Optimal Charging and Vending of Shared Electric Bikes

Andrew A. Campbell
Karry Jiao
Jean-Baptiste Sibille
University of California, Berkeley

May 9, 2014

Abstract

In this paper we introduce a new shared bike optimization problem: the e bikeshare charging problem. The current global trajectory of bikeshare adoption, the explosive popularity of personal e bikes in China, and the presence of e bikeshare pilot projects in several countries all support a future that will contain large scale e bikesharing. Although much research has been devoted to understanding the role of bikeshare, considerably less thought has been put towards the operation of shared electric bike systems. We explore the cost optimal charging of shared e bikes at a single station while operating under demand constraints. We introduce two formulations for solving the e bikeshare charging problem: a binary linear program and a dynamic program. Even under simplifying assumptions, the e bikeshare charging problem is shown to be NP complete.

1 Introduction

1.1 Motivation & Background

Public bikeshare systems are among the world's fastest growing modes of public transportation, if not the fastest (Midgley, 2011). Bikeshare is the automated distribution of a public-use bike fleet for short term (usually less than 30 minute) rental, aimed at providing a one-way urban transportation service (DeMaio, 2009). Asia is the world's fastest growing bikeshare market, as defined by number of shared bikes, and China constitutes the majority of that growth (Shaheen et al., 2012). China is also the world's largest adopter of electric

bicycles (e bikes). E bikes feature a battery powered motor to assist locomotion. Due to government support, and their often superior performance to public bus systems, e bikes have been the fastest growing mode of private transportation in China since the early 2000s (Cherry and Cervero (2007); Montgomery (2010)). Annual e-bike sales are on the scale of tens of millions and the fleet size in China is estimated to be between 100 and 150 million (Cherry et al., 2012). Taken together, bikeshare and e bikes are two of the world's most rapidly growing transportaiton technologies. Our research is motivated by the emerging convergence of the two in the form of shared electric bicycle (e bikeshare) systems.

The exponential adoption of bikeshare worldwide has been well documented (Shaheen et al., 2010), but the recent emergence of e bikeshare has received less attention. There have been several small e bikeshare pilots (most notably in in Japan, Copenhagen, and on the campus of the University of Tennessee, Knoxville (City Bike, 2013; GoBike 2014, Cycle-U-Share, 2013)), and commercial e bikeshare products are offered by companies in Europe and China (Mobility Parc, 2013; OmniPay, 2013). The Bike sharing Blog, coauthored by prominent bikeshare researcher, Paul DeMaio, documents at least nine e bikeshare systems worldwide, many of which are augmentations of standard bikeshare systems in Europe (DeMaio and Meddin, 2014). In the case of e bikeshare, the market is outpacing the research; there is only one known e bikeshare study (Langford and Cherry, 2013). Despite the extra layer of complexity that battery charing introduces, we know of no study, besides our own, that investigates optimization of e bikeshare charging.

The ability of e bikes to expand shared bike markets is likely motivating the emergence of e bikeshare, but this ability is contingent upon sufficient battery charge. Researchers find that e bikes help overcome many of the barriers to cycling. The motor assistance reduces fatigue, sweating, and travel time, especially in hilly regions, and also opens cycling to people with physical limitations (Dill and Rose, 2012). However, e bikes are considerably heavier than standard bicycles, weighing anywhere from 25 to 100 kg (Jamerson and Benjamin, 2013). This makes them extremely difficult to operate without motor assistance. Therefore, for an e bikeshare system to be successful, it is critical that users be able to confidently rent e bikes with sufficient battery charge to carry them to their destinations. One possible strategy is to simply charge all batteries to capacity whenever possible. However, this strategy is not cost optimal, may conflict with the environmental goals of policy makers, and may not be feasible in the context of solar powered e bikeshare stations. We therefore explore charging strategies that minimize total power consumption while meeting battery charge and demand constraints.

1.2 Relevant Literature

A variety of literature relating to bikeshare and e bike history, planning, and policy is introduced in the previous section. There is also a large volume of research exploring optimization of shared fleet operation and electric vehicle charging.

The majority of bikeshare optimization research is focused on the problem of redistributing bikes between stations due to the imbalance of user flows. One approach employs network traversal optimization techniques. Benchimol et al. (2011) approach static rebalancing (meaning no bikes are moving) as an adaptation of the C delivery Traveling Salesman Problem and demonstrate that a polynomial time solution does exist in the simplified case of a tree network. In Raviv et al. (2013), multiple heuristic solutions to the NP hard static rebalancing problem are presented, including the time indexed algorithm, which is demonstrated to be capable of solving the dynamic rebalancing problem. Work by Contardo et al. (2012) also explores the more difficult dynamic rebalancing problem. An alternative to approach to using network routing algorithms is to consider station level decisions. This is exemplified by Nair et al. (2013) who treat the marginal costs of inter station movements as constant. Optimal rebalancing movements between station pairs are identified in the context of stochastic demand constraints.

The decision variables in the aforementioned approaches are: routing, pickup and drop of quantity, and inter station movement volumes. In all of these approaches, bicycles are treated as interchangeable. In the context of e bikeshare, this cannot be done since we need to consider the battery state of charge of each bike.

There is a wide body of literature that explores the optimal charging of electric vehicles, especially in the context of smart grid interaction (Deilami et al., 2011) and dynamic price signals (Bashash et al., 2013). Since the addition of battery charge already significantly increases the complexity of shared bike optimization, we simplify the scenario by ignoring grid interaction and assuming constant cost of electricity.

1.3 Focus of this Study

The focus of this study is to develop algorithms for optimizing the cost of charging shared e bikes at a single station under demand constraints. The work is distinct from existant bikeshare optimization research because the inclusion of battery charging adds a new source of cost and makes it infeasible to treat bikes as interchangeable. An important point here is that the problem of redistribution considered in Nair et al. (2013) is not treated in this report.

2 Technical Description

2.1 Data Analysis

The problem formulations in the following sections assume that e bikeshare demand is known. In order to investigate the limits of this assumption, data from real world shared bikes are analyzed. Due to the fact that shared e-bike systems are all in relatively small scale that could not provide us with sufficient data, we decide to use the demand data from Capital Bikeshare system to study the demand and distribution of the bike (CapitalBikeshare, 2014). The reason we choose Capital Bikeshare system is that the dataset contains the specific details on each trip, including the destination station, starting station, time, duration, bike number, etc. In our study, we choose one of the largest stations "Massachusetts Ave & DuPont Cycle" (Station M) which could generate around 20,000 trips per 3 month season and used the data in the second season of 2013 for the analysis. The station displays a highly periodic bi modal distribution of daily demand, corresponding to peak commute hours, Figure 1. For stations with this behavior, stochastic dynamic programming will be a viable solution. Please refer to the Appendix for a more detailed discussion of Capital Bikeshare data.

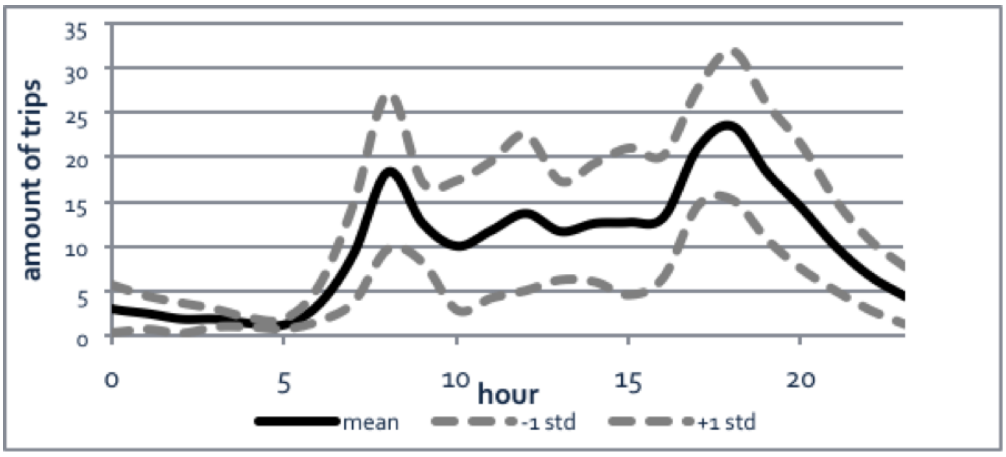

Figure 1: Average daily demand for trips arriving at Station M during Spring, 2013, season

2.2 Problem Formulation: Binary Linear Programming

Let's start by formulating an assumption that shaped most of the project. It is assumed that the system knows at the moment when a user requests a bike what level of charge the user needs. Without this assumption, optimizing the charging pattern for the system is transformed into having a distribution of charged bikes at the station that follows the

expected demand and the system cannot assign the bikes itself anymore. Even though this assumption can sound restrictive, it actually makes sense in the following ways:

- We can request to the customer at which station he plans to return his bike before he starts his trip, and estimate the adequate level of charge. This is what is done in Paris with the *Autolib* system (except that it is with electric cars).
- In the case of a subscription system, where the system keeps track of previous trips for each user, it can learn from the habits of the user and try to predict where he is going next and with that the level of charge he will need.

With this assumption in hand, it appears that the optimization of the charging of the bikes also involves the optimization of the bikes distribution. Indeed, and this is where the model built in this paper differs from what already exists in the literature, bikes cannot be considered all the same. Each bike has a different state of charge, and this distinction has to appear in the mathematical formulation of the problem. It actually adds a lot to the modeling complexity, because the vending of bikes has to be done in an intelligent way. As an example, let's consider a station with four bikes that have the following states of charge (in the following notation, each element in the row vector represent a bike's state of charge):

$$\begin{pmatrix} 90\% & 70\% & 40\% & 10\% \end{pmatrix}$$

If a user requests a bike with 60 % charge, a way to distribute the bike would be to assign him the one with 70 % charge, because it is the least charged bike that has enough charge to meet the user's demand. This assignment needs to be taken into account in the optimization problem. However, this example only shows one part of the optimization (which seems trivial: why would the system distribute an overcharged bike?). There also is an optimization on the whole time horizon that can be thought of. Depending on the system's strategy, it may actually distribute another bike to this user. Let's assume the system is allowed to vend bikes that are not sufficiently charged (for example we allow to vend the bike with 40 % of charge to this user), and it is penalized on the "amount of charge" that doesn't meet the demand ($60 - 40 = 20\%$ loss in that case), then the optimization has to be considered on the whole time horizon and not only on a given time step. Continuing on the previous example, if two users request bikes with 90 % and 70 % charge at the following time step, it would be less penalized to

1. distribute to the first user the bike with 40 % charge
2. then the two following users with bikes that meet their request (loss of 20 %)

rather than

1. distribute the bike with 70 % charge to the first user
2. then be left with 90 % and 40 % charged bikes to distribute to a 90 % and 70 % request (loss of 30 %)

It appears here that even in this very simple example, a clear mathematical problem formulation has to be written in order to properly solve a real-life situation. In the first model presented in this paper, the focus is mostly on understanding how to properly formulate this assignment and charging optimization problem. The first assumptions are hence pretty strong and simplify the problem a lot, but help get a sense of what is happening.

Assumptions Here is a list of all the assumptions to be considered for the first problem formulation.

1. Time is discretized in time steps. The actual length of a time step will depend on the demand data. A longer time step will tend to be less sensitive to the granularity of the demand and "even out" the peaks for instance, hence approaching demand with its expectation. A smaller time step would require more refined modeling of the demand. Regarding the assumptions made later and the data in hand, a reasonable value for the time step would be an hour.
2. All demand is for 100 % charged bikes. This is one of the strongest simplifications done here. It basically means that demand can be aggregated into a simple scalar. However the vector notation will be kept to be able to re-use this formulation for a more complex model where this assumption is removed (See below for the notations).
3. All bikes come back empty to the docks. This assumption is similar to the previous one. It simplifies the treatment of bikes that come back.
4. At each time step, there are never more users that request a bike than the number of docks at the station. In other terms, if the station has 4 docks, there cannot be a demand for 5 bikes at a given time step. This helps the mathematical formulation, as seen later.
5. Similarly, there are never more bikes that come back than the number of docks at the station.
6. A bike is considered available if it has 50 % or more of charge. This is also a strong assumption, but it pairs well with the first one. Indeed the notion of availability for a bike should depend on the user that requests the bike (because each user requests a different state of charge). However with the first assumption, it is possible to define a generic notion of availability. This assumption changes the nature of the mathematical formulation of the problem, as revealed in the second model from Section 4.

7. It takes 2 time steps to fully charge a bike. Hence a bike that was empty at time k will be 50 % full at time $k+1$ and 100 % full at time $k+2$. This assumption is easy to change. A longer charging time basically only requires to write down more contraints, but the concept is similar. Moreover, with time steps equal to one hour, it is a realistic assumption.
8. The formulation only addresses the optimization for one station with N docks. As the redistribution problem is not considered here, stations can be considered individually with users requesting bikes and bikes coming back as external inputs modeled through the data analysis previously carried.

Notations The following notations will be used in the problem formulation:

- N is the number of bikes at the considered station.
- T is the number of time steps in our time horizon.

Then, for each time step $\boxed{k \geq 1}$, let's define:

- D^k the vector of demand for bikes at time step k. It is an **input** to the system, that is assumed to be deterministic for now (i.e. we know the demand for the whole time horizon in advance). With assumption 4, D^k is always of size the number of docks, i.e. $D^k \in \mathcal{M}_{N\times 1}(\mathbb{R})$. As stated in assumption 2, all demand is for fully charged bikes. Hence each component of D^k can only take two values: 0 or 1. Also, this variable will be formatted in a way such that the demand is always "pushed to the top of the vector". For instance, if there are 4 docks in our station, with two users requesting a bike,

$$D^k = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

is not a correct formulation for the demand, but

$$D^k = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

is correct.

- R^k the vector of returned bikes. It is also assumed to be a deterministic **input** for now. It is the size the number of docks, because of assumption 5, and works the same way as D^k.
- A^k is the bikes assignment matrix. It is a **decision variable**. As explained before, the system needs to assign the bikes, depending on the demand. Hence there need to be a way to say which user is assigned to which bike. Here, assumption 4 is crucial: it allows to keep the size of A^k fixed at each time step: it is a square matrix of the size the number of docks in our station: $A^k \in \mathcal{M}_{N\times N}(\mathbb{R})$. It is designed in a way such that $A^k D^k$ is the actual bike assignment. Let's give a simple example to understand what this matrix does exactly. Keeping the same demand vector as previously, having

$$A^k = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

  will lead to

$$A^k D^k = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix},$$

  which means that bikes at docks 2 and 3 are vended to the users who demanded bikes. More precisely, the 1 in position $(2, 2)$ in A^k assigns user 2 to bike 2. The 1 in position $(3, 1)$ assigns user 1 to bike 3. One can remember that the columns of A^k represent the docks and its rows represent the users.
- Similarly, B^k is the docks assignment matrix. It is also a **decision variable**. It is also a square matrix and works the same way as A^k, but with the dock demand vector R^k.
- c^k is a vector representing charging docks. It is a $\mathcal{M}_{N\times 1}(\mathbb{R})$ vector, each element representing a dock. It is a **decision variable**. For instance

$$D^k = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

  means that for a station with 4 docks, docks 1 and 3 are charging the bikes that are docked to them.

- b^k is a vector representing the docks at which there is a bike. It is a **state variable**. If there is a bike at dock i, then $b_i^k = 1$. It is important to note that b^k represents the state of the station at the end of time step k, i.e. after the bikes are checked out and checked in.
- d^k is a vector representing if a bike is available or not. It is a **state variable**. In this model there is no need to keep track of the bikes' state of charge, but only their availability, because of assumptions 7 and 8.
- α represents the marginal cost of charging a dock, considered fixed so far.
- β and γ are design variables and represent how much not meeting the demand in bikes or docks is penalized. It was fixed to 1 for the simulations.

Cost Function With the notations defined above, we can now formulate the cost function:

$$\min_{c,A,B} J = \min_{c,A,B} \alpha \sum_{k=1}^{T} \sum_{i=1}^{N} c_i^k + \beta \sum_{k=1}^{T} \sum_{i=1}^{N} \left[D_i^k - (A^k D^k)_i \right] + \gamma \sum_{k=1}^{T} \sum_{i=1}^{N} \left[R_i^k - (B^k R^k)_i \right]$$

The first term represents the cost of charging: the sum over all time steps and all docks of the docks that are charging.

The second term represents the penalization for the demand in bike that cannot be met. This term has to be considered per time step: let's assume that demand at time step k is

$$D^k = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

and the actual assignment is

$$A^k D^k = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

then it means that one of the requested bikes cannot be distributed (not enough available bikes at the station). In this case, $\sum_{i=1}^{N} D_i^k = \sum_{i=1}^{4} D_i^k = 3$ and $\sum_{i=1}^{N} (A^k D^k)_i = \sum_{i=1}^{4} A^k D_i^k = 2$, hence $\sum_{i=1}^{N} D_i^k - (A^k D^k)_i = 1$, which indeed translates into the cost function the fact that all demand for bikes is not met.

The third term represent the same type of penalization, but for not meeting demand in free docks.

Constraints Constraints are the tricky part of the model. Here is a list of all of them with their physical meaning. For all the constraints listed here, $i \in \{1, \dots, N\}, j \in \{1, \dots, N\}, k \in \{1, \dots, T\}$. As a side note, a dummy time step 0 is created to initialize the system's state variables (i.e. c, b, and d). It is defined by equality constraints that are presented after those inequality constraints. All the constraints listed here are hence well defined for $k \in \{1, \dots, T\}$. Let's start with the **inequality constraints**.

1. All variables are binary, i.e.:

$$\begin{aligned} A_{ij}^k &\in \{0,1\} \\ B_{ij}^k &\in \{0,1\} \\ c_i^k &\in \{0,1\} \\ b_i^k &\in \{0,1\} \\ d_i^k &\in \{0,1\} \end{aligned}$$

2. Don't apply a charge to a dock that has no bike docked in.

$$c^k \leq b^k$$

   It is important to note that this equation, as well as the two following ones, are actually inequalities on vectors, that we define as each component on the left hand side of the equation has to be inferior to its corresponding component on the right hand side.

3. If there is no bike at a dock, the dock doesn't have any bike available.

$$d^k \leq b^k$$

4. A bike is available only if it was charging at the last time step (i.e. has more than 50 % of charge).

$$d^k \leq c^{k-1}$$

5. Only assign the necessary number of bikes. This constraint needs to be there because the demand vector is of the size $N \times 1$, and will hence have some zeros inside representing a non-demand.

$$\sum_{i=1}^{N} A_{ij}^k \leq D_j^k$$

   The inequality takes into account the fact that the demand can possibly not be met. To understand what is happening in this constraint, let's consider two cases:

- $D_j^k = 0$. That means that there is no more than $j-1$ bikes demanded at time step k. The constraint then means that all the components of the jth column of A^k have to be zeros: we are not assigning bikes to non-existing demand.
- $D_j^k = 1$. That means that there is a request for at least j bikes. If the constraint was an equality constraint, it would simply mean that the j-th demand is assigned to at least one bike and no more than one bike (one of the components in the j-th column of A^k is one). The fact that the constraint is an inequality only allows to take into account the situation when there are not enough bikes available. It authorizes to assign no bikes to a given demand

6. Only assign docks at which a bike is available. The model contains a notion of availability for a bike at each dock (d^k vector). The idea is to use this information to decide the values inside A^k:
$$\sum_{j=1}^{N} A_{ij}^k \leq d_i^{k-1}$$

7. There are very similar constraints for assignment matrix for the bikes that come back to the station. First, don't assign a "position" to a bike that is not coming back.
$$\sum_{i=1}^{N} B_{ij}^k \leq R_j^k$$

8. Then, don't assign a bike that comes back to a dock that has no space.
$$\sum_{j=1}^{N} B_{ij}^k \leq 1 - b_i^{k-1}$$

Now, the **equality constraints**:

9. Update law for the state variable b, which is a vector representing the presence of bikes at the station or not. The new state is the past state minus the bikes that we have assigned plus the bikes that have been returned.
$$b^k = b^{k-1} - A^k D^k + B^k R^k$$

10. This is the initial state of the system: all bikes are charging, all bikes are available.
$$b^0 = d^0 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Condition (4) means that a bike is available only if it was charging at the last time step. Indeed, if the bike was charging at the last time step, (2) will ensure that it was present. In an earlier formulation, this inequality constraint that defines the availability of bikes was expressed as an equality constraint on previous state variables. However this made the problem non-linear and this is why this equivalent inequality constraint is preferred. However, at first thoughts, one may think that removing the exact equality constraint on d also removes a lower bound on d and therefore one could think that the solver will simply assign $d = 0$ at each time step. But one must keep in mind that there also is a link between A^k and d^k, through constraint (6). This constraint, together with assigning $d^k = 0$ at each time step would lead to assigning $A^k = 0$ (matrice of zeros). And this would lead to a penalization through the second term of the cost function. Having only an inequality constraint to define d is therefore not an issue in this situation, and the problem is not under constrained.

Let's now count how many inequality constraints we have total. The binary constraints (1) will be directly implemented into the solver and are not counted here.

- (2) represents $T \times N$ constraints.
- (3) represents $T \times N$ constraints.
- (4) represents $T \times N$ constraints.
- (5) represents $T \times N$ constraints.
- (6) represents $T \times N$ constraints.
- (7) represents $T \times N$ constraints.
- (8) represents $T \times N$ constraints.

We hence have a total of $7 \times T \times N$ inequality constraints. Similarly, we have a total of $N \times (T-1) + 3 \times N$ equality constraints.

With the cost function and all the constraints in hand, it is possible to formulate the problem under a canonical form, in order to enter it into a solver:

$$\begin{aligned} & \min e^T X \\ s.t. \quad & CX \leq F \\ & C_{eq} X = F_{eq} \end{aligned}$$

Concatenating all constraints together, it appears that X is a vector with $T(2N^2 + 3N)$ rows. It is hard to represent it in the general case but in the example of a station with 2 docks, X will look like the following:

$$X = \begin{bmatrix} A_{11}^1 \\ A_{21}^1 \\ A_{12}^1 \\ A_{22}^1 \\ B_{11}^1 \\ B_{21}^1 \\ B_{12}^1 \\ B_{22}^1 \\ d_1^1 \\ d_2^1 \\ b_1^1 \\ b_2^1 \\ c_1^1 \\ c_2^1 \\ \vdots \\ A_{11}^T \\ A_{21}^T \\ A_{12}^T \\ A_{22}^T \\ B_{11}^T \\ B_{21}^T \\ B_{12}^T \\ B_{22}^T \\ d_1^T \\ d_2^T \\ b_1^T \\ b_2^T \\ c_1^T \\ c_2^T \end{bmatrix}$$

Matrix C has $7TN$ rows and $T(2N^2 + 3N)$ columns. Writing down an example for C is of poor interest here and the interested reader can figure out how to formulate it knowing the shape of X. F is a $7TN$ rows vector. C_{eq} is a $N \times (T-1) + 3 \times N$ rows and $T(2N^2+3N)$ columns matrix. F_{eq} is a $N \times (T-1) + 3 \times N$ rows vector. Finally, e is a $T(2N^2 + 3N)$ rows vector.

The problem is clearly a binary linear optimization program. Several techniques exist to try to optimize the resolution of such NP-hard problems. Granted the scope of the project, and the fact that most of the assumptions are going to be removed for a more general formulation, our focus has been more on testing this solution rather than digging too much into binary optimization. Matlab includes in its optimization toolbox a solver for binary linear problems such as ours that we used to run a simulation. The process and results are described in the following section.

2.3 Implementation & Results

Using Matlab binary solver, we implemented a program that can solve the problem formulation above for any N and any T. The hardest part of the implementation was to find a way to define the matrices C, C_{eq}, F, and F_{eq} for any value of N, and T. Sparse matrices were used for that purpose with the following logic: a block of each of those matrices is created for each time step, and then is added to the total matrix at the correct position. For instance, running the simulator with the following inputs

- $N = 4$
- $T = 6$
- $D = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
- $R = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

returns a big X vector that needs to be formatted and interpreted. The formatted version of X looks like Figure 2, but this is given for illustration purposes only. The interpretation and analysis of this result is given right after.

Let's look how we can interpret that result. In Figure 3 is presented the data from Figure 2 in a more understandable way.

The results from Figure 3 are interesting to analyze. Let's go back to the original goal of the study: optimize the cost of charging bikes in a bike sharing system. With the color code from Figure 3, every dock that is in orange state corresponds to a dock where there is a bike that is not fully charged. Hence it corresponds to a state where a dummy system would apply a charge to the dock. However our system only applies a charge to 3 of those 11 orange

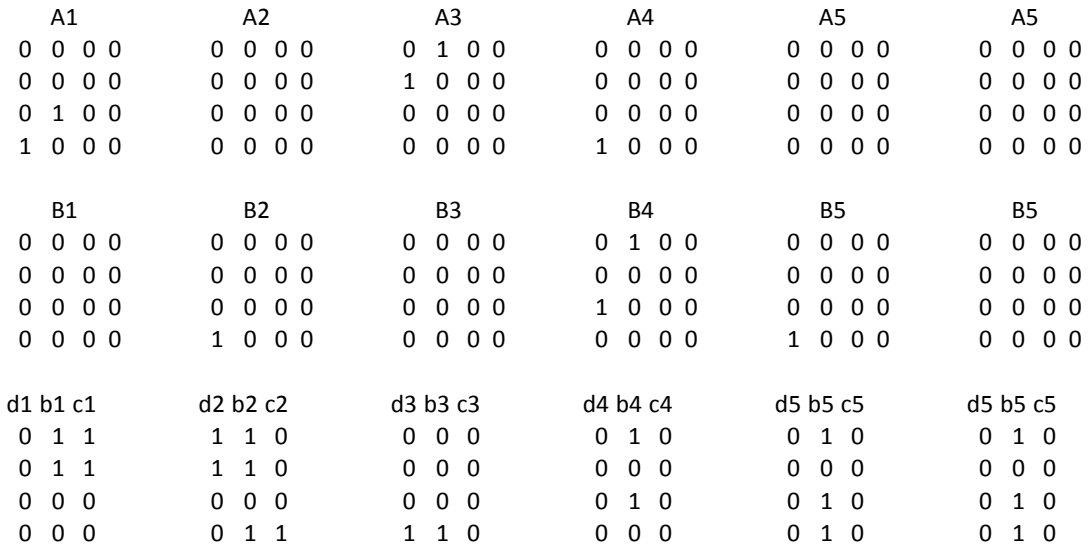

Figure 2: The result returned by Matlab for $N = 6$ and $T = 4$, which corresponds to the X vector formatted in a readable way.

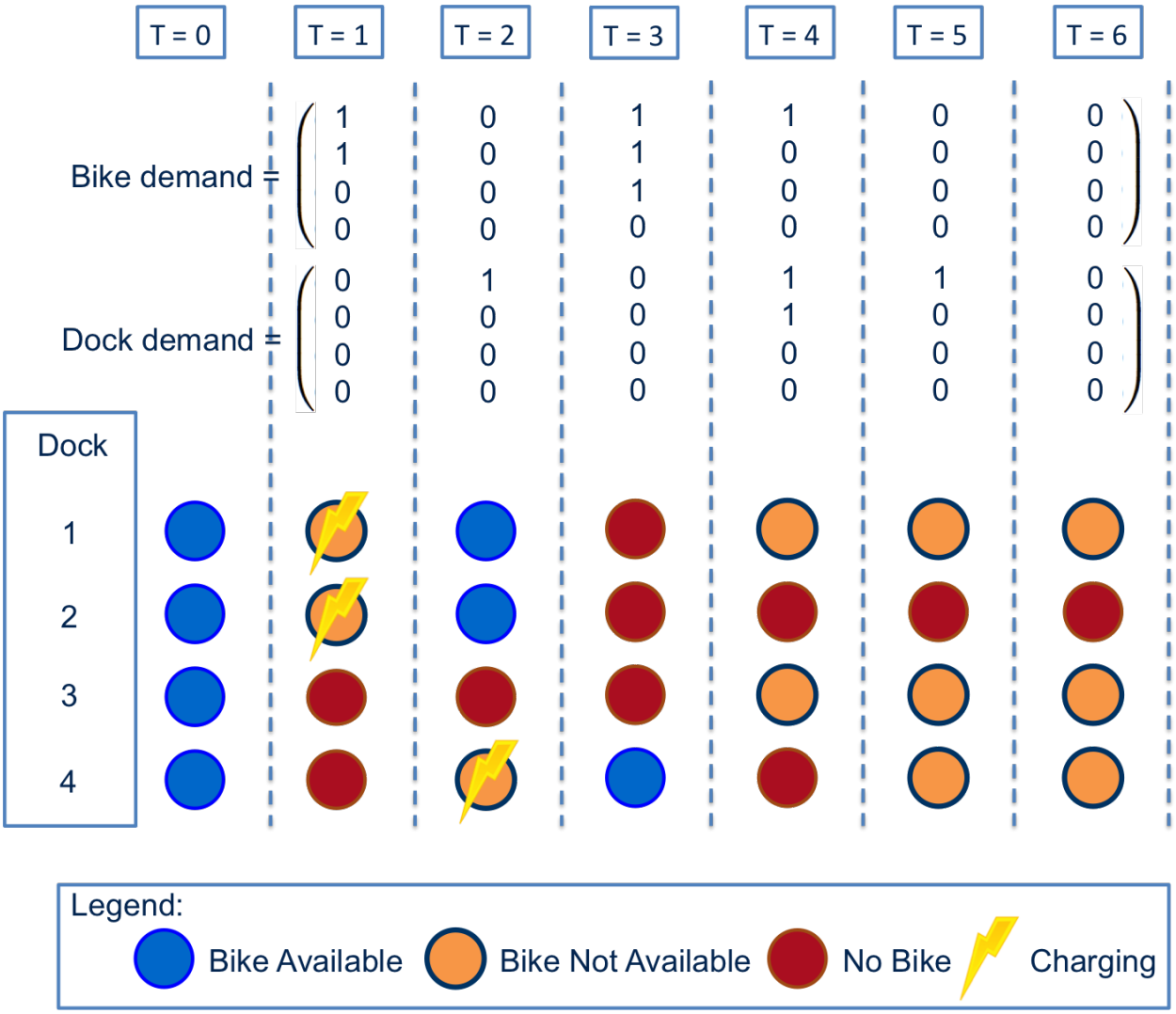

Figure 3: **The interpretation of the result from Figure 2.** As a reminder, the state variables c, d, b represent the state of the station at the **end** of each time step. At the first time step, there is a demand for two bikes, and no bike comes back. There are four bikes available at the end of the previous time step (time step 0), hence it is possible to distribute two bikes. This is why at the end of time step 1, two docks are marked as red: bikes have been checked out of those docks. The first two docks are orange, which can seem strange at first but can be considered as a slight initialization bug: we consider that at time step 0, all bikes are present and available but not charging. As bikes 1 and 2 were not charging at the previous time step (0), they are considered unavailable at time step 1. It doesn't harm the availability of bikes. When a bike comes back, like at time step 2, it is taken into account by the system and one dock that was red becomes orange (and charging, if necessary). One last thing to note here is that demand for bikes at time step 3 exceeds the number of available bikes. The program behaves as expected: it distributes as many bikes as possible, and then penalizes the cost function for the bike that hasn't been distributed.

states. Hence, even in this very simple example, there is a clear need for optimization in the charging pattern. This first result is very encouraging for the refinements of the model to come.

3 Discussion

Despite this encouraging first result, there is bad news in sight. Indeed, the number of variables and constraints scales badly, especially with N, and one must keep in mind that we are dealing with an NP hard problem here (even if the built-in solver has some optimizations included, it cannot perform miracles). Our tests showed that it is unrealistic to run such an optimization program on a personal computer for any $N > 20$ and $T > 12$. This is a first reason that motivates a move to Dynamic Programming, in order to really understand what is happening under the hood of the solver, and not simply call a Matlab routine. Another motivation for switching to Dynamic Programming is that the nature of the problem evolves when removing some of the initial assumption, especially the one regarding the uniform demand. The problem shifts to a binary/integer linear optimization problem for which a solver is available in Matlab only for the version 2014, which wasn't available to us. Also, implementing our own algorithm will allow us to understand what optimizations can be done in order to reduce computation time.

Hence, we tried to go to a lower level for the implementation and move to dynamic programming. The next section describes the dynamic programming formulation of the problem, along with some assumptions removal.

4 Dynamic Programming

4.1 Problem reformulation

For this part to bring something new to the previous formulation, the assumption that all demand is for the same level of charge (Assumption 2) is removed. Therefore some new variables have to be introduced and and variables from the last part have to be modified. Everything that is new here is written in **bold**. Here are the variables, for a given time step k.

Input variables

- $\mathbf{D^k}$ is the demand vector. The vector keeps the same dimension (number of docks), but

this time it represents the actual levels of charge requested by the users. For instance

$$D^k = \begin{pmatrix} 90 \\ 60 \\ 0 \\ 0 \end{pmatrix}$$

represents a demand for two bikes, one with 90 % charge and one with 60 % charge.

- $\mathbf{D^k_{bin}}$ is the equivalent as the vector D^k in the previous section. It is a binary vector representing the requested bikes at time step k. It can be simply formulated as a function of the new D^k we have just introduced using the following formula:

$$\forall i \in \{1, \ldots, N\}, D^k_{bin,i} = \mathbb{1}\left\{D^k_i \neq 0\right\}$$

  which can be interpreted as follows: if a component in the D^k vector is not zero, that means that there is a demand for a bike, and we want to take it into account in D_{bin}.
- R^k is the same as previously: it is the demand in free docks at a given time step. As the assumption that bikes come back with zero charge has not been removed yet, there is nothing to change regarding this vector.

State variables State variables represent the state of the system at the end of each time step.

- b^k is the same as in Section 2.2: it is a vector that represents if bikes are present at given docks or not.
- $\mathbf{s^k}$ is a new state variable that replaces d^k. It is a vector that represents the state of charge of the bikes that are docked at the station, for the docks where there is a bike. For instance

$$s^k = \begin{pmatrix} 70 \\ 40 \\ 0 \\ 80 \end{pmatrix}$$

  means that there are 3 or 4 bikes at the station (we don't know if the zero represents no bike or an uncharged bike. That is the purpose of the other state variable b^k). What we know is that there is a bike with 70 % charge at dock 1, a bike with 40 % charge at dock 2, and a bike with 80 % charge at dock 4.

Control variables Those variables are the same as the ones defined in 2.2. They are:

- A^k: bikes assignment matrix.
- B^k: docks assignment matrix.
- c^k: charging docks vector.

Cost Function The cost function doesn't change much either compared to the previous formulation from Section 2.2. It can be expressed as:

$$\min_{c,A,B} J = \min_{c,A,B} \alpha \sum_{k=1}^{T} \sum_{i=1}^{N} c_i^k + \beta \sum_{k=1}^{T} \sum_{i=1}^{N} \left[D_i^k - (A^k D^k)_i \right] + \gamma \sum_{k=1}^{T} \sum_{i=1}^{N} \left[R_i^k - (B^k R^k)_i \right]$$

However one must keep in mind that here the variable D^k is different from what was defined in Section 2.2. Indeed, this time users can be distributed bikes that don't meet the requested level of charge. However this distribution is authorized in a limited way that will be defined in the constraints.

Constraints Let's first look at the **equality constraints**. Now that the state of charge is expressed explicitly in the equations, an update law for this state variable has to be formulated. It is important to notice here that as the implementation of this problem should be done *via* a solver built from ground up, ensuring the linearity of the constraints doesn't seem as important as previously, where we used a built-in solver that only supported linear constraints.

1. Update law for the presence of bikes at the docks. This constraint doesn't change from what was formulated in Section 2.2:

$$b^k = b^{k-1} - A^k D_{bin}^k + B^k R^k$$

2. **Update law for the state of charge of the bikes.** This constraint is new and translates the dynamics of the state of charge of a bike **when it is docked**. The scope of the project is the charging of bikes when they are at the station and the state of charge of bikes when they are not docked is not of interest here. Bikes are only considered as inputs to the system when they come back, and their level of charge is then considered as an input as well (zero with the assumptions we have made).

$$\forall i \in \{1, \dots, N\}, s_i^k = (s_i^{k-1} + CRc_i^{k-1}) b_i^k$$

where CR is the Charging Rate (the higher CR, the less time steps it takes to fully charge the bike). This equation translates several things:

- When a bike is not present at the end of time step k at dock i ($b_i^k = 0$), the state of charge of a given dock is considered to be zero.
- When a bike is present at the end of time step k at dock i ($b_i^k = 1$), the state of charge of bike i at the end of time step k is equal to the previous state of charge of bike i (s_i^{k-1}), plus the amount of charge that has been applied to the bike during time step k (the bike is charging during time step k if $c_i^{k-1} = 1$, because c_i^{k-1} represents if the bike is charging from the end of time step $k-1$ to the end of time step k).
- A bike can never be overcharged, for the simple reason that there is never demand for more than 100 % charged bikes. Hence once s_i^k reaches 100, it would only penalize the system to charge it more, and the optimal solution will then set $c_i^k = 0$.
- Bikes in this model are considered not to be discharging if they are idle (i.e. neither in use or charging). This is a simplifying assumption to help us implement a first model of dynamic programming but one can easily figure a way to incorporate a discharge rate in the update equation.

3. Initial conditions:

$$b^0 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$s^0 = \begin{pmatrix} 100 \\ \vdots \\ 100 \end{pmatrix}$$

Now let's look at the **inequality constraints**. Again, most of the work done in Section 2.2 can be re-used, but there are a couple of subtleties that need to be put in light:

1. There is a mix of binary and integer variables:

$$\begin{aligned} A_{ij}^k &\in \{0,1\} \\ B_{ij}^k &\in \{0,1\} \\ c_i^k &\in \{0,1\} \\ b_i^k &\in \{0,1\} \\ \mathbf{s_i^k} &\in \{\mathbf{0}, \mathbf{10}, \mathbf{20}, \ldots, \mathbf{90}, \mathbf{100}\} \end{aligned}$$

2. Don't apply a charge to a bike that is not present stays the same:

$$c^k \leq b^k$$

3. Only assign the necessary number of bikes stays the same:

$$\sum_{i=1}^{N} A_{ij}^k \leq D_j^k$$

4. **Have a 20 % tolerance factor on the bikes that are distributed**. This condition echoes what was said earlier in this part regarding the penalization of distributing bike that are not charged enough. Such distribution is authorized but only in a limited way: a tolerance factor is introduced (fixed at 20 % here, but this value can be changed) to translate that condition. It handles the optimization problem stated at the beginning of Section 2.2 regarding the optimal distribution of bikes over all the times period rather than over single time periods.

$$\forall i \in \{1, \ldots, N\}, (A^k D^k)_i \leq 1.2 s_i^{k-1}$$

   To understand this equation, one must remember that $A^k D^k$ represents the actual bike assignments. Hence $(A^k D^k)_i$ represents the demand in charge that is actually assigned at dock i. The actual state of charge of dock i is overestimated (factor 1.2) to allow the distribution of bike even if it doesn't have sufficient charge, to a given extent.

5. The constraints for assignment matrix for the bikes that come back to the station are the same. First, don't assign a "position" to a bike that is not coming back.

$$\sum_{i=1}^{N} B_{ij}^k \leq R_j^k$$

6. Then, don't assign a bike that comes back to a dock that has no space.

$$\sum_{j=1}^{N} B_{ij}^k \leq 1 - b_i^{k-1}$$

4.2 Implementation

This section is not fully finished at the time when this report is written.

Complexity considerations To implement our model using dynamic programming, we got inspired by what was done in homework 5. First of all, the constraints for the state variables have to be grouped and written in a way understandable by a computer. The following constraints are valid for any $(i, j, k) \in \{1, \ldots, N\} \times \{1, \ldots, N\} \times \{1, \ldots, T\}$:

$$\begin{aligned} 0 \leq \quad s_i^k &\leq 100 \\ s_i^k &= (s_i^{k-1} + CRc_i^{k-1})b_i^k \\ 0 \leq \quad b_i^k &\leq 1 \\ b_i^k &= b_i^{k-1} - (A^k D_{bin}^k)_i + (B^k R^k)_i \end{aligned}$$

What happens here is that those variables will actually only be able to take a set of values pre-defined on a grid with the linspace function. The control variables can only take a limited number of values. This will be crucial in the implementation of the program as it will reduce the complexity:

$$\begin{aligned} 0 \leq \quad c_i^k &\leq 1 \\ 0 \leq \quad A_{ij}^k &\leq 1 \\ 0 \leq \quad B_{ij}^k &\leq 1 \end{aligned}$$

Those equations look good, but are actually one of the worst scenario one could imagine. They basically mean that the complexity of our program will be $\mathcal{O}(T2^{2*N^2+N}) = \mathcal{O}(T) + \mathcal{O}(2^{2*N^2})$, which turns out to be unrealistic to implement even for very small values of N. However, the other inequalities are very helpful in reducing the complexity of the problem:

$$\sum_{i=1}^{N} A_{ij}^k \leq D_j^k$$

means that there can at most be only one "1" on each column of A^k. Actually, if $D_j^k = 0$, then a whole column of A^k will be zeros. If $D_j^k = 1$, the problem consists of placing a "1" somewhere in the j-th column of A^k. Hence, instead of trying every binary combination for the j-th column (for which the complexity would be $\mathcal{O}(2^N)$), the program will only have to test every position for this "1". This reduces the complexity of filling this column to $\mathcal{O}(N)$ (!). A similar observation can be made for B^k. The only tricky part here is c^k, for which the only constraint available is:

$$c_i^k \leq b_i^k$$

which is not sufficient to change the complexity of the problem. Indeed, for every bike that is at the station ($b_i^k = 1$), we will have to test the binary combinations for c_i^k. For example, if we have P bikes at the station, the complexity is $\mathcal{O}(2^P)$. With those reductions in complexity, the problem's complexity is now under $\mathcal{O}(T) + \mathcal{O}(N^2 2^N) = \mathcal{O}(T) + \mathcal{O}(2^N)$.

Value Function, Principle of Optimality Let's define the value function $V_k(s^k, b^k)$ as the optimal cost to go from time step k to time step N. To formulate the principle of optimality, we define:

- The instantaneous cost $g_k(s^k, b^k, A^k, B^k, c^k)$ (where we put the state variables first, then the control variables) as:

$$g_k(s^k, b^k, A^k, B^k, c^k) = \sum_{i=1}^{N} \left[\alpha c_i^k + \beta[D_i^k - (A^k D^k)_i] + \gamma[R_i^k - (B^k R^k)_i]\right]$$

- The boundary condition. So far the system is considered as being able to finish in any state, without any penalty:

$$\forall s^N, b^N, V_N(s^N, b^N) = 0$$

The principle of optimality now states that:

$$V_k(s^k, b^k) = \min_{A^k, B^k, c^k} \left\{ g_k(s^k, b^k, A^k, B^k, c^k) + V_{k+1}(s^{k+1}, b^{k+1}) \right\}$$

The implementation on Matlab of this program has been started, but not finished yet.

5 Summary

In this paper we have introduced a new kind of shared bicycle optimization problem, the e bikeshare charging problem. This is motivated by the recent emergence of several e bikeshare systems. E bikeshare is a convergence of two of the world's fastest growing transportation modes, bikeshare and e bikes, and can potentially greatly expand geographic and social markets of shared bike systems.

We have formulated two solutions to the e bikeshare charging problem. Even after imposing many simplifying assumptions, the problem remains challenging. A binary linear formulation is tested and shown to be computationally difficult. Not only are binary linear programs known to be NP hard, but the need to assign specific bicycles leads to variable and coefficient matrices that scale as a polynomial of the number of bikes at a station and the number of time steps. For this reason, a dynamic program is also formulated.

After testing and refining our dynamic program formulation, it will be possible to further relax assumptions and to incorporate stochastic demand forecasting. Real world energy prices will be used to benchmark cost savings in various demand and charging scenarios. Despite the change in formulation, the problem remains NP hard and will require brute force calculations to optimize the charging pattern. This limits the optimization possibilities to stations with a limited number of docks.

References

Bashash, Saeid, and Fathy. Cost-optimal charging of plug-in hybrid electric vehicles under time-varying electricity price signals. *Intelligent Transportation Systems, IEEE Transactions*, 99:1–11, 2013.

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

CapitalBikeshare. Trip history. `http://capitalbikeshare.com/trip-history-data`, 2014. Accessed: May 08th, 2014.

C. Cherry and R. Cervero. Use characteristics and mode choice behavior of electric bike users in china. *Transp Policy*, 14:247–257, 2007.

C. Cherry, H. Yang, Jones, L., and H. Min. Dynamics of electric bike use over six years in kunming china. *Transport Res A-Pol (in review)*, 2012.

C. Contardo, C. Morency, and L.-M. Rousseau. Balancing a dynamic public bike-sharing system. *CIRRELT*, 4, 2012.

S. Deilami et al. Real-time coordination of plug-in electric vehicle charging in smart grids to minimize power losses and improve voltage profile. *Smart Grid, IEEE Transactions*, 2 (3):456–467, 2011.

P. DeMaio. Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, 12, 2009.

P. DeMaio and R. Meddin. The bike-sharing blog. `http://bike-sharing.blogspot.com/`, 2014. Accessed: May 08th, 2014.

J. Dill and G. Rose. Electric bikes and transportation policy. *Transportation Research Record: Journal of the Transportation Research Board*, 2314(1):1–6, 2012.

F.E. Jamerson and E. Benjamin. *Electric Bikes Worldwide Reports*. Thirteenth edition, 2013.

B.C. Langford and C. Cherry. North america's first electric bicycle share: A year of experience. Washington, D.C., 2013. Proceedings of TRB 2013 Annual Meeting.

P. Midgley. Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas. *United Nations Department of Economic and Social Affairs*, 2011.

B.N. Montgomery. Cycling trends and fate in the face of bus rapid transit. *Transportation Research Record: Journal of the Transportation Research Board*, 2193:28–36, 2010.

R. Nair et al. Large-scale vehicle sharing systems: Analysis of vélib'. *International Journal of Sustainable Transportation*, 7(1):85–106, 2013.

Tal Raviv, Michal Tzur, and Iris A. Forma. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3): 187–228, 2013.

S.A. Shaheen, S. Guzman, and H. Zang. Bikesharing in europe, the americas, and asia: Past, present, and future. *Transportation Research Record: Journal of the Transportation Research Boar*, 2143(159-167), 2010.

S.A. Shaheen, E.W. Martin, A.P. Cohen, and R.S. Finson. Public bikesharing in north america: Early operator and user understanding. *Mineta Transportation Institute*, 2012.