The bike request scheduling problem

Kenneth Sörensen Nicholas Vergeylen

March 29, 2016

Abstract

In this paper we introduce the *bike request scheduling problem*, a new approach to city bike repositioning problems. The rationale behind this approach is explained, and a mixed-integer programming formulation is given. We prove that the bike request scheduling problem is NP-hard and formulate recommendations for future research.

1 Introduction: The bike request scheduling problem

Bicycle sharing systems (BSSs) are popping up in cities all over the world. These initiatives allow individuals to rent a bike from an automated rental station, use it for a short period of time, and return it to any other rental station in the city. A 2011 report of the Commission on Sustainable Development of the United Nations Department of Economic and Social Affairs (Midgley [2011]) put the number of BSSs at 375, using around 236,000 bikes, and there is very little doubt that these numbers have only increased since. Clearly, “bikesharing” has evolved from an interesting experiment to a viable addition to the modal mix of public passenger transport, even in large cities. For a historical perspective on BSSs, as well as some future trends we refer to DeMaio [2009].

Demand and supply at specific rental stations are rarely balanced, and fluctuations in supply and demand in the long or the short term might cause stations to fill up or deplete, preventing users from collecting or returning bikes. BSSs therefore typically use a fleet of light vehicles to transfer bikes between stations, attempting to rebalance the system. The vehicles are in constant contact with the dispatching station, where the current inventory of each station is monitored, and from where the repositioning activities are directed.

In the literature, several contributions have tackled the problem of determining the optimal routing of the repositioning vehicles. The family of related optimization problems solved in these papers is generally referred to as the *bike repositioning problem* (BRP). The *static* BRP (SBRP) is applicable when use of the BSS is negligible (i.e., at night). Examples of papers that tackle this problem are Benchimol et al. [2011], Chemla et al. [2012], Erdoğan et al. [2013], Raviv et al. [2012], Rainer-Harbach et al. [2014]. Other authors , like Contardo et al. [2012], Kloimllner et al. [2014], focus on the *dynamic* BRP (DBRP), in which demand and supply at each station during the day is taken into account. Their objective is to minimize the total unmet demand (which is expressed as the number of users who tried to collect bikes from empty stations or to bring back bikes to full stations).

A large majority of approaches in the literature combine two distinct problems into one single optimization problem: (1) deciding how many bikes to load or unload at the different stations throughout the day, and (2) the vehicle routing problem of the repositioning vehicles. This forces them to either model user demand and supply of bikes at the various stations as zero (the static BRP) or as a highly simplified process (the dynamic BRP). In reality, however, determining the expected demand and supply of bikes at each station and deriving from this information the number of bikes to load or unload at the different stations, as well as the best moment to do so, is a difficult problem that deserves attention in its own right. Ignoring the stochastic nature of this problem may lead to solutions that are not implementable. Moreover, it is unlikely that the process of determining the number of bikes to load and unload can and should be always fully automated: the complexity of the real-life situation will most likely call for some human interaction in the planning process. For these reasons, we propose an alternative modelling approach, which we call the *bike request scheduling problem* (BRSP) is a better alternative.

The remainder of this article is organized as follows. Our methodology for tackling repositioning problems is explained in section 2. A mathematical model of the BRSP is presented in section 3. In section 4 a proof is given that the BRSP is NP-hard. Finally, conclusions and recommendations for further work are given in section 5.

2 Methodology

We propose a novel approach that tackles the bicycle repositioning problem by decomposing it into two distinct subproblems: (1) the generation of loading or unloading *requests* (essentially an order to pick up or drop off a certain number of bikes at a certain station within a certain time window), and (2) the (dynamic) assignment of these requests to vehicles and the scheduling of requests within each vehicle. In this contribution we focus on the latter problem, which has been called the *bike request scheduling problem* (BRSP). This approach differs from the traditional approaches (using the BRP) in that it explicitly separates the process of determining the demand and supply of bikes at the different stations from the problem of routing the repositioning vehicles. The objective of the BRSP is to determine the assignment and sequencing of requests to vehicles that minimizes the priority-weighted number of unscheduled requests, subject to the time windows of the requests, as well as the capacities of the replenishment vehicles.

Decomposing the problem of request generation and the problem of scheduling those requests into two distinct subproblems requires that an information

Table 1: Attributes of a request

Name	**Explanation**
Issue time	The time at which the request is issued
Type	Pick-up (load) or deliver (unload)
Quantity	Number of bikes to load or unload
Station	Identity of the station at which to load or unload bikes
Priority	A weight that indicates the relative importance of the request
Earliest time	Start of the time window during which the request can be fulfilled
Latest time	Expiration time of the request
Drop time	Time required to execute the request

exchange protocol is established between both subproblems. At the core of the boundary are the so-called *requests*. They encapsulate the necessary information for the scheduling algorithm to create a set of vehicle routes such that preferably all requests are handled. The attributes of a request are defined in Table 1.

All common taxonomies of vehicle routing problems distinguish between static and dynamic vehicle routing problems (Pillac et al. [2013]), which differ in the availability of all information at the start of the planning period (static problems) or not (dynamic problems). Both static and dynamic variants of the BRSP can be defined through the issue time of the requests. The static BRSP will be agnostic of future requests, which is equivalent to all requests having an issue time of zero (assuming that the planning period starts at time zero or after). The dynamic BRSP can introduce positive issue times that fall within the interval defined by the start and end of the planning period. The rest of this paper will discuss the static BRSP, although dynamic variants of the BRSP will be investigated in future research.

3 Problem definition and mathematical model

Given a set of requests, the objective of the (static) BRSP is to minimize the priority-weighted number of unscheduled requests. To this end, requests can be scheduled on a (given) set of vehicles, each having identical capacity. Executing a request requires a vehicle to visit the station where this request occurs. Travel times between stations are assumed to be known. Each request has a time window (earliest and latest time), and the request can only be executed during this time. Executing a request takes a certain amount of time, which needs to be spent before the vehicle can start traveling towards the station at which its next request occurs. Vehicles are allowed to wait before starting service if they arrive before the earliest time of the request. It is assumed that a vehicle can always start executing their first request at its earliest time and return to the depot after the latest time of their last request. This makes it unnecessary to model a depot in the mathematical model. Each request requires a number of

Table 2: Parameters of the static BRSP

I	set of requests
N	set of positions within a vehicle tour
K	set of vehicles
b_i	number of bikes picked up or delivered at request i ($> 0 \rightarrow$ delivery, $< 0 \rightarrow$ pick-up)
w_i	weight of request i
o_i	working time at request i
a_i^e	earliest arrival time at request i
a_i^l	latest arrival time at request i
t_{ij}	travel time from request i to request j
C	capacity of the vehicles

Table 3: Decision variables of the static BRSP

x_{in}^k	1 if request i is served as the n-th request of vehicle k, 0 otherwise
y_i	1 if request i is not served, 0 otherwise
a_i	arrival time at request i
z_{ij}	1 if request j is visited immediately after request i by the same vehicle

bikes to be either picked up or dropped off. The number of bikes on a vehicle after each pick-up or drop-off can never fall below zero or exceed the vehicle's capacity. Each request has a priority, and the priority of all requests that have not been assigned to a vehicle, are added to the objective function.

In this section a mixed-integer programming model is developed for the (static) BRSP. The parameters used in the model are shown in Table 2. The decision variables can be found in Table 3.

Using this notation, the problem can be written as follows.

$$\min \sum_i y_i w_i \tag{1}$$

s.t.

$$\sum_k \sum_n x_{in}^k + y_i = 1 \qquad \forall i \in I \tag{2}$$

$$z_{ij} \geq x_{jn+1}^k + x_{in}^k - 1 \qquad \forall i, j \in I, k \in K, n \in N \tag{3}$$

$$\sum_i x_{i(n+1)}^k \leq \sum_i x_{in}^k \qquad \forall k \in K, n \in N \tag{4}$$

$$\sum_i x_{in}^k \leq 1 \qquad \forall n \in N, k \in K \tag{5}$$

$$\sum_i \sum_{n' \leq n} x_{in'}^k b_i \leq C \qquad \forall n \in N, k \in K \tag{6}$$

$$\sum_i \sum_{n' \leq n} x^k_{in'} b_i \geq 0 \qquad \forall n \in N, k \in K \qquad (7)$$

$$a_j \geq a_i + o_i + t_{ij} - (1 - z_{ij})M \qquad \forall i, j \in I \qquad (8)$$

$$a^e_i \leq a_i \leq a^l_i \qquad \forall i \in I \qquad (9)$$

$$x^k_{in}, z_{ij}, y_i \in (0, 1) \qquad \forall i \in I, k \in K, n \in N \qquad (10)$$

$$a_i \geq 0 \qquad \forall i \in I \qquad (11)$$

The objective function (1) minimizes the total weight of all unscheduled requests. Constraints (2) ensure that each request is not served more than once. Constraints (3) link the decision variables z_{ij} and x^k_{in} so that z_{ij} is forced to be equal to 1 if a vehicle serves request j immediately after request i. Constraints (4) ensure that a contiguous set of adjacent positions in the vehicle is used, starting from the first position. Constraints (5) force each position in the vehicle to be used only once. For a vehicle moving between requests i and j, constraints (8) ensure that the vehicle can only serve request j after its arrival time at i plus its drop time at i plus the time it takes to travel between i and j. Constraints (6) and (7) ensure that the number of bikes on a vehicle never exceed the vehicles' capacity or goes below zero respectively. Constraints (9) force the arrival time at request i to be between its earliest and latest arrival time. Finally, constraints (10) to (11) define the domains of the decision variables.

4 NP hardness of the BRSP

In this section we show that the BRSP is NP-hard by demonstrating that a subset of BRSP instances are knapsack problems which are studied in Martello and Toth [1990] and Dasgupta et al. [2008]. A subset of BRSP instances can be transformed to 0-1 knapsack problems and any 0-1 knapsack problem can be transformed into a BRSP instance. Solving the BRSP implies solving the knapsack problem. Therefore the BRSP is at least as hard as the knapsack problem, or, in other words, the BRSP is NP-hard.

Instances of the BRSP that can be modeled as knapsack problems are those where (1) only one vehicle is used, (2) every request has a positive quantity of bicycles, and (3) time windows are non-constraining. The latter will be the case when all early time limits are zero and all late time limits are at least as late as the length of the longest vehicle trip, which we denote T_m. This means the time windows, as enforced by constraints 3 and 8 - 9 are always satisfied. To see this, notice the arrival time for any request in any instance will necessarily be between these bounds:

$$a^e_i = 0 \leq a_i < T_m = a^l_i, \forall i \in I \qquad (12)$$

This means we can leave those constraints explicitly out of consideration when solving the subproblem.

All b_i are positive. This means we can drop constraints 7 explicitly as well. It also means that vehicle load will increase monotonously with the request sequence in constraints 6:

$$\sum_i \sum_n x_{in} b_i = \sum_i \sum_{w=1}^{n'} x_{iw} b_i + \sum_i \sum_{w=n'+1}^{n} x_{in} b_i \leq C, \quad \forall n \in N, 1 \leq n' < n$$

So

$$\sum_i \sum_n x_{in} b_i \leq C \Rightarrow \sum_i \sum_{w=1}^{n'} x_{iw} b_i \leq C, \qquad \forall n \in N, 1 \leq n' < n$$

This means we can only consider the constraint where n is the position of the last scheduled request. For a certain schedule, we can permute the requests without changing the left hand side of the constraint. This means the order of the schedule becomes irrelevant in terms of the capacity constraints and the time window constraints. It was already irrelevant in terms of the objective function. Therefore we can now remove order information from the model together with constraints (4) and (5) as they become redundant.

The remaining model minimizes the priority-weighted number of unscheduled requests. The decision variables x_i are the complement of the y_i variables: $x_i + y_i = 1, \forall i \in I$. This means we can further simplify the model by removing y_i and expressing the model only in terms of x_i. This removes the need to express constraints (2) explicitly, but changes the sense of the objective function. The resulting model after applying the simplifications is given by:

$$\max \quad P_i \cdot x_i \tag{13}$$

s.t.

$$\sum_i x_i b_i \leq C \tag{14}$$

$$x_i \in \{0, 1\} \qquad \forall i \tag{15}$$

This is exactly the formulation of the *0-1 knapsack problem* with profits P_i and weigths b_i. We can therefore conclude that the highly restricted case of the BRSP we consider here is identical to the knapsack problem.

On the other hand, every knapsack problem can be transformed to a special case of the BRSP. This can be done as follows. Given a set I of items, each with a profit P'_i and a weight b'_i, a request is created with the following attributes:

- $b_i = b'_i, \forall i \in I$
- $w_i = P'_i, \forall i \in I$

Further, a set of stations should exists, to which requests may be assigned in any possible way. Travel times between stations are irrelevant, as are working times for each requests. Further, a single vehicle is defined with the same capacity as the knapsack. Time windows for each request are set to zero for the earliest time and an arbitrarily large number of the latest time. The optimal solution of the BRSP with these characteristics is the same as the optimal solution of the original knapsack problem.

We set up experiments to verify how the model scales when implemented in a general purpose solver (Gurobi). Without going into much detail, we report that the presented model is prone to combinatorial explosion, even for small instances. When enforcing a time limit of 3600 seconds, we see that the number of time constraint violations increases exponentially ranging from 3 for 9 requests to 189 for 12 requests. Notice that an instance of 12 requests is extremely small when looking at real life situations where we have more than 100 stations in networks of medium size.

5 Conclusion

In this contribution we have described the problem of bicycle repositioning and highlighted the problems of the approaches that exist today. We have proposed a new approach to tackle this problem, that separates the process of determining the number of bikes and the time window within which to pick them up or deliver them at each station, and the vehicle routing of the repositioning vehicles. The concept of a *request* was introduced as the core communication between both subproblems. We have proven that the static version of the bike repositioning problem is NP-hard and developed a mixed-integer programming model.

One possible avenue for future research is an improvement of the exact model, including strategies to reduce computational effort such as introducing problem-specific cutting planes, an intelligent column generation strategy, or the use of lazy constraints. The aim of this line of research would be to solve realistic instances in a more acceptable time limit. The empirical analysis presented in this contribution could then be extended to these cases, yielding results of more practical significance.

Another path for further research will focus on the development of heuristic solution algorithms to obtain a better ratio of resource consumption to objective value. In practice, for example, the unpredictable calculation time of the exact model will be unacceptable, especially in a dynamic situations where the changing list of requests requires a constant re-evaluation of the current solution. Heuristics are the only alternative in this case. This research path could also focus on matheuristics, trying to combine the best from the exact models and heuristics.

Future research also will consider the request generation algorithm. Requests should be generated to reflect patterns found in user behavior. Modeling usage behavior could be of great value for the research on the BRSP as well, as it will allow to generate more realistic test cases compared to the extended Solomon cases used in this research.

References

M. Benchimol, P. Benchimol, B. Chappert, A.D.L. Taille, F. Laroche, F. Meunier, and L. Robinet. balancing the stations of a self service bike hire system. *RAIRO - Operations Research*, 45(1):37–61, 2011.

D. Chemla, F. Meunier, and R. Wolfler Calvo. Bike sharing system: solving the static rebalancing problem. *Discrete Optimization*, 2012. Accepted for publication in Discrete Optimization.

C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. *Centre Interuniversitaire de Recherche sur les Réseaux d'entreprise, la Logistique et le Transport*, 2012.

S. Dasgupta, C. H. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2008.

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

G. Erdoğan, G. Laporte, and R. Wolfler Calvo. The one-commodity pickup and delivery traveling salesman problem with demand intervals. Submitted for publication to Transportation Science, 2013.

C. Kloimllner, P. Papazek, B. Hu, , and G.R. Raidl. Balancing bicycle sharing systems: An approach for the dynamic case. In C. Blum and G. Ochoa, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 8600 of *Lecture Notes in Computer Science*, pages 73–84, Berlin Heidelberg, 2014. Springer.

S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

P. Midgley. Bicycle sharing schemes: enhancing sustainable mobility in urban areas. Background Paper CSD19/2011/BP 8, United Nations Department of Economic and Social Affairs, Commission on Sustainable Development, 2011.

V. Pillac, M. Gendreau, C. Guret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 255(1): 1–11, 2013.

M. Rainer-Harbach, P. Papazek, G.R. Raidl, B. Hu, and C. Kloimllner. Pilot, grasp, and vns approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, pages 1–33, 2014.

T. Raviv, M. Tzur, and I.A. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, pages 1–43, 2012.