1 Introduction

Bike sharing systems are a very popular means to provide bikes to citizens in a simple and cheap way. The idea is to install bike stations at various points in the city, from which a registered user can easily loan a bike by removing it from a specialized rack. After the ride, the user may return the bike at any station (if there is a free rack). Services of this kind are mainly public or semi-public, often aimed at increasing the attractiveness of non-motorized means of transportation, and are usually free, or almost free, of charge for the users.

Depending on their location, bike stations have specific patterns regarding when they are empty or full. For instance, in cities where most jobs are
located near the city centre, the commuters cause certain peaks in the mor-
ing: the central bike stations are filled, while the stations in the outskirts are
empty. Furthermore, stations located on top of a hill are more likely to be
empty, since users are less keen on cycling uphill to return the bike, and often
leave their bike at a more reachable station. These issues result in substan-
tial user dissatisfaction which may eventually cause the users to abandon
the service. This is why nowadays most bike sharing system providers take
measures to rebalance them. Balancing a bike sharing system is typically
done by employing a fleet of trucks that move bikes overnight between unbal-
canced stations. More specifically, each truck starts from a depot and travels
from station to station in a tour, executing loading instructions (adding or
removing bikes) at each stop. After servicing the last station, each truck
returns to the depot.

Over the last few years, balancing bike sharing systems (BBSS) has
become increasingly studied in optimization \[1\, 3\, 7\, 2\, 8\, 5\, 4\, 6\]. As such,
generating meaningful instance to serve as a benchmark for the proposed
approaches is an important task. In this technical report we describe the
procedure we used to generate BBSS problem instances from data of the
CitiBike NYC bike sharing system.

2 Instance format

We employ the instance format defined and popularized by the ADS group
at the Technische Universität Wien (TU Wien) and the mobility department
at Austrian Institute of Technology (AIT). Note, however, that our scope is
limited to the static variant of the problem, as such we only consider a valid
subset of the instance format.

The format for the static BBSS specifies, for each station \( s \in S \)

- the current number of bikes \( b_s \),
- the target number of bikes \( \hat{b}_s \),
- the distance from the depot \( d_{s,d} \), and
- the distance from each other station \( k d_{s,k} \).

Note that this format only describes a state of the bike sharing system,
therefore, in a sense, it describes a family of instances. Specific instances
can be generated by specifying

- the stations capacities \( C_s \),
- the vehicles capacities \( c_v, v \in V \),
- the number of vehicles from the depot \( V \), and
- the vehicles time budget \( \hat{t}_v \).

\[1\] We refer the reader to https://www.ads.tuwien.ac.at/w/Research/Problem_Instances
for a complete description of the format and all its variants, e.g., the dynamic formulation.
3 Data collection

The first step of instance generation, is gathering a sufficient amount of usage data about a bike sharing system. Our choice system of choice is CitiBike NYC\textsuperscript{2}, since they provide full access, through a web service, to the state of the network in JSON format at any time\textsuperscript{3}. This is an example of output from the web service

\begin{verbatim}
{
  "executionTime": "2013-11-04 12:09:01 AM",
  "stationBeanList": [
    {
      "availableDocks": 21,
      "totalDocks": 39,
      "longitude": -73.99392888,
      "testStation": false,
      "stAddress1": "W 52 St & 11 Ave",
      "stationName": "W 52 St & 11 Ave",
      "landMark": "",
      "latitude": 40.76727216,
      "statusKey": 1,
      "availableBikes": 17,
      ...
      "id": 72
    },
    ...
  ]
}
\end{verbatim}

From the output, it can be noted that three pieces of information about capacity are reported: the \texttt{availableDocks}, \texttt{totalDocks}, and the \texttt{availableBikes}. One interesting thing is that

\[ \text{totalDocks} \neq \text{availableDocks} + \text{availableBikes} \tag{1} \]

i.e., there is a displacement of one bike, which is, however, constant through the stations. Moreover, a field \texttt{statusKey} encodes the status of the station, e.g., operational or non-operational.

We have stored a snapshot of the network every 10 minutes for 6 months (since May 2013 to November 2013). A snapshot, among other information, contains the current and the maximum number of bikes of each station, and its address. This data was necessary in order to provide realistic \( b_s \) and \( \hat{b}_s \) for every station. Moreover, we employed the address data to query the Google

\begin{itemize}
  \item \textsuperscript{2}CitiBike NYC: \url{http://www.citibikenyc.com}
  \item \textsuperscript{3}Web service URL: \url{http://citibikenyc.com/stations/json}
\end{itemize}
Maps API[^1] about the distance, both in minutes and meters, between each pair of stations \( s, k \in S \). Note that this also includes the distances from a depot, as we consider one of the central stations as a depot.

4 Data processing

We considered the \( \approx 25'000 \) snapshots, and, for each station, we computed the distributions of stored bikes at every hour of the day. The week-ends were not considered because the bike usage is much noisier than during working days. From the analysis it is clear that, at certain times of the day, there are stations acting as sources (see Figure 1) and others acting as sinks (see Figure 2). The following box plots, that show the distributions of bikes on a single station throughout the day show this behavior.

![Figure 1: Example of source station](image)

From some of the distributions, it was also clear that there was some artificial rebalancing happening overnight between 00 : 00 and 06 : 00 AM (mildly visible in Figure 3 at 04 : 00 AM).

For each station \( s \in S \), we first computed the 1\(^{st}\) and 3\(^{rd}\) quartiles for all the hours of the day. Then we found the minimum first quartile and the maximum third quartile across the day, which we denote, respectively, by \( \text{min}_s \) and \( \text{max}_s \). Ideally, these are values which we would like to be as far as possible, respectively from 0 (empty station) and from \( C_s \) (full station). We thus computed a displacement value for each station \( s \), as

\[
disp_s = \lfloor C_s - (C_s - (\text{max}_s - \text{min}_s))/2 \rfloor - \text{max}_s
\]

[^1]: Google Maps API: [https://developers.google.com/maps](https://developers.google.com/maps)
ideally, displacing the distributions by $\text{disp}_s$ brings $\text{min}_s$ and $\text{max}_s$ as far as possible from 0 and $C_s$, so that the probability of finding an empty or a full station is minimized.

5 Instance generation

Once the displacement of each station is known, generating an instance from a snapshot is rather easy. But there are some aspects which one should take into account.
Selection of the snapshot. In order for the generated instances to be realistic, one should consider when the rebalancing is likely to be done. A good guess for this is that the rebalancing happens overnight. This is also supported by the rebalancing step which is visible on some stations (e.g., Figure 3). For our instances, we have chosen midnight as the expected time for the start of the rebalancing, thus we have used the 30 midnight snapshots from September 2013 as starting points. For each station \( s \in \mathcal{S} \), the initial number of bikes \( b_s \) is thus the actual number of bikes in the station at midnight.

Selection of the depot. The information released by CitiBike NYC does not contain any data about depots. Of course, this station must be excluded from the choice of the other stations to include in the instance. We selected the station with CitiBike ID 294 as it was quite central with respect to all the other stations.

Selection of stations. Our generator accepts a size parameter that controls the number of stations that are included in the instance. The stations are then partitioned in sinks and sources and a random station from each set is added uniformly at random to the instance. This strategy tries to balance sinks and sources, so that the objective function range is broader (and the generated instances are more interesting). Note that, because of the randomness in the generation process, the generated instance use, in principle, a different set of stations. However, since the number of stations in the CitiBike NYC system is limited (\( \approx 330 \)) the larger instances are more likely to include similar sets of stations.

6 Generator and instances

We have built an instance generator for BBSS based on the ideas described in the previous section. The generator is available under the permissive MIT license at the address https://bitbucket.org/tunnuz/citibike-nyc-generator. Moreover, we have generated 180 instances of increasing size \( \in \{40, 80, \ldots, 240\} \), which are publicly available as well, under the same license at the address https://bitbucket.org/tunnuz/citibike-nyc-sept-13. Note that, unlike the instances available from the ADS and AIT websites, the one generated by our software do not already consider the bicycle loading / unloading times inside the traveling times. The distances are thus real distances, and the loading and unloading times must be added to the cost function. This allows to implement various policies, e.g., fixed loading / unloading times, or loading / unloading time dependent on the number of transferred bicycles.
References

[1] Mike Benchimol, Pascal Benchimol, Benoît Chappert, Arnaud De la Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. Balancing the stations of a self service bike hire system. *RAIRO – Operations Research*, 45(1):37–61, 2011.

[2] Daniel Chemla, Frédéric Meunier, and Roberto Wolff Calvo. Bike sharing systems: Solving the static rebalancing problem. *To appear in Discrete Optimization*, 2012.

[3] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. Balancing a Dynamic Public Bike-Sharing System. Technical Report CIRRELT-2012-09, CIRRELT, Montreal, Canada, 2012.

[4] Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. Constraint-based approaches for balancing bike sharing systems. In *CP’13 - The 19th International Conference on Principles and Practice of Constraint Programming*, pages 758–773. Springer Berlin Heidelberg, 2013.

[5] Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. A hybrid aco+cp for balancing bicycle sharing systems. In *HM’13 - The 8th International Workshop on Hybrid Metaheuristics*, pages 198–212. Springer Berlin Heidelberg, 2013.

[6] Marian Rainer-Harbach, Petrina Papazek, Bin Hu, and Günther R. Raidl. Balancing bicycle sharing systems: A variable neighborhood search approach. In Martin Middendorf and Christian Blum, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 7832 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin Heidelberg, 2013.

[7] Tal Raviv, Michal Tzur, and Iris A. Forma. Static repositioning in a bike-sharing system: Models and solution approaches. *To appear in EURO Journal on Transportation and Logistics*, 2012.

[8] Jasper Schuijbroek, Robert Hampshire, and Willem-Jan van Hoeve. Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems. Technical Report 2013-E1, Tepper School of Business, Carnegie Mellon University, 2013.