buffer visits</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Difference of best result to initial cost (%)</td>
<td>66.5</td>
<td>30.2</td>
<td>58.1</td>
<td>75.6</td>
<td>47.78</td>
</tr>
</tbody>
</table>

February 4\textsuperscript{th} is the only instance whose optimal solution has not been found under the acceptable time window, and its intractability is of obscure cause. While it is the one with more stops, it is also the one rebalancing the least amount of bikes and the one with more “inspection” stops, which do not interfere with capacity restrictions. Nonetheless, it has been shown that a single vehicle is capable of performing all the accounted actions in an overall shorter route, proving the claim of the model as a future decision-aiding tool. Besides, three of the five solutions found have benefitted from the introduction of a buffer vertex.

The solution of February 4\textsuperscript{th} is traced in Figure 3. Bear in mind that the drawings do not comprehend the actual distances considered by the program, which are affected by road infrastructures, but the Euclidean distances. This detail may result in some arc overlapping that is actually inexistent in the respective final solutions and vice-versa.

The number of stations visited appears not to fluctuate enough to pose a decisive factor on the model’s performance. An increase in total number of repositioned bikes per instance, however, apparently causes routes to become more intricate, sometimes even introducing arc overlaps. Such behaviour is not surprising, as capacity restrictions tend to be more relevant for problems with more repositioned bikes, pushing the final solution further and further away from the one that would solve the Travelling Salesman Problem (recall that the CVRP is a generalization of the TSP, both coinciding when vehicle capacity is great enough). Unsurprisingly, these solutions have made use of the respective buffer vertexes.

6. Conclusions, Limitations and Future

Building up from a previous formulation this work has proved that it is possible to improve nightly routes of rebalancing vehicles operating at the Lisbon bike sharing system under a practical amount of time. Having been generalized to include inspection stops and different types of bikes and vehicles, it comprehends technical aspects that bring the results a step closer to being useful in real decisions. The possibility of preemption at one node, named the “buffer vertex”, has also been introduced and, in some instances, reduced the effect of vehicle capacity restrictions in the total route length.

Some simple acceleration heuristics have been tested out. The ones that proved to be the most powerful were those that raised the lower bound by preventing key subtours.

Finally, the final model has run five case study instances. The solution routes found under the established time-stretch of half an hour posed a shorter length than that taken by the operator’s vehicles on the respective dates. In fact, significant route length savings are proposed, ranging between 30\% and 75\%. Nonetheless, one of the instances has not provably reached the respective best possible solution, while another one was stopped at a 1.66\% difference to it, indicating that further acceleration procedures should be pursued. If a merely practical model is sought, abdicating of model generality and...
looking in the given network for key subtours to eliminate would almost certainly go a long way in improving the overall performance.

Hopefully, the actual implementation of this model as a route-tracing tool will not only result in significant cost reduction but also in greater rebalancing efficiency, which may lead to more balanced networks and an increased perception of usefulness of bike sharing systems as a transport choice by the respective communities.

In order to truly become helpful in BSS decision-taking moments, the model must be improved to further comprehend crew, equipment and task limitations, live demand and traffic and, perhaps, station inventory flexibility. Approaches focusing on the users, such as the economic incentive of user-operator cooperation in the rebalancing actions or user dissatisfaction minimization models can also be studied and integrated within the present one.

7. References


