# Exploring the use of hyper-heuristics to solve a dynamic bike-sharing problem

Emily Myers, Ghofran Shaker

July 21, 2015

**Abstract**

In a self-service bike-share system we have a finite number of stations with limited capacity. In any real-world situation, some stations will have more demand than others and will accumulate or lose bikes. The system operator must use vehicles to pick up the bikes from full stations and redistribute them in empty ones. We define the problem and explore a hyper-heuristic approach previously used successfully to solve the Vehicle Routing Problem.

## 1 Introduction

The implementation of public bike-sharing systems has had a significant increase worldwide in the recent years. As of 2014, they are available in more than 700 cities including Barcelona, Montreal, New York City, and Paris and are also currently being introduced in a number of cities like Pittsburgh. Bike-sharing systems amplify public transportation by offering people an affordable, convenient way to move around large crowded cities without having to worry about motorized traffic or parking. Furthermore, they boost environmental sustainability and public health.

Typically, a bike-sharing system consists of multiple stations distributed over favorable spots in the service area. A user may rent a bike from any station and return it later to any other station. Due to demographic characteristics and usage patterns throughout the day, many stations tend to overload with bikes whereas other stations will run completely empty. An overloaded station will disable any user from delivering a bike back and an empty station will disable any user from picking up a bike. In both cases, this will result in user dissatisfaction and imbalance in the whole bike system. Therefore, a balancing procedure becomes necessary, and action must be taken by the system operator to satisfy the users needs. The system operator sends out a fleet of vehicles that visit each station, picking bikes up from overloaded stations and delivering them to empty stations to balance the system. This can be done either statically or dynamically. In the former, the procedure is done overnight when the system is not in use and user demand is negligible. In the latter case, the procedure is done during service hours, where user demand is constantly changing.

Many have addressed the static case of the bike-sharing problem; however, most research in the dynamic case is relatively recent. Our work explores a new approach to solving the dynamic public bike-sharing system by adapting a dynamic vehicle routing problem hyper-heuristic. First, we discuss methods previously used to approach the problem. Then, we explore the hyper-heuristic method. Finally, we attempt to apply the aforementioned method to the bike-sharing problem.

## 2 Previous work

Given the extremely short time frame afforded us, we knew that we wanted to pursue an alternate approach of an existing problem. However, as we read the many papers already written about the bike-share problem, we noticed a few distinct trends: most involved a mixed integer problem (MIP), metaheuristics, or some combination of the two. We felt that these approaches had been exhausted and wanted to pursue something more unique.

### 2.1 Vehicle routing problem

In our reading of past work and related problems–in particular the vehicle routing problem– we came across an evolutionary-based hyper heuristic approach called the EH-DVRP proposed by Garrido and Riff in their paper, DVRP: a hard dynamic combinatorial optimization problem tackled by an evolutionary hyper-heuristic [2]. The EH-DVRP solves hard instances for the dynamic vehicle routing problem, and in our work we explore its application to the dynamic bike-sharing problem.

The VRP is very similar to the bike-sharing problem in that there is a fleet of vehicles making pickups and deliveries to a set of customers. This makes it incredibly easy to adapt the bike-sharing problem into a VRP. For example, in the dynamic VRP we can have new customers appear. Translated to the bike-sharing problem, each 'customer' is now a bike station, and the updated customer list now represents the updated demand of the corresponding station.

### 2.2 Bike-sharing instances

Though ideally it would be possible to apply the hyper-heuristic method to a real-world situation, perhaps even the bike-sharing system in Pittsburgh, the systems in major cities that we have data for all have upwards of 100 stations, and we wanted a smaller instance that would be easier to parse. It was for this reason that we used the generated instances given by Contardo, Rousseau, and Morency in their paper "Balancing a dynamic public bike-sharing system" [1].

## 3 Problem description

Before reading the problem description, let's take a brief look at the static bike-sharing problem, which we've adapted heavily to fit a VRP instance. Our bike-sharing problem can be modeled as a complete weighted digraph $G = (V, E)$ with $V = \{v_0, v_1, ..., v_n\}$ as the set of bike stations where $v_0$ is the central depot and $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ as the arcs between pairs of stations. These arcs can be represented by the cost matrix $C = (c_{ij})(c_{ij} \in \mathbb{R}^+)$ where $c_{ij}$ is just the distance between stations $i$ and $j$. In the vehicle depot $v_0$ there are $m$ vehicles with capacities $Q_1, ..., Q_m$ which start their routes $R_0, ..., R_m$ from $v_0$. The remaining nodes $v_1, ..., v_n$ represent the stations that either give or receive $q_1, ..., q_n$ bikes. We want to find feasible vehicle routes to pick up and deliver bikes. A solution is feasible if and only if it satisfies these constraints:

- Each vehicle route starts and ends at the depot $v_0$.
- The length of each route does not exceed an upper bound $D$, the length of the working day.
- Each station $v \in V - \{v_0\}$ is visited exactly once and the service time (loading/unloading the bikes) is bounded between $[0, D]$.
- The total demand of each route cannot exceed truck capacity $Q$.

The optimal solution for this problem minimizes total vehicle travel time.
In the dynamic case, the initial problem can change as stations become full and empty throughout the day, while the vehicles are on their routes. This will be represented as 'new' stations $v_1', ..., v_n'$. The problem then acquires new constraints:

- 'New' stations $v_1', ..., v_n'$
- Stations $v$ with a bike imbalance from the end of the previous work day
- New imbalances can be found
- We don't know how many bikes will be delivered/picked up at each 'new' station $v_k'$

Using our adapted VRP and and the new constraints from the dynamic version, we can express our dynamic bike-sharing problem as a DVRP:

*-Definitions:*

- $t$: elapsed service time such that $t \in [0, D]$
- $m$: number of vehicles
- $V'$: unserved stations

- ${v_s}^k$: last station served by vehicle
- ${S_k}^t$: sequence of stations ${v_0}^k, ..., {v_l}^k$ served by the $k$-th vehicle before $t$

-*Constants:*

- $C_{il}$: travel distance from $v_i$ to $v_l$ where $v_i \in V' \cup \{{v_s}^l, ..., {v_s}^m\}$
- $q_l$: bikes delivered/picked up at station $v_l$
- ${Q_t}^k$: capacity of vehicle $k$ at $t$
- ${D_t}^k$: remaining time of vehicle at time $t$ before reaching $D$

-*Variables:*

$$x_{lk} = \begin{cases} 1, & \text{if vehicle } k \text{ serves station } v_l : v_l \in V' \cup \{{v_s}^k\} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ilk} = \begin{cases} 1, & \text{if vehicle goes directly from station } v_i \text{ to } v_l, v_i \in V' \cup \{{v_s}^k\} \text{ and } v_l \in V' \\ 0, & \text{otherwise} \end{cases}$$

-*Objective function:*

$$\min \sum_{ilk, i \neq l} c_{il} * y_{ilk}$$

-*Constraints:*

$$\sum_{k=1}^{m} x_{lk} = 1, \forall v_l \in V', v_l \neq v_0 \tag{1}$$

$$x_{0k} = \begin{cases} 1, & \forall k = 1...m | {S_k}^0 = \emptyset \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\sum_{l} x_{lk} * q_l \leq {Q^t}_k, \forall k = 1...m, \forall v_l \in V' \cup \{{v^k}_s\} \tag{3}$$

$$\sum_{i} y_{ilk} + \sum_{j} y_{ljk} = x_{lk}, = 1...m, \forall v_l \in V' \cup \{{v_s}^l ... {v_s}^m\} \tag{4}$$

$$\sum_{i,l} c_{i,l} * y_{ilk} \leq {D_k}^t, \forall k = 1...m \tag{5}$$

$$\sum_{i,l} y_{ilk} \leq |S|, \begin{array}{c} S \subseteq V' \cup \{{v_s}^1, ...{v_s}^m\} \\ 2 \leq |S| \leq n-1 \\ \forall k = 1...m \end{array} \tag{6}$$

(1) Limits the service of each remaining station to one visit from a single vehicle. (2) means that all vehicles start their journey at the depot $v_0$. The third constraint represents the remaining capacity for each vehicle, updated over every time window. (4) means that a vehicle $k$ serves a station $v_l$ if and only if it belongs to that vehicle's particular schedule. (5) Means that to finish the route before $D$, travel times of each vehicle $k$ are limited to ${D_k}^t$. (6) establishes the maximum number of stations that can be served by vehicle $k$.

## 4 The hyper-heuristic approach

### 4.1 Definition of hyper-heuristic

A hyper-heuristic is a high-level method used to solve hard computational search problems. Heuristics are commonly used to obtain a good enough solution in reasonable time when the optimization problem at hand is intractable. Because heuristics are problem-specific, this leads us to the use of metaheuristics, which are more general and can find sufficient solutions to a wide range of optimization problems without having to deeply adapt to each problem. Metaheuristics are recognized as efficient approaches for many optimization problems. However, research and experiments reveal that certain metaheuristics perform better for certain problems. In the same problem, different metaheuristics perform better for different instances. Furthermore, for that same instance, different metaheuristics perform better at different stages of the solution process. This introduces us to hyper-heuristics, where we can generate the sequence of metaheuristics that best fits the problem at hand. The term hyper-heuristic is relatively new and the idea is to construct an algorithm that combines different metaheuristics in a way such that each would compensate, to some extent, for the weaknesses of others. Where metaheuristics have a search space of solutions, a hyper-heuristic has a search space of heuristics. So rather than trying to solve the problem directly, we attempt to find the right method or sequence of heuristics in a given situation. Thus, a hyper-heuristic is thought of as a heuristic to choose heuristics.

### 4.2 Why we chose the hyper-heuristic method

The public bike-sharing problem has been tackled by several approaches as mentioned in the previous section. The static case of the problem has been exhausted with different strategies, with the most popular being Mixed Integer Programming, Constraint Programming techniques, and Metaheuristics. On the other hand, the much harder dynamic case has had only few approaches so far. To the best of our knowledge, hyper-heuristics have never been used in either case to solve the public bike sharing system. Thus, a hyper-heuristic approach to the public bike-sharing system is indeed novel.

In addition to its novelty, we feel the hyper-heuristic approach is very appropriate for the dynamic case of the public bike-sharing system: Using a constantly evolving genetic program is ideal for a dynamic problem that is always changing. In the dynamic case, throughout the service hours of the system, users are concurrently picking up and delivering bikes to stations. So for one instance of the problem, demand will regularly change at different points in time and therefore a route assumed to be optimal in one point in time might not be in the next. Due to this dynamic behavior, certain metaheuristics may perform better than others at different points in time and different stages of the solution process. A hyper-heuristic comes in handy here because it is dynamic in nature, it can easily adapt to changing scenarios like the dynamic bike-sharing system.

## 4.3 The EH-DVRP

Before we apply the evolutionary based hyper-heuristic approach to the public bike-sharing problem, let us first explain how it works with the Vehicle Routing Problem(VRP) as demonstrated in Garrido.

The hyper heuristic framework proposed by Garrido includes three classes of low-level heuristics that were meticulously selected based on design considerations. These three classes are constructive, repairing and ordering heuristics. The four constructive heuristics included in this approach were chosen based on their ability to communicate and construct feasible solutions together and tackle the problem with different service area distribution. Likewise, the repairing heuristics incorporated were chosen based on their collaboration but also on the type of repairing mechanism such that it allows the discovery of new potential insertions. Similar considerations were taken when selecting the ordering heuristics, as they are used prior the construction step to arrange the customers insertion list. Because there exists several implementation differences between the heuristics, Garrido and Riff define the following two main data structures: a list of new unassigned customers and a set of routes. Since all heuristics can discern partial states of the problem using these basic structures, this will enhance cooperation among all heuristics in the framework.

In this framework, the hyper-heuristic will generate a sophisticated sequence of low-level heuristics where this sequence is represented as a chromosome and each low-level heuristic is represented as a gene. Thus, a chromosome is made of a sequence of genes and is regarded as an individual. Each gene has a constructive and repairing heuristic as its two main components. The constructive heuristic constructs and optimizes partial routes and consists of parameters, which are the ordering heuristic (O), constructive-improvement heuristic (CI) and the number of customers to be inserted. On the other hand, the repairing heuristic improves all the routes constructed so far. The genes are applied sequentially to gradually insert customers and repair the set of routes, improving partial states of the problem.

The approach uses the traditional fitness function to find and preserve the best individuals through the search. New individuals are found by applying four operators, three of which are mutation-like operators: the asexual operator, the add operator and the replace operator. The fourth is the cross operator, which is a recombination operator. The goal of these operators is to increase exploration capability, include new heuristics in different steps to obtain better cooperation, and allow the possibility of eliminating heuristics that wont contribute to improving candidate solutions.

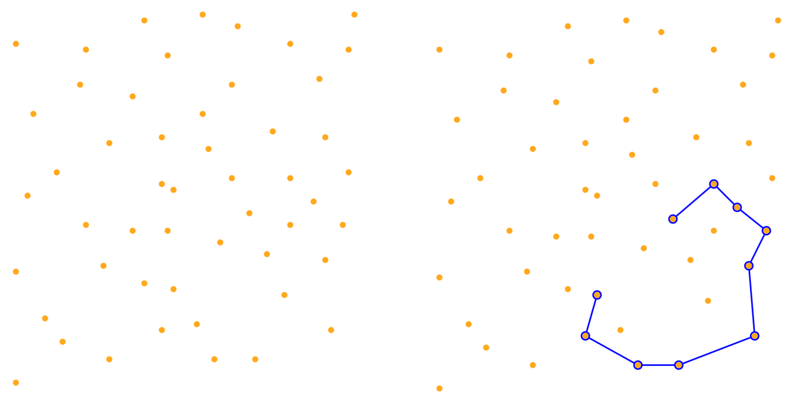
The procedure of the EH-DVRP is as follows: An initial population P is generated to include all combinations of heuristics at random positions. The hyper-heuristic explores the heuristics space starting from the initial population. At each generation, one operator is applied with equal probability to allow the production of new offspring O to the current population.

Garridos and Riffs approach is designed to perform three tasks: obtain new orders that appear during service hours, optimize and update routes, and send vehicles to serve customers. This is all done within a strategic system that consists of two main components to handle these tasks: the Event Manager and the Hyper-heuristic. The Event Manager serves as an interface between the real world and the optimization process that is done by the Hyper-heuristic. It is important to mention that the working day is divided into discrete time slices of equal length to deal with the dynamism of the problem. During the working day, the Event Manager collects new orders, stores them in the system and then creates a corresponding static VRP-like instance at each time slot. Based on this information, the Hyper-heuristic sets out to find the best sequence of heuristics. The Event Manager then creates the new optimized solutions and commits orders to the respective vehicles. Finally, it updates the positions of the customers and the states of the vehicles.

## 5 Experimental Results

### 5.1 Reproducing dynamic VRP results

In our attempt to apply the EH-DVRP to the dynamic bike-sharing system, we first set out to reproduce Garridos results. We run an instance of 50 randomly distributed stations with 50 available vehicles. The results are shown below, depicting the stages of the optimized route solutions for 5 vehicles.



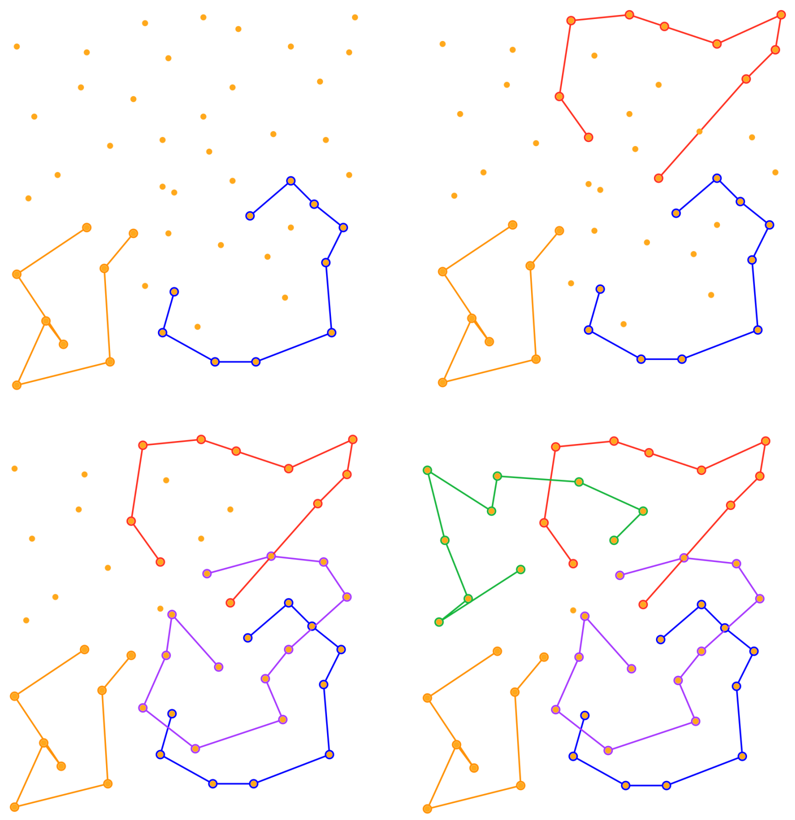


*Figure 1.* The EH-DVRP applied to a Garrido instance. The different colors represent different routes.

As can be seen, the EH-DVRP is able to obtain sufficient routes for the dynamic VRP. Compared with low-level heuristics and MIPs, this approach is better able to adapt to dynamic problems and is flexible enough to allow for additional heuristics and constraints. Thus, the performance of the EH-DVRP on the VRP is proven to be quite successful and certainly outperforms other heuristic-based methods and state-of-the-art algorithms in several cases.

## 5.2 Application to dynamic bike-sharing system

We consider instances from Contardo, one of the first works to address the dynamic case of the bike-sharing system. Contardo generated many instances with different numbers of stations, namely 25, 50, and 100 which are either clustered or randomly distributed in a plane with x and y coordinates in the interval [0, 60]. They define a time horizon of 2 hours that is divided into 24 periods of 5 minutes each or 60 periods of 2 minutes each [1]. We run the EH-DVRP with a clustered instance of 50 stations and time discretization of 24 periods.

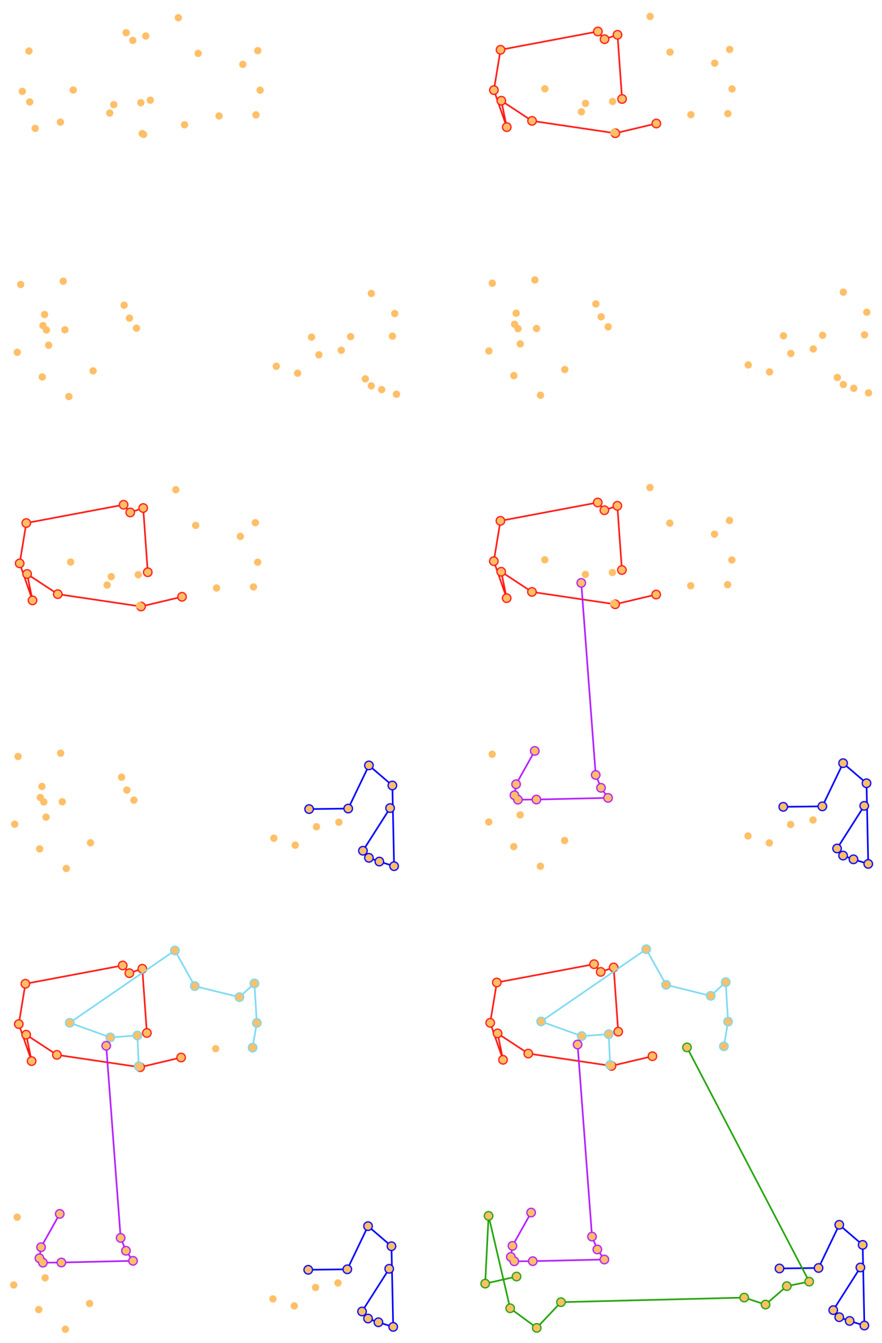


*Figure 2.* The EH-DVRP applied to a Contardo instance of 50 clustered stations.

Our results demonstrate that the EH-DVRP can indeed be applied to the bike-sharing problem. Treating the dynamic bike-sharing problem as a vehicle routing problem, the hyper-heuristic generated sufficient routes for the vehicles. Although the routes don't appear to be ideal, we believe this is due to capacity constraints. Further experimentation using different vehicle capacities, service area distribution, constraints etc. could yield a more optimal solution.

## 6 Conclusions and future work

To conclude, we believe that Garrido and Riff's hyper-heuristic method is extremely useful and applicable to many different kinds of optimization problems. However, it is extremely complicated and hard to use. Any future research should involve taking some time to better understand the method as well as adapting it to better fit the bike-sharing problem. Another possible topic would be to either introduce additional constraints or select different low-level heuristics.

## 7 Acknowledgements

We would like to thank our advisor, Brian Kell, for his help and support. We would also like to thank the director of SUAMI, Deb Brandon, the CMU Center for Nonlinear Analysis, as well as the organizations from which SUAMI receives its funding: the NSF and NSA.

## References

[1] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. *Balancing a dynamic public bike-sharing system*, volume 4. CIRRELT, 2012.

[2] Pablo Garrido and María Cristina Riff. DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics*, 16(6):795–834, 2010.