A 3-Step Math Heuristic for the Static Repositioning Problem in Bike-Sharing Systems

Iris A. Forma, Tal Raviv, Michal Tzur

Industrial Engineering Department

Tel Aviv University, Tel Aviv, 69978 ISRAEL

irisf@afeka.ac.il, talraviv@eng.tau.ac.il, tzur@eng.tau.ac.il

Revised August 2014

Abstract

Over the last few years, *bike-sharing* systems have emerged as a new mode of transportation in a large number of big cities worldwide. This new type of mobility mode is still developing, and many challenges associated with its operation are not well addressed yet. One such major challenge of bike-sharing systems is the need to respond to fluctuating demands for bicycles and for vacant lockers at each station, which directly influences the service level provided to its users. This is done using dedicated repositioning vehicles (light trucks) that are routed through the stations, loading and unloading bicycles to/from them. Performing this operation during the night when the demand in the system is negligible is referred to as the *static repositioning problem*. In this paper, we propose a 3-step mathematical programming based heuristic for the static repositioning problem. In the first step, stations are clustered according to geographic as well as inventory (of bicycles) considerations. In the second step the repositioning vehicles are routed through the clusters while tentative inventory decisions are made for each individual station. Finally, the original repositioning problem is solved with the restriction that traversal of the repositioning vehicles is allowed only between stations that belong to consecutive clusters according to the routes determined in the previous step, or between stations of the same cluster. In the first step the clusters are formed using a specialized saving heuristic. The last two steps are formulated as Mixed Integer Linear Programs and solved by a commercial solver. The method was tested on instances of up to 200 stations and three repositioning vehicles, and was shown to outperform a previous method suggested in the literature for the same problem.

1. Introduction

Bike-sharing systems offer a low cost and environmentally friendly mean of transportation for short travels. It can also be used as a complementary mode to other public transit such as buses, trams or subways. In order to provide high quality service, bicycles and vacant lockers should be available to the users of these systems when and where they are demanded. This can be achieved by *repositioning* the bicycles in the system according to forecasted demand rates for bicycles and lockers at the stations, using a dedicated fleet of light trucks. These trucks, referred to as *repositioning vehicles*, travel among the stations, where bicycles are loaded or unloaded subject to bicycle availability and capacity limitations at the repositioning vehicles and at the stations. When the repositioning is carried out during the day when the system is active, it is referred to as *dynamic repositioning* and when it is carried out during the night when the demand for bicycles is negligible it is referred to as *static repositioning*.

In this paper we consider the static repositioning problem (SBRP), as defined in Raviv et al. (2013). In particular, the goal is to minimize a weighted sum of the expected number of unserved users (due to shortage of both bicycles and lockers) during the next working day, and the total traveling distance. The planning horizon of the repositioning operation is limited in time, typically five to six hours, to reflect the fact that it should be completed before dawn. Loading and unloading times of bicycles at the stations are taken into account; in practice, the loading and unloading operation amounts to most of the working time of the repositioning crew. Raviv et al. (2013) propose several mixed integer linear program formulations for the SBRP. Their numerical experiment shows that the formulation referred to as the *arc-indexed* (AI) is very effective for solving real life instances with up to 100 stations and two repositioning vehicles. In this paper we propose a 3-step math heuristic that is based on a new clustering concept and on the AI formulation. We show that our heuristic produces solutions with small optimality gaps for large instances.

In the first step of our heuristic, stations are clustered according to geographic as well as inventory (of bicycles) considerations. In the second step the repositioning vehicles are routed through the clusters while tentative inventory decisions are made for each individual station. Finally, in the third step the exact routing of the repositioning vehicles is determined, along with the (possibly revised) number of bicycles to be loaded and unloaded at each station. The routing decisions are subject to restrictions imposed by the solution of the previous step.

The goal of the first step is to create clusters of stations that are geographically close together and that are likely to be visited successively within the same route. Moreover, they are formed such that their requirements for loading and unloading bicycles are complementary. The purpose of the second step is to determine the set of clusters that are visited by each repositioning vehicle and the sequence in which they are visited. These routing decisions are made together with some preliminary inventory decisions that take into consideration the time needed for the loading and unloading operations. These inventory decisions are subject to refinement in the third step, in which the original repositioning problem is solved with restrictions on the traversal of the repositioning vehicles. Namely, traversal is allowed only between stations that belong to consecutive clusters according to the routes determined in the previous step, or between stations of the same cluster. Since in this step the assignment of stations to repositioning vehicles is already fixed, the repositioning problem is solved for each vehicle separately. The last two steps are formulated as Mixed Integer Linear Programs that are similar to the AI formulation introduced in Raviv et al. (2013), and solved using a commercial solver.

The applicability of the proposed method is numerically demonstrated by solving a variety of instances that are based on real data. The results indicate that near optimal solutions of real life instances with up to 200 stations and three repositioning vehicles can be obtained. These results significantly outperform the results of applying the AI formulation on the original problem, in the same computational environment.

The contribution of this paper is in presenting a successful math-heuristic solution method to the SBRP, which is suitable for large real life instances of the problem. Although our heuristic is based on decomposing the problem into smaller sub-problems that are solved separately, our method is quite different from typical clustering approaches known in the literature for vehicle routing problems, see Section 2. We believe that our new decomposition approach may be useful for other rich vehicle routing problems.

The rest of the paper is organized as follows: in Section 2, we review the literature, describing related work from several application areas. In Section 3, we formally state the SBRP and present some notation. In Section 4, we present our algorithm in detail, where a separate sub-section is dedicated to each of its three steps. In Section 5, we describe our numerical experiments, the results, and their analysis. Finally, in Section 6, we conclude and discuss possible extensions and directions for further research.

2. Literature Review

Bike sharing systems rapidly gained popularity during the last decade, first in West Europe and East Asia and more recently in North America. DeMaio and Meddin (2014) maintain a geographical database that describes the location and the characteristic of all public bike-sharing systems around the world. Larsen (2013) reports that as of April 2013 more than 500 cities host advanced bike-sharing systems with a combined fleet of more than half a million bicycles. Many cities are in the process of planning and deploying new systems or extending existing ones.

Design and operation of bike sharing systems raised a set of new interesting problems to be addressed. First, the extent and the nature of the demand of a new bike-sharing system should be forecasted. Then, some strategic problems need to be addressed. These include the coverage, capacity and density of the system. Next, the actual location of the stations and their capacity should be designed. At a more tactical level, pricing and reservation policies should be determined by the operator. Finally, the day-to-day operation of the system requires rebalancing the system continuously in order to meet the demand for bicycles and vacant lockers at the location and times desired by the users. While practitioners in the bike-sharing industry are already dealing with these challenging problems, with various levels of success, the scientific literature on the above topics is still relatively sparse. Below we describe the state of the art in the scientific literature.

Several studies try to estimate and forecast the demand in existing bike-sharing systems using data mining and classical empirical methodologies. For some examples see Kaltenbrunner et al. (2010), Vogel et al. (2011), Côme et al. (2013) and Rudloff and Lackner (2014). Lin and Yang (2011) address the combined problem of selecting optimal locations for bike-sharing stations and designing the bicycle paths network in a city so as to minimize the total travel costs of all users in the city. Romero et al. (2012) propose a bi-level optimization model for the optimal location of bike-sharing stations where the goal is to maximize the number of travelers that use the system. Chow and Sayarshad (2014) proposed a new framework to design transportation networks in the presence of coexisting networks. The framework was applied to a bike-sharing system in the presence of a coexisting transit system and was shown to result in a capacitated multicommodity flow problem. Shu et al. (2005) study the design and management of a stochastic vehicle-sharing system. Shu et al. (2013) develop a network flow model for a system in which the flow is determined by the random demand of the travelers. They examine the effect of redistribution of bicycles in the network and the capacity of the stations on the flow in the system. Nair et al. (2013) studied the Vélib' system in Paris from several aspects, including system characteristics, utilization patterns, the connection between public transit and bicycle-sharing systems, and flow imbalances between stations. Rudloff and Lackner (2014) build a statistical model for the demand for bikes and lockers by studying the influence of weather and full/empty neighboring stations on demand.

In order to cope with the inherent asymmetry and stochasticity of trips in the system, two general methods were proposed for diverting the demand towards more balanced patterns. Waserhole and Jost (2014) present a dynamic pricing mechanism that forces a balanced demand and hence save the need for repositioning altogether. Such an approach clearly comes at the cost of reducing the level of service provided to the users of the system but nevertheless seems the only viable alternative to balancing a car sharing system where repositioning is too expensive. Fricker and Gast (2014) use mean-field approximation to analyze the effect of a simple incentive mechanism on the service provided by the system and the optimal fleet size. Kaspi et al. (2014a, 2014b) explore operational policies in which the users of the system reserve lockers at the destination upon the beginning of their journey. Trips can be denied or the destination can be slightly diverted if no vacant lockers are expected to be available at the true destination.

Many studies on bike-sharing systems deal with some variants of the static repositioning problem, which is also the topic of this work. Some authors refer to the same class of problems as *rebalancing problems* but since, as mentioned above, a balance in the system can also be achieved by pricing incentives and reservations, we prefer the term repositioning. Note that the static repositioning problem is not uniquely defined since various studies make different modeling assumptions. Most of the authors viewed the problem as an extension of the single commodity many-to-many pickup and delivery problem introduced by Hernández-Pérez and Salazar-González (2004). In particular, most studies consider the total distance traveled by the repositioning vehicles, as the objective function. Berbeglia et al. (2007) provide a survey and classification of pickup and delivery problems; see also the discussion below regarding the differences between the problem considered here and pickup and delivery problems.

Chemla et al. (2013a) propose a math-heuristic for the single vehicle version of the problem, assuming each station can be visited more than once. Their algorithm is based on a solution of a relaxation of the problem that is solved by a branch-and-cut procedure. The objective function is to find a route that minimizes the total traveling distance. Erdoğan et al. (2013) extend the pickup and delivery model to allow the final inventory at each node to be within a prescribed interval instead of at a unique level. This model reflects the idea that there are some degrees of flexibility in the desired number of bicycles at each station at the end of the static repositioning operation. They present a branch-and-cut algorithm and a Bender decomposition that allows solving instances with up to 50 stations. Schuijbroek et al. (2013) study the static repositioning problem with a service level constraint. Similarly to Erdoğan et al. (2013) they assume that the service level constraint is met when the desired inventory level at the end of the repositioning operation is within a given interval. They present a queuing model that allows calculating the required interval given the desired service level. Next, the paper introduces a cluster-first route-second algorithm where the clustering problem is solved by a heuristic that is based on the maximum star approximation of the TSP. The routing problem of each repositioning vehicle is then formulated as an integer program and solved by a commercial solver. Rainer-Harbach et al. (2013) and Raidl et al. (2013) present variable neighborhood search heuristics for a variant of the static repositioning problem that take the loading and unloading times into account.

Contrary to the approach of all the above studies, we believe that the desired number (or range) of bicycles at the stations after the static repositioning operation is completed should constitute a soft constraint, expressed as a penalty in the objective function. From the operator's point of view, the target inventory levels of bicycles are only idealistic goals to aspire to.

Raviv et al. (2013), based on Forma et al. (2010) further extend the many-to-many pickup and delivery problem introduced by Hernández-Pérez and Salazar-González (2004) by including the following characteristics that are not considered by other studies: (a) the objective function incorporates a service level component directly, that is, the goal is to minimize a weighted sum of the expected number of un-served users (due to shortage of both bicycles and lockers) during the next working day, and the total traveling distance. The former part is calculated based on Raviv and Kolka (2013); (b) the planning horizon of the repositioning operation is limited in time; (c) the loading and unloading times of bicycles at the stations are taken into account. Near optimal solutions were reported for instances created based on the demand of actual systems with up to 104 stations and two vehicles, but as shown in this study, this marks the limits of the model size that can be solved with a reasonable optimality gap subject to a reasonable computing time constraint.

The solution method presented in this paper is built to solve the problem under the assumptions presented in Raviv et al. (2013) even though it can be easily adapted to the other optimization models presented in the literature. In the current study we significantly increase the size of the problem instances that can be solved to near optimality.

Dynamic repositioning is much more intricate to manage because it incorporates a scheduling component derived from the users' activity during the operation, together with a routing component. Contardo et al. (2012) assume dynamic deterministic demand and formulate an optimization model to route a single vehicle that moves bicycles between stations so as to minimize the number of shortage events. They propose a decomposition scheme that allows obtaining a solution with a reasonable optimality gap for instances with up 100 stations and 60 discretized periods, in ten minutes. Pessach et al. (2014) use a formulation that is similar to the time-indexed formulation of Raviv et al. (2013) within a rolling horizon framework in order to heuristically solve the actual stochastic and dynamic repositioning problem. Their method was shown to outperform a variety of heuristic method based on dispatching rules, including some of those used by practitioners. Chemla et al. (2013b) and Pfrommer et al. (2014) propose methods for dynamic rebalancing of the system that utilize both pricing incentives and repositioning. Kloimüllner et al. (2014) suggest solving the dynamic balancing problem by greedy and PILOT construction heuristics, as well as variable neighborhood search and GRASP improvement heuristics.

As mentioned above, Schuijbroek et al. (2013) present a heuristic algorithm to the repositioning (rebalancing) problem that can be classified as a cluster-first route-second approach, which is a quite prevalent approach in solving various multi-vehicle routing problems. It is based on the idea of building clusters of nodes/customers first, and then solving a series of independent single-vehicle TSP (or other routing) problems, one for each cluster. Early algorithms in this class include, for example, the sweep heuristic by Gillett and Miller (1974) and the well-known heuristic of Fisher and Jaikumar (1981), which uses the generalized assignment problem to create the clusters. A review of classical heuristics approaches can be found in Laporte and Semet (2002).

While our heuristic also builds clusters first and then constructs routes, it is quite different from the above literature. The main difference is that a vehicle/route according to our approach travels through a sequence of clusters, whereas a vehicle/route according to the cluster-first route-second approach travels through a single cluster. The purpose of creating clusters in our heuristic is to reduce the network size, so that routing of multiple vehicles can be performed on the reduced network; in the cluster-first route-second approach, the clusters are created in order to decompose the network so that a single vehicle serves each of them. We are not aware of other heuristic approaches that similarly to ours, reduce the network size in such a manner.

Battarra et al. (2014) studied the *clustered vehicle routing problem* (CluVRP), where the customers are clustered into pre-defined clusters, so that a vehicle visiting one customer in the cluster must visit all the remaining customers in the cluster before leaving it. Although this is similar to our heuristic in the fact that a vehicle visits several clusters, there are two main differences between the routing decisions of the CluVRP problem and ours: (i) in the CluVRP problem, the clusters of customers are given as part of the problem's input, while in our problem they are created as part of the solution procedure in order to reduce the complexity of the problem; (ii) in the CluVRP problem, all nodes must be visited, while in our problem it is not a requirement; Other differences between the problems clearly exist, as the operation performed in our repositioning problem is quite more involved than supplying a given quantity to the customers, as in the CluVRP. Baldacci et al. et al. (2010) introduce the *generalized vehicle routing problem* (GVRP), in which the clusters are also pre-defined. Each customer has a non-negative demand and a given number of vehicles must visit exactly one customer per cluster, such that the total cost is minimized, and the routes obey capacity and time constraints. It is demonstrated that the GVRP provides a useful modeling framework for a wide variety of routing applications.

In conclusion, our 3-step algorithm advances the literature on repositioning problems and forms a practical approach for medium-large bike-sharing systems. Moreover, it presents a new type of decomposition method for a variety of vehicle routing and inventory routing problems.

3. Problem Definition

The SBRP addressed in this paper was first introduced by Raviv et al. (2013). For the sake of completeness we restate this problem here. Their arc-indexed formulation, which is used in this paper, is given in Appendix A.

The input of the SBRP consists of a set of stations, N, and a depot where all the repositioning vehicles are initially located. The set of the stations and the depot together are referred to as N_0 where the depot is indexed by 0. Each station $i \in N$ is characterized by:

- Capacity c_i, which is the maximum possible number of bicycles that it may store.
- Initial bicycle inventory, s_i^0.
- Penalty function, $f_i(s)$, which represents the expected shortage of bicycles and lockers incurred by users in the station during the next day as a function of the inventory at the station after the repositioning is carried out. Let $f_i(s) = \infty$ for values out of the domain $\{0, \dots, c_i\}$. Raviv and Kolka (2013) show how to calculate this function based on a demand forecast for the station and prove its convexity. However any penalty function can be used as long as it is convex and separable between stations.

The depot may have its own capacity and initial inventory but these are typically assumed to be non-binding. The depot faces no demand and hence incurs no penalty.

In addition, the input consists of:

- A set of vehicles V. Each vehicle has capacity k_v.
- The total time allocated for static repositioning, T. Typically, 5-6 hours during the night.
- The time L (resp., U) needed for the loading (resp., unloading) operation of each bicycle to (resp., from) the vehicle, typically 1-2 minutes.
- Vehicle driving time matrix, with elements t_{ij} between each pair of stations i and j, including the depot.
- A weight, α, of the travel cost per time unit in the objective function. Without loss of generality, the weight of the expected number of unserved users in the objective function is set to one.

In particular, the goal is to minimize a weighted sum of the expected number of unserved users (due to shortage of both bicycles and lockers) during the next working day, and the total traveling distance. A solution of the problem constitute a a route for each vehicle that starts and ends at the depot, along with decisions on the number of bicycles to load/unload at each station along the route, while satisfying the time constraint. The objective is to minimize a weighted sum of the penalty function over all stations and the total travel cost of all the vehicles.

Let s_i^* be the inventory level at station i that minimizes $f_i(s)$. Clearly, $\sum_i f_i(s_i^*)$ is a lower bound on the value of the objective function of the above problem. We refer to s_i^* as the ideal inventory level of station i and to a solution that achieves the ideal inventory level at all stations as an *ideal solution*. Due to the time limitation, T, it is generally infeasible to achieve an ideal solution. Moreover, even if an ideal solution is feasible, it is generally sub-optimal due to the travel cost component in the objective function. Therefore, in general, the optimal solution differs from the ideal solution.

4. A 3-Step Heuristic Approach

Our proposed algorithm is composed of three steps: first, stations are clustered according to geographic and inventory considerations. Then, the vehicles are routed through the clusters while inventory decisions are made for each individual station separately. Finally, the original static repositioning problem is solved for all stations, but traversal of the vehicles is allowed only between stations of the same cluster or stations that belong to two consecutive clusters, according to the decisions made in the previous step. These steps are described next.

In this section we use the following definitions: Given a cluster of stations I, let its initial inventory, denoted by S_I^0, be the sum of its stations' initial inventories (i.e., before repositioning), and its capacity, denoted by C_I, be the sum of its stations' capacities. That is:

$$S_I^0 = \sum_{i \in I} s_i^0 \tag{1}$$
$$C_I = \sum_{i \in I} c_i \tag{2}$$

4.1 Step 1: A saving heuristic for building the clusters

The purpose of the first step is to create clusters of stations in order to solve a repositioning problem on the reduced network, whose nodes would refer to clusters. Each cluster may include several stations, although a cluster of a single station is also possible.

To create clusters such that the repositioning solution on the reduced network will serve the objective of the original problem, two issues need to be considered: the geographic location and the initial inventory. Since in the next steps of the algorithm, the stations are visited cluster by cluster, we would like the stations in a given cluster to be geographically close to each other. Therefore, we limit the diameter of each cluster, i.e., the maximal distance (in terms of travel time) between stations that belong to the cluster. The maximal diameter allowed is denoted by D. Since as D decreases, we expect the number of clusters to increase, the value of D is determined to be as low as possible such that the repositioning problem on the reduced network can still be solved in Step 2 in a reasonable amount of time.

The second issue to consider is the intial inventory of the clusters, which determine their costs. We define the total cost of a cluster, $F_I(S)$, as the expected number of shortages that will be incurred in its stations when an inventory of S bicycles is arranged in the stations of cluster I in the best possible way, i.e., so as to minimize the sum of the expected number of shortages in all of its stations. This definition is motivated by the fact that the stations of a cluster are visited by the same vehicle, which can easily transfer bicycles between them. For example, if we include in a certain cluster only stations in which the inventory level is lower than their ideal levels, tranferring bicycles between these stations can not improve the service provided by these stations significantly. On the other hand, if the inventory of some stations is higher than their ideal level, and of others is lower, then tranferring bicycles between stations within this cluster can be more beneficial. In the former (latter) case the cluster's cost would be high (low).

$F_I(S)$ is formally defined by the following mathematical program:

$$F_I(S) = \min_{s_i : i \in I} \sum_{i \in I} f_i(s_i) \tag{3}$$

$s.t$:

$$\sum_{i \in I} s_i = S \tag{4}$$

In the above optimization problem, the decision variables are s_i ($\forall i \in I$). The variable s_i denotes the inventory level at station i after distributing S bicycles among the stations of the cluster. Note that in the solution obtained, $0 \le s_i \le c_i \; \forall i$ since by definition, $f_i(s_i) = \infty$ for $s_i > c_i$ or $s_i < 0$. We also remark that this distribution is not actually performed, it is determined only for the sake of computing the value of $F_I(S)$. Later in this section we show how program (3)-(4) can be efficiently solved.

We are now ready to formulate the clustering problem. A valid cluster has a diameter no larger than D. The objective function, to be minimized, consists of two components: the number of clusters, and the sum of the expected costs of the clusters. Let $I_1, \dots, I_m$ be a partition of the stations into m clusters. Then, finding the optimal partition is represented by the following mathematical program, where m and $I_1, \dots, I_m$ are the decision variables:

$$\min_{m;\ I_1,\dots,I_m}\{m + \beta\ \textstyle\sum_{k=1}^{m} F_{I_k}(S_{I_k}^0)\} \tag{5}$$

$s.t$:

$$t_{ij} \le D \qquad \forall i,j \in I_k,\ k = 1,\dots,m \tag{6}$$

Note that the two terms of the objective function (5) are expressed in different units: the first is the number of clusters and the second is a cost. Therefore, we multiply the second term by a scaling parameter β. We explain later how to choose the value of β. Constraints (6) assure that the diameter constraint is satisfied, i.e., stations assigned to the same cluster are close geographically. It is possible to formulate (5)-(6) as an integer programming model. However, solving it exactly is computationally intractable and hence we devised a saving heuristic to solve it.

Our saving algorithm is based on two observations presented and proved in Lemma 1 and Lemma 2 below. In Lemma 1, we show how the cost function $F_I(S)$ can be efficiently evaluated and in Lemma 2 we prove that combining two clusters always saves costs.

The following lemma demonstrates that the optimization problem (3)-(4) is easily solved to optimality by a greedy procedure. Such a procedure assigns the S units one by one, where each unit is assigned to the station where its contribution to the objective function is minimal.

*Lemma 1: Problem (*3*)-(**4**) that finds the cost of a cluster can be solved by the following recursive procedure:*

Let $\tilde{s}_i(I,S)$ ($\forall i \in I$) be the optimal solution of (3)-(4) *for given I and S. Then,*

$$F_I(S+1) = min_{j\in I:\tilde{s}_j(I,S)<c_j}\{f_j(\tilde{s}_j(I,S)+1) + \textstyle\sum_{i\neq j} f_i(\tilde{s}_i(I,S))\} \tag{7}$$

$$\tilde{s}_i(I,0) = 0\ \ \forall i, \forall I \tag{8}$$

Proof:

The proof is by induction on the value of S. Clearly (8) holds. For $S = 1$, the solution must assign one unit to one of the stations. According to (7), the unit would be assigned to the station whose contribution to the objective function is minimal (note that the contribution can be negative), which is clearly the optimal solution for this case.

Now, define $\Delta_i(s) \equiv f_i(s) - f_i(s-1)$. The convexity of the integer valued function $f_i(\cdot)$ implies that $\Delta_i(1), \Delta_i(2), \Delta_i(3), \dots$ is increasing. The relation (7) can be rewritten as

$$F_I(S+1) = \min_{j\in I}\{\Delta_j(\tilde{s}_j(I,S)+1)\} + \textstyle\sum_i f_i(\tilde{s}_i(I,S)) \tag{9}$$

Now, we assume that the optimal solution for S was obtained by recursively applying (9), and we claim that the optimal solution for $S+1$ is also obtained by (9). To prove that, let $k = \text{argmin}_{j\in I}\{\Delta_j(\tilde{s}_j(I,S)+1)\}$. That is, $\Delta_k(\tilde{s}_k(I,S)+1) \leq \Delta_j(\tilde{s}_j(I,S)+1)\ \forall j \neq k$.

Next we will show that: (a) $\tilde{s}_k(I,S+1) \leq \tilde{s}_k(I,S)+1$; (b) $\tilde{s}_k(I,S+1) \geq \tilde{s}_k(I,S)+1$. (a) and (b) imply that $\tilde{s}_k(I,S+1) = \tilde{s}_k(I,S)+1$ and therefore $\tilde{s}_j(I,S+1) = \tilde{s}_j(I,S)\ \forall j \neq k$.

Part (a) is true since any increase of $\tilde{s}_k(I,S+1)$ beyond $\tilde{s}_k(I,S)+1$ would imply decreasing $\tilde{s}_k(I,S+1)$ in some other station(s) $j \neq k$. However, due to the convexity of all $f_j(.)$ functions and the procedure in (9) that imply $\Delta_k(\tilde{s}_k(I,S)+l) \geq \Delta_j(\tilde{s}_j(I,S)-p)\ \forall j \neq k$ for any number of units $l \geq 2$ and $p \geq 1$, such a solution cannot be optimal.

Part (b) is true since if $\tilde{s}_k(I,S+1)$ is not increased beyond $\tilde{s}_k(I,S)$, then $\tilde{s}_j(I,S+1)$ for at least one other station $j \neq k$ must be increased. However, $\Delta_k(\tilde{s}_k(I,S)+1) \leq \Delta_j(\tilde{s}_j(I,S)+p)$ for any number of units $p \geq 1$. Thus, such a solution cannot be optimal.

We conclude that $\tilde{s}_k(I,S+1) = \tilde{s}_k(I,S)+1$ and $\tilde{s}_j(I,S+1) = \tilde{s}_j(I,S)\ \forall j \neq k$, by the induction assumption. This implies that (7) is satisfied. ■

Consider now fixed values of initial inventories in the stations, $(s_1^0, \ldots, s_n^0)$, and recall that $S_I^0 = \sum_{i\in I} s_i^0$. In order to calculate the value of $F_I(S_I^0)$, the minimization of (7) needs to be carried out for all $\text{S} = 1, \ldots, S_I^0$. Although the complexity of this procedure is $O(S_I^0)$ which is pseudo-polynomial, we note that the value S_I^0 of a typical cluster would be in the range of tens or a few hundred at most, therefore it can be solved very quickly. When solving this problem for each value of S from zero up to C_I units, we obtain a full characterization of the $F_I(\cdot)$ function. An immediate corollary of Lemma 1 is that this function is convex. The collection of functions $F_I(\cdot)$ also satisfies the following property:

Lemma 2: Given disjoint clusters I and J, $F_{I\cup J}(S_I^0 + S_J^0) \leq F_I(S_I^0) + F_J(S_J^0)$.

Proof: The proposition is proved by,

$F_{I\cup J}(S_I^0 + S_J^0) = \sum_{i\in I\cup J} f_i(\tilde{s}_i(I\cup J, S_I^0 + S_J^0)) \leq \sum_{i\in I} f_i\left(\tilde{s}_i(I, S_I^0)\right) + \sum_{i\in J} f_i\left(\tilde{s}_i(J, S_J^0)\right) =$
$F_I(S_I^0) + F_J(S_J^0)$.

The inequality above is due to the fact that in the combined cluster, one possible solution for the inventory level at the stations, is to arrange it in the same way as it is done in the separate clusters. However, additional arrangements are possible, which may reduce the cost of the combined cluster. ■

Our *saving heuristic* algorithm for the clustering problem, (5)-(6), is based on a saving function, denoted by $F_{saving}(I,J)$, which represents the saving in the objective function value (5) when combining two distinct clusters I and J into one new cluster. This function is defined for any pair of clusters I and J, whose stations satisfy the distance constraint (6). The saving function is calculated as follows:

For all I and J such that (6) is satisfied, i.e., $t_{ij} \leq D \quad \forall i \in I, j \in J$,

$$F_{saving}(I,J) = 1 + \beta[F_I(S_I^0) + F_J(S_J^0) - F_{I\cup J}(S_I^0 + S_J^0)] \tag{10}$$

The expression in (10) represents the saving because when combining two clusters, the number of clusters in the solution is reduced by one, and a joint cost $F_{I\cup J}(S_I^0 + S_J^0)$ is incurred instead of the sum of the separate costs $F_I(S_I^0) + F_J(S_J^0)$, with the respective weight.

The saving heuristic iterations are performed as follows: Initially, each cluster is represented by a single station and the saving is calculated according to (10) for each pair of clusters whose stations comply with the traveling distance constraint, (6). The pair of clusters with the maximum saving is chosen to be combined into one cluster. The saving and the compliance with the distance constraint is re-calculated for all pairs that include the newly created cluster, and the process is repeated until no pair of clusters satisfies constraint (6). In fact, since by Lemma 2, $F_I(S_I^0) + F_J(S_J^0) - F_{I\cup J}(S_I^0 + S_J^0) \geq 0$, both terms of (10) are always non-negative for any pair of clusters. Moreover, the first term of (10) is constant (and equals one) for any pair of I and J, therefore the pair that is selected to be combined is simply the one that maximizes $F_I(S_I^0) + F_J(S_J^0) - F_{I\cup J}(S_I^0 + S_J^0)$. Interestingly, the value of β is irrelevant for this proposed heuristic and we believe that the value of this parameter admits very minor effect on the optimal solution of the clustering problem. Note that the proposed saving heuristic is inspired by the idea of the saving heuristic for the vehicle routing problem, introduced by Clarke and Wright (1964).

4.2 Step 2: The repositioning problem on the clusters

In the second step, the routes of the vehicles through the clusters are determined, along with tentative decisions on the amount of bicycles to be loaded or unloaded at each station (in the visited clusters). The routing within each cluster is ignored, so that the operations are planned as if all the stations of each cluster are at exactly the same location. For simplicity, we chose this location to be at the station that is closest to the centroid of the cluster. Note that the difficulty in solving large-scale repositioning problems stems from the routing decisions and not from the inventory decisions. Therefore we allow at this stage performing detailed (for each station) inventory decisions while the routing decisions are made only at the cluster level.

We formulate the problem of this step as a mixed integer linear model by adapting the arc-indexed (AI) formulation of Raviv et al. (2013) to this reduced and relaxed version of the problem. We include the revised model in Appendix B. Since the number of clusters is significantly smaller than the number of stations in the original problem, it is possible to solve this model using a commercial solver for fairly large instances.

In order to allow decomposition of the problem to separate single vehicle problems at the next step, additional constraints were imposed on the model at this step. Namely, each cluster can be served by at most one vehicle. This is implemented by constraint (33), see appendix B. While this additional constraint may eliminate some good solutions of the problem, it simplifies Step 3 greatly and thus allows solving larger instances of the problem with smaller optimality gaps under limited computational resources.

The results obtained from the second step do not correspond with a feasible solution to the original problem, since the travel times between stations belonging to the same cluster are ignored. The purpose of this step is merely to determine the order in which the clusters will be visited by each vehicle in Step 3 of the algorithm. The inventory decisions made at Step 2 are only tentative and are included in the model only to facilitate reasonable routing decisions between the clusters.

4.3 Step 3: Obtaining the solution

In this step, the solution of the problem is obtained. The input to this step includes the routing decisions obtained in Step 2, that specify which clusters were visited by each vehicle and in what order. Note that since at Step 2 we allow each cluster to be visited by a single vehicle, the sets of clusters that are visited by each vehicle are disjoint. This creates separate problems, one for each vehicle.

For each vehicle, the constraints imposed by Step 2's decisions specify which clusters will be visited by the vehicle and in what order. Contrary to Step 2, the routing decision variables between stations that belong to the same cluster are also included in the model, but now this poses lesser computational difficulty due to the separation between the vehicles and the smaller number of stations associated with each vehicle's problem. Finally, inventory decisions for all stations are made according to the actual (original) constraints of the problem.

Next we define the separate network for each vehicle, on which the repositioning problem is solved. The problem for each vehicle $v \in V$ is defined on a directed graph $D_v = (N_v \cup \{0\}, A_v)$. The node set N_v is the set of all stations that belong to clusters visited by vehicle v according to Step 2. The depot is added to this node set. The set of arcs, A_v, consists of four categories:

1. All arcs between nodes in N_v that represent stations in the same cluster. That is, if I is a cluster visited by vehicle v in the solution of Step 2, then for all pairs of nodes $i, j \in I$, arc (i, j) is in A_v.
2. All arcs between ordered pairs of nodes in N_v that belong to two consecutive clusters in the routing of vehicle v in Step 2. That is, if vehicle v travels from cluster I to cluster J, and $i \in I$ and $j \in J$ are stations in those clusters, then arc (i, j) is in A_v.
3. All arcs between the depot node and the nodes that belong to the first visited cluster in the route of vehicle v in Step 2.

4. All arcs between nodes that belong to the last visited cluster in the route of vehicle v in Step 2, to the depot node.

The above definition of the set A_v for vehicle v reflects our belief that the route generated between the clusters of this vehicle, provide good guidelines for the final route. In particular, the sequence of traversing the clusters is kept almost unchanged but more flexibility is added by allowing traveling within the clusters and between any two stations of consecutive clusters. Moreover, loading and unloading decisions are reconsidered and finalized at this stage. The problems are solved sequentially, one problem for each vehicle, since the networks of different vehicles are completely disjoint. An example that demonstrates the steps of the algorithm is presented in Figure 1.

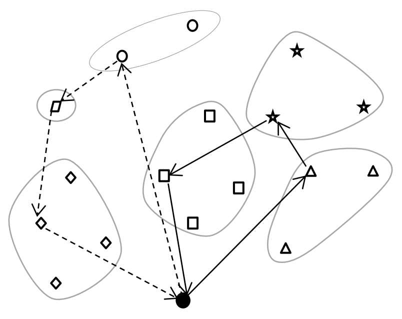
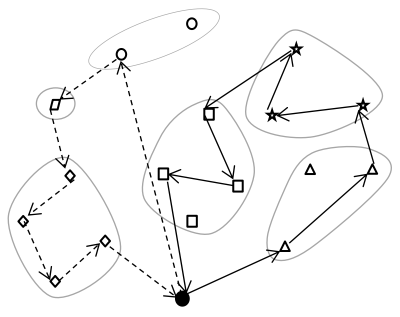
(a)	(b)
	
Step 1: Creating the clusters Step 2: Routing the vehicles through them	Step 3: Routing each vehicle through the stations in the sequence of clusters determined in Step 2.

Figure 1: Example of the solutions obtained by the 3-step algorithm

In Figure 1, the clusters created in Step 1 are circled and the stations of each cluster are denoted by a different shape. In Figure 1a, we present the route of each vehicle through the clusters assigned to it in Step 2. The location of each cluster is represented by a single station that is closest to its centroid. In Figure 1b, we present the final route of each vehicle through the stations it visits.

5. Numerical Experiments

In this section, we present results that were obtained when solving instances of practical size with the 3-step heuristic described in Section 4. We compare the results to those obtained by solving the AI formulation of the original network, using CPLEX. We start by describing and analyzing instances that are based on data from the Vélib system (in Paris). We used the actual locations of some 200 Vélib stations located in the first through the fifth arrondissements. The input of our instances is available online at: http://www.eng.tau.ac.il/~talraviv/Publications/.

We conducted an experiment with all the combinations of the following parameter values:

- Number of stations – 75, 100, 125, 150 and 200 stations, where the smaller instances are subsets of the larger ones.
- Penalty function – representing the expected number of shortages function ($f_i(.)$ for station i), based on a fictitious (but likely representative) demand pattern. The penalty functions were constructed as follows: The stations of the system were divided into three types, namely, 1) residential area stations where the demand for bicycles is high in the morning and the demand for lockers is high in the afternoon; 2) Business area stations with complementary pattern of demand; 3) "touristic" stations where the intensity of the demand for bicycles and lockers behave similarly throughout the day. The penalty functions were calculated using the method of Raviv and Kolka (2013) based on the actual capacity of each station.
- Workload (light, real, heavy) - the initial inventory levels at the stations influence the amount of work necessary in order to reach their ideal inventory levels. The further the initial inventory level of a station is from its ideal level, the more work is necessary. That is, the number of bicycles to load/unload to/from the station is higher. Therefore, we test three different cases of initial inventories called: "light ", "real" and "heavy" referring to different workloads. For the "light" case we generated initial inventories from a normal distribution with mean that corresponds to the ideal inventory and a standard deviation that equals to $0.2c_i$. In the "heavy" case, the standard deviation is equal to $0.6c_i$. In this way we generate data with a higher expected workload. The "real" case corresponds to real initial inventories that were observed in the system at midnight on a random day.
- Number of vehicles – two and three vehicles.
- Total time allocated to repositioning – 5 hours (18,000 seconds).
- α, the weight of the operating/travel costs per second relative to an expected shortage of one unit – a value of 1/900 was used. This means that traveling 900 seconds (=15 minutes) is equivalent (in cost) to an expected shortage of one user.

The travel time matrix was calculated based on the L_1 (Manhattan) metric. The loading and unloading times were set to be one minute/bicycle. The vehicle's capacity was set to 25 bicycles, which is the capacity of the light trucks used by Vélib. The location of the depot was selected to correspond to the location of station number one, situated in the 1st quarter. The capacity and the initial inventory of the depot were set to be large enough so that they were not binding. The dataset for our benchmark problems is available from the authors upon request.

After some tuning, we set the parameter that defines the maximal distance between stations in the same cluster, D, to be 800 seconds.

The saving heuristic used for the first step of our algorithm is implemented by a code computed with MATLAB. We implemented the second and third steps using IBM-Ilog OPL, and solved the above instances using IBM-Ilog CPLEX 12.3 on an Intel i7 2600 @ 3.4GHz with 16GB of RAM. In all our experiments, we used CPLEX's default settings.

While the saving heuristic of Step 1 took few seconds to run and the model of Step 3 was solved by CPLEX in a very short time as well (up to 42 seconds for the hardest instances), we could not solve the model of Step 2 for the larger instances in a reasonable time. Therefore, we set a time limit of one hour and used the best integer solution obtained by this time. In real scenarios one hour is approximately the time that can be allotted since the problem should be solved before the repositioning vehicles start their duties but after the state of the stations is revealed.

For the sake of benchmarking the 3-step algorithm we also solved the AI formulation subject to the same time limitation of one hour. Table 5, which contains detailed results of our experiments with both the 3-step algorithm and the AI formulation, is provided in Appendix C. In the rest of this section we summarize various aspects of these results and discuss their implications.

The outcome of Step 1 for each instance was a set of clusters where the average number of stations per cluster was about four. Therefore, the dimension of the routing problem solved at Step 2 decreased significantly.

In order to explore the limits of Step 2 we report in Table 1 on the quality of the results of this step in terms of optimality gaps obtained within one hour. For the smaller instances that were solved to optimality in less than one hour we report on the solution time in seconds, instead. The first two columns of the table describe the characteristics of the instances in terms of number of stations and number of vehicles. The next three columns present the relative optimality gaps for the light, real and heavy workloads. The relative optimality gap, as calculated by CPLEX, is defined as follows:

$$Relative\ Optimality\ Gap = \frac{Solution\ value\ \ - Lower\ Bound}{Solution\ value}$$

The optimality gap is calculated relative to the lower bound obtained from the MILP solver for the problem solved in Step 2. Note that, strictly speaking, this is not a valid lower bound for the original problem because the route between the centers of the clusters may be, in some rare cases, longer than the route between the stations that are actually served. This may occur, if not all the stations of the visited clusters are served in the optimal solution. A valid lower bound for the original problem is presented below. An asterisk (*) is used to denote instances that were solved within the CPLEX default optimality tolerance of 0.01%. The three rightmost columns present the solution time in seconds. A value of 3600 seconds is reported for instances that could not be solved to optimality within one hour.

We observe that while most of the larger instances could not be solved to optimality within the allotted hour, the optimality gaps of all tested problems are not larger than 2.52%. Since Step 2 is the most computationally demanding component of our heuristic, these results imply its applicability to large problems. As expected, the optimality gap and solving time increase as the network size and the number of vehicles increases. However, these values are less sensitive to the workload, whereas the real workload is the easiest to solve in most cases.

Stations	Vehicles	Relative Optimality Gap Step 2 (%)			Solution time Step 2 (sec.)		
		Light	Real	Heavy	Light	Real	Heavy
75	2	*	*	*	108	91	110
	3	*	*	0.28	114	188	3600
100	2	*	*	*	1154	237	1117
	3	0.11	*	1.59	3600	614	3600
125	2	*	*	*	2119	727	423
	3	0.65	0.12	1.78	3600	3600	3600
150	2	0.34	*	0.31	3600	1684	3600
	3	1.10	0.84	1.57	3600	3600	3600
200	2	1.05	2.16	0.99	3600	3600	3600
	3	2.52	1.77	2.12	3600	3600	3600

Table 1 – Relative optimality gap and solution time for Step 2

In Table 2 we present the relative optimality gaps obtained for the 3-step algorithm and the AI formulation. The gaps for both solution methods are defined relative to a valid lower bound of the original problem, obtained from the MILP solver for the AI formulation. The structure of the table is identical to that of Table 1.

We observe from Table 2 that the instances with real workload are solved by the 3-step algorithm with a small optimality gap, between 0.95% and 5.11%. In all the harder instances, the optimality gaps of the solutions obtained by the 3-step algorithm are significantly smaller than those obtained by the AI formulation. In fact, the AI formulation could not obtain a feasible solution for 17 out of the 30 test instances. The optimality gap of these instances is reported in the table as "-". The AI formulation outperformed the 3-step algorithm only in two instances, with 75 stations and light workload. The optimality gaps of the 3-step algorithm are greatly affected by the workload, with higher optimality gaps for heavier workload. Interestingly, the workload obtained from observing the system on a random day (the “Real” instances) is slightly lighter than the workload of the “Light” instances that we generated, see Table 5 in Appendix C. Consequently, these instances are solved with smaller optimality gaps.

Stations	Vehicles	Relative Optimality Gap (%)					
		3-step algorithm			AI Formulation		
		Light	Real	Heavy	Light	Real	Heavy
75	2	2.01	0.95	4.94	0.87	1.62	8.39
	3	2.15	1.13	6.72	0.92	1.46	12.54
100	2	1.67	1.58	3.34	-	9.31	7.25
	3	1.95	1.30	9.92	2.78	2.03	-
125	2	2.06	1.62	3.89	5.65	9.03	-
	3	2.53	1.52	9.73	-	4.76	-
150	2	2.55	1.15	6.93	-	-	-
	3	4.69	3.30	9.23	-	-	-
200	2	4.61	2.85	4.52	-	-	-
	3	5.65	5.11	7.21	-	-	-

Table 2 – Relative optimality gap for the 3-step algorithm and AI formulation

In order to measure the performance of both solution methods using a normalized scale, we consider the optimality gap of the solution obtained within one hour, relative to the extent of the potential improvement. The potential improvement, which is independent of the solution method, is defined as the difference between an upper bound and a lower bound. The former is the expected penalty assuming no repositioning is carried out and the latter is a lower bound obtained from the AI formulation when run on CPLEX for two hours. The above ratio is defined as the *normalized optimality gap* (NOG):

$$NOG = \frac{Solution\ value - Lower\ bound}{Upper\ bound - Lower\ bound} = \frac{Optimality\ gap}{Potential\ improvement}$$

This notion is depicted in Figure 2.

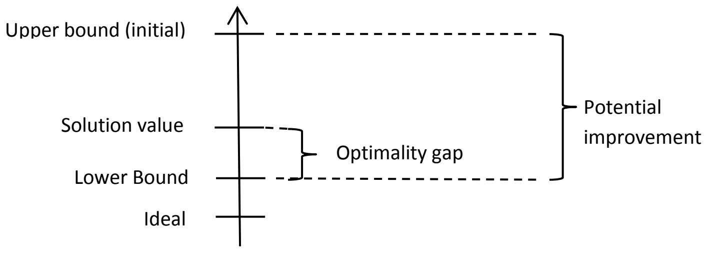


Figure 2: Normalized Optimality Gap (NOG)

The NOG measures the benefit of applying a solution method on a given problem instance because it reveals the improvement achieved relative to what potentially could be improved. We note that while the optimality gap is sensitive to the definition of the penalty function, the NOG is more robust. For example, by adding a positive constant to the penalty functions the relative optimality gap is reduced while the NOG is not affected.

Table 3 reports on the NOG for the 3-step algorithm and for the AI formulation for all instances that were reported in Table 1. The first three columns of the table describe the characteristics of the instances in terms of number of stations and number of vehicles. The next three columns present the NOG value (in %) obtained by the 3-step algorithm for the light, real and heavy workloads. The rightmost three columns present the NOG value (in %) obtained by the AI formulation. In cases when a feasible solution could not be obtained within the time limit, no NOG value is reported.

Stations	Vehicles	NOG (%)					
		3-step algorithm			AI formulation		
		Light	Real	Heavy	Light	Real	Heavy
75	2	22.27	8.59	12.53	9.33	14.42	19.91
	3	23.47	10.20	13.27	9.84	13.04	23.10
100	2	23.27	15.70	11.02	-	90.95	23.07
	3	26.69	12.76	23.12	37.26	19.74	-
125	2	23.53	17.33	15.44	63.05	95.11	-
	3	26.83	16.17	24.51	-	46.12	-
150	2	29.69	10.92	28.42	-	-	-
	3	38.02	25.83	26.22	-	-	-
200	2	43.17	29.96	26.38	-	-	-
	3	45.18	37.86	28.12	-	-	-

Table 3 – Normalized optimality gap for the 3-step algorithm and AI formulation

The results presented in Table 3 are visualized in Figure 3 - Figure 5 for the light, real and heavy workload, respectively. The vertical axis represents the normalized optimality gap, in %, where a value of 100% means that no feasible solution was found. Each bar presents an instance with its number of stations and repositioning vehicles.

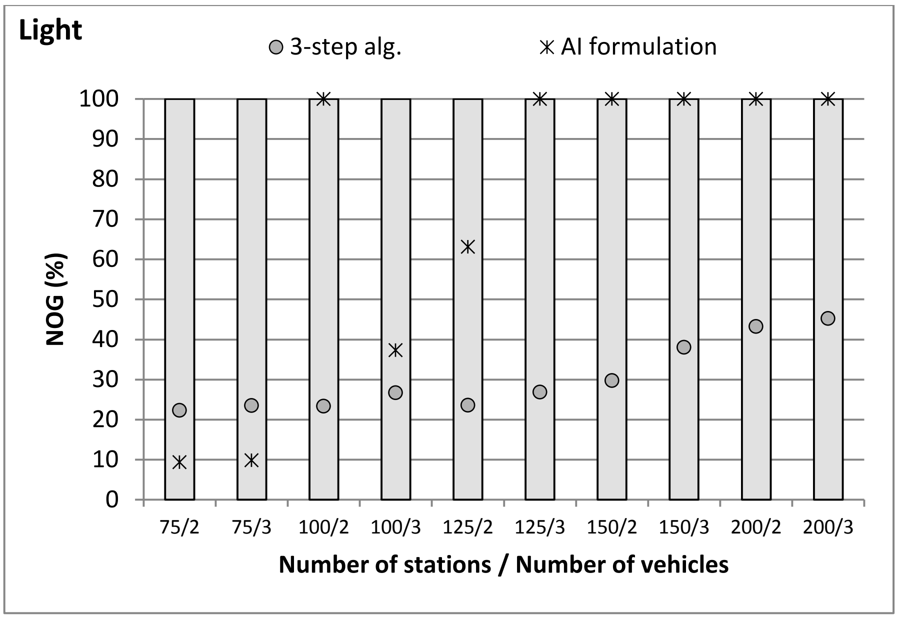


Figure 3: “Light” case: NOG for the 3-step algorithm and the AI formulation

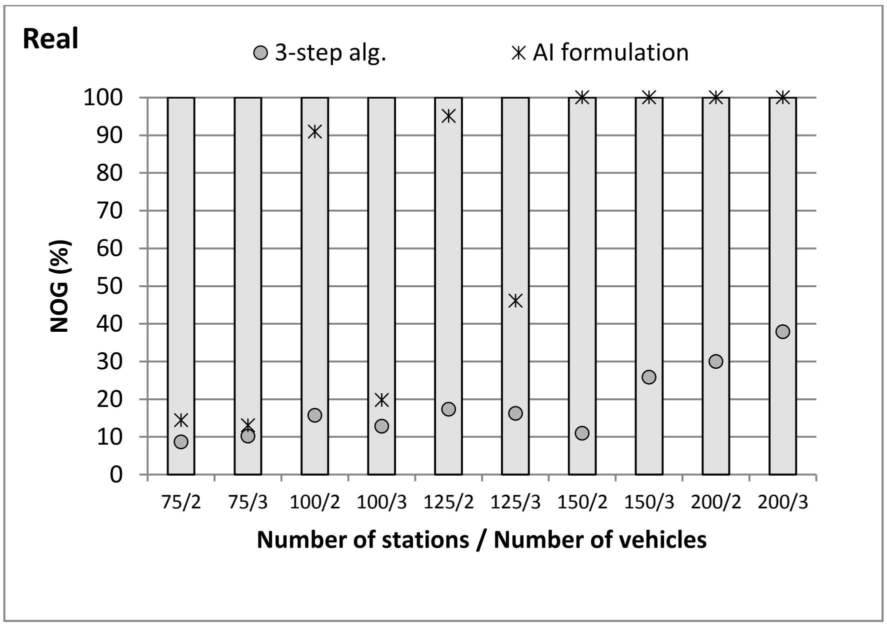


Figure 4: “Real” case: NOG for the 3-step algorithm and the AI formulation

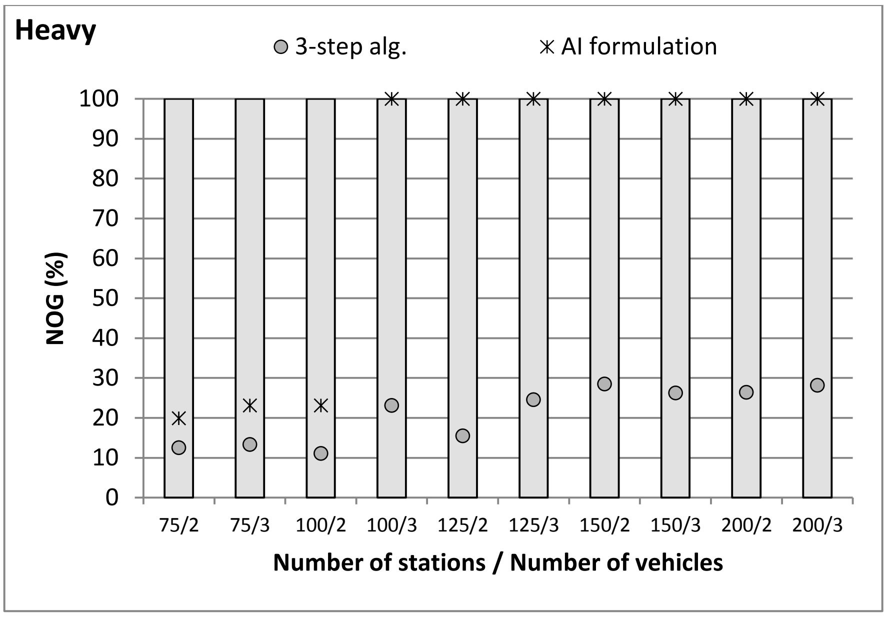


Figure 5: “Heavy” case: NOG for the 3-step algorithm and the AI formulation

We observe that in most cases the 3-step heuristic algorithm outperforms the result obtained from CPLEX using the AI formulation subject to the one hour time limit. There are only two exceptions, out of 30 instances, which refer to the instances with the smallest number of stations examined, and in which the workload was light. In fact, with the AI formulation, for most of the larger instances, we could not even obtain a feasible solution after one hour. The 3-step algorithm, on the other hand, always delivered feasible solutions that materialized substantial share of the potential improvement. As expected, for both methods the NOG is generally lower for smaller instances in terms of the number of stations and vehicles.

We also observe that the NOG obtained by the 3-step algorithm is not very sensitive to the workload. In particular, the NOG values of the heavy workload are always lower than those of the light workload, and in some cases also lower than those of the real workload.

The above results show that our 3-step algorithm is a good heuristic for the SBRP with at least 200 stations and three repositioning vehicles, which represents bike sharing systems of moderate-large size. This is an improvement over previous results for the same problem, which were able to solve the problem with up to about 100 stations and two repositioning vehicles.

6. Conclusions and discussion

This paper presents a new math-heuristic for the SBRP, referred to as a 3-step algorithm. It enables operators of bike sharing systems to obtain a near optimal solution of the problem. For larger systems, a practical approach is to decompose the system into several geographical areas, such that each area is served by two or three vehicles. The 3-step algorithm could be a building block of such a solution approach. For example, the clustering approach suggested by Schuijbroek et al. (2013) can be enhanced by including in each cluster as many stations and repositioning vehicles as can be handled by our heuristic.

One of the innovative components of our heuristic is the idea of reducing the size of the routing problem by clustering together sets of points that are likely to be visited consecutively. Given the newly defined network, the routes among these clusters are optimized and the solution is used to guide the search for a good feasible route in the original network. Such a decomposition approach can be useful to many other rich vehicle routing problems, for example, inventory routing problems.

Acknowledgment

This research was supported by the Israel Science Foundation (ISF) grant no 1109/11.

References

Baldacci, R., Bartolini, E. and Laporte, G. (2010). Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, 61, 1072-1077.

Battarra, M., Erdoğan, G. and Vigo, D. (2014). Exact Algorithms for the Clustered Vehicle Routing Problem. *Operations Research*, 62(1), 58–71.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I. and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. TOP, 15:1–31.

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

Chemla, D., Meunier, F., Pradeau, T., Wolfler Calvo, R. and Yahiaoui, H. (2013b). Self-Service Bike Sharing Systems: Simulation, Repositioning, Pricing. Working paper. http://hal.archives-ouvertes.fr/docs/00/82/40/78/PDF/RealTime-BikeSharing_final.pdf.

Chow, J.Y.J. and Sayarshad, H.R. (2014). Symbiotic network design strategies in the presence of coexisting transportation networks. Transportation Research Part B, 62, 13-34.

Clarke, G. and J.W. Wright. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, Vol. 12, 568-581.

Côme, E., Randriamanamihaga, A., Oukhellou, L., Aknin, P., (2013) Spatio-temporal analysis of Dynamic Origin-Destination data using Latent Dirichlet Allocation. Application to the Vélib' Bike Sharing System of Paris. Working paper Université Paris-Est, IFSTTAR, COSYS-GRETTIA, F-77447 Marne-la-Vallée, France, http://www.comeetie.fr/pdfrepos/LDATRB.pdf

Contardo, C., Morency, C. and Rousseau, L.-M. (2012). Balancing a dynamic public bike-sharing system. Working paper. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf.

DeMaio, P. (2009). Bike-sharing: history, impacts, models of provision, and future, *Journal of Public Transportation*, Vol. 12(4), 41-56.

DeMaio, P. and Meddin, R., (2014). The bike sharing blog, http://bike-sharing.blogspot.com/, observed at April 11, 2014.

Erdoğan, G., Laporte, G. and Calvo, R.W. (2013). The One-Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals, working paper, https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2013-46.pdf.

Fisher, M. L. and Jaikumar, R. (1981). A Generalized Assignment Heuristic for Vehicle Routing. *Networks*, 11, 109-124.

Forma I., Raviv T., Tzur M. (2010). The static repositioning problem in a bike-sharing system. In: Proceeding of the 7th triennial symposium on transportation analysis (TRISTAN), Tromsø, Norway, 279–282

Fricker, C. and Gast, N. (2014). Incentives and regulations in bike-sharing systems with stations of finite capacity. *European Journal of Transportation and Logistics.*

Gillett, B. E. and Miller, L. R. (1974). A Heuristic Algorithm for the Vehicle Dispatch Problem. *Operations Research*, 22, 340-349.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145, 126-139.

Kaltenbrunner, A., Meza, R., Grivolla, J., Codina, J., and Banchs, R. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system, *Pervasive and Mobile Computing*, 6 (4), pp. 455-466.

Kaspi, M., Raviv. T. and Tzur, M. (2014a). Parking Reservation Policies in One-Way Vehicle Sharing Systems. *Transportation Research B*, 62, 35-50.

Kaspi, M., Raviv, T., Tzur, M., and Galili, H. (2014b). Analysis and performance bounds of parking reservation policies in vehicle sharing systems. Working paper.

Kloimüllner, C., Papazek, P., Hu, B. and Raidl, G.R. (2014). Balancing Bicycle Sharing Systems: An Approach for the Dynamic Case. In: C. Blum and G. Ochoa (Eds.): EvoCOP 2014, LNCS 8600, 73–84.

Laporte, G. and Semet, F. (2002). Classical Heuristics for the Capacitated VRP, in Toth, P. and Vigo, D. editors, *The Vehicle Routing Problem*, 109-128, SIAM,

Larsen, J. (2013), Plan B Updates - Bike-Sharing Programs Hit the Streets in Over 500 Cities Worldwide, http://www.earth-policy.org/plan_b_updates/2013/update112, published April 25, 2013, observed on April 11, 2014.

Lin, J-R. and Yang, T.-H. (2011). Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E*, 47(2), 284–294.

Miller, C.E., Tucker, A.W. and Zemlin, R.A. (1960). Integer programming formulations and traveling salesman problems. *J. ACM*, **7**, 326–329.

Nair, R., Miller-Hooks, E., Hampshire, R.C. and Bušić, A. (2013). Large-Scale Vehicle Sharing Systems: Analysis of Vélib'. International Journal of Sustainable Transportation. 7, 85–106.

Pessach, D., Raviv, T., Tzur, M., 2014. Dynamic Repositioning in a Bike Sharing System. Working Paper, Tel-Aviv University.

Pfrommer, J. Warrington, J., Schildbach, G. and Morari, M. (2014). Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6754195.

Raidl, G.R., Hu, B., Rainer-Harbach, M., and Papazek, P. (2013), Balancing bicycle sharing systems: Improving a VNS by efficiently determining optimal loading operations . In M. J. Blesa et al., editors, Hybrid Metaheuristics, 8th International Workshop, volume 7919 of *Lecture Notes in Computer Science*, pages 130-143. Springer Berlin Heidelberg, 2013.

Rainer-Harbach, M., Papazek, P., Hu, B. and Raidl, G.R. (2013). Balancing Bicycle Sharing Systems: A Variable Neighborhood Search Approach Evolutionary Computation in Combinatorial Optimization. Lecture Notes in Computer Science, Vol. 7832, pp. 121-132.

Raviv, T., Tzur, M. and Forma, I. (2013). Static Repositioning in a Bike-Sharing System: Models and Solution Approaches. *European Journal of Transportation and Logistics*, 2, 187-229.

Raviv, T. and Kolka, O. (2013). Optimal Inventory Management of a Bike-Sharing Station, *IIE Transactions* 45(10), 1077-1093.

Romero, J.P., Ibeas, A., Moura, J.L., Benavente, J., and Alonso, B. (2012). A Simulation-optimization Approach to Design Efficient Systems of Bike-sharing, Procedia - *Social and Behavioral Sciences*, 54, 646-655.

Rudloff, C., and Lackner, B. (2014). Modeling Demand for Bicycle Sharing Systems – neighboring stations as a source for demand and a reason for structural breaks, TRB 2014 Annual Meeting

Schuijbroek, J., Hampshire, R., van Hoeve W-J., (2013). Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems, working paper, http://repository.cmu.edu/cgi/viewcontent.cgi?article=2490&context=tepper.

Shu, J., Chou, M., Liu, Q., Teo, C-P. and Wang, I-L. (2013). Models for Effective Deployment and Redistribution of Bicycles Within Public Bicycle-Sharing Systems. *Operations Research*, 61(6), 1346-1359.

Shu, J., Teo, C.-P., and Shen, Z.J.M. (2005). Stochastic Transportation-Inventory Network Design Problem. Operations Research, 53(1), 48-60.

Vogel, P., Greiser, T., and Mattfeld, D.K., (2011). Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns, *Procedia - Social and Behavioral Sciences* , Volume 20, 2011, Pages 514-523, ISSN 1877-0428, http://dx.doi.org/10.1016/j.sbspro.2011.08.058.

Waserhole A. and Jost V. (2014), Pricing in Vehicle Sharing Systems: Optimization in queuing networks with product forms, to appear in *Euro Journal of Transportation and Logistics*

Appendix A – The Arc Indexed Formulation

Notation

The SBRP is described by the following sets and parameters:

N	Set of stations, indexed by $i = 1, \dots, \lvert N \rvert$
N_0	Set of nodes, including the stations and the depot (denoted by $i = 0$), $i = 0, \dots, \lvert N \rvert$
V	Set of vehicles, $v = 1, \dots, \lvert V \rvert$
s_i^0	Number of bicycles at node *i* before the repositioning operation starts
c_i	Number of lockers installed at station $i \in N_0$, referred to as the station's capacity
k_v	Capacity (number of bicycles) of vehicle $v \in V$
$f_i(s_i)$	A convex penalty function for station $i \in N$, the function is defined over the integers $s_i = 0, \dots, c_i$
t_{ij}	Traveling time from station *i* to station *j*
α	Weight / scaling factor (in the objective function) of the operating costs relative to the penalty costs
T	Repositioning time, i.e., time allotted to the repositioning operation
L	Time required to remove a bicycle from a station and load it onto the vehicle
U	Time required to unload a bicycle from the vehicle and hook it to a locker in a station

We use the following decision variables:

x_{ijv}	Binary variable which equals one if vehicle v travels directly from node i to node j, and zero otherwise
y_{ijv}	Number of bicycles carried on vehicle v when it travels directly from node i to node j. y_{ijv} is zero if the vehicle v does not travel directly from i to j
y_{iv}^L	Number of bicycles loaded onto vehicle v at node i
y_{iv}^U	Number of bicycles unloaded from vehicle v at node i
q_{iv}	Auxiliary variables used for sub-tour elimination constraints
s_i	Inventory level at node i at the end of the repositioning operation
M	An upper bound on the number of arcs in a vehicle's tour whose length is at most T time units, where the vehicle visits each station at most once.

The Arc Indexed (AI) formulation is the following:

$$Min \sum_{i \in N} f_i(s_i) + \alpha \sum_{i \in N_0} \sum_{j \in N_0} \sum_{v \in V} t_{ij} x_{ijv} \tag{11}$$

s.t.

$$s_i = s_i^0 - \sum_{v \in V} (y_{iv}^L - y_{iv}^U) \qquad \forall\, i \in N_0 \tag{12}$$

$$y_{iv}^L - y_{iv}^U = \sum_{j \in N_0, j \neq i} y_{ijv} - \sum_{j \in N_0, j \neq i} y_{jiv} \qquad \forall\, i \in N_0,\ \forall\, v \in V \tag{13}$$

$$y_{ijv} \leq k_v x_{ijv} \qquad \forall\, i, j \in N_0, i \neq j,\ \forall\, v \in V \tag{14}$$

$$\sum_{j \epsilon N_0, j \neq i} x_{ijv} = \sum_{j \epsilon N_0, j \neq i} x_{jiv} \qquad \forall\, i \in N_0, \forall\, v \in V \qquad (15)$$

$$\sum_{j \epsilon N_0, j \neq i} x_{ijv} \leq 1 \qquad \forall\, i \in N,\ \forall\, v \in V \qquad (16)$$

$$\sum_{v \in V} y_{iv}^L \leq s_i^0 \qquad \forall\, i \in N_0 \qquad (17)$$

$$\sum_{v \in V} y_{iv}^U \leq c_i - s_i^0 \qquad \forall\, i \in N_0 \qquad (18)$$

$$\sum_{i \in N_0} \left(y_{iv}^L - y_{iv}^U \right) = 0 \qquad \forall\, v \in V \qquad (19)$$

$$\sum_{i \in N} \left(L y_{iv}^L + U y_{iv}^U \right) + \sum_{i \in N} \left(L y_{0iv} + U y_{i0v} \right) + \sum_{i,j \in N_0 : i \neq j} t_{ij} x_{ijv} \leq T \qquad \forall\, v \in V \qquad (20)$$

$$q_{jv} \geq q_{iv} + 1 - M\left(1 - x_{ijv}\right) \qquad \forall\, i \in N_0, j \in N, i \neq j, \forall\, v \in V \qquad (21)$$

$$x_{ijv} \in \{0,1\} \qquad \forall\, i,j \in N_0 \colon i \neq j,\ \forall\, v \in V \qquad (22)$$

$$y_{iv}^L \geq 0,\ y_{iv}^U \geq 0, \text{ integer} \qquad \forall i \in N_0\,, \forall v \in V \qquad (23)$$

$$y_{ijv} \geq 0 \qquad \forall\, i,j \in N_0 \colon i \neq j,\ \forall\, v \in V \qquad (24)$$

$$s_i \geq 0 \qquad \forall\, i \in N_0 \qquad (25)$$

$$q_{iv} \geq 0 \qquad \forall i \in N_0\,, \forall v \in V \qquad (26)$$

The objective function (11) minimizes the total cost of the system, consisting of the sum of the penalties incurred at all stations and the total operating costs, appropriately weighted by a factor of α. Note that the objective function consists of a sum of convex functions over a discrete set, which can be linearized as described as described below, see (27) and (28). Constraints (12)) are inventory-balance constraints at the nodes (the stations and the depot). Constraints (13) represent the conservation of inventory on the vehicles, and constraints (14) limit the quantity carried on each vehicle to its capacity. These constraints also set the quantity carried on a vehicle to zero when it travels directly from i to j if the vehicle does not use that arc. Constraints (15) are vehicle flow-conservation equations. Constraints (16) ensure that each station is visited at most once by each vehicle and constraints (17) (resp., (18)) limit the quantity picked-up by all vehicles from a station (resp., delivered to a station) to the quantity available there initially (resp., the residual capacity of the station). Note that constraints (17) also imply non-negativity of the inventory variables, while constraints (18) also ensure that the inventory at each station and at the depot is bounded by its capacity; therefore, these two restrictions are not written explicitly. Constraints (19) stipulate that all the bicycles that are loaded onto the vehicles are also unloaded. Constraints (20) limit the total loading and unloading times plus the travel times to the total time available for the repositioning operation. Constraints (21) are sub-tour elimination constraints that are similar to those of Miller et al. (1960). Finally, (22) and (23) are binary and general integrality constraints, respectively, and (24)-(26) are non-negativity constraints. Note that the integrality of y_{ijv} and s_i is implied by the integrality of y_{iv}^L, y_{iv}^U and s_i^0.

As mentioned above, we can replace the previous convex terms in the objective function by linear terms:

$Min \sum_{i\in N} g_i + \alpha \sum_{i\in N_0} \sum_{j\in N_0} \sum_{v\in V} t_{ij} x_{ijv}$ (27)

and add the following constraints to the formulation:

$g_i \geq a_{iu} + b_{iu} s_i \qquad \forall\, i \in N, u = 0, \dots, c_i - 1$ (28)

Where g_i is the penalty incurred at station i, $b_{iu} \equiv f_i(u+1) - f_i(u)$, $a_{iu} \equiv f_i(u) - b_{iu} \cdot u \;\; \forall i, \forall u$. We also added the same cut constraints as detailed in to Raviv et al. (2013), in order to speed the solution time.

<u>Appendix B – the Arc indexed (AI) formulation adapted to clusters routing (step 2)</u>

We explain how we adapt the notation to the problem definition of this step. We let $B = \{I_1, \dots, I_n\}$ represent the set of clusters obtained in the first step, $B_0 = \{I_0, I_1, \dots, I_n\}$ where $I_0 = \{0\}$ is a cluster which consists of the depot only, and we use I and J to represent general clusters. In a pre-processing calculation, we set the location of a cluster to be the location of the closest station to the center of gravity of the cluster, and denote the travel distance between clusters I and J by t_{IJ}.

In the definitions below, since routing decisions are considered between clusters only, routing decision variables are also defined between clusters only. In particular, no routing decision variables are defined between stations. On the other hand, all decision variables that concern the inventory decisions still exist.

The adapted decision variables (relative to the original AI formulation, which is stated in Appendix A):

x_{IJv}	binary variable that equals one if vehicle v travels directly from cluster I to cluster J, and zero otherwise;
y_{IJv}	number of bicycles carried on vehicle v when it travels directly from cluster I to cluster J. y_{IJv} is zero if the vehicle v does not travel directly from I to J;
q_{Iv}	auxiliary variables, used for sub-tour elimination constraints;

The same decision variables (that are the same as in the original AI formulation):

y_{iv}^L	number of bicycles loaded onto vehicle v at station i;
y_{iv}^U	number of bicycles unloaded off of vehicle v at station i;
s_i	inventory level at station i at the end of the repositioning operation (mentioned before);

Arc indexed (AI) formulation adapted to clusters routing

$Min \sum_{i\in N} f_i(s_i) + \alpha \sum_{I\in B_0} \sum_{J\in B_0} \sum_{v\in V} t_{IJ} x_{IJv}$ (29)

s.t.

$s_i = s_i^0 - \sum_{v\in V}(y_{iv}^L - y_{iv}^U) \qquad \forall\, i \in N_0$ (12)

$\sum_{i\in I}(y_{iv}^L - y_{iv}^U) = \sum_{J\in B_0, J\neq I} y_{IJv} - \sum_{J\in B_0, J\neq I} y_{JIv} \qquad \forall\, I \in B_0,\ \forall\, v \in V$ (30)

$y_{IJv} \leq k_v x_{IJv} \qquad \forall\, I, J \in B_0, I \neq J,\ \forall\, v \in V$ (31)

$\sum_{J\epsilon B_0,J\neq I} x_{IJv} = \sum_{J\epsilon B_0,J\neq I} x_{JIv}$ $\quad \forall\, I \in B_0, \forall\, v \in V$ (32)

$\sum_{J\epsilon B_0,J\neq I} \sum_{v\in V} x_{IJv} \ \leq\ 1$ $\quad \forall\, I \in B$ (33)

$\sum_{v\in V} y_{iv}^{L} \leq s_i^0$ $\quad \forall\, i \in N_0$ (17)

$\sum_{v\in V} y_{iv}^{U} \leq c_i - s_i^0$ $\quad \forall\, i \in N_0$ (18)

$\sum_{i\in N_0}\left(y_{iv}^{L} - y_{iv}^{U}\right) = 0$ $\quad \forall\, v \in V$ (19)

$\sum_{i\in N}\left(Ly_{iv}^{L} + Uy_{iv}^{U}\right) + \sum_{I\in B}\left(Ly_{0Iv} + Uy_{I0v}\right) + \sum_{I,J\in B_0:I\neq J} t_{IJ}x_{IJv} \leq T$ $\quad \forall\, v \in V$ (34)

$q_{Jv} \geq q_{Iv} + 1 - M\left(1 - x_{IJv}\right)$ $\quad \forall\, I \in B_0, J \in B, I \neq J, \forall\, v \in V$ (35)

$x_{IJv} \in \{0,1\}$ $\quad \forall\, I,J \in B_0 \colon I \neq J,\ \forall\, v \in V$ (36)

$y_{iv}^{L} \geq 0,\ y_{iv}^{U} \geq 0$, integer $\quad \forall i \in N_0\,, \forall v \in V$ (23)

$y_{IJv} \geq 0$ $\quad \forall\, I,J \in B_0 \colon I \neq J,\ \forall\, v \in V$ (37)

$s_i \geq 0$ $\quad \forall\, i \in N_0$ (25)

$q_{Iv} \geq 0$ $\quad \forall I \in B_0\,, \forall v \in V$ (38)

In the formulation above, the first term of the objective function (29), which refers to the inventory-related costs, is the same as in the original AI formulation. (This term is later linearized, similar to (27) and (28) above.) The second term refers to the operating costs of routing between clusters, rather than between stations as in the original AI formulation. Note that the value of the objective function is a lower bound to the real costs of the repositioning problem, since the operating costs don't include the routing between stations within clusters.

As for the constraints, similarly to the definition of the decision variables, those that refer to inventory at the stations have not changed (Constraints (9), (17)-(18), (19), (23), (25), whose reference number did not change). The constraints that refer to the routing are similar to the original ones, only with a cluster index rather than a station index (Constraints (31)-(32), (35)-(38)). Finally, Constraints (30), (33) and (34) have changed slightly. Constraint (30) is a flow conservation constraint, which applies now to clusters rather than to stations. In Constraint (33) we assume that each cluster may be visited at most once by all vehicles; this is different from the original formulation in which this constraint is applied to each vehicle separately; this change is motivated by Step 3 of our algorithm, see Section 4.3. Constraints (34) limit the total loading and unloading times at stations as well as at the depot, plus the travel times between clusters to the length of time available for the repositioning operation. In this constraint, routing variables refer again to clusters rather than to stations.

We add the constraints (cuts) that were included in the original formulation of Raviv et al. (2103), adapted to the formulation above, i.e., where routing decision variables refer to clusters rather than to stations, and inventory decision variables are unchanged. We also add the following constraints, which follows from (33):

$\left|\sum_{i\in I}\sum_{v\in V}(y_{iv}^{L} - y_{iv}^{U})\right| \leq Max_{v\in V}k_v \cdot \sum_{J\epsilon B_0:J\neq I}\sum_{v\in V} x_{IJv}$ $\quad \forall\, I \in B$ (39)

While (39) is not linear due to the absolute value at the left hand side, it can easily be linearized.

Appendix C – Detailed numerical results

In this appendix, we first provide some characteristics of the instances considered in our experimental study, see Table 4. Note that in this table we ignore the number of vehicles used to carry out the repositioning. Next, we present information on the results obtained for each instance in this experiment, including the data from which the optimality gaps presented in Table 1-Table 3 in Section 5 were calculated.

The first and second columns in Table 4 denote the number of stations and the workload of each instance. The next column, entitled "Initial", presents the expected number of shortage events assuming no repositioning is done. This value is used as an upper bound for the solution value. The column entitled "Ideal" presents the expected number of shortage events assuming the number of bicycles at each station before the demand commences is optimal (i.e., ideal, ignoring the repositioning effort costs and constraints). The rightmost column in the table specifies the total number of bicycles that need to be added or removed at the stations in order to change the initial state of the system to its ideal state. Note that this number is significantly smaller for the "Light" and "Real" instances compared with the "Heavy" instances. The latter were artificially created in order to examine the effect of the workload on the performance of the proposed solution method.

# of Stations	Workload	Initial	Ideal	To Add/ Remove
75	Light	556.32	498.78	264
	Real	562.72		259
	Heavy	829.17		656
100	Light	734.50	647.71	337
	Real	737.13		328
	Heavy	1048.19		807
125	Light	932.44	809.10	455
	Real	916.98		410
	Heavy	1334.44		1056
150	Light	1135.74	981.54	553
	Real	1131.30		499
	Heavy	1621.20		1270
200	Light	1556.69	1329.22	797
	Real	1548.77		735
	Heavy	2300.68		1869

Table 4 – Characterization of the numerical experiment instances

Table 5 contains the results of the 3-step algorithm and the AI formulation. The first three columns present the problem characteristics of each instance, namely, number of stations, number of vehicles and workload. In the next column we report on the lower bound of the optimal solution obtained from the MILP solver applied on the original AI formulation, see Appendix A. In the next five columns we report on the results of running the 3-step algorithm. First the solution values of Step 2 and Step 3 are presented. Note that the former is consistently slightly lower since the model solved in this step does not take into account the travel time of the vehicles between the stations of each cluster. In the next two columns the two components of the objective function are presented: the expected shortage and the travel time. Recall that the value of the solution is the expected shortage plus α multiplied by the travel time. The number of bicycles that are added and removed at the stations in the solution obtained by the 3-step algorithm is presented next. In the last four columns similar information is presented with respect to the AI formulation. No information is provided when the problem could not obtain a feasible solution within the time limit of one hour.

Problem characteristics				3-step algorithm					AI formulation			
Stations	Vehicles	Work-load	LB	Step 2 value	Step 3 value	Shortage	Travel time (sec.)	Added / Removed	AI value	Shortage	Travel time (sec.)	Added / Removed
75	2	light	509.52	504.08	519.95	500.87	17163	183	513.89	495.41	16623	222
		real	506.34	503.34	511.19	495.63	14004	207	514.48	496.27	16389	209
		heavy	586.02	575.05	616.49	599.94	14895	342	634.44	616.76	15912	316
	3	light	508.79	504.28	519.95	500.87	17163	183	513.47	495.43	16236	220
		real	506.10	503.76	511.88	495.05	15147	214	513.49	494.30	17271	212
		heavy	537.52	535.88	576.22	548.58	24876	442	604.91	578.47	23796	383
100	2	light	684.59	678.73	696.21	673.68	20277	212	-	-	-	-
		real	668.68	664.28	679.43	658.34	18981	230	730.94	729.09	1656	32
		heavy	798.16	783.62	825.72	809.64	14472	345	855.86	836.16	17757	303
	3	light	683.49	679.21	697.11	675.13	19782	201	702.50	675.56	24246	180
		real	668.33	665.07	677.11	654.24	20583	263	681.92	665.24	15012	218
		heavy	709.95	735.17	788.17	760.67	24750	479	-	-	-	-
125	2	light	855.77	849.66	873.81	851.33	20241	238	904.11	894.48	8667	134
		real	837.48	833.40	851.26	828.87	20142	250	913.09	911.38	1539	29
		heavy	1057.31	1050.49	1100.12	1082.39	15957	334	-	-	-	-
	3	light	850.22	849.03	872.28	836.71	32004	286	-	-	-	-
		real	837.12	831.63	850.04	824.50	22995	293	873.95	862.32	10476	164
		heavy	926.97	969.10	1026.84	999.95	24201	482	-	-	-	-
150	2	light	1043.80	1039.86	1071.10	1047.69	21069	224	-	-	-	-
		real	1022.81	1017.30	1034.66	1014.03	18567	224	-	-	-	-
		heavy	1284.75	1328.57	1380.39	1362.09	16470	324	-	-	-	-
	3	light	1005.51	1029.52	1055.03	1016.60	34587	315	-	-	-	-
		real	999.30	1015.46	1033.40	1000.11	29970	346	-	-	-	-
		heavy	1168.28	1207.27	1287.05	1261.24	23238	502	-	-	-	-
200	2	light	1400.12	1442.49	1467.71	1444.24	21123	235	-	-	-	-
		real	1410.73	1434.68	1452.10	1429.36	20457	255	-	-	-	-
		heavy	1950.69	2006.24	2043.04	2025.69	15615	341	-	-	-	-
	3	light	1374.46	1431.21	1456.80	1418.57	34407	325	-	-	-	-
		real	1355.83	1396.08	1428.88	1393.15	32157	347	-	-	-	-
		heavy	1802.71	1872.58	1942.73	1916.14	23931	496	-	-	-	-

Table 5 – Information on the numerical experiment results of each instance