November 2017

Improving Service Level of Free-Floating Bike Sharing Systems

Aritra Pal
University of South Florida, aritra1@mail.usf.edu

Improving Service Level of Free-Floating Bike Sharing Systems

by

Aritra Pal

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Date of Approval:
October 2, 2017

Keywords: Bike Sharing, Last Mile, Mobility Patterns, Rebalancing, Anomaly Detection

Dedication

This dissertation is dedicated to my parents Abhijit and Rupa Pal, for their unconditional support in all my endeavors.

Acknowledgements

My heartfelt thanks to Dr. Tapas Das for giving me the opportunity to pursue PhD at University of South Florida.

I would like to thank my co-major professors: Dr. Yu Zhang and Dr. Changhyun Kwon, and the rest of my committee members: Dr. Tapas Das, Dr. Grisselle Centeno and Dr. Gangaram S. Ladde for their generous advice, support and guidance during my doctoral study.

I would like to thank Garrick Aden-Buie for introducing me to the Julia programming language; and to Dr. Bo Zeng and Dr. Hadi Charkhgard for teaching me the beautiful art and science of optimization.

This dissertation would not have been possible without the unconditional support of my parents: Abhijit and Rupa Pal, my brother: Avik Pal, my beautiful girlfriend: Kelly Serban, my other family members: Anjana Pal, Chandrima Bose, Anirban Bose, Sreetama Guha, Sanjib Guha and my naughty nephews.

I would like to thank my colleagues at USF, Tampa namely Daniel Romero, Sharmin Mithy, Jasper Quach, Debtanu Maiti, Sovik Nath, Vignesh Subramanian, Nazmus Sakib and XuXue Sun for making my four years at USF, Tampa memorable.

Last but not the least to my friends in India and abroad: Rahul Dey, Subrata Das, Atanu Santra, Krishnendu Banerjee and Debayan Chandra, who I miss dearly everyday.

Table of Contents

List of Tables

List of Figures

Abstract

Bike Sharing is a sustainable mode of urban mobility, not only for regular commuters but also for casual users and tourists. Free-floating bike sharing (FFBS) is an innovative bike sharing model, which saves on start-up cost, prevents bike theft, and offers significant opportunities for smart management by tracking bikes in real-time with built-in GPS. Efficient management of a FFBS requires: 1) analyzing its mobility patterns and spatio-temporal imbalance of supply and demand of bikes, 2) developing strategies to mitigate such imbalances, and 3) understanding the causes of a bike getting damaged and developing strategies to minimize them. All of these operational management problems are successfully addressed in this dissertation, using tools from Operations Research, Statistical and Machine Learning and using Share-A-Bull Bike FFBS and Divvy station-based bike sharing system as case studies.

1 Introduction

Bike sharing allows people a healthy, enjoyable and emission-free way to commute across small distances free from the worries of owning a bike. It also provides an alternative and attractive solution for the first and last-mile problem in multi-modal transportation. Over the years, various schemes of bike sharing have been presented, with the earliest generation dating back to July, 1965, in Amsterdam with Witte Fietsen (White Bikes). The next generation, coin-deposit systems, first were introduced in Denmark in Farse and Grena in 1991 and then in Nakskov in 1993. A major breakthrough came when people could use a magnetic stripe card to rent a bike. This generation of bike sharing, known as the IT-based system started with Bikeabout in 1996 at Portsmouth University, England. Interested readers are referred to DeMaio (2009), for an overview of various generations of bike sharing.

Free-floating bike sharing (FFBS), also known as station-less bike sharing is a new generation of bike sharing system (BSS) that allows bikes be locked to ordinary bike racks (or any solid frame or standalone), eliminating the need for specific stations. In comparison to the prevailing Station-based Bike Sharing (SBBS), FFBS saves on start-up cost by avoiding the construction of expensive docking stations and kiosk machines required for SBBS. With built-in GPS, customers can find and reserve bikes via a smart phone or a web app and operators can track the usage of the bikes in real-time. These has two primary benefits. First, users satisfaction levels increase as renting and returning bikes become extremely convenient and second, operators have the basis for smart management of the system.

SocialBicycles (SoBi) is one of the providers of FFBS bikes. SoBi bikes are used as an example to further illustrate how FFBS works. Each registered SoBi member gets a unique PIN and can use a smartphone app to locate available bikes. After reserving a bike, the user has 15 minutes to get to its location. Once the user finds the bike, (s)he enters the PIN on the bike's built-in keypad to unlock the bike. If the user wants to stop somewhere quickly, the bike can be locked and placed on hold. Upon reaching the destination, the user can simply lock the bike to a bicycle rack (or any solid frame or standalone) and the bike becomes available for the next user.

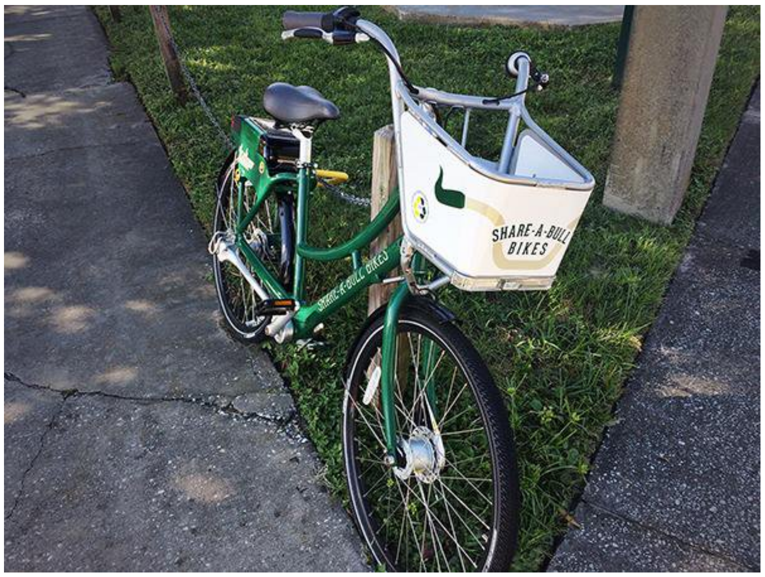


Figure 1.1: A Share-A-Bull Bike Locked to a Wooden Frame

The core problem faced by the operator of a bike sharing system, is to maximize its service level (ability to serve users) and decrease its maintenence cost. The major causes for decrease in service level of a BSS, is imbalance of usable bikes and presence of unusable/damaged bikes in the system. Imbalance of usable bikes in the system can be addressed by maintaining an (sub-)optimal quantity of bikes in each station in case of station-based systems or each zone in case of free-floating systems, because excess supply may hamper the return of bikes, whereas shortage in supply may result in increased access cost for users (e.g. elongated walking distance) and in lost demand. To mitigate the overall or a station/zonal imbalance, the operator may use different rebalancing strategies depending on the situation. Maintenance cost of a BSS pertaining to usable bikes is primarily owing to operator-based rebalancing, which can be minimized by computing high quality tours of the fleet of rebalancing vehicles. Solving the core problem of an established BSS requires understanding the mobility patterns of its users. This enables the operator to estimate an approximate target distribution of bikes for rebalancing as well as gain insights necessary for developing appropriate rebalancing strategies, e.g. whether static rebalancing is sufficient or is dynamic rebalancing needed, when are the different types of rebalancing appropriate and how much time is available for each type of rebalancing.

Usable bikes become unusable for two major reasons: 1) from over usage by subset of regular users and 2) from mishandling or vandalism by a subset of casual users, who will be refered to as malevolent users henceforth. Once a usable bike becomes unusable, a user is unable to use it until it is repaired, decreasing his/her level of satisfaction. Now, the operator has to repair these unusable bikes either on-site or at a remote location, both of which involve routing and labor costs. These costs owing to the presence of unusable bikes can be minimized, if the operator employs strategies to prevent usable bikes from being converted to unusable bikes owing to over usage by taking proper maintenance measures and to prevent damage from mishandling / vandalism by identifying malevolent users.

In chapter 2 of this dissertation, we present a novel mixed integer linear program for solving the static complete rebalancing problem. The proposed formulation, can not only handle single as well as multiple vehicles, but also allows for multiple visits to a node by the same vehicle. We present a hybrid nested large neighborhood search with variable neighborhood descent algorithm, which is both effective and efficient in solving static complete rebalancing problems for large-scale bike sharing programs. Computational experiments were carried out on the 1-commodity pickup and delivery traveling salesman problem (1-PDTSP) instances previously used in the literature and on three new sets of instances, two (one real-life and one general) based on Share-A-Bull Bikes (SABB) FFBS program recently launched at the Tampa campus of University of South Florida and the other based on Divvy SBBS in Chicago. Computational experiments on the 1-PDTSP instances demonstrate that the proposed algorithm outperforms a tabu search algorithm and is highly competitive with exact algorithms previously reported in the literature for solving static rebalancing problems in SBSS. Computational experiments on the SABB and Divvy instances, demonstrate that the proposed algorithm is able to deal with the increase in scale of the static rebalancing problem pertaining to both FFBS and SBBS, while deriving high-quality solutions in a reasonable amount of CPU time.

In chapter 3 of this dissertation, we try to understand the mobility patterns and imbalance of an FFBS by analyzing its historical trip and weather data. Resulting outcomes provide insights to assist the system operator to make more informed decisions. Researchers have studied mobility patterns by analyzing historical trip and weather data of station-based bike sharing systems (SBBS) using data visualization and or generalized linear models. However, none of these studies considered interaction between independent variables or study imbalance as a dependent variable. In this chapter, we demonstrate that by considering such interactions, more insights can be obtained about the mobility patterns and imbalance of an FFBS. We propose a simple method to decompose continuous variables into binary variables and two stage models that consider interactions between independent variables. The proposed decomposition method significantly improves the (quasi-)Poisson regression model commonly used in the literature and has the ability to identify intervals of a continuous variable for which they are statistically significant.

In chapter 4 of this dissertation, we develop strategies to minimize the overall costs of a FFBS due to damage by: 1) determining the relationship between the number of breakdowns of a bike to its total distance traveled, total duration of travel and total number of pickups, 2) identifying users who are responsible for breakdowns of bikes and 3) developing strategies to minimize damage to bikes based on the previous two outcomes. The first and second problem are formulated as a supervised and an unsupervised learning problem respectively and tested on the Share-A-Bull FFBS (SABB), an FFBS on the main campus of the University of South Florida (USF). Finally, based on the above two outcomes, we provide some strategies that the operator of SABB FFBS can use to minimize costs owing to damage of bikes. It is worth mentioning that our proposed method is easy to implement and can be easily ported to other bike sharing systems without any changes.

Finally, Chapter 5 concludes the dissertation with directions for future research.

2 Solving Large-Scale Static Rebalancing Problems in Free-Floating Bike Sharing Systems

2.1 Note to Reader

This chapter has been previously published © 2017 Elsevier. Reprinted with permission from Pal and Zhang (2017).

2.2 Introduction

During daily operation, the distribution of bikes in the system becomes skewed, often leading to a low quality of service and user dissatisfaction. To prevent such a scenario from prevailing, operators move bikes across the network to achieve a desired distribution. The operation of redistributing bikes across the network using a fleet of vehicle(s) is known as bike rebalancing. Rebalancing at night, when user intervention is negligible, is called static rebalancing. If user intervention is considered, the problem is called dynamic rebalancing. Bike rebalancing being a variant of vehicle routing problem, is a challenging combinatorial optimization problem, with the objective function being, to minimize the financial and environmental costs of rebalancing. Different variants of the bike rebalancing problem have been proposed in the literature, see Section 2.3 for a detailed literature review.

Dependent on how rigorous of a rebalancing needs to be performed, it can be classified into two categories, complete and partial rebalancing. In complete rebalancing the rebalancing operation terminates only when target inventory of all nodes in the network have been met. However, if complete rebalancing is not feasible (for example: if the time taken to completely rebalance the bike sharing system is more than the actual time available for rebalancing), the operator may consider partial rebalancing. In partial rebalancing, not all nodes will meet their target inventory. In SBBS, nodes are working stations with status of having a deficit or surplus of bikes, or being self-balanced. FFBS has no stations like in SBBS, so nodes in FFBS include regular bike racks and standalone locations where bikes are parked by users, and bike racks and standalone locations where bikes have not been parked but are perceived as important locations for bikes to be present by the operator. For FFBS, there are no stations like in SBBS. In this chapter, we focus on Static Complete Rebalancing Problem (SCRP). Partial rebalancing are an on-going effort of our research team and will be addressed in a future article. The static complete rebalancing problem (SCRP) is computationally more challenging than static partial rebalancing problem, because the number of times a node is visited, in the optimal solution can not be determined apriori. We are also performing studies on understanding demand patterns of bike sharing system and explore how dynamic rebalancing can be applied in real world cases.

For the same configuration, i.e., number of stations (in case of SBBS and bike racks in case of FFBS), number of bikes and capacity of the rebalancing fleet, computational complexity of SCRP is higher for FFBS than for SBBS. To illustrate this, let us consider a bike sharing system with 100 stations (or bike racks) and 200 bikes. In case of SBBS, number of locations that the rebalancing vehicle(s) has to visit to completely rebalance the system will at most be 100. This is because some stations may be self rebalanced. However, in case of FFBS, number of locations that the rebalancing vehicle(s) has to visit to completely rebalance the system can at times be $>>$ 100. To illustrate this, let us consider the scenario, when all 200 bikes are parked outside of bike racks in standalone locations but the operator wants each bike rack to have 2 bikes each. In this scenario, number of locations that the rebalancing vehicle(s) has to visit to completely rebalance the system is 300, out of which 200 are standalone locations (for pickup) where bikes are parked and 100 are bike racks (for drop offs). Now, let us consider the scenario when instead of 100 bike racks, the system has 300 bike racks and all 200 bikes are parked outside of bike racks in standalone locations. In this case the operator can have at most 200 bike racks to be filled with 1 bike each. In this scenario, number of locations that the rebalancing vehicle(s) has to visit to completely rebalance the system is at most 400, out of which 200 are standalone locations (for pickup) where bikes are parked and 200 out of 300 bike racks where at least 1 bike needs to be dropped off. Thus we can conclude that nodes in the system that a rebalancing vehicle(s) has to visit is $\leq$ *Number of Working Stations* in case of SBBS and is $\leq$ *min {Number of Bike Racks + Number of Bikes,* $2\times$ *Number of Bikes}* in case of FFBS. This fact is also evident from the real life instances introduced in Section 2.8.

Chemla et al. (2013a) was the first to introduce SCRP for SBBS and proposed tabu search algorithms for solving it. Erdoan et al. (2015) proposed a time extended network formulation of SCRP for SBBS and solved it exactly using mixed integer programming. However, the mathematical formulations proposed in Chemla et al. (2013a) and Erdoan et al. (2015) can handle only a single vehicle. Further, the time extended network formulation of SCRP were designed in a manner that it could not be extended for multiple vehicles. This is evident when in Alvarez-Valdes et al. (2016) a heuristic method is proposed for solving SCRP in SBBS for a fleet of multiple vehicles, however the authors are unable to present any mathematical formulation. This issue is being addressed by a mathematical formulation based on spacial decomposition of the network into nodes of unit imbalance each, except for the depot whose imbalance is 0. The proposed formulation, can not only handle single and multiple vehicles, but also allows for multiple visits to a node by the same vehicle. For more details on the proposed formulation see Section 2.4.

Tabu search and exact algorithms proposed in Chemla et al. (2013a) and Erdoan et al. (2015) respectively, are not effective for solving static rebalancing problems even in small or medium scale FFBS or SCRP with multiple vehicles. Thus, in this chapter a heuristic algorithm is proposed, to derive high quality solutions of SCRP with both single as well as multiple vehicles, in a reasonable amount of CPU time. The proposed heuristic consists of creating an initial solution using a greedy construction heuristic and improving it until no further improvement is possible. The improvement heuristic is a hybridization of variable neighborhood descent with large neighborhood search. Seven granular descent operators (Toth and Vigo (2003)) are used for variable neighborhood descent. Four perturbation and three repairing operators were developed, resulting in a total of twelve large neighborhoods. Each of these is explored exhaustively before moving on to the next large neighborhood until no further improvement is possible, resulting in a nested large neighborhood search. For more details on the proposed heuristic see Section 2.5.

Computational experiments on the 1-PDTSP instances from the literature, demonstrate that the presented algorithm outperforms the tabu search algorithm (Chemla et al. (2013a)) and is highly competitive with the exact algorithms (Erdoan et al. (2015)) for solving SCRP in SBBS. It is able to find new solutions for 59 of 148 instances for which the optimal solution is not known and on average 400 and 36 times faster than the exact and the tabu search algorithms proposed in the literature. Computational experiments on the new SABB FFBS instances (consisting of up to 400 nodes, 300 bikes, and a eet size of up to 3 vehicles) and Divvy instances (consisting of 450 stations, 3000 bikes, and a eet size of up to 30 vehicles), demonstrate that NLNS+VND is able to deal with the increase in scale of SCRP for both FFBS and SBBS. It also shows that SCRP is feasible for both SABB program at USF, Tampa and Divvy SBBS at Chicago with the given size of the rebalancing fleet.

The remainder of this chapter is organized as follows. Section 2.3 describes SCRP in detail and presents the literature review of rebalancing operations on bike sharing systems. Section 2.4 describes the mathematical formulation proposed for SCRP. Section 2.5 describes our proposed heuristic for deriving high quality solutions of SCRP. Section 2.6 discusses our recommended strategies for solving different types of SCRP using our proposed methodology. Sections 2.7, 2.8 and 2.9 summarizes the experimental results and conclusions of the three case studies. Section 2.10 concludes the chapter with final remarks.

2.3 Problem Description and Related Work

In recent years, with the boom of SBBS, extensive bike-share related research has been conducted and documented. Related to the operational management of a bike-sharing system, the literature can be grouped into three major research sub-streams: demand analysis, service-level analysis and rebalancing strategies. Service-level and demand Analysis are beyond the scope of this chapter and will be summarized in a future article we are working on. In this article, we focus only on the literature relevant to rebalancing operations. Rebalancing a bike sharing system can be achieved in various ways, illustrated in Figure 2.1. In Figure 2.1, operator and user based strategies refers to manually rebalancing the bikes in the network using a fleet of rebalancing trucks and incentivizing users to encourage them to self-rebalance the bikes in the network respectively. If the manual rebalancing is done when the user intervention is negligible, it is known as static rebalancing, whereas if it is done when there is significant user intervention, the rebalancing is known as dynamic rebalancing. Further, in SCRP the operator meets the target inventory level at all the nodes in the network exactly. However, if the resources available (rebalancing time or number of rebalancing vehicles) to the operator is not sufficient for complete rebalancing, partial target inventory levels are met at certain or at all the nodes. This is known as partial rebalancing (SPRP).

A summary of the recent literature of operator based rebalancing strategies in SBBS is provided in Table 2.1. In terms of user based rebalancing strategies for SBBS, Chemla et al. (2013b) and Pfrommer et al. (2014) presented dynamic pricing strategies, which encourage users to return bikes to empty (or unsaturated) nodes. Singla et al. (2015) extended the model of Pfrommer et al. (2014) by incorporating a learning mechanism to shape the user utility function, and enriched it by taking into account a budget constraint

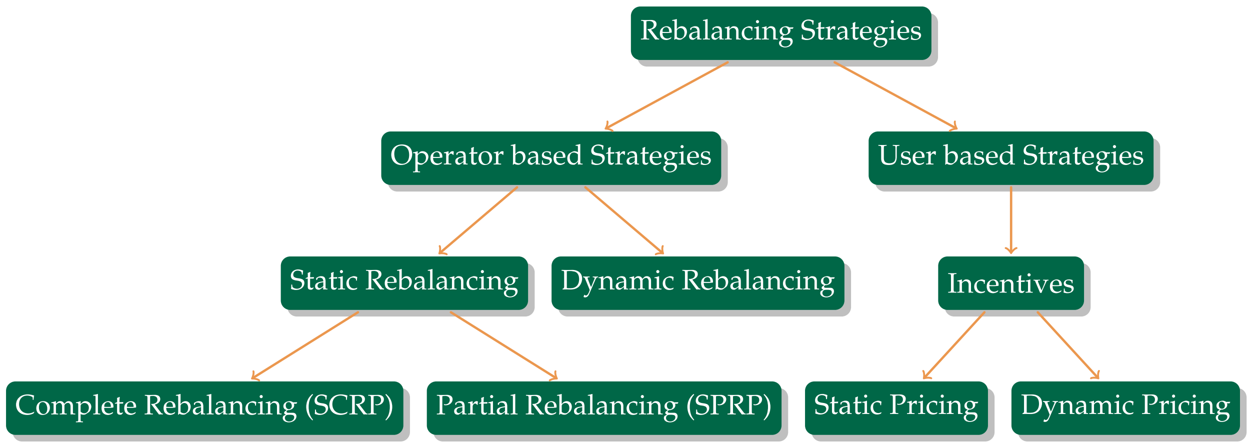


Figure 2.1: Subdivisions of Rebalancing Strategies

for the operator. For a more detailed literature review of rebalancing strategies in station based shared mobility systems, the readers are referred to Laporte et al. (2015). From the literature review, it can be concluded that only Reiss and Bogenberger (2015) reported a study conducted on FFBS. Weikl and Bogenberger (2013) and Boyacı et al. (2015) report SPRP schemes for one-way station-based (SBCS) and Free Floating Car Sharing (FFCS) systems respectively. However, no article has reported any SCRP scheme for either FFBS or FFCS.

Table 2.1: Summary of Literature of Operator Based Rebalancing Strategies

Research Article	Type of Rebalancing	Subtype	Fleet Size	Stations	Methodology
Chemla et al. (2013a)	Static Rebalancing	Complete Rebalancing	1	100	Tabu Search
Erdoan et al. (2015)			1	60	Exact algorithm based on Bender's Cuts
Alvarez-Valdes et al. (2016)			2	28	Heuristic based on Minimum Cost Flow
Raviv et al. (2013)		Partial Rebalancing	2	60	Mixed Integer Programming (MIP)
Forma et al. (2015)			3	200	3-Step Matheuristic
Ho and Szeto (2014)			1	400	Iterated Tabu Search
Schuijbroek et al. (2017)			5	135	Constraint Programming and MIP
Erdoan et al. (2014)			1	50	Branch and Cut and Bender's Decomposition
Dell'Amico et al. (2014)			1	116	Branch and Cut
DellAmico et al. (2016)			1	500	Metaheuristic based on Destroy and Repair
Rainer-Harbach et al. (2015)			21	700	Combination of Greedy Heuristics, GRASP and VNS
Szeto et al. (2016)			1	300	Chemical Reaction Optimization
Ho and Szeto (2017)			5	518	Hybrid Large Neighborhood Search
Pfrommer et al. (2014)	Dynamic Rebalancing		1	-	Greedy Heuristics
Contardo et al. (2012)			1	100	A hybrid MIP approach using Dantzig-Wolfe and Benders decomposition
Regue and Recker (2014)			1	-	MIP
Kloimüllner et al. (2014)			5	90	Combination of Greedy Heuristics, GRASP and VNS

In this chapter, the objective function of SCRP is minimizing the makespan of the fleet of rebalancing vehicles. This is equivalent to minimizing the maximum rebalancing time of the fleet of rebalancing vehicles, as is done in Schuijbroek et al. (2017). However, if the fleet consist of a single vehicle, it is equivalent to minimizing the total distance traversed by the rebalancing vehicle, as is done in Chemla et al. (2013a) and Erdoan et al. (2015). Different studies in the literature have minimized different objective functions, including weighted sum of two or more measures. The most important measures being:

1. Travel cost
2. Total redistribution (travel + loading and unloading) cost
3. Total absolute deviation from the target number of bikes
4. Total unmet customer demand or a penalty cost
5. Makespan of the rebalancing fleet - Measure used in this chapter

For SCRP, options 3 and 4 are not applicable as the system is completely rebalanced. In this chapter, the objective function of SCRP is to minimize the make-span of the fleet of rebalancing vehicles. If the fleet consists of a single vehicle, it is equivalent to minimizing the total distance (travel cost as well) traversed by the rebalancing vehicle. If multiple vehicles are needed, minimizing the make-span is better than minimizing the total travel distance (travel cost) because the latter may create problems that one vehicle is alloted a rebalancing trip whose time is significantly longer than that of another vehicle. Minimizing make-span determines the needed time allotted to bike rebalancing and make sure that rebalancing workload more evenly distributed to multiple vehicles. It decreases the variance of the rebalancing time of the fleet while at the same time minimizing the overall rebalancing time to a great extent.

Preemption is not allowed in the tours of the rebalancing vehicles, which means that nodes can not be used as buffers for storing bikes. For more details pertaining to preemption in SCRP, the readers are referred to Chemla et al. (2013a) and Erdoan et al. (2015). Allowing preemption only increases the computational complexity of the mixed integer linear program (Section 2.4) and the solution algorithm (Section 2.5) without much improvement in solution quality. The computational complexity increases because the number of (possible) nodes to visit increases when preemption is allowed. If preemption is not allowed, nodes which are already balanced can be removed in the preprocessing phase. The solution space also decreases because the inventory level can only either increase or decrease monotonically from the initial inventory levels to the target inventory levels. Erdoan et al. (2015) empirically showed that, preemption adds a value of 0.6%, at most in the solution quality, for the 1-PDTSP instances used in the literature.

Mathematical formulations proposed in the literature for SCRP can only handle a single vehicle. The formulations were designed in a manner that they could not be extended for multiple vehicles. This is evident when in Alvarez-Valdes et al. (2016) a heuristic method is proposed for SCRP in SBBS using multiple vehicles, however the authors are unable to present any mathematical formulation of SCRP with multiple vehicles. This issue is addressed by introducing a mathematical formulation based on spacial decomposition of the network into nodes of unit imbalance each, except for the depot whose imbalance is 0. Our formulation, can not only handle single and multiple vehicles, but also allows for multiple visits to a node by the same vehicle. In the existing literature, to deal with the scale of the static rebalancing problem, researchers have sacrificed solution quality by limiting the number of visits to a node to, at most, once. If multiple visits to a node are allowed, the solution algorithms are unable to cope with the scale of the problem and become ineffective for nodes greater than 50. The proposed heuristic addresses both of these issues, i.e., retaining solution quality with increase in the scale of the static rebalancing problem while allowing multiple visits to a node. Our solution algorithm can also handle a fleet size of 30 vehicles.

2.4 Mathematical Formulation of SCRP

Bike rebalancing network consists of nodes with non-zero imbalance and a depot with 0 imbalance. Figure 2.2 is an example of a network consisting of three nodes, Node 1 or the Depot, Node 2 with a positive imbalance of 2 (at Node 2 there is a surplus of two bikes) and Node 3 with a negative imbalance of 2 (at Node 3 there is a deficit of two bikes). Erdoan et al. (2015) was able to formulate the SCRP for a fleet of a single vehicle using such a network (named original network thereafter). However, it is mathematically challenging to formulate a SCRP with a fleet of multiple vehicles using the original network, as the number of visits to a node by a vehicle is unknown apriori. To address this issue, each node (other than the depot) in the original network is decomposed into nodes each with unit imbalance, but at same geographic location. Figure 2.3 is the decomposed network of the original network in Figure 2.2. In the decomposed network, Node 2 from the original network is decomposed into two nodes 2_1 and 2_2, with the same geographic location as Node 2 but each with a positive unit imbalance. Similarly, Node 3 from the original network is decomposed into two nodes 3_1 and 3_2, with the same geographic location as Node 3 but each with a negative unit imbalance. Table 2.2 describes the notations used in the rest of the chapter.

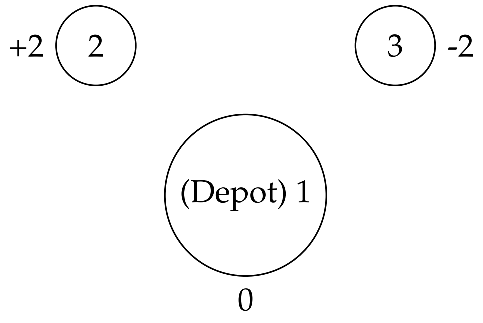


Figure 2.2: Original Network

In the decomposed network, SCRP becomes feasible to formulate, because the number of visits to a node in the decomposed network must equal one. This is because, multiple visits to a node may be required, if and only if its absolute imbalance is strictly greater than one. However, every node in the decomposed network has a *unit absolute imbalance*. Now, SCRP with multiple vehicles can be formulated as a multiple traveling salesman problem (m-TSP) with additional constraints, on the decomposed network. For a detailed overview on m-TSP, readers are referred to Laporte and Nobert (1980) and Bektas (2006).

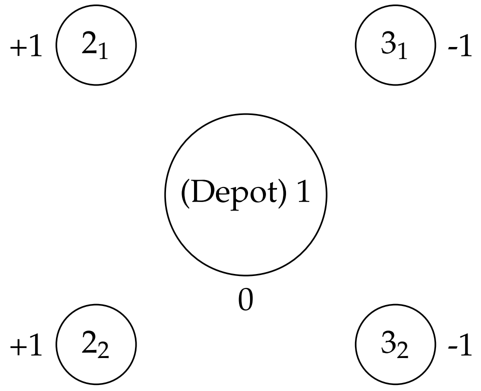


Figure 2.3: Decomposed Network

Let us now consider two other variants of the Traveling Salesman Problem (TSP) - 1 Commodity Pickup and Delivery TSP (1-PDTSP) and Q-TSP or also known as Capacitated TSP with Pickups and Deliveries (CTSPPD). 1-PDTSP (Hernández-Pérez and Salazar-González (2004)) is a generalization of TSP, in which nodes (providing or requiring known amounts of a product) must be visited exactly once by a vehicle (with a given capacity) serving the imbalances, while minimizing the total travel distance. Q-TSP (Chalasani and Motwani (1999)) or CTSPPD (Anily and Bramel (1999)) is a special case of 1-PDTSP, where the delivery and pickup quantities are all equal to one unit.

SCRP with a single rebalancing vehicle on the original network is not equivalent to 1-PDTSP, because in 1-PDTSP number of visits to a node is limited to exactly once, irrespective of the imbalances at the node. This limitation on the number of visits to once, makes 1-PDTSP relatively easy to solve compared to SCRP. However, SCRP on the decomposed network with a single vehicle is equivalent to formulating a 1-PDTSP, a Q-TSP or a CTSPPD problem. However, in order to extend the formulation to handle a fleet of (homogeneous) multiple vehicles, our formulation is based on m-TSP rather than on 1-PDTSP, Q-TSP or CTSPPD, as doing so reduces the quantity of decision variables. Thus, our contribution in terms of the mathematical programming formulation is coming up with a simple, yet effective scheme to decompose the original network, so that existing formulations of different variants of TSP and Vehicle Routing Problem (VRP) can be used to formulate SCRP with both single and multiple vehicles.

After a feasible solution of SCRP on the decomposed network is obtained, it can be converted to a feasible solution for SCRP on the original network, by switching the indices of nodes in the tour(s) of the rebalancing vehicles, with their respective indices in the original network. For example, let us consider SCRP with a single vehicle of capacity 2, on the original network in Figure 2.2. However, if instead of choosing to use the original network, its corresponding decomposed network (Figure 2.3) is used to compute a feasible solution (Figure 2.4), it can be converted to a solution feasible on the original network (Figure 2.5) by switching the indices of nodes in the tour with their respective indices in the original network. One more thing that can be done, is to combine consecutive locations with same node indices together into one location and their instruction equal to the sum of that of those corresponding locations. Now, let us consider SCRP with a single vehicle of capacity 1, on the original network in Figure 2.2. Figure 2.6 represents a feasible solution on the decomposed network and Figure 2.7 represents the corresponding feasible solution on the original network. This example illustrates how multiple visits to a node in the original network translates into single visit to a node in the decomposed network.

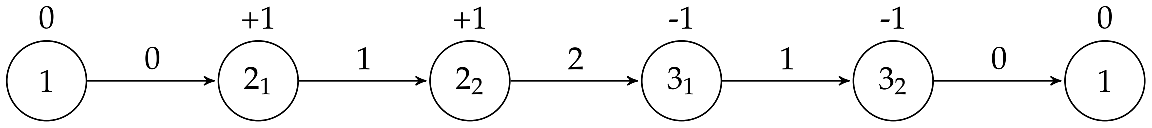


Figure 2.4: Feasible Solution for the Decomposed Network with a Single Vehicle of Capacity 2

Before moving onto the Mixed Integer Linear Program (MILP) of SCRP on the decomposed network, we have to understand that in this article, we only address SCRP and not SPRP. For a particular instance to be feasible for SCRP, the total positive imbalance must equal the total negative imbalance, i.e, $\sum_{i \in \mathcal{N}} d_i = 0$ on the original network and by ex-

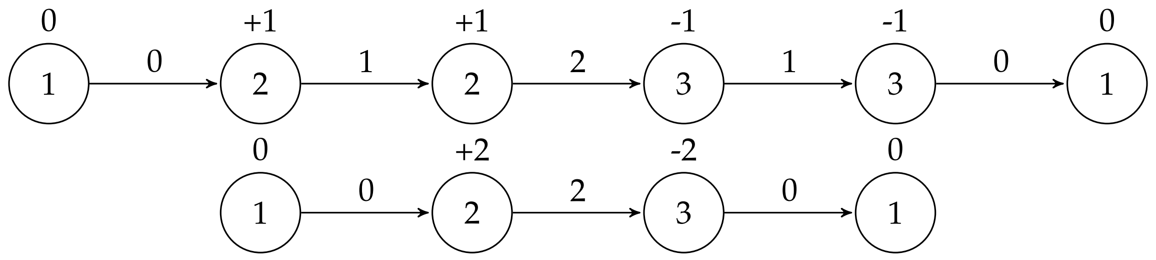


Figure 2.5: Corresponding Feasible Solution for the Original Network with a Single Vehicle of Capacity 2

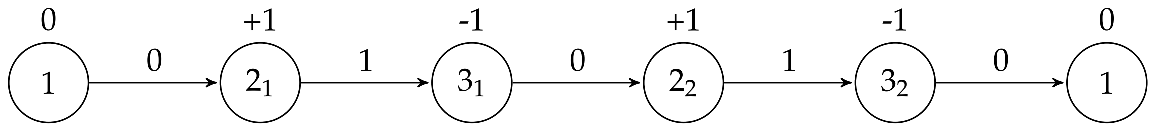


Figure 2.6: Feasible Solution for the Decomposed Network with a Single Vehicle of Capacity 1

tension $\sum_{i \in \mathcal{N}_o} \bar{d}_i = 0$ on the decomposed network. From a modeling point of view, SCRP for SBBS is exactly similar to that for FFBS, i.e, same set of equalities and inequalities define the constraints. The only difference is from a computational point of view. In case of FFBS, the scale (number of nodes to rebalance and or number of bikes to rebalance) of SCRP can become very large compared to that of SBBS with a similar scale (total number of nodes and total number of bikes in the system).

2.1 - 2.12 is the MILP of SCRP on the decomposed network. There are three decision variables in this formulation:

- τ_i is the arrival time of a rebalancing vehicle at node $i, \forall i \in \mathcal{N}_o$
- $x_{ij} = 1$, if Edge (i, j) is traversed by a rebalancing vehicle, otherwise it is $0, \forall (i, j) \in \mathcal{E}_o$.
- q_{ij} is the quantity of bikes carried by a rebalancing vehicle on Edge $(i, j), \forall (i, j) \in \mathcal{E}_o$.

In this formulation, the objective function (2.1) is to minimize the make-span of the rebalancing fleet, i.e, τ_1. Constraints 2.2 and 2.3 are the Miller-Tucker-Zemlin Subtour

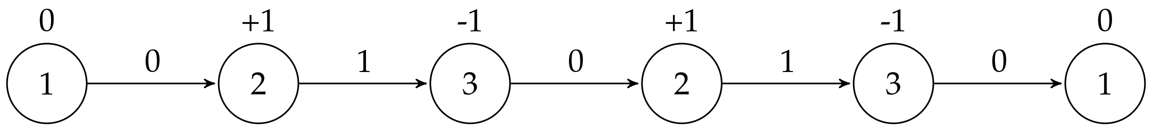


Figure 2.7: Corresponding Feasible Solution for the Original Network with a Single Vehicle of Capacity 1

Elimination constraints. Constraint 2.4 makes sure, that the number of visits to equals the number of exits from a node. Constraint 2.5 makes sure, that every node, other than the depot is visited exactly once. As there are $|\mathcal{V}|$ rebalancing vehicles and the flow is directional (because of capacity constraints), the number of uniques flows through the depot must be equal to $|\mathcal{V}|$, which is what Constraint 2.6 is. Each of the unique flow out of $|\mathcal{V}|$ flows, starting and ending at the depot corresponds to the tour for a rebalancing vehicle. Constraint 2.7 prevents self loops in the tour of rebalancing vehicles. Constraint 2.8 are the complete rebalancing constraints for each node in the graph. Constraint 2.8 ensures that the imbalance is met exactly at each node during rebalancing. Constraint 2.9 are the capacity constraints of the rebalancing vehicles.

$$\min \tau_1 \tag{2.1}$$

$$\text{s.t. } \tau_i \geq \tau_j + \bar{c}_{ji} x_{ji} + \bar{\tau} - M(1 - x_{ji}), \forall i \in \mathcal{N}_o, j \in \mathcal{N}_o \setminus \{1\} \tag{2.2}$$

$$\tau_i \geq \bar{c}_{1i} x_{1i} - M(1 - x_{1i}), \forall i \in \mathcal{N}_o \tag{2.3}$$

$$\sum_{j \in \mathcal{N}_o} x_{ji} = \sum_{j \in \mathcal{N}_o} x_{ij}, \forall i \in \mathcal{N}_o \tag{2.4}$$

$$\sum_{j \in \mathcal{N}_o} x_{ji} = 1, \forall i \in \mathcal{N}_o \setminus \{1\} \tag{2.5}$$

$$\sum_{j \in \mathcal{N}_o} x_{j1} = |\mathcal{V}| \tag{2.6}$$

$$\sum_{i \in \mathcal{N}_o} x_{ii} = 0 \tag{2.7}$$

$$\sum_{j \in \mathcal{N}_o} q_{ij} - \sum_{j \in \mathcal{N}_o} q_{ji} = \bar{d}_i, \forall i \in \mathcal{N}_o \tag{2.8}$$

$$q_{ij} \leq \mathcal{Q} x_{ij}, \forall (i,j) \in \mathcal{E}_o \tag{2.9}$$

$$\tau_i \geq 0, \forall i \in \mathcal{N}_o \tag{2.10}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{E}_o \tag{2.11}$$

$$q_{ij} \in \mathcal{Z}^+, \forall (i,j) \in \mathcal{E}_o \tag{2.12}$$

To illustrate how the above MILP can handle multiple vehicles, let us consider the network presented in Figure 2.8. Figure 2.9 presents a feasible solution on the network of Figure 2.8 by a fleet consisting of 2 rebalancing vehicles with $\mathcal{Q} = 1$. Figures 2.10 and 2.11 are the tours of Vehicle 1 and 2 respectively, for the feasible solution of Figure 2.9

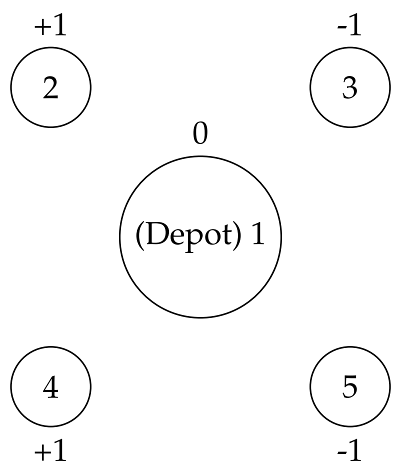


Figure 2.8: Example Network

respectively. Further from Figures 2.10 and 2.11, we can conclude that

$$\tau_2 \geq \bar{c_{12}} \tag{2.13}$$

$$\tau_3 \geq \tau_2 + \bar{c_{23}} + \bar{\tau} \tag{2.14}$$

$$\tau_1 \geq \tau_3 + \bar{c_{31}} + \bar{\tau} \tag{2.15}$$

$$\tau_4 \geq \bar{c_{14}} \tag{2.16}$$

$$\tau_5 \geq \tau_4 + \bar{c_{45}} + \bar{\tau} \tag{2.17}$$

$$\tau_1 \geq \tau_5 + \bar{c_{51}} + \bar{\tau} \tag{2.18}$$

As our problem is a minimization problem, the above set of inequalities becomes equations as follows:

$$\tau_2 = \bar{c_{12}} \tag{2.19}$$

$$\tau_3 = \tau_2 + \bar{c_{23}} + \bar{\tau} \tag{2.20}$$

$$\tau_4 = \bar{c_{14}} \tag{2.21}$$

$$\tau_5 = \tau_4 + \bar{c_{45}} + \bar{\tau} \tag{2.22}$$

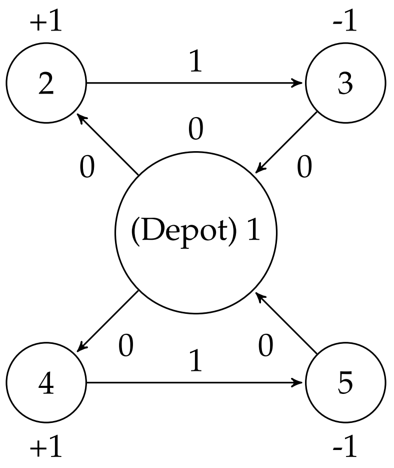


Figure 2.9: A Feasible Solution of the Network in Figure 2.8 for a Fleet Consisting of 2 Rebalancing Vehicles

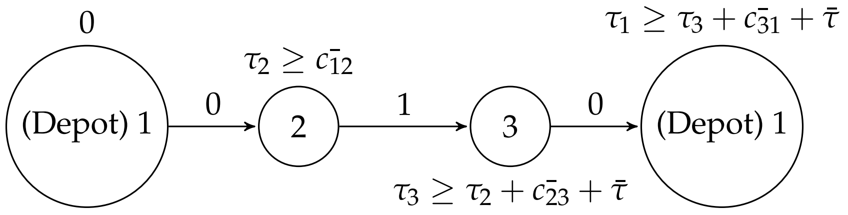

0
$\tau_1 \geq \tau_3 + \bar{c_{31}} + \bar{\tau}$
$\tau_2 \geq \bar{c_{12}}$
(Depot) 1
0
2
1
3
0
(Depot) 1
$\tau_3 \geq \tau_2 + \bar{c_{23}} + \bar{\tau}$

Figure 2.10: Tour of Vehicle 1 for the Feasible Solution in Figure 2.9

The above set of equations is pretty obvious except for the case of τ_1. If $\tau_3 + \bar{c_{31}} + \bar{\tau} > \tau_5 + \bar{c_{51}} + \bar{\tau}$, $\tau_1 = \tau_3 + \bar{c_{31}} + \bar{\tau}$, otherwise $\tau_1 = \tau_5 + \bar{c_{51}} + \bar{\tau}$. In case $\tau_3 + \bar{c_{31}} + \bar{\tau} = \tau_5 + \bar{c_{51}} + \bar{\tau}$, τ_1 can take either values. Thus τ_1 equals the arrival time of the rebalancing vehicle that arrives last at the depot.

The proposed formulation is computationally intractable even for small scale instances owing to the presence of Big M in the constraints and (geographic) symmetry in the decision variables, which make the linear programming relaxation of the formulation extremely weak. The Big M constraints are used for subtour elimination. Another reason for the computational intractability of the formulation is the significant increase in the number of variables in the formulation owing to spacial decomposition. It is directly proportional to square of the number of bikes to be rebalanced. Several strategies for

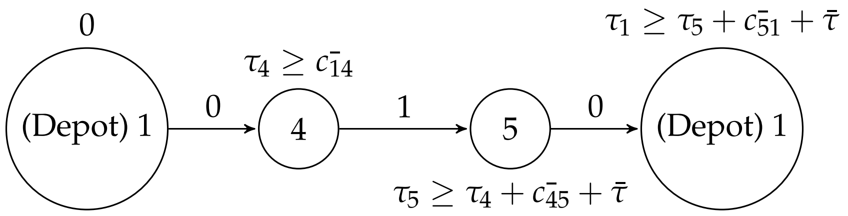

0
(Depot) 1
0
$\tau_4 \geq \bar{c_{14}}$
4
1
5
$\tau_5 \geq \tau_4 + \bar{c_{45}} + \bar{\tau}$
0
$\tau_1 \geq \tau_5 + \bar{c_{51}} + \bar{\tau}$
(Depot) 1

Figure 2.11: Tour of Vehicle 2 for the Feasible Solution in Figure 2.9

strengthening and making the formulation computationally tractable is mentioned in Section 2.10.

2.5 Proposed Heuristic

The algorithm proposed in this section is a hybrid of Nested Large Neighborhood Search and Variable Neighborhood Descent. It will be referred to as NLNS+VND in the rest of the article. NLNS+VND was greatly influenced by the success of metaheuristic algorithms based on perturbation and repair (Helsgaun (2000), Ahuja et al. (2002), Applegate et al. (2003), Ghilas et al. (2016), and Helsgaun (2009)) for solving large scale traveling salesman and vehicle routing problems. NLNS+VND can solve SCRP on both the original and the decomposed network. There are major differences between NLNS+VND and previous algorithms reported in the literature (Chemla et al. (2013a), Erdoan et al. (2015)) for solving SCRP in bike sharing systems. NLNS+VND consists of three primary components each of which have various sub-components. The three primary components are creating the initial solution, variable neighborhood descent for intensification and large neighborhood search for diversification. Each of these components and their corresponding sub-components are elaborated in great details in the subsequent sections. NLNS+VND has been coded in Julia (v 0.5.0) (Bezanson et al. (2012)). The sourcecode of the implementation is available at https://github.com/aritrasep/NLNS+VND.jl.

2.5.1 Initial Solution

2.5.1.1 Single Vehicle

For a fleet consisting of a single rebalancing vehicle, an initial solution is created using a greedy construction heuristic. Unlike Chemla et al. (2013a), both $(\mathcal{T}_v, \mathcal{I}_v)$ are created simultaneously. Algorithm 1 is the pseudo-code of the greedy construction heuristic. On lines 4 and 10, in Algorithm 1, the function *Maximum operations for every other node()* computes the maximum operation that can be performed at a node other than the current node, if that node is visited next from the current node. When computing maximum operation, only operations remaining at a node are taken into consideration. The procedure for computing Maximum Operations (or MaxOps) is very simple. First, we have to understand that $\text{sum}(I) =$ current number of bikes in the corresponding rebalancing vehicle. With this in mind, we can define $\text{MaxOps}_i = \begin{cases} \min\{d_i^o, Q - \text{sum}(I)\} & \text{if } d_i^o > 0 \\ \max\{d_i^o, -\text{sum}(I)\} & \text{if } d_i^o < 0 \\ 0 & \text{if } d_i^o = 0 \end{cases}, \forall i \in \mathcal{N}$. The *Nearest Neighbor Function()* in Algorithm 1 can be computed in three different ways. For all three different functions, only nodes with a non-zero maximum operation ($\text{MaxOps}_i \neq 0$), are considered for determining *Next node*.

- Nearest Neighbor 1: The nearest neighbor 1 of a node, is the node with the minimum (traveling) cost from the current node ,i.e, arg min (c_{ij}) , $\forall j \in \mathcal{N} \setminus \{i\}$; i being the current node.

- Nearest Neighbor 2: The nearest neighbor 2 of a node, is the node that has the maximum value of $\frac{|\text{Max Ops}_j|}{c_{ij}}, \forall j \in \mathcal{N} \setminus \{i\}$; i being the current node.

- Nearest Neighbor 3: The nearest neighbor 3 of a node, is a random node (where a non-zero operation is left) other than the current node and the depot.

From Line 5 in Algorithm 1, it is evident that the Next Node of the Depot is chosen randomly irrespective of what *Nearest Neighbor Function()* is used to generate the initial solution. This is because, in our experiments *Nearest Neighbor 2* is used to generate 10 different starting solutions. If the Next Node of the Depot is chosen according to *Nearest Neighbor 2*, all the 10 starting solutions will exactly be the same.

Algorithm 1: Greedy Construction Heuristic

```
Data: $\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$, Nearest Neighbor Function
Result: $(\mathcal{T}_v, \mathcal{I}_v)$
1  $\mathcal{T}_v \longleftarrow$ [Depot]
2  $\mathcal{I}_v \longleftarrow$ [0]
3  $\mathrm{d_i^o} \longleftarrow \mathrm{d}_i$
4  Max Ops $\longleftarrow$ Maximum operations for every other node($\mathrm{d_i^o}$, $\mathcal{Q}$, $sum(\mathcal{I}_v)$)
5  Next node $\longleftarrow$ Randomly select any node from 2 to $|\mathcal{N}|$
6  Add Next node at the end of $\mathcal{T}_v$
7  Add Max $\mathrm{Ops_{Next\ node}}$ at the end of $\mathcal{I}_v$
8  $\mathrm{d^o_{Next\ node}} = \mathrm{d^o_{Next\ node}} - \mathrm{Max\ Ops_{Next\ node}}$
9  while Number of non zero elements in $d_i^o > 0$ do
10     Max Ops $\longleftarrow$ Maximum operations for every other node($\mathrm{d_i^o}$, $\mathcal{Q}$, $sum(\mathcal{I}_v)$)
11     Next node $\longleftarrow$ Nearest Neighbor Function($\mathrm{d_i^o}$, $\mathrm{c_{ij}}$, Next node, Max Ops)
12     Add Next node at the end of $\mathcal{T}_v$
13     Add Max $\mathrm{Ops_{Next\ node}}$ at the end of $\mathcal{I}_v$
14     $\mathrm{d^o_{Next\ node}} = \mathrm{d^o_{Next\ node}} - \mathrm{Max\ Ops_{Next\ node}}$
15 end
16 Add Depot at the end of $\mathcal{T}_v$
17 Add 0 at the end of $\mathcal{I}_v$
```

2.5.1.2 Multiple Vehicles

When $|\mathcal{V}| > 1$, a partition first rebalance second approach is used. For partitioning, two strategies have been tried, first partitioning based on geographic locations of the nodes and second partitioning randomly. During our experiments, it was observed that, on average partitioning randomly is not only faster but also results in higher quality solutions for large scale instances. Thus, partitions are created randomly. However, the partition created in this stage is by no means the final partition, because the overall solution may get stuck in a local optima. To address this, local search operators *INTER*

Crossover, INTER Node Swapping, INTER Edge Exchange have been proposed, to improve the quality of the solution by modifying the partitions as needed.

Exactly $|\mathcal{V}|$ partitions are made from the set of nodes $\mathcal{N}$, such that for each partition, the total deficit of bikes equals the total surplus of bikes. While partitioning, a node $\in \mathcal{N}$ may be split into 2 to up to $|\mathcal{V}|$ nodes, each of them being in a separate partition, however the sum of their absolute imbalance in each partition, must equal their total original absolute imbalance. This is necessary to satisfy the condition that, total deficit of bikes equals total surplus of bikes in each partition. Once $|\mathcal{V}|$ partitions have been created, Algorithm 1 is used to create an initial solution for each partition.

2.5.2 Variable Neighborhood Descent (VND)

In VND, the feasibility of the solution is maintained during the subsequent local search operations, i.e., no capacity constraints are violated. To keep a solution feasible in the following operations, when an operation is performed on T_v, the corresponding operation is also performed on I_v and vice versa. This has significant advantages, as it results in a decrease in the size of the neighborhood of an incumbent solution to a great extent, taking considerably less time to explore them. Further, a candidate list based on nearest neighbors is used to rank the edges in a tour. Only operations that result in a tour whose highest edge rank is less than or equal to the highest edge rank of the current tour are allowed. This prevents exploration of unwanted regions in a neighborhood and subsequently reduces exploration time, making the operation extremely granular (Toth and Vigo (2003)). In total, seven such granular local search operators have been developed and used successively inside Variable Neighborhood Descent (VND). Operations which have an *INTRA* or an *INTER* prefix signifies that the operation takes place inside a solution of a single vehicle or the operation takes place between solutions of two vehicles respectively. Further a *K-OPT Operation* means that *K* variables are altered from their present state, in this case *K* edges are changed. The higher the value of *K*, the more is the likely improvement, but it comes at the cost of computing time. For a detailed understanding and analysis of *K-OPT Operations* the readers are referred to Helsgaun (2000) and Helsgaun (2009).

2.5.2.1 Local Search Operators for Single Vehicle

1. INTRA Delete-Reinsert: In this case, one node is deleted from a location and inserted in another location in $\mathcal{T}_v$ while maintaining feasibility of the solution. At each iteration, starting from the left (or beginning) of $\mathcal{T}_v$, each node (other than the depot) is inspected to determine if it can be deleted and reinserted either ahead or behind its current location in $\mathcal{T}_v$. If a valid location(s) is (are) found that reduces the cost of $\mathcal{T}_v$, then the (best) operation is made. An iteration is completed on reaching the penultimate location in $\mathcal{T}_v$. This procedure is continued until no further improvement is possible.

2. INTRA Node Swapping: Two 4-OPT neighborhoods based on Intra Node Swapping have been developed. Let k & l be locations of two nodes in $\mathcal{T}_v$, such that $k \neq l, k < l, k \neq 1$ & $l \neq |\mathcal{T}_v|$. If $\mathcal{I}_{v_k} = \mathcal{I}_{v_l}$ and $\mathcal{T}_{v_k} \neq \mathcal{T}_{v_l}$, then the two locations k and l can be swapped to obtain a new feasible tour. If the swapping results in a tour with a lower cost, the swapping is confirmed. At each iteration, each location (k) of $\mathcal{T}_v$ is inspected for a possible swap. If a valid swap(s) is(are) found that reduces the current cost of $\mathcal{T}_v$, the (best) swap is made. An iteration is completed on reaching the penultimate location in $\mathcal{T}_v$. This procedure is continued until no further improvement is possible. Another variation of the above procedure is, let k and l be locations of two nodes in $\mathcal{T}_v$, such that $k \neq l, k < l, k \neq 1$ and $l \neq |\mathcal{T}_v|$. If $\mathcal{I}_{v_k} \neq \mathcal{I}_{v_l}$ and $\mathcal{T}_{v_k} \neq \mathcal{T}_{v_l}$, and swapping the two nodes at k and l results in a new feasible tour with a lower cost, the swapping is confirmed. Everything else is exactly same as the above procedure.

3. INTRA Edge Exchange: Let us consider four edges e_k, e_l, e_m, e_n, connecting nodes at location k, l, m, n to nodes at location $k+1, l+1, m+1, n+1$ in $\mathcal{T}_v$ respectively, such that $k < l < m < n$. If the flow of bikes (= number of bikes carried by the rebalancing vehicle) on edges e_k and e_l and that on edges e_m & e_n are equal, the tour segment $(k+1) \rightarrow l$ can be swapped with the tour segment $(m+1) \rightarrow n$ without violating any capacity constraints. If such a operation results in a tour with a lower cost than the current one, that operation is confirmed. At each iteration, a step ($= l - k - 1$) is fixed and all possible values of m and n are checked for possible exchanges. If no improvement is possible for the current value of step, its value is incremented by 1. The value of step is initialized with 1 and can at most be increased to $|\mathcal{T}_v| - 3$, otherwise there will be overlapping of the tour segments. On reaching this value of step, the operation is terminated.

4. INTRA Adjust Instructions: In this operation, number of edges broken, which is variable equals number of edges inserted. The number of edges broken or inserted, can be represented as $2n$, where $n \in Z^+$. This operation comprises of nodes, that are visited multiple times in the current tour. The objective of this neighborhood is to drive the loading-unloading instructions, at viable locations in the current tour, of such nodes towards 0. If this can be achieved for one or more locations, the corresponding locations can be removed from the tour. This is because, the instances (used in this chapter) are metric, and removing nodes at locations in the current tour, for which the instruction is 0, reduces the cost of the tour without making it infeasible. Let a node be present at the k^{th} and $l^{th}, k < l$ locations in $\mathcal{T}_v$. If $|I_{v_k}| \leq |I_{v_l}|, I_{v_k}$ is driven towards 0 by transferring instructions between I_{v_k} and I_{v_l}, while maintaining feasibility of the flow of bikes in the tour segment $k \longrightarrow l$ and vice versa. This procedure is executed for all possible combinations of k and l for each node with multiple visits in $\mathcal{T}_v$. This operation also serves a secondary purpose.

Instructions at some locations in the current tour are altered, which might create new operations for other local search operators described earlier.

2.5.2.2 Local Search Operators for Multiple Vehicles

1. INTER Crossover: This is a 2-OPT operation in which part of the tour of a vehicle is swapped with part of the tour of another vehicle while maintaining feasibility of the solution. At each iteration, starting from the left (or beginning) of $\mathcal{T}_{v1}$ (tour of first vehicle), flow of bikes on each edge is inspected to determine if it can be swapped with an edge on $\mathcal{T}_{v2}$ (tour of another vehicle) with exactly same flow of bikes. If a valid edge(s) is (are) found that reduces the current make-span, then the (best) operation is made. An iteration is completed on reaching the penultimate location in $\mathcal{T}_{v1}$. This procedure is continued until no further improvement is possible.

2. INTER Node Swapping: Let k and l be location of two nodes in $\mathcal{T}_{v1}$ and $\mathcal{T}_{v2}$ respectively. If $\mathcal{I}_{v1_k} = \mathcal{I}_{v2_l}$ and $\mathcal{T}_{v1_k} \neq \mathcal{T}_{v2_l}$, then the two locations k and l can be swapped without violating any capacity constraints. If the swapping results in a decrease in the current make-span, the swapping is confirmed. At each iteration, each location (k) of $\mathcal{T}_{v1}$ is inspected for a possible swap. If a valid swap(s) is(are) found that reduces the current cost of the tour, the (best) swap is made. An iteration is completed on reaching the penultimate location in $\mathcal{T}_{v1}$. This procedure is continued until no further improvement is possible.

3. INTER Edge Exchange: Let us consider four edges e_k, e_l, e_m, e_n, connecting nodes at location k, l, m, n to nodes at location $k+1, l+1, m+1, n+1$ in $\mathcal{T}_{v1}$ and $\mathcal{T}_{v2}$ respectively, such that $k < l$ and $m < n$. If the flow of bikes (= number of bikes carried by the rebalancing vehicle) on edges e_k and e_l and that on edges e_m and e_n are equal, the tour segment $(k+1) \rightarrow l$ in $\mathcal{T}_{v1}$ can be swapped with the tour segment $(m+1) \rightarrow n$ in $\mathcal{T}_{v2}$ without violating any capacity constraints. If such a operation results in a

decrease of the make-span, that operation is confirmed. At each iteration, a step ($= l - k - 1$) is fixed and all possible values of m and n are checked for possible exchanges. If no improvement is possible for the current value of step, its value is incremented by 1. The value of step is initialized with 1 and can at most be increased to length of the current tour - 5, otherwise there will be overlapping of the tour segments. On reaching this value of step, the procedure is terminated.

Algorithm 2 is the pseudo-code of VND, used in NLNS+VND. In VND, local search operations (described in the above sub-sections) are done sequentially in an iterative manner on an incumbent solution, until no further improvement is possible. The order in which the operations is carried out is a crucial factor for the improvement to be substantial. In VND, it is based on the computational complexity and the execution time of an individual operation. The order used in VND is as follows:

- INTRA Adjust Instructions (Operation_1)
- INTRA Delete-Reinsert (Operation_2)
- INTRA Node Swap (Operation_3)
- INTRA Edge Exchange (Operation_4)
- INTER Crossover (Operation_5)
- INTER Node Swap (Operation_6)
- INTER Edge Exchange (Operation_7)

2.5.3 Large Neighborhood Search (LNS)

With the decrease in the size of the neighborhood used in VND, finding high-quality solutions becomes extremely challenging. To overcome this, VND is hybridized with

Algorithm 2: Variable Neighborhood Descent

```
Data: $\mathcal{T}, \mathcal{I}, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}, \bar{\tau}$
Result: $(\mathcal{T}, \mathcal{I})$
STOP $\longleftarrow$ FALSE
while $STOP = FALSE$ do
    for $v \leftarrow 1$ to $|\mathcal{V}|$ do
        while $Cost(\mathcal{T}_v) \neq Cost(\mathcal{T}_v^{previous})$ do
            for $i_o \leftarrow 1$ to 4 do
                $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow \mathrm{Operation}_{i_o}(\mathcal{T}_v, \mathcal{I}_v, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}})$
                if $Cost(\bar{\mathcal{T}}_v) < Cost(\mathcal{T}_v)$ then
                    $\mathcal{T}_v \longleftarrow \bar{\mathcal{T}}_v$
                    $\mathcal{I}_v \longleftarrow \bar{\mathcal{I}}_v$
                end
            end
        end
        STOP $\longleftarrow$ TRUE
    end
    if $|\mathcal{V}| > 1$ then
        while $Makespan(\mathcal{T}) \neq Makespan(\mathcal{T}^{previous})$ do
            for $i_o \leftarrow 5$ to 7 do
                $(\bar{\mathcal{T}}, \bar{\mathcal{I}}) \longleftarrow \mathrm{Operation}_{i_o}(\mathcal{T}, \mathcal{I}, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}, \bar{\tau})$
                if $Cost(\bar{\mathcal{T}}) < Cost(\mathcal{T})$ then
                    $\mathcal{T} \longleftarrow \bar{\mathcal{T}}$
                    $\mathcal{I} \longleftarrow \bar{\mathcal{I}}$
                end
            end
        end
        if $Makespan(\mathcal{T}) \neq Makespan(\mathcal{T}^{previous})$ then
            STOP $\longleftarrow$ FALSE
        end
    end
end
```

Large Neighborhood Search (LNS). Multiple large neighborhoods are explored to find local optimas that, either are clustered together or are present in valleys far away from each other. Further, these large neighborhoods are nested together to increase the effectiveness of LNS. The perturbation and repairing operators that comprise LNS are described in the subsequent sections. They are only applicable for single vehicles. In case the fleet comprises of multiple vehicles, it is applied to each partition corresponding to each vehicle. Thus in Algorithms 3, 4 and 5, when $|\mathcal{V}| > 1, d_i$ represent the imbalance of the nodes for the partition, where the corresponding vehicle is performing the rebalancing.

2.5.3.1 Repairing Operators

The repairing operator proposed in this section is capable of repairing a partial or an infeasible solution of a single vehicle. Algorithm 3 is the pseudo-code of the repairing operator. It is based on the greedy construction heuristic described in Section 2.5.1. As, three different functions (Nearest Neighbor 1,2 and 3 described in Section 2.5.1.1) can be used as the nearest neighbor function (line 19), often three distinct solutions can be constructed from an initial partial solution. This feature comes in handy while repairing an infeasible solution from a perturbation (Section 2.5.3.2) on a feasible solution.

2.5.3.2 Perturbation Operators

Perturbation operators used in NLNS+VND are greatly influenced by Chained Lin-Kernighan used for solving large-scale Traveling Salesman Problems (Applegate et al. (2003)). However, a major difference of the two methods is that in Chained Lin-Kernighan selected edges are destroyed whereas in the proposed perturbation operators selected locations in a tour are destroyed. Locations in a tour are ranked for the purpose of perturbation.

$$\text{Rank of location k} = \frac{\text{Rank of Edge}_{k-1,k} + \text{Rank of Edge}_{k,k+1}}{2}, \forall k \in [2, \text{length of tour} - 1]$$

Algorithm 3: Repair Tour

```
Data: $\mathcal{T}_v$, $\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$, Nearest Neighbor Function
Result: ($\mathcal{T}_v$,$\mathcal{I}_v$)
1  if $\mathcal{T}_v[1]$ is not Depot then
2      Depot is added at the beginning of $\mathcal{T}_v$
3  end
4  $\mathcal{I}_v \longleftarrow [0]$
5  $\mathrm{d_i^o} \longleftarrow \mathrm{d}_i$
6  $k \longleftarrow 2$
7  while $k \leq |\mathcal{T}_v|$ do
8      Max Ops $\longleftarrow$ Maximum operations for every other node($\mathrm{d_i^o}$, $\mathcal{Q}$, $sum(\mathcal{I}_v)$)
9      if $Max\ Ops_{\mathcal{T}_v[k]} = 0$ then
10         Delete node at location $k$ of $\mathcal{T}_v$
11     else
12         Add Max $\mathrm{Ops}_{\mathcal{T}_v[k]}$ at the end of $\mathcal{I}_v$
13         $\mathrm{d}^{\mathrm{o}}_{\mathcal{T}_v[k]} \longleftarrow \mathrm{d}^{\mathrm{o}}_{\mathcal{T}_v[k]} - \mathrm{Max\ Ops}_{\mathcal{T}_v[k]}$
14         $k \longleftarrow k + 1$
15     end
16 end
17 while Number of non zero elements in $d_i^o > 0$ do
18     Max Ops $\longleftarrow$ Maximum operations for every other node($\mathrm{d_i^o}$, $\mathcal{Q}$, $sum(\mathcal{I}_v)$)
19     Next node $\longleftarrow$ Nearest Neighbor Function($\mathrm{d_i^o}$, $\mathrm{c_{ij}}$, Next node, Max Ops)
20     Add Next node at the end of $\mathcal{T}_v$
21     Add Max $\mathrm{Ops}_{\text{Next node}}$ at the end of $\mathcal{I}_v$
22     $\mathrm{d}^{\mathrm{o}}_{\text{Next node}} = \mathrm{d}^{\mathrm{o}}_{\text{Next node}} - \mathrm{Max\ Ops}_{\text{Next node}}$
23 end
24 Add Depot at the end of $\mathcal{T}_v$
25 Add 0 at the end of $\mathcal{I}_v$
```

and

$$\text{Average Rank of a Tour} = \frac{\sum_{k \in [2,\text{length of tour}-1]} \text{Rank of location k}}{\text{length of tour} - 2}$$

In the upcoming perturbation operators, two functions *Sorted Location Rank List* and *Reverse Sorted Location Rank List* are used extensively. *Sorted Location Rank List* takes a tour and a rank list as its inputs and computes the rank of all the locations (except for the first and for the last) in the tour. It then sorts the locations in the tour in a descending order of their respective ranks. The sorted list and the number of locations whose rank is above the average rank of the tour is returned. *Reverse Sorted Location Rank List* is similar to *Sorted Location Rank List*, except that the locations in the tour are sorted in an ascending order of their respective ranks. The sorted list and the number of locations whose rank is less than or equal to the average rank of the tour is returned.

2.5.3.2.1 Perturbation Operators 1 and 2 Perturbation operators 1 and 2 are complements of each other. In Perturbation Operator 1, local optimas in valleys clustered together, are explored systematically by destroying locations in a tour with undesirable configurations, followed by repairing of the tour and VND. If the cost of the new tour is lower than that of the current tour, the new tour becomes the current tour, otherwise the value of perturbation is incremented. The process continues until the value of perturbation equals that of maximum perturbation. Similarly, in Perturbation operator 2, local optimas in valleys far away from each other, are explored systematically by destroying locations in a tour with undesirable configurations, followed by repairing of the tour and VND. If the cost of the new tour is lower than that of the current tour, the new tour becomes the current tour, otherwise the value of perturbation is incremented. The process continues until the value of perturbation equals that of maximum perturbation.

With increase in $|\mathcal{N}|$, execution time of VND and the *maximum perturbation* increases significantly. Thus, to keep the exploration time of these large neighborhoods reasonable, without hampering the quality of the solutions found, number of perturbations is limited to only $\frac{15}{|\mathcal{N}|}$% of total perturbations possible. Further, the value of perturbation is varied between its minimum and maximum values simultaneously. This is based on the observation that, perturbation is most effective when it is close to its extreme values.

Algorithm 4 is the pseudo-code for Perturbation Operator 1. The pseudo-code for Perturbation Operator 2 is similar to Algorithm 4, except *Sorted Location Rank List* and *Number of Locations above Average Rank* are replaced by *Reverse Sorted Location Rank List* and *Number of Locations below Average Rank* respectively.

Algorithm 4: Perturbation Operator 1

```
   Data: $\mathcal{T}_v, \mathcal{I}_v, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}$,Repairing Operator
   Result: $(\mathcal{T}_v, \mathcal{I}_v)$
1  Location Rank List,Number of Locations above Average Rank = Sorted Location
     Rank List($\mathcal{T}_v$,Rank List)
2  Perturbation $\longleftarrow$ 1
3  while Perturbation $\leq \frac{\text{Number of Locations above Average Rank} \times 15}{|\mathcal{N}|}$ do
4      if Perturbation is Odd then
5          Locations to Destroy $\longleftarrow$ Location Rank List $\left[1\,\textbf{to}\,\frac{\text{Perturbation+1}}{2}\right]$
6      else
7          Locations to Destroy $\longleftarrow$ Location Rank
             List $\left[1\,\textbf{to}\,\text{Number of Locations above Average Rank + 1} - \frac{\text{Perturbation}}{2}\right]$
8      end
9      $\bar{\mathcal{T}}_v \longleftarrow$ Delete Nodes at locations [Locations to Destroy] of $\mathcal{T}_v$
10     $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow$ Repair Tour($\bar{\mathcal{T}}_v, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}$,Repairing Operator)
11     $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow$ VND($\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}$)
12     if Cost($\bar{\mathcal{T}}_v$) < Cost($\mathcal{T}_v$) then
13         $\mathcal{T}_v \longleftarrow \bar{\mathcal{T}}_v$
14         $\mathcal{I}_v \longleftarrow \bar{\mathcal{I}}_v$
15         Location Rank List,Number of Locations above Average Rank = Sorted
             Location Rank List($\mathcal{T}_v$,Rank List)
16         Perturbation $\longleftarrow$ 1
17     else
18         Perturbation $\longleftarrow$ Perturbation+1
19     end
20 end
```

2.5.3.2.2 Perturbation Operators 3 and 4 As with Perturbation Operators 1 and 2, Perturbation Operators 3 and 4 are also complements of each other. In Perturbation Operator 3, node(s) on the right of a location in the tour is (are) destroyed, followed by repairing of the tour and VND. If the cost of the new tour is lower than that of the current tour, the new tour becomes the current tour, otherwise the value of perturbation is incremented. The process continues until the value of perturbation equals that of maximum perturbation. Similarly, in Perturbation Operator 4, node(s) on the left of a location in the tour is (are) destroyed, followed by repairing of the tour and VND. If the cost of the new tour is lower than that of the current tour, the new tour becomes the current tour, otherwise the value of perturbation is incremented. The process continues until the value of perturbation equals that of maximum perturbation. The locations chosen in these large neighborhoods are locations with rank greater than the average rank of the tour.

Algorithm 5 is the pseudo-code for Perturbation Operator 3. Pseudo-code for Perturbation Operator 4 is similar to Algorithm 5, except $>$ in lines 5 and 23 and *Delete nodes in $\mathcal{T}_v$ right of (Location to Destroy[Perturbation])* in line 13 is replaced by $<$ and *Delete nodes in $\mathcal{T}_v$ left of (Location to Destroy[Perturbation])* respectively.

2.5.4 NLNS+VND

Algorithm 6 is the pseudo-code of NLNS+VND. The initial solution is constructed using Algorithm 1 (line 2). *Repairing Operator$_{j_o}$* in line 9 in Algorithm 6, denotes that *Nearest Neighbor j_o()* is used in Algorithm 3 for repairing the perturbed solution. $\mathcal{T}_v^{-12}$ is the 12th previous tour after perturbation.

2.6 Recommended Solution Strategies

Instances of SCRP can be classified into two categories:

1. Instances with zero imbalance at the Depot, i.e., $d_1 = 0$,

Algorithm 5: Perturbation Operator 3

```
   Data: $\mathcal{T}_v$, $\mathcal{I}_v$, $\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$,Repairing Operator
   Result: $(\mathcal{T}_v, \mathcal{I}_v)$
1  Location Rank List,Number of Locations above Average Rank = Sorted Location
     Rank List($\mathcal{T}_v$,Rank List)
2  Location Rank List $\longleftarrow$ Location Rank List[1 to Number of Locations above
     Average Rank]
3  $k \longleftarrow 2$
4  while $k \leq$ | Location Rank List | ) do
5      if Location Rank List[k] $>$ Location Rank List[$k-1$] then
6          delete at(Location Rank List,$k$)
7      else
8          $k \longleftarrow k+1$
9      end
10 end
11 Perturbation $\longleftarrow$ 1
12 while Perturbation $\leq$ | Location Rank List | do
13     $\bar{\mathcal{T}}_v \longleftarrow$ Delete Nodes in $\mathcal{T}_v$ right of (Location to Destroy[Perturbation])
14     $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow$ Repair Tour($\bar{\mathcal{T}}_v$, $\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$,Repairing Operator)
15     $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow$ VND($\bar{\mathcal{T}}_v$, $\bar{\mathcal{I}}_v$, $\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$)
16     if Cost($\bar{\mathcal{T}}_v$) $<$ Cost($\mathcal{T}_v$) then
17         $\mathcal{T}_v \longleftarrow \bar{\mathcal{T}}_v$
18         $\mathcal{I}_v \longleftarrow \bar{\mathcal{I}}_v$
19         Location Rank List,Number of Locations above Average Rank = Sorted
             Location Rank List($\mathcal{T}_v$,Rank List)
20         Location Rank List $\longleftarrow$ Location Rank List[1 to Number of Locations above
             Average Rank]
21         $k \longleftarrow 2$
22         while $k \leq$ | Location Rank List | do
23             if Location Rank List[k] > Location Rank List[$k-1$] then
24                 delete at(Location Rank List,$k$)
25             else
26                 $k \longleftarrow k+1$
27             end
28         end
29         Perturbation $\longleftarrow$ 1
30     else
31         Perturbation $\longleftarrow$ Perturbation+1
32     end
33 end
```

Algorithm 6: Nested Large Neighborhood Search + Variable Neighborhood Descent - NLNS+VND

```
   Data: $\mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}, \bar{\tau}$
   Result: $(\mathcal{T}, \mathcal{I})$
 1 for $v \leftarrow 1$ to $|\mathcal{V}|$ do
 2     $(\mathcal{T}_v, \mathcal{I}_v) \longleftarrow$ Greedy Construction Heuristic($\mathrm{d_i}$, $\mathcal{Q}$, $\mathrm{c_{ij}}$,Nearest Neighbor 2)
 3 end
 4 STOP $\longleftarrow$ FALSE
 5 while $STOP = FALSE$ do
 6     for $v \leftarrow 1$ to $|\mathcal{V}|$ do
 7         for $i_o \leftarrow 1$ to $4$ do
 8             for $j_o \leftarrow 1$ to $3$ do
 9                 $(\bar{\mathcal{T}}_v, \bar{\mathcal{I}}_v) \longleftarrow$ Perturbation Operator $i_o(\mathcal{T}_v, \mathcal{I}_v, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}$,Repairing
                    Operator $j_o)$
10                 if $Cost(\bar{\mathcal{T}}_v) < Cost(\mathcal{T}_v)$ then
11                     $\mathcal{T}_v \longleftarrow \bar{\mathcal{T}}_v$
12                     $\mathcal{I}_v \longleftarrow \bar{\mathcal{I}}_v$
13                 end
14                 if $Cost(\bar{\mathcal{T}}_v) = Cost(\mathcal{T}_v^{-12})$ then
15                     STOP $\longleftarrow$ TRUE
16                     break
17                 end
18             end
19             if $STOP = TRUE$ then
20                 break
21             end
22         end
23     end
24     if $|\mathcal{V}| > 1$ then
25         $(\bar{\mathcal{T}}, \bar{\mathcal{I}}) \longleftarrow$ VND($\mathcal{T}, \mathcal{I}, \mathrm{d_i}, \mathcal{Q}, \mathrm{c_{ij}}, \bar{\tau}$)
26         if $Makespan(\bar{\mathcal{T}}) < Make\text{-}span(\mathcal{T})$ then
27             $\mathcal{T} \longleftarrow \bar{\mathcal{T}}$
28             $\mathcal{I} \longleftarrow \bar{\mathcal{I}}$
29             STOP $\longleftarrow$ FALSE
30         end
31     end
32 end
```

2. Instances with non-zero imbalance at the Depot, i.e., $d_1 \neq 0$,

The proposed MILP and NLNS+VND can only handle instances with zero imbalance at the depot. In this section, it is shown how an instance with non-zero imbalance at the depot can be converted into an instance with zero imbalance at the depot, so that the proposed methodology becomes applicable.

2.6.1 Instances with Zero Imbalance at the Depot

The recommended method is using NLNS+VND to solve the instance on the original network. Currently, the proposed MILP is computationally intractable. However, if using some techniques the MILP can be made computationally tractable, it can be used to solve the instance in the following way:

1. Decompose the original network of the instance into its decomposed network
2. Solve the MILP on the decomposed network
3. Convert the solution found by the MILP on the decomposed network to a solution on the original network

2.6.2 Instances with Non-zero Imbalance at the Depot

For instances, where the depot has a non-zero imbalance, i.e., $d_1 \neq 0$, a pseudo node is created in the network. The modified network has $|\mathcal{N} + 1|$ nodes, $|\mathcal{N} + 1|$ being the index of the pseudo node created. In the modified network,

$$\tilde{d}_1 = 0 \tag{2.23}$$

$$\tilde{d}_{|\mathcal{N}+1|} = d_1 \tag{2.24}$$

$$\tilde{d}_i = d_i, \qquad \forall i \in \mathcal{N} \setminus \{1\} \tag{2.25}$$

$$\tilde{c}_{ij} = c_{ij}, \qquad \forall i \in \{1, ..., |\mathcal{N}+1|\}, j \in \mathcal{N} \tag{2.26}$$

$$\tilde{c}_{i|\mathcal{N}+1|} = c_{i1}, \qquad \forall i \in \{1, ..., |\mathcal{N}+1|\} \tag{2.27}$$

$$\tilde{c}_{|\mathcal{N}+1|i} = c_{1i}, \qquad \forall i \in \{1, ..., |\mathcal{N}+1|\} \tag{2.28}$$

For a feasible solution on the modified network, all $|\mathcal{N}+1|$ in the tour must be replaced by *1* to obtain a feasible solution on the original network. The recommended method in this scenario is the following:

1. When NLNS+VND is used to solve the SCRP:

    1. Convert the original network of the instance into its modified original network, by adding a pseudo node and making the imbalance of the depot equal to zero as mentioned above.
    2. Solve SCRP on the modified original network using NLNS+VND
    3. Change index of the pseudo node in the solution obtained by NLNS+VND to *1* so that it becomes a feasible solution for the original network.

2. When MILP is used to formulate and solve the SCRP:

    1. Convert the original network of the instance into its modified original network, by adding a pseudo node and making the imbalance of the depot equal to zero as mentioned above.
    2. Decompose the modified original network of the instance into its decomposed network

3. Solve the MILP on the decomposed network
4. Convert the solution obtained by the MILP on the decomposed network to a solution on the modified original network
5. Change index of the pseudo node in the solution for the modified original network to *1* so that it is a feasible solution for the original network.

2.7 Case Study 1: 1-PDTSP Instances

The purpose of Case Study 1 is to compare the performance of *NLNS+VND* with exact algorithms from Erdoan et al. (2015) and Tabu Search algorithms from Chemla et al. (2013a).

The instances are those used in Chemla et al. (2013a) and Erdoan et al. (2015) and adapted from the 1-PDTSP instances introduced in Hernández-Pérez and Salazar-González (2004). The instances are available at https://github.com/aritrasep/BSSLib.jl. Computational experiments are carried out on instances with $\alpha = \{1, 3\}$, $|\mathcal{N}| = \{20, 30, 40, 50, 60\}$, $Q = \{10, 15, 20, 25, 30, 35, 40, 45, 1000\}$ and $|\mathcal{N}| = 100$, $Q = \{10, 30, 45, 1000\}$. Further, each of these configurations have 10 independent instances, resulting in a total of 980 instances. For each instance, we compute $d_i = \text{Original Imbalance} \times \alpha, \forall i \in \mathcal{N}$ and c_{ij} as the euclidean distance between the nodes i and $j, \forall i, j \in \mathcal{N}$.

For comparing NLNS+VND with Exact (Erdoan et al. (2015) as well as Tabu Search (Chemla et al. (2013a)) Algorithms, the test cases are divided into 3 sets. $\alpha \in \{1, 3\}$ for all the instances in all 3 sets.

The best known lower bound reported in the literature as mentioned in Table 2.5, is the maximum of the lower bounds reported by Erdoan et al. (2015) and Chemla et al. (2013a) for each instance. Negative values in column Gap under PEA in Table 2.6 denotes the added value of preemption.

Our computational experiments are carried out on a workstation powered by an Intel Core i7-4790 CPU @ 3.6 GHz with a total RAM of 16 GB running an Ubuntu 16.04 64 bit operating system. The performance of Iridis 4 Computing Cluster and Intel i7-4790 is quite similar, so the computational times for the Exact Algorithms are by itself standardized. Chemla et al. (2013a) on the other hand, ran experiments a workstation powered by an AMD Athlon 64 X2 Dual Core 5600+ processor. To compare the runtime performance of the Tabu Search algorithms proposed in Chemla et al. (2013a) and NLNS+VND, an approximation factor is estimated based on the single thread rating values reported in http://www.cpubenchmark.net/compare.php?cmp[]=86&cmp[]=2226. From the above information, it can be concluded that, a workstation powered by an AMD Athlon 5600+ processor is approximately 2.705325444 times slower than a workstation powered by an Intel i7-4790 CPU. Thus, computational times of the Tabu Search Algorithms (TS1 and TS2), are standardized by dividing their reported computational times by 2.705325444.

Table 2.6 summarizes our experimental results for 1-PDTSP instances. The results (Gap , Best Gap, Avg Gap and Time) are the average of all the combination of $\mathcal{N}$, Q and α for each set. Corresponding breakdowns for each set is provided in Figure 2.12. In Figure 2.12, the scale of the y-axis for Time (seconds), for all 3 sets are in $\log_{10}$ scale. Detailed experimental results for this case study is available from the authors. In our experiments, NLNS+VND was able to find new solutions for 59 out of 148 instances, for which the optimal solution is not known.

From Table 2.6 it is evident that:

- PEA outperforms NLNS+VND in terms of finding higher quality solutions for instances with $|\mathcal{N}| \leq 60$. One of the reason being, PEA allows preemption, whereas NLNS+VND does not.

- NLNS+VND is highly competitive with NPEA is terms of quality of solutions found.

- NLNS+VND outperforms TS1 for all instances.

- NLNS+VND is more effective than TS2, for instances with $|\mathcal{N}| > 60$.

- NLNS+VND is the most efficient of all the five algorithms, as on average it has the lowest runtime.
- On average, NLNS+VND is 300, 500, 11 and 66 times faster than PEA, NPEA, TS1 and TS2 respectively.

Other important information not evident from Table 2.6 but evident from Figure 2.12 are:

- NLNS+VND is more effective than the algorithms presented in the literature for realistic instances,i.e., instances with $|\mathcal{N}| \geq 50$ and $Q \leq 20$
- NLNS+VND is 3 orders of magnitude faster than both PEA and NPEA, for instances with either $\alpha = 1$ or $Q \geq 30$.
- NLNS+VND is 2 orders of magnitude faster than both TS1 and TS2, for instances with $\alpha = 1$ or $Q \geq 30$ or $|N| = \{40, 60\}$

2.8 Case Study 2: Share-A-Bull (SABB) FFBS

The University of South Florida's Tampa campus covers 1,700 acres and houses more than 320 buildings. Walking from one building to another during the short breaks between classes is challenging, owing to weather conditions and the heavy weight of textbooks. An annual Tampa campus transportation and parking survey shows that those who drive to campus make, on average, one across-campus trip per day (between buildings or to lunch). Given that there are more than 38,000 students and 1,700 faculties and staffs on the Tampa campus, across-campus driving trips can lead to significant fuel consumption and greenhouse gas (GHG) emissions. Thus, USF collaborated with Social Bicycles (SoBi) and developed the Share-A-Bull FFBS program (SABB). Phase I of the program was launched in September 2015 with 100 bicycles. With Phases II and III in next few years, the program will be expanded to 300 bicycles and cover both the Tampa campus and student housing in the vicinity of the campus. The program is expected to be integrated with parking management and other multi-modal transportation initiatives on the campus. SABB provides an excellent case study for our bike rebalancing research.

One of the objectives of this case study is to determine whether SCRP with single and multiple vehicles is feasible for SABB. Another important objective is to determine if NLNS+VND is capable of dealing with increase in complexity of SCRP for FFBS, i.e., when $|\mathcal{N}| \geq 100, \mathcal{Q} \leq 10$ and $|\mathcal{V}| > 1$.

Table 2.7 summarizes the experimental results for five actual rebalancing operations that took place in SABB FFBS. Total Time in Table 2.7 is the summation of the Computing and the Rebalancing Times. μ and σ in Table 2.7 are the mean and standard deviation of the respective variable. From Table 2.7, it is evident that even for a small scale FFBS, SCRP can be computationally challenging. Algorithms proposed in Chemla et al. (2013a), Erdoan et al. (2015) and Alvarez-Valdes et al. (2016) are not capable for handling the corresponding SCRPs except for the first instance with $|\mathcal{N}| = 43$. For each instance, ten trials of NLNS+VND (with Nearest Neighbor 2() as the initial solution creator) are taken, with a fleet of two vehicles each with a capacity $\mathcal{Q}$ of five. The average speed of the rebalancing vehicles was varied between $\{10, 15\}$ miles per hour, and the average loading unloading time per bike ($\bar{\tau}$) was varied between $\{30, 60\}$ seconds. Each of the these rebalancing operations were completed in less than two hours by a fleet of two vehicles each with a capacity $\mathcal{Q}$ of five in real life. This also shows that the accuracy of NLNS+VND in computing the makespan is also very high.

Originally, there were 147 nodes (or ordinary bike racks) with a total capacity to hold 1688 bikes, on the USF Tampa campus. However, an additional 10 nodes (SoBi Hubs) were added at important locations on the USF Tampa campus, to increase user satisfaction. On top of this 157 (original) nodes present in the USF Tampa campus, an additional 293 (artificial) nodes were created at realistic locations on the USF Tampa campus, to simulate parking of bikes outside of SoBi Hubs and ordinary bike racks. The capacities of these artificial nodes were generated from a normal distribution whose mean and standard deviation equals the mean and standard deviation of the capacities of the original 157 nodes. Campus recreation center is the Depot of SABB. Test cases are designed to simulate Phase I, II and III of the Share-A-Bull program, i.e., for $100, 200$ and 300 bikes in the network respectively. For each Phase, the maximum $|\mathcal{N}| = \min\{2 \times |\mathcal{B}|, |\mathcal{B}| + 157\}$. Thus for $100, 200$ and 300 bikes the maximum $|\mathcal{N}|$ are $200, 357$ and 457 nodes respectively.

For configurations with $|\mathcal{N}| \leq 157$, $|N| - 1$ nodes were selected randomly from the set of 156 original nodes (excluding the Depot). Otherwise all the original 156 nodes (excluding the depot) were selected, and the remaining $|\mathcal{N}| - 157$ nodes, were selected randomly from the 293 artificial nodes. The distance matrix (c_{ij}) was computed from an Open Street Map of the USF, Tampa campus. Loading-unloading operations (d_i) at each node was randomly assigned such that the following conditions are satisfied:

$$d_1 = 0 \tag{2.29}$$

$$d_i \neq 0, \forall i \in \mathcal{N} \setminus \{1\} \tag{2.30}$$

$$|d_i| \leq \text{Number of Bikes Node i can hold}, \forall i \in \mathcal{N} \setminus \{1\} \tag{2.31}$$

$$\sum_{i \in \mathcal{N}} d_i = 0 \tag{2.32}$$

$$\sum_{i \in \mathcal{N}} |d_i| = 2 \times |\mathcal{B}| \tag{2.33}$$

In total 9 such independent instances were created. These instances are available at https://github.com/aritrasep/BSSLib.jl.

Let us denote the make-span of a fleet of $\mathcal{V}$ rebalancing vehicles be $\mathcal{M}_{|\mathcal{V}|}$. Then we can approximate

$$\mathcal{M}_{|\mathcal{V}|} \approx \frac{\text{Traveling Time if fleet consist of 1 vehicle} + \text{Number of Operations} \times \tau}{|\mathcal{V}|}$$

Thus for a fixed $|\mathcal{V}|$ and τ, $\mathcal{M}_{|\mathcal{V}|}$ will be close to its maximum value, when the *Traveling Time if fleet consist of 1 vehicle* and *Number of Operations* is maximum or pretty close to their maximum values. From Equation 2.33, one can see that, the total number of (loading and unloading) operations $= 2 \times |\mathcal{B}|$, which is the maximum possible operation for a fixed $|\mathcal{B}|$, i.e, pickup and drop off of every bike in the system. Further, Equation 2.30 ensures that there is at least an unit (positive or negative) imbalance is associated with each node (other than the depot) in the network. This ensures that all nodes in the network is visited at least once, by one or more of the rebalancing vehicles, if not more depending on their respective imbalance. This ensures, that the Traveling Time is also pushed to its maximum value. Thus, the test cases generated simulate extreme scenarios, so that the performance of NLNS+VND and feasibility of SCRP for SABB can be tested in the worst possible scenario.

Given, capacity of the rebalancing vehicle $\mathcal{Q} \in \{5, 10\}$, average speed of the rebalancing vehicle $\in \{10, 15\}$ miles per hour, and average loading unloading time per bike $(\bar{\tau}) \in \{30, 60\}$ seconds, there are a total of 72 different configurations. Ten trials of NLNS+VND (with Nearest Neighbor 2() as the initial solution creator) are taken for each configuration. For each configuration, $|\mathcal{V}| = \dfrac{|\mathcal{N}|}{100}$. The summary of the experimental results are reported in Table 2.8.

Total Time in Table 2.8 is the summation of the Computing and the Rebalancing Times. μ and σ in Table 2.8 are the mean and standard deviation of the respective major column. From Table 2.8, it is evident that SCRP is feasible for SABB, as the Total Time is less than 6 hours (time available per day for computation and rebalancing) for all the configurations and NLNS+VND is able to compute high quality solutions of SCRP for FFBS in a short period of CPU Time.

2.9 Case Study 3: Divvy SBBS

Divvy is a large scale SBBS system in the city of Chicago with 476 nodes (stations with a total capacity to hold approximately 7900 bikes), and 4760 bikes. The objectives for conducting this case study are two fold, first, to determine if SCRP with multiple vehicles is feasible for Divvy and second, to determine if NLNS+VND is capable of dealing with increase in complexity of SCRP for large scale SBBS, i.e., when $|\mathcal{N}| > 400, 500 \leq |\mathcal{B}|$ and $5 \leq |\mathcal{V}|$. The test cases are created using the same method used for creating the SABB general instances as described in Section 2.8. In total 6 such independent test cases were created. For each test case generated, capacity of the rebalancing vehicle $\mathcal{Q} \in \{10, 20\}$, average speed of the rebalancing vehicle $\in \{40, 50\}$ miles per hour, and average loading unloading time per bike $(\bar{\tau}) \in \{30, 60, 90\}$ seconds, creating a total of 72 different configurations. Five trials of NLNS+VND (with Nearest Neighbor 2() as the initial solution creator) was taken for each configuration. For each configuration, $|\mathcal{V}| = \dfrac{|\mathcal{N}|}{100}$. These instances are available at https://github.com/aritrasep/BSSLib.jl. The summary of the experimental results are reported in Table 2.9.

Total Time in Table 2.9 is the summation of the Computing and the Rebalancing Times. μ and σ in Table 2.9 are the mean and standard deviation of the respective major column. From Table 2.9, it is evident that SCRP is also feasible for Divvy instances, as the Total Time is less than 7 hours for all configurations and NLNS+VND is able to compute high quality solutions for SCRP for large scale Bike Sharing Systems in a short period of CPU Time.

2.10 Final Remarks

In this chapter, a Novel MILP for formulating SCRP in FFBS and SBBS based on spacial decomposition is reported. The proposed formulation, can not only handle single and multiple vehicles, but also allows for multiple visits to a node by the same vehicle. The proposed formulation is computationally intractable even for small scale instances owing to the presence of Big M, used for subtour elimination in the constraints. It makes the linear programming relaxation of the formulation extremely weak. Another reason for the computational intractability of the formulation is the significant increase in the number of decision variables owing to spacial decomposition.

A hybrid nested large neighborhood search with variable neighborhood descent algorithm (NLNS+VND) for solving SCRP both effectively and efficiently for FFBS and SBBS is also presented. Computational experiments on 1-PDTSP instances, previously used the literature, demonstrate that NLNS+VND outperforms tabu search and is highly competitive with exact algorithms reported in the literature. The fact that NLNS+VND was able to find all solutions for 148 instances, with 59 of which did not have optimal solutions from applying solution algorithms in existing literature show that, it is more robust than the algorithms previously reported in the literature. Further, NLNS+VND is, on average, 300 times faster than the exact algorithm that allows preemption and 500 times faster than the exact algorithm does not allow preemption. To the best of our knowledge, NLNS+VND is the rst to solve SCRP, with instances having nodes greater than or equal to 50 and vehicle capacity less than 30 both effectively and efficiently. Computational experiments on the new SABB real and general instances (consisting of up to 400 nodes, 300 bikes, and a eet size of up to 3 vehicles) and Divvy instances (consisting of 450 stations, 3000 bikes, and a eet size of up to 30 vehicles), demonstrate that NLNS+VND is able to deal with the increase in scale of SCRP for both FFBS and SBBS. It also shows that SCRP is feasible for both SABB program at USF, Tampa and Divvy SBBS at Chicago.

In future research, we consider strengthening the linear programming relaxation of our proposed formulation by extending valid inequalities proposed in the literature for m-TSP, 1-PDTSP and Q-TSP to our proposed formulation. To deal with increase in the number of variables, strategies based on column generation will be explored. Other interesting strategies can be using the high quality solution provided by NLNS+VND as a starting solution (upper bound) for MIP solvers once some mechanism for strengthening the formulation has been implemented. We also study partial rebalancing in FFBS and SBBS with single and multiple vehicles.

Table 2.2: Notations Used in the Remainder of the Chapter

	Notation	*Description*	*Network*		*MILP*	*Heuristic*
			Original	*Decomposed*		
Parameters	$\mathcal{G} = (\mathcal{N}, \mathcal{E})$	Graph of the original network.	✓	×		✓
	$\mathcal{N}$	Set of nodes in the original network, including the depot. The depot is denoted by 1, where the tour of the rebalancing vehicles start and end. Rest of the nodes are numbered from $2, 3, ..., \lvert\mathcal{N}\rvert$.				
	$\mathcal{E}$	Edges in the original network.				
	c_{ij}	Time taken to travel from node i to node $j, \forall (i, j) \in \mathcal{E}$ in seconds.				
	d_i	Imbalance at node $i, \forall i \in \mathcal{N}$. $d_i > 0$ when node i is a pickup node or has a surplus of bikes, $d_i < 0$ when node i is a delivery node or has a deficit of bikes and 0 when node i is the depot.				
	$\mathcal{G}_o = (\mathcal{N}_o, \mathcal{E}_o)$	Graph of the decomposed network.	×	✓		×
	$\mathcal{N}_o$	Set of nodes in the decomposed network, including the depot. The depot is denoted by 1, where the tour of the rebalancing vehicles start and end. Rest of the nodes are numbered from $2, 3, ..., \lvert\mathcal{N}_o\rvert$.				
	$\mathcal{E}_o$	Edges in the decomposed network.				
	$\bar{c}_{ij}$	Time taken to travel from node i to node $j, \forall (i, j) \in \mathcal{E}_o$ in seconds.				
	$\bar{d}_i$	Imbalance at node $i, \forall i \in \mathcal{N}_o$. $\bar{d}_i = +1$ when node i is a pickup node or has a surplus of one bike, $\bar{d}_i = -1$ when node i is a delivery node or has a deficit of one bike and $\bar{d}_i = 0$ when node i is the depot.				
	$\bar{\tau}$	Average loading unloading time per bike in seconds.	✓			
	$\mathcal{V}$	Fleet of homogeneous rebalancing vehicles.				
	$\mathcal{Q}$	Capacity of each vehicle in the fleet of homogeneous rebalancing vehicles.				
Variables	τ_i	Arrival time of a rebalancing vehicle at node $i, \forall i \in \mathcal{N}_o$.	×	✓		×
	x_{ij}	Equals 1, if Edge (i, j) is traversed by a rebalancing vehicle, otherwise it is $0, \forall (i, j) \in \mathcal{E}_o$.				
	q_{ij}	Quantity of bikes carried by a rebalancing vehicle on Edge $(i, j), \forall (i, j) \in \mathcal{E}_o$.				
	$\mathcal{T}_v$	Tour of rebalancing vehicle $v, \forall v \in \mathcal{V}$	×			✓
	$\mathcal{T}$	$\{\mathcal{T}_v, \forall v \in \mathcal{V}\}$				
	$\mathcal{I}_v$	Loading unloading instructions of rebalancing vehicle $v, \forall v \in \mathcal{V}$.				
	$\mathcal{I}$	$\{\mathcal{I}_v, \forall v \in \mathcal{V}\}$				
	k, l, m, n	Indices used for locations inside a Tour ($\mathcal{T}_v$) of a rebalancing vehicle.				
	i_o, j_o	Iterators used in Algorithms 2, 3 and 6.				

Table 2.3: Terminology of the Algorithms

Terminology	Description
TS1	Tabu search algorithm initialized with solution from greedy heuristic (Chemla et al. (2013a))
TS2	Tabu search algorithm initialized with Eulerian circuit from MILP relaxation (Chemla et al. (2013a))
RB	Reliability branching used for exploring the Branch-and-bound tree while solving relaxation of the static rebalancing problem (Chemla et al. (2013a))
DB	Degree branching used for exploring the Branch-and-bound tree while solving relaxation of the static rebalancing problem (Chemla et al. (2013a))
PEA	Exact algorithm that allows preemption (Erdoan et al. (2015))
NPEA	Exact algorithm that does not allow preemption (Erdoan et al. (2015))
NLNS+VND	Hybrid nested large neighborhood search with variable neighborhood descent algorithm presented in this chapter

Table 2.4: Number of Trials of each Algorithm

Algorithm	Number of Trials
TS1	2
TS2	1 trial each, with RB and DB as branching strategy
NPEA	1
PEA	1
NLNS+VND	10 (each with Nearest Neighbor 2 as initial solution creator)

Table 2.5: Description of Parameters Used in Table 2.6

Type of Measure	Algorithms	Description	Unit
Gap	Exact	Absolute Gap of Upperbound found by Exact Algorithms from best known Lowerbound reported in literature.	%
Best Gap	Tabu Search and NLNS+VND	Absolute Gap of Best Upperbound found by corresponding Algorithms in all of its trials, from best known Lowerbound reported in literature.	%
Avg Gap	Tabu Search and NLNS+VND	Absolute Gap of Average Upperbound found by corresponding Algorithms from all of its trials, from best known Lowerbound reported,in literature.	%
Time	Exact, Tabu Search and NLNS+VND	Computation time of corresponding Algorithms standardized to our workstation.	seconds

Table 2.6: Summary of Overall Results for 1-PDTSP Instances

Set	Exact Algorithms				Tabu Search Algorithms						NLNS+VND		
	PEA		NPEA		TS1			TS2					
	Gap	Time	Gap	Time	Best Gap	Avg Gap	Time	Best Gap	Avg Gap	Time	Best Gap	Avg Gap	Time
I	*-0.075*	1047.32	2.691	1847.936	5.601	6.748	174.054	0.382	0.742	1020.411	2.435	3.776	*5.003*
II	*-0.041*	978.06	2.003	1663.736	-	-	-	-	-	-	2.47	3.971	*3.224*
III	-	-	-	-	14.786	16.279	296.442	5.21	7.148	1686.578	*5.104*	*6.776*	*25.469*

Table 2.7: Summary of Overall Results for SABB Real Instances

$\|\mathcal{B}\|$	$\|\mathcal{N}\|$	*Makespan (Seconds)*		*Computing Time (Seconds)*		*Total Time (Hours)*	
		μ	ρ	μ	ρ	μ	ρ
42	43	3512	845.13	0.71	0.31	0.98	0.23
44	79	3670.75	845.57	1.37	0.29	1.02	0.23
51	98	4207.5	989.6	2.13	1.05	1.17	0.27
57	96	4358.25	1040.29	4.56	1.99	1.21	0.29
63	118	4842.75	1182.68	6.1	2.62	1.35	0.33

Table 2.8: Summary of Overall Results for SABB General Instances

$\|\mathcal{B}\|$	$\|\mathcal{N}\|$	$\|\mathcal{V}\|$	Q	Makespan (Seconds)		Computing Time (Seconds)		Total Time (Hours)	
				μ	σ	μ	σ	μ	σ
100	100	1	5	13519.5	3175.51	32.99	13.24	3.76	0.88
			10	12426.62	3121.92	15.17	6.11	3.46	0.87
	200		5	16769.1	3427.65	40.21	12.58	4.67	0.95
			10	16584.18	3401.45	18.64	9.56	4.61	0.94
200	100	2	5	12774.75	3070.98	22.82	10.4	3.55	0.85
			10	11750.3	3114.54	8.29	3.07	3.27	0.87
	200		5	15426.42	3394.37	27.58	14.42	4.29	0.94
			10	14535.1	3313.93	13.4	7.22	4.04	0.92
	300		5	16526.18	3542.88	65.8	37.97	4.61	0.98
			10	15562	3263.39	28.25	9.54	4.33	0.91
300	100	3	5	12215.9	3112.13	18.39	6.68	3.4	0.86
			10	11346.32	3081.06	5.21	2.04	3.15	0.86
	200		5	14842.05	3286.22	17.26	6.94	4.13	0.91
			10	13685.98	3205.2	7.24	3.42	3.8	0.89
	300		5	14778.64	3306.76	47.64	17.61	4.12	0.92
			10	13933.52	3243.78	19.35	8.64	3.88	0.9
	400		5	15803.7	3365.43	80.76	44.22	4.41	0.94
			10	14832.58	3381.04	38.94	20	4.13	0.94

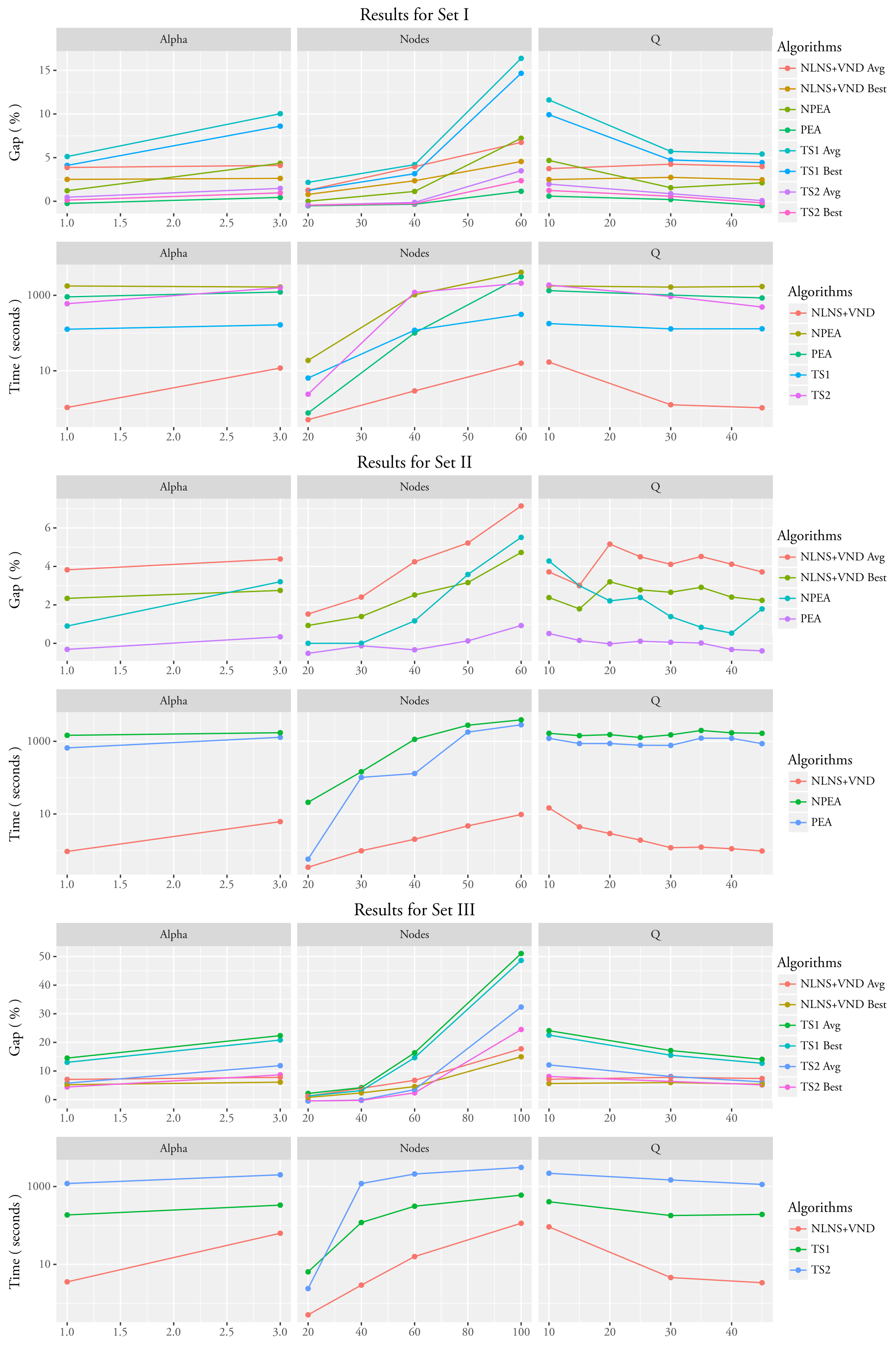


Figure 2.12: Summary of Results for Set I, II and III Respectively

Table 2.9: Summary of Overall Results for Divvy SBBS Instances

$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{V}\rvert$	Q	Makespan (Seconds)		Computing Time (Seconds)		Total Time (Hours)	
				μ	σ	μ	σ	μ	σ
500	450	5	10	18009.47	5119.49	37.1	14.2	5.01	1.42
			20	17840.97	4999.4	20.87	8.27	4.96	1.39
1000		10	10	16790.53	5104.62	29.29	4.59	4.67	1.42
			20	16491.13	5070.54	15.18	2.3	4.59	1.41
1500		15	10	15807.07	5096.04	31.55	3.24	4.4	1.42
			20	15970.4	5124.05	17.86	0.75	4.44	1.42
2000		20	10	15768.13	5127.47	40.53	4.1	4.39	1.42
			20	15781.7	5163.12	23.15	0.83	4.39	1.43
2500		25	10	15268.4	5093.73	54.41	6.02	4.26	1.41
			20	15306.67	5131.3	28.2	0.93	4.26	1.43
3000		30	10	15119.53	5069.09	65.62	5.75	4.22	1.41
			20	15136.53	5113.23	34.44	1.23	4.21	1.42

3 Analyzing Mobility Patterns and Imbalance of Free Floating Bike Sharing Systems

3.1 Problem Description

Solving the core problem of an established BSS requires the understanding of the mobility patterns of its users. It enables the operator to estimate an approximate target distribution of bikes for rebalancing as well as gain insights necessary for developing appropriate rebalancing strategies by addressing issues such as whether static rebalancing is sufficient or dynamic rebalancing is needed, when the different types of rebalancing should start, and how much time is available for each type of rebalancing. In this chapter, we demonstrate our proposed methods of understanding mobility patterns and extracting management insights, using the historical trip data of Share-A-Bull BSS (SABB), an FFBS on the Tampa campus of the University of South Florida (USF). The knowledge and insights gained using our proposed method can be used by operators of both FFBS and SBBS to improve their respective service levels.

Existing studies on mobility patterns analysis focus primarily on SBBS by analyzing historical trip and weather data. Authors take system outputs (rentals and or returns) as dependent variables and environmental factors, socio-demographic features and cycling infrastructure as independent variables. However, none of these studies, consider imbalance (difference between returns and rentals) as a dependent variable or *interaction* between the independent variables. In this chapter, we demonstrate that by considering imbalance as a dependent variable and the *interaction* between independent variables, more knowledge and insights can be obtained about the mobility patterns of an FFBS, than by using conventional methods like data visualization and generalized linear models.

To be consistent with other studies in the literature, rentals and returns of a BSS are referred to as pickups and dropoffs respectively, in the rest of the chapter. To be more specific, in this chapter, we are trying to determine how the demand (dropoffs and pickups) and imbalance of an FFBS vary with time and how they are affected by exogenous variables such as holidays, weather conditions, etc. To accomplish this, we propose a simple method to decompose continuous variables into binary variables that improves the base model (Poisson and negative binomial regression models) commonly used in the literature as well as consider all feasible (second and third order) interactions between binary variables. The purpose of adding such interactions is to extract additional insights from the data for operational management purposes. It is obvious that considering *interactions* could result in a significant increase in the number of independent variables, sometimes even significantly larger than the number of observations. This makes it inappropriate to use (generalized) linear models directly. To address this issue, we first use a regularization operator to shrink the variable space and then estimate an appropriate linear model on the shrunk variable space. Although our case study is an FFBS, our proposed method can be used for SBBS without any modifications.

The remainder of the chapter is organized as follows. Section 3.2 summarizes and highlights gaps in the literature. Section 3.3 describes the proposed method. Section 3.4 introduces the case study and presents the experimental results of our proposed methods. Section discusses how knowledge and operational management insights about the SABB FFBS can be drawn from the statistical models. We also demonstrates, how some of this insights can be used for making useful recommendations to the operator of the system. Finally, Section 3.6 concludes this chapter with some final remarks.

3.2 Literature Review

Papers related to analytics of a BSS (primarily SBBS) can be broadly classified into two categories, based on their objective(s): 1) chapters whose primary objective is to predict the future demand of the system and 2) chapters whose primary objective is to understand and describe a system(s), so that either its service level can be improved or the system can be expanded. The most important chapters related to predicting the future demand of a BSS (or car sharing systems) are Cheu et al. (2006); Kaltenbrunner et al. (2010); Borgnat et al. (2011); Regue and Recker (2014) and Alvarez-Valdes et al. (2016). It is interesting to note that, chapters focused on predicting future demand almost always rely on non-parametric statistical methods, like neural networks (Cheu et al. (2006)), gradient boosted machines (Regue and Recker (2014)), non-homogeneous Poisson process (Alvarez-Valdes et al. (2016)), etc. Further, recent chapters on predicting demand (Regue and Recker (2014); Alvarez-Valdes et al. (2016)) also use the outputs of their demand prediction model as inputs to a rebalancing optimization model.

On the other hand, chapters in the second category always use generalized linear and generalized linear mixed models as their core statistical method. This is because linear models are easy to interpret compared to non-linear and non-parametric models. Papers in the second category can be further subdivided into two subcategories: 1) chapters that try to understand factors affecting the demand of a BSS and 2) chapters that propose metrics either to compare several BSS among themselves or to measure the performance of a BSS. In the first subcategory, the most common factors considered in the literature are:

1. temporal factors (season, month, day of week, holiday and hour of day) - Gebhart and Noland (2014); Faghih-Imani et al. (2014); Faghih-Imani and Eluru (2016); Wagner et al. (2016)

2. meteorological factors (temperature, relative humidity, wind speed, etc) - Gebhart and Noland (2014); Faghih-Imani et al. (2014); Faghih-Imani and Eluru (2016)

3. socio-demographic factors - Faghih-Imani et al. (2014, 2017b)

4. infrastructure of BSS and other modes of transportation - Faghih-Imani et al. (2014); Faghih-Imani and Eluru (2016); Faghih-Imani et al. (2017b,a)

5. size of operating area (large, medium or small-scale city) - Caulfield et al. (2017)

6. effect of expansion on demand - Wagner et al. (2016); Zhang et al. (2016)

Contrary to the above mentioned papers, Fishman et al. (2015) studied factors that affect membership instead of demand of a BSS. In the second subcategory, papers such as OBrien et al. (2014); de Chardon and Caruso (2015); de Chardon et al. (2017) propose methods to compare several BSS using daily trip data, whereas de Chardon and Caruso (2015); de Chardon et al. (2017) propose metrics to measure the quality and performance of a BSS without using the daily trip data.

To the best of our knowledge, none of the papers in the literature, consider imbalance as a dependent variable or interactions between independent variables. Thus, this is the first research on an FFBS, which takes imbalance as a dependent variable and considers *interactions between independent variables* in a statistical model. We propose two stage models to address the increase in the number of independent variables when *interactions between independent variables* are considered. Although in this chapter, we are focused on extracting knowledge and insight, often smart use of interactions between independent variables can lead to significant improvement in prediction accuracy (or decrease in out of sample testing error). We also propose a simple method to decompose continuous variables into binary variables, which significantly improves the negative binomial regression model commonly used in the literature, and has the ability to identify intervals of a continuous variable that are statistically significant. Further, our proposed methodology provides an unique opportunity to study an FFBS and make recommendations to the operator from various vantage points.

3.3 Methodology

In this section, we describe the variables used in this chapter, method of collecting and cleaning the data, strategy for discretizing continuous variables into binary variables, method for creating interactions between independent binary variables, and two stage models for scenarios when number of independent variables outnumbers number of observations.

3.3.1 Variables

In this chapter, the dependent variables are daily and hourly dropoffs and pickups as well as hourly imbalance. Hourly imbalance equals the difference of the number of dropoffs and the number of pickups in that hour. Unlike dropoffs and pickups, we do not study daily imbalance as its mean and variance is zero and close to zero respectively. This makes perfect sense, as the daily dropoffs and pickups will be close to each other unless bikes are added to or removed from the system by the operator. Daily and hourly dropoffs and pickups are non-negative count variables whereas hourly imbalance is a variable which can take any value from the set of real numbers.

Independent variables used in this chapter include temporal variables (season, month, day and hour) and holiday and weather variables (temperature, apparent temperature, relative humidity, wind speed, cloud cover and dew point). Season, month and day are nominal variables whereas hour is an ordinal variable. To have correct estimates, we decompose both nominal and ordinal variables in to binary (or dummy) variables for each level. Holiday is a binary variable and the six weather variables are continuous. Tables 3.1, 3.2 and 3.3 provide a more detailed description of the dependent variables, binary independent variables and continuous independent variables respectively.

Table 3.1: Dependent Variables Used in this Chapter

Variable Name	*Variable Description*
Daily Dropoffs	Number of dropoffs in that day
Hourly Dropoffs	Number of dropoffs in that hour
Daily Pickups	Number of pickups in that day
Hourly Pickups	Number of pickups in that hour
Imbalance	Difference of the number of dropoffs and pickups in that hour

3.3.2 Data Descriptions

We test our proposed methods on the SABB FFBS program at USF, Tampa. Phase I of the program was launched in August 2015, providing 100 bikes to students, staff and faculty at no charge if the users limited their cumulative usage time to less than two hours per day. An hourly fee was imposed for the extra time beyond the daily two hour free quota. With Phases II and III in the coming years, the program will be expanded to 300 bikes and cover both the Tampa campus and student housing in the vicinity of the campus. The program is expected to be integrated with parking management and other multi-modal transportation initiatives on the campus. USF researchers collaborated with the bike sharing company and developed the program in 2015. Given it is a program operated and managed internally, USF researchers had full access to the usage data, including trajectory data, of the program. With built-in GPS and the application developed by Social Bicycles, the trip data (trajectory of bikes) of each usage of the bikes is recorded in the operation management system. All trips have a unique ID. Further, each trip has a user ID, bike ID, starting timestamps, starting latitude, starting longitude, ending timestamps, ending latitude, ending longitude, trip duration (in minutes) and trip distance (in miles). Thus, the SABB program provided the perfect setting to test our proposed method. The time frame of this study was from August 28, 2015, the launch date of the program to March 30, 2017. During this time frame, a total of $189,082$ trips were recorded. However, many of these trips were noise; hence, they had to be identified and subsequently

Table 3.2: Binary Independent Variables Used in this Chapter

Variable Name	*Variable Description*
Spring Season Indicator	1 if Spring, 0 otherwise
Autumn Season Indicator	1 if Autumn, 0 otherwise
Summer Season Indicator	1 if Summer, 0 otherwise
Fall Season Indicator	1 if Fall , 0 otherwise
January Indicator	1 if January, 0 otherwise
February Indicator	1 if February, 0 otherwise
March Indicator	1 if March, 0 otherwise
April Indicator	1 if April, 0 otherwise
May Indicator	1 if May, 0 otherwise
June Indicator	1 if June, 0 otherwise
July Indicator	1 if July, 0 otherwise
August Indicator	1 if August, 0 otherwise
September Indicator	1 if September, 0 otherwise
October Indicator	1 if October, 0 otherwise
November Indicator	1 if November, 0 otherwise
December Indicator	1 if December, 0 otherwise
Monday Indicator	1 if Monday, 0 otherwise
Tuesday Indicator	1 if Tuesday, 0 otherwise
Wednesday Indicator	1 if Wednesday, 0 otherwise
Thursday Indicator	1 if Thursday, 0 otherwise
Friday Indicator	1 if Friday, 0 otherwise
Saturday Indicator	1 if Saturday, 0 otherwise
Sunday Indicator	1 if Sunday, 0 otherwise
Holiday Indicator	1 if Saturday or Sunday or a US Holiday, 0 otherwise
Hour 0 Indicator (00:00)	1 if after 12:00 AM and before 1:00 AM, 0 otherwise
Hour 1 Indicator (01:00)	1 if after 1:00 AM and before 2:00 AM, 0 otherwise
Hour 2 Indicator (02:00)	1 if after 2:00 AM and before 3:00 AM, 0 otherwise
Hour 3 Indicator (03:00)	1 if after 3:00 AM and before 4:00 AM, 0 otherwise
Hour 4 Indicator (04:00)	1 if after 4:00 AM and before 5:00 AM, 0 otherwise
Hour 5 Indicator (05:00)	1 if after 5:00 AM and before 6:00 AM, 0 otherwise
Hour 6 Indicator (06:00)	1 if after 6:00 AM and before 7:00 AM, 0 otherwise
Hour 7 Indicator (07:00)	1 if after 7:00 AM and before 8:00 AM, 0 otherwise
Hour 8 Indicator (08:00)	1 if after 8:00 AM and before 9:00 AM, 0 otherwise
Hour 9 Indicator (09:00)	1 if after 9:00 AM and before 10:00 AM, 0 otherwise
Hour 10 Indicator (10:00)	1 if after 10:00 AM and before 11:00 AM, 0 otherwise
Hour 11 Indicator (11:00)	1 if after 11:00 AM and before 12:00 PM, 0 otherwise
Hour 12 Indicator (12:00)	1 if after 12:00 PM and before 1:00 PM, 0 otherwise
Hour 13 Indicator (13:00)	1 if after 1:00 PM and before 2:00 PM, 0 otherwise
Hour 14 Indicator (14:00)	1 if after 2:00 PM and before 3:00 PM, 0 otherwise
Hour 15 Indicator (15:00)	1 if after 3:00 PM and before 4:00 PM, 0 otherwise
Hour 16 Indicator (16:00)	1 if after 4:00 PM and before 5:00 PM, 0 otherwise
Hour 17 Indicator (17:00)	1 if after 5:00 PM and before 6:00 PM, 0 otherwise
Hour 18 Indicator (18:00)	1 if after 6:00 PM and before 7:00 PM, 0 otherwise
Hour 19 Indicator (19:00)	1 if after 7:00 PM and before 8:00 PM, 0 otherwise
Hour 20 Indicator (20:00)	1 if after 8:00 PM and before 9:00 PM, 0 otherwise
Hour 21 Indicator (21:00)	1 if after 9:00 PM and before 10:00 PM, 0 otherwise
Hour 22 Indicator (22:00)	1 if after 10:00 PM and before 11:00 PM, 0 otherwise
Hour 23 Indicator (23:00)	1 if after 11:00 PM and before 12:00 PM, 0 otherwise

Table 3.3: Continuous Independent Variables Used in this Chapter

Variable Name	*Variable Description*
Apparent Temperature	Numerical value representing apparent ("feels like") temperature at a given time in degrees Fahrenheit
Cloud Cover	Numerical value between 0 and 1 (inclusive) representing percentage of sky occluded by clouds
Dew Point	Numerical value representing dew point at a given time in degrees Fahrenheit
Relative Humidity	Numerical value between 0 and 1 (inclusive) representing relative humidity
Temperature	Numerical value representing temperature at a given time in degrees Fahrenheit
Wind Speed	Numerical value representing wind speed in miles per hour

removed before any further analysis could be conducted. Trips with the following properties were removed:

- if trip duration $\leq$ 30 seconds, in such case, the user might be checking the bike without using it.
- if trip duration $\geq 1.5\times$ inter-quantile range of the trip duration + mean of trip duration, in such case, the user might have forgotten to lock the bike after completion of the trip.
- if trip distance $\leq$ .000621371 miles or 1 meter, in such case, the bike might be damaged after short usage and the user may not able to complete his/her trip.
- if the trip either started or ended outside the USF, Tampa campus.
- if the trip is owing to a rebalancing operation.
- if the trip was conducted for testing the system.

After removing trips with the above mentioned properties, there was a total of $147,438$ trips. From this cleaned trip data, first daily and hourly dropoffs and pickups were extracted, followed by hourly imbalance. In the case of dropoffs and pickups, their corresponding time was the starting timestamps and the ending timestamps of that particular trip respectively. From the respective timestamps, the nominal temporal variables *Season, Month, Day and Hour* were computed using date and time functions in the Julia standard library (jul, 2017) and to check whether it was a holiday, the *BusinessDays.jl* package (bus, 2017) was used. Once the nominal temporal variables were created, they were converted into binary (or dummy) variables, to prevent erroneous statistical estimation.

Daily and hourly weather data for the USF, Tampa campus from August 28, 2015 to March 30, 2017 were obtained using the dark sky api (dar, 2017a), which offers historical weather data for both daily and hourly time-frames. (dar, 2017a) is backed by a wide range of data sources, which are detailed in (dar, 2017b). Daily and hourly weather data were then joined with the daily and hourly dropoffs and pickups as well as hourly imbalance data to obtain the final data that was used for the statistical analysis in this chapter.

3.3.3 Decomposing Continuous Independent Variables

Each continuous variable was decomposed into four binary variables, each of which represents a quantile range. For example, if we have a continuous variable ContVar whose quantiles are Q_1, Q_2, Q_3, Q_4, Q_5, we create four binary variables ContVar 1, ..., ContVar 4, such that ContVar $1 = 1$ if $Q_1 \leq$ ContVar $< Q_2$, 0 otherwise. Table 3.4 describes the quantiles of the six continuous variables. Thus when $36.51^o F \leq$ Temperature $< 67.25^o F$, Temperature $1 = 1$ and Temperature $2 =$ Temperature $3 =$ Temperature $4 = 0$.

This operation has four major advantages. First, binary variables are easier to interpret. Second, a continuous variable by itself may not be statistically significant but one of its corresponding binary variables may be. This is in fact true in the case of the SABB dataset and is demonstrated in Section 5. Third, adding such binary variables in (quasi-) Poisson and linear regression models may improve their out-of-sample performance. This is again true in case of the SABB dataset and is demonstrated in Section 4. Finally, it is difficult to derive interactions between independent variables if one or more

Table 3.4: Quantiles of Continuous Variables

Continuous Variables	*Quantile*				
	Zeroth	First	Second	Third	Fourth
Apparent Temperature	28.11	67.25	75.09	82.495	107.23
Cloud Cover	0.0	0.03	0.1	0.22	1.0
Dew Point	16.55	58.16	66.0	73.08	82.14
Relative Humidity	0.16	0.62	0.79	0.89	1.0
Temperature	35.61	67.25	75.09	80.37	94.99
Wind Speed	0.0	3.87	5.66	7.82	26.55

are continuous. So, adding binary variables corresponding to continuous variables make interactions involving continuous variables indirectly possible.

3.3.4 Interactions between Binary Independent Variables

Now that we have made sure that there are binary variables corresponding to each continuous variable, we can proceed to derive interaction among binary variables. In this chapter, we refer to the product of any two or any three independent binary variables, as second order and third order interactions respectively. If $\text{BinVar 1}, \text{BinVar 2}, \text{BinVar 3}$ are three independent primary binary variables, $\text{BinVar 1} \times \text{BinVar 2}, \text{BinVar 2} \times \text{BinVar 3}$, $\text{BinVar 3} \times \text{BinVar 1}$ and $\text{BinVar 1} \times \text{BinVar 2} \times \text{BinVar 3}$ are second and third order interactions respectively of the three independent binary variables. Further, by definition all second and third order interactions are also binary variables.

It is important to note that, some of the above mentioned second and third order interactions will have zero variance. Such interactions should not be considered. Any interactions between binary variables for the same original variable will have zero variance, i.e, the product of any two season indicator variable will have zero variance. The same holds true for binary/indicator variables corresponding to continuous variables. Further, to prevent creation of unnecessary interactions, interactions between season and month, weekends and holiday are not considered. To ease in the variable selection procedure, certain interactions whose variance is below a predetermined threshold may also be removed. However, we do not employ any such procedure in this chapter.

It is also not very clear *a priori* up to what order of interactions should be considered to achieve a desirable performance. One way of determining the highest order of interactions to be considered is via discussions and inputs from the operator, the primary user of such an analysis. Another approach is by comparing the out of sample testing errors of models with different orders of interactions used for training them. The order after which the testing error starts increasing significantly is an indication of overfitting and should be chosen as the best order of interactions.

3.3.5 Variable Sets Used in this Chapter

In this chapter, *Var Set* refers to the set of independent variables used for training a statistical model. Four such sets are considered. The first and second sets consist of only primary (binary and continuous) variables and primary variables with decomposed binary variables of the primary continuous variables respectively. The third and fourth sets consist of all variables in the second set with all feasible second order interactions and all variables in the second set with all feasible second and third order interactions respectively.

3.3.6 Baseline Models

To study how pickups or dropoffs vary with time and or are affected by external events such as holidays or weather conditions, negative binomial regression is commonly used in the literature (Gebhart and Noland (2014)). Negative binomial regression is more appropriate than Poisson regression for the SABB dataset, as the variance of both daily and hourly dropoffs and pickups is significantly larger than their respective means. Negative binomial regression, like Poisson regression, can also be modeled as a zero-inflated or a zero-truncated model. However, in this chapter no such modification is required, as we are only interested in the process that generates non-zero count variables (pickups or dropoffs). To study how hourly imbalance varies with time and or is affected by external events such as holidays or weather conditions, linear regression is used. This is because, unlike dropoffs and pickups, imbalance can also assume a negative value.

Unlike linear regression, it is difficult to interpret the coefficients of the independent variables in a negative binomial regression model directly. For this purpose, two other parameters are commonly estimated for the independent variables to determine their effects on the dependent variable. They are known as elastic and marginal effects. Elasticity of an independent variable provides an estimate of the effect of a 1% change in the independent variable on the expected frequency of the dependent variable. They provide a measure of evaluating the relative impact of each independent variable in the model. However in this chapter we focus on using marginal effects rather than elastic effects owing to the ease of interpretation of marginal effects over elastic effects. Marginal effects can be more easily interpreted than elastic effects, particularly for binary variables, which are extensively present in the models used in this chapter. Unlike elastic effects, marginal effects measure the effect of one unit change in the independent variable on the dependent variable. For more details on negative binomial regression models, refer to Washington et al. (2010). We use the *pscl* psc (2015) and *mfx* mfx (2015) packages in R to estimate all the negative binomial regression models and their respective average marginal effects respectively.

It is interesting to note that, when *Var Set 3* and *4* are used, the number of independent variables outnumbers the number of observations. In such a scenario, estimating the coefficients of a negative binomial regression using maximum likelihood estimation or a linear regression using least squares cannot be used. To deal with such scenarios, we propose two stage models. In the first stage, at most *n* statistically significant variables are selected from the set of independent variables using a variable selection method. Once a set of variables less than the number of observations has been selected, these selected variables are used to estimate either a negative binomial or a linear regression model.

3.3.7 Regularization

In this section we describe two regularization strategies used in this chapter:

1. Least Absolute Shrinkage and Selection Operator (LASSO) Tibshirani (1996)

2. ElasticNet Zou and Hastie (2005)

LASSO was introduced in Tibshirani (1996). LASSO performs both shrinkage and variable selection over a set of variables to improve the prediction accuracy and interpretability of the model. Despite having some attractive properties and features, LASSO has some disadvantages that may end up being problematic for this study. For example, if there are correlated variables, LASSO will arbitrarily select only one variable from a group of correlated variables.

ElasticNet, in certain instances, may be a better choice for regularization than LASSO, because of its above mentioned limitations. ElasticNet incorporates both L1 and L2 regularization which makes the coefficients of correlated variables shrink towards each other, while retaining the feature selection property of LASSO. This often results in selection of subsets of correlated variables. This property of ElasticNet makes it a competitive choice for variable selection along with LASSO. For more details on LASSO, ElasticNet and other regularization strategies refer to James et al. (2013) and Friedman et al. (2009).

We use the *glmnet* (Friedman et al., 2010) package in R to compute the regularization paths for both LASSO and ElasticNet for all models in this chapter. The *glmnet* package has no implementation of LASSO and ElasticNet corresponding to negative binomial distribution, so we use the implementation corresponding to Poisson distribution for daily and hourly dropoffs and pickups. This does not affect the variable selection procedure, as over-dispersion does not affect the estimates for the conditional mean. This is because, the estimating equations for the coefficients of the conditional mean are equivalent for both Poisson and negative binomial regression models. Therefore the point estimates are identical for both Poisson and negative binomial regression models when using either LASSO or Elastic Net.

Two primary parameters α and λ in glmnet need to be tuned. When $\alpha = 1$, glmnet only uses L1 regularization (LASSO) and when $0 < \alpha < 1$, glmnet uses a combination of L1 and L2 regularization (ElasticNet). Thus we vary α from 0.1 to 1.0 with a step size of 0.1. The parameter λ for both LASSO and ElasticNet is selected using 5-fold cross validation. All other parameters in glmnet are set to its default values.

3.3.8 Models Used in this Chapter

Three distinct models *Model 1*, *Model 2* and *Model 3* are used in this chapter. In case of daily and hourly dropoffs and pickups, *Model 1* refers to the commonly used negative binomial regression model in the literature. In case of hourly imbalance, *Model 1* refers to the linear regression model. *Model 1* is valid only for *Var Sets 1* and *2* as for *Var Sets 3* and *4* the number of independent variables is greater than the number of observations. The other two models *Model 2* and *Model 3* used in this chapter are two stage models. In the first stage, a regularization strategy is used to select at most n statistically important variables from the respective variable set. This is then followed by either negative binomial regression for dropoffs and pickups or linear regression for imbalance on the set of selected variables. The first stage in *Model 2* and *Model 3* is using LASSO ($\alpha = 1$) and ElasticNet ($0 < \alpha < 1$) as the respective regularization strategy.

3.3.9 Model Selection

Various metrics can be used to measure the quality of a negative binomial regression model. Two commonly used metrics are ρ^2 and out of sample testing error. ρ^2 statistic, also sometimes referred to as the McFadden ρ^2 is $1 - \dfrac{LL(\beta)}{LL(0)}$ where $LL(\beta)$ is the log-likelihood at convergence and $LL(0)$ is the initial log-likelihood. The ρ^2 statistic for a negative binomial regression model is always between zero and one. The closer it is to one, the better the model is. Similarly, the two most commonly used metrics for selecting linear regression models are Adjusted R^2 and out of sample testing error. The Adjusted R^2 statistic for a linear regression model is always between zero and one. The closer it is to one the better the model is.

Although ρ^2 and Adjusted R^2 statistics for negative binomial and linear regression are commonly used and provide some valuable information about the quality of a model, they fail to ascertain how well the model generalizes out of the training set. In other words, these metrics are unable to detect overfitting as they measure the quality of the model on the training set. Thus, the other measure, i.e., the root mean square error (RMSE) of the models on the hold out / testing set will be used for selecting the final models.

The dataset used in this chapter, is split into two sets, the training and the testing set. The training set is used for estimating the models and comprises of trips from August 28, 2015 to February 28, 2017. The testing set is used for selecting the models. It measures how well the models generalizes out of the training set. It comprises of trips from March 1, 2017 to March 30, 2016.

3.4 Experimental Results

This section summarizes the experimental results of the proposed methods on the SABB FFBS dataset. Tables 3.5 and 3.6 summarizes the training and testing error measures for all statistical models of dropoffs and pickups and of imbalance respectively. Tables 3.7 and 3.8 reports the total number of variables and the number of variables selected corresponding to each model of dropoffs and pickups and of imbalance respectively. In Tables 3.7 and 3.8, *Vars Sel* and *SS Vars* refers to number of variables selected and the number of statistically significant variables (with 90% confidence intervals) among the variables selected for the corresponding model respectively.

Table 3.5: Summary of Training and Testing Error Measures for All Models of Dropoffs and Pickups

Variable	*Time-frame*	*Var Set*	*Model Used*					
			Model 1		*Model 2*		*Model 3*	
			ρ^2	*RMSE*	ρ^2	*RMSE*	ρ^2	*RMSE*
Dropoffs	*Daily*	*1*	0.0438	256.9260	0.0362	189.5043	0.0366	188.2041
		2	0.0470	253.3899	0.0439	152.6411	0.0378	*150.3104*
		3	-		0.0702	224.1921	0.0617	231.6895
		4			0.0854	*148.4511*	0.0873	186.2352
Pickups		*1*	0.0437	256.8616	0.0437	256.8616	0.0365	184.3904
		2	0.0470	253.3143	0.0378	150.3913	0.0378	*150.3913*
		3	-		0.0620	231.1562	0.0661	252.7010
		4			0.0955	190.7476	0.0903	*144.1414*
Dropoffs	*Hourly*	*1*	0.1161	11.9317	0.1161	11.9317	0.1161	11.9317
		2	0.1179	11.2325	0.1179	11.2325	0.1179	*11.2325*
		3	-		0.1668	18.7437	0.1668	18.7437
		4			0.1945	15.1279	0.1915	*14.5176*
Pickups		*1*	0.1159	11.9516	0.1159	11.9516	0.1159	11.9516
		2	0.1178	11.2552	0.1178	11.2552	0.1178	*11.2552*
		3	-		0.1667	17.2632	0.1667	17.2632
		4			0.1982	14.5979	0.1940	*14.0161*

Table 3.6: Summary of Training and Testing Error Measures for All Models of Hourly Imbalance

Variable	*Time-frame*	*Var Set*	*Model Used*					
			Model 1		Model 2		Model 3	
			Adjusted R^2	*RMSE*	Adjusted R^2	*RMSE*	Adjusted R^2	*RMSE*
Imbalance	*Hourly*	1	0.0422	0.6503	0.0442	0.6484	0.0441	0.6487
		2	0.0420	0.6495	*0.0444*	*0.6483*	0.0444	0.6484
		3	-		*0.1250*	*0.7262*	0.1259	0.7448
		4			0.1857	0.7326	0.1857	0.7326

Table 3.7: Summary of Variable Selection for All Models of Dropoffs and Pickups

Variable	*Time-frame*	*Var Set*	*Total Vars*	*Model Used*					
				Model 1		*Model 2*		*Model 3*	
				Vars Sel	*SS Vars*	*Vars Sel*	*SS Vars*	*Vars Sel*	*SS Vars*
Dropoffs	*Daily*	*1*	27	27	19	12	6	16	6
		2	44	44	19	33	16	*18*	*11*
		3	928	-		100	36	70	23
		4	8160			*127*	*45*	132	44
Pickups		*1*	27	27	19	27	19	14	7
		2	44	44	19	18	11	*18*	*11*
		3	928	-		75	23	89	26
		4	8160			160	51	*149*	*45*
Dropoffs	*Hourly*	*1*	50	50	47	50	47	50	47
		2	66	66	57	66	57	*66*	*57*
		3	2146	-		922	378	922	378
		4	31734			1348	617	*1271*	*578*
Pickups		*1*	50	50	46	50	46	50	46
		2	66	66	57	66	56	*66*	*56*
		3	2146	-		906	371	906	371
		4	31734			1486	695	*1350*	*617*

Table 3.8: Summary of Variable Selection for All Models of Imbalance

Variable	*Time-frame*	*Var Set*	*Total Vars*	*Model Used*					
				Model 1		Model 2		Model 3	
				Vars Sel	*SS Vars*	*Vars Sel*	*SS Vars*	*Vars Sel*	*SS Vars*
Imbalance	*Hourly*	*1*	50	50	12	18	14	19	14
		2	66	66	19	*24*	*16*	23	15
		3	2146	-		*184*	*131*	201	133
		4	31734			170	137	170	136

Table 3.9: Selected Models

Variable	*Time-frame*	*Selected Model*	
		No Interactions	*With Interactions*
Dropoffs	*Daily*	*Model 3* with *Var Set 2*	*Model 2* with *Var Set 4*
Pickups		*Model 3* with *Var Set 2*	*Model 3* with *Var Set 4*
Dropoffs	*Hourly*	*Model 3* with *Var Set 2*	*Model 3* with *Var Set 4*
Pickups		*Model 3* with *Var Set 2*	*Model 3* with *Var Set 4*
Imbalance		*Model 2* with *Var Set 2*	*Model 2* with *Var Set 3*

Models in this chapter were selected based on their testing errors, because they are a better indicator of how a model performs out of the training set, i.e., how well it generalizes out of the training set. Needless to say, the lower the testing error, the better the model is. However, if two models have similar testing errors, their training error measures can be used for breaking the tie. Unlike the testing error measure, the higher the ρ^2 or Adjusted R^2 of a model the better it is. The best models for each category are summarized in Table 3.9 based on the results from Tables 3.5 and 3.6.

From Tables 3.5 and 3.6, it is evident that *Var Set 2* always performs better than *Var Set 1* for all models on the SABB dataset. This indicates that it is advantageous to use *Var Set 2* instead of *Var Set 1* for training a model with no interactions on the SABB dataset, as opposed to the current trend in the literature. We also observe that, *Model 3* outperforms *Model 2* when the dependent variable is a count variable (dropoffs and pickups) except for daily dropoffs. However, the reverse is true when the dependent variable is a real number (imbalance). This indicates that 1) it is always advantageous to use either *Model 2* or *Model 3* instead of *Model 1* for training a model on the SABB FFBS dataset and 2) for training models related to dropoffs and pickups, *Model 3* is the recommended option whereas for training models related to imbalance, *Model 2* is the recommended option.

Another interesting observation is that, the sparsest model is always performing the best. By the sparsest model, we refer to the model whose *Vars Sel* is the lowest. This in a way is an indication that the simpler the model is, the better it tends to perform. Hence, we can conclude that two stage models proposed in this chapter, generates models that are not only simple/sparse (models with fewer number of variables) but also closer to the ground truth (as their testing errors are lower) than the baseline *Model 1* with *Var Set 1*, commonly used in the literature. It is interesting to note that, when interactions are added to the model, it sometimes performs better than models with no interactions and sometimes does not. However, it is almost always true that the quality of the model improves when the order of the interactions is increased, except for hourly imbalance.

Although, we limit ourselves to third order interactions in this chapter, this indicates that increasing the order of the interactions from third to fourth or even fifth may improve the quality of the model, but it will come at a higher cost of computational complexity and difficultly in interpreting the resulting model.

Adding interactions does not always improve the testing error of a model (it always improve the training error). For example: from Table 3.5, it is evident that for daily time-frame, the best models with interactions outperform the best models without interactions, however the same cannot be said for hourly time-frames. This leads to some interesting insights. For daily time-frame, *Model 2* and *Model 3* with *Var Set 4* for dropoffs and for pickups respectively, have some third order interactions (mentioned in Table 10) which by themselves are not statistically significant in *Model 3* with *Var Set 2* for both dropoffs and pickups. This is a clear indication that the best models with interactions are able to capture information, which were missed by the corresponding best models with no interactions. This characteristic of the best models with interactions being able to capture information that the best models without interactions cannot becomes more evident in Section 3.5.3. Thus, it important that instead of choosing a model with or without interactions over another, both models are used in conjunction to complement each other weaknesses with their strengths.

3.5 Discussion

In this section, we demonstrate how to interpret and draw inferences from visualization of historical data, best models with no interactions, best models with interactions and by combining all these methods. Then, we demonstrate how to provide appropriate recommendations to the operator, based on these respective inferences. In this chapter, we use only pickups and imbalance for drawing inferences and providing recommendations. The reason for this is two-fold: 1) to prevent repetition and 2) in the case of free-floating systems, dropoffs have very little effect on the demand of system as they have no explicit

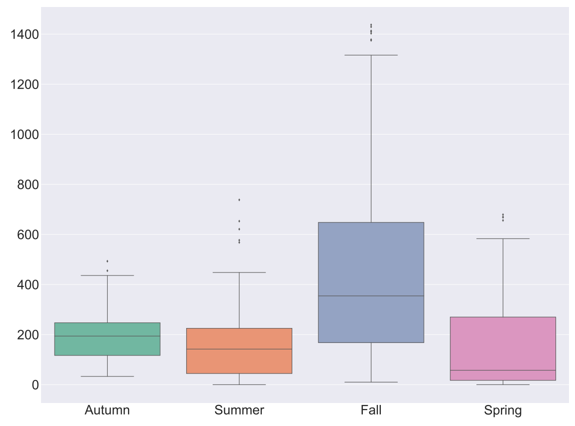
Table 3.10: Variables that Become Significant when Combined Together

Independent Variable	*Time-frame*	*Dependent Variables*		
		Variable 1	*Variable 2*	*Variable 3*
Dropoffs	*Daily*	Spring	Wind Speed 2	Cloud Cover 3
		September	Tuesday	Cloud Cover 3
		February	Tuesday	Relative Humidity 4
		Spring	Temperature 2	Cloud Cover 2
		Monday	Cloud Cover 2	Relative Humidity 2
		September	Wind Speed 3	Cloud Cover 2
		September	Temperature 4	Wind Speed 2
		Tuesday	Cloud Cover 2	Relative Humidity 4
		Tuesday	Temperature 1	Wind Speed 3
		February	Monday	Cloud Cover 4
Pickups		September	Tuesday	Cloud Cover 3
		February	Wind Speed 1	Cloud Cover 1
		November	Wind Speed 1	Cloud Cover 2
		February	Tuesday	Relative Humidity 4
		October	Dew Point 2	Cloud Cover 4
		September	Temperature 4	Wind Speed 2
		September	Dew Point 3	Relative Humidity 4
		February	Monday	Cloud Cover 4
		Tuesday	Cloud Cover 2	Relative Humidity 4
		Apparent Temperature 3	Dew Point 3	Cloud Cover 1

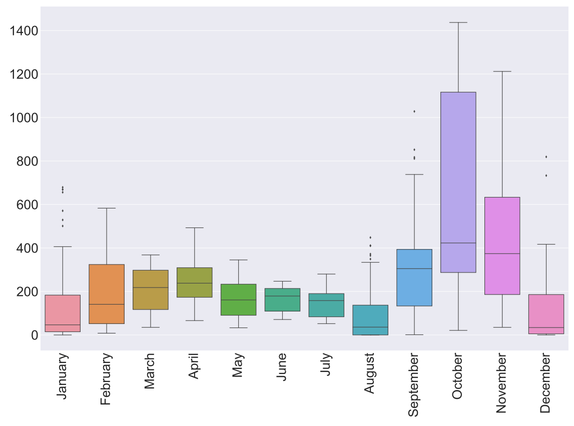
(capacity) restriction, unlike in the case of station-based systems. Further, pickups for both free-floating and station-based systems is a far better indicator of the approximate demand of the system. In case of station-based systems, dropoffs may also be considered in conjunction to pickups.

3.5.1 Data Visualization

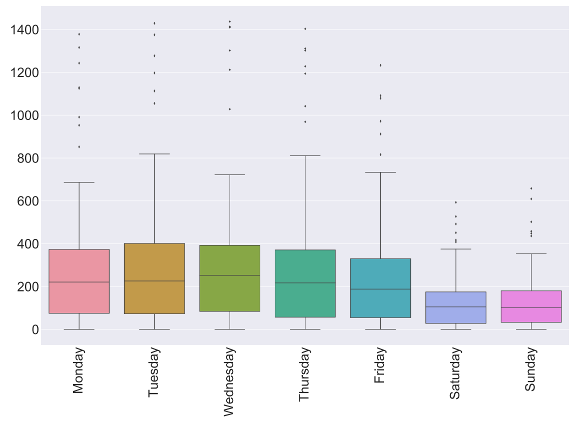
Figures 4.9 through 3.1d visualize how daily pickups vary with season, month, day and holiday respectively, in the SABB dataset. Figures 3.1e and 3.1f visualize how hourly pickups and imbalance vary with hours in a day respectively, in the SABB dataset. From Figures 4.9 and 3.1b, we can infer that there is significant variation in pickups owing to both season and month. The two primary causes for this phenomenon, are the correlation of both season and month with the timing of semesters at USF and weather conditions. Most trips are reported in the Fall semester, when the weather is pleasant. There is a dip in



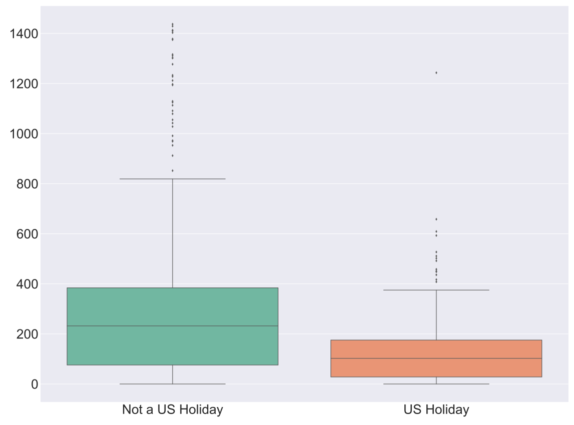
(a) Variation of Daily Pickups with Season



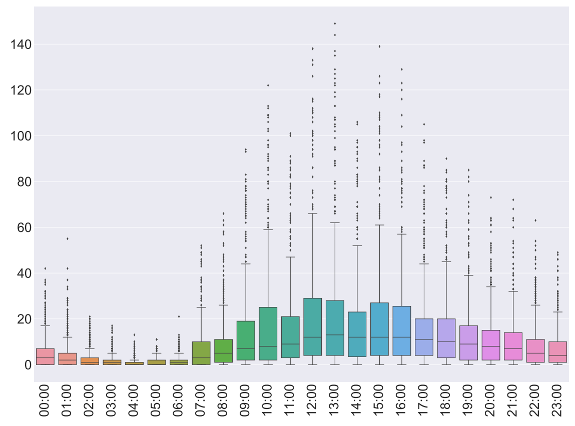
(b) Variation of Daily Pickups with Month



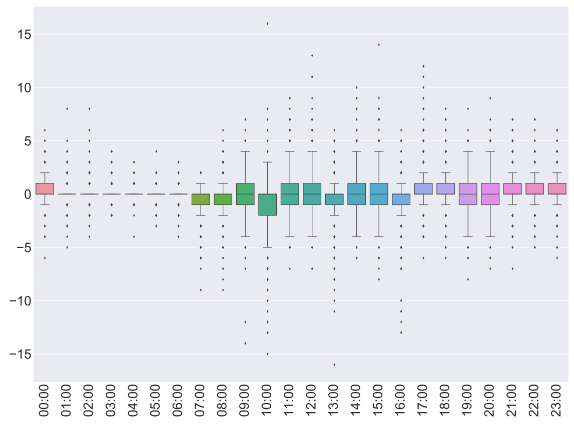
(c) Variation of Daily Pickups with Day



(d) Variation of Daily Pickups with Holiday



(e) Variation of Hourly Pickups with Hour



(f) Variation of Hourly Imbalance with Hour

Figure 3.1: Variation of Dependent Variables with Temporal Variables

usage for both the Spring and Summer semesters because the weather in the beginning of both of these semesters is a bit more severe compared to that in the fall semester. Further, fewer students are present on campus during the Summer semester. From Figures 3.1c and 3.1d, we can conclude that pickups are higher on weekdays than on weekends or holidays. This is owing to more activity (inter class or dorm to class or class to dorm trips) on campus on weekdays than on weekends. Pickups are maximum on Tuesday, followed by Wednesday, Monday, Thursday and Friday. This is because, most USF classes are held on Tuesday, followed by Wednesday, Monday, Thursday and Friday. From Figure 3.1e, we can conclude that pickups start increasing at 7:00 AM (when classes start), and peak around 1:00 PM. From Figure 3.1f, we can conclude that there is negative imbalance in the system from 7:00 AM to 9:00 AM, 10:00 AM to 11:00 AM, 1:00 PM to 2:00 PM and 4:00 PM to 5:00 PM. This phenomenon is because of class timings and extracurricular activity patterns of students and staff at USF. Based on Figures 4.9 through 3.1f, we recommend to the operator of the SABB FFBS that, the best time-frame for static rebalancing or on-site maintenance is 1:00 AM to 7:00 AM, because the pickups on average are almost close to zero during this time period and the appropriate time-frames for dynamic rebalancing are 9:00 AM to 10:00 AM, 11:00 AM to 1:00 PM and 2:00 PM to 4:00 PM.

3.5.2 Models with No Interactions

Figures 3.2 and 3.3, visualize the average marginal effects of statistically significant variables for the best models with no interaction for daily and hourly pickups respectively. From Figures 3.2 and 3.3, we can conclude that fall season (and its corresponding months) has a significant positive impact on both daily and hourly pickups. On the contrary, for both Spring and Summer seasons and for their corresponding months, there is a sudden dip for both daily and hourly pickups. From figure 3.3, it is clear that 11:00 AM to 12:00 PM is the peak time frame, which is a bit different than that obtained from data visualization. Further, the time frames 7:00 AM to 9:00 PM and 11:00 PM to 6:00 AM have

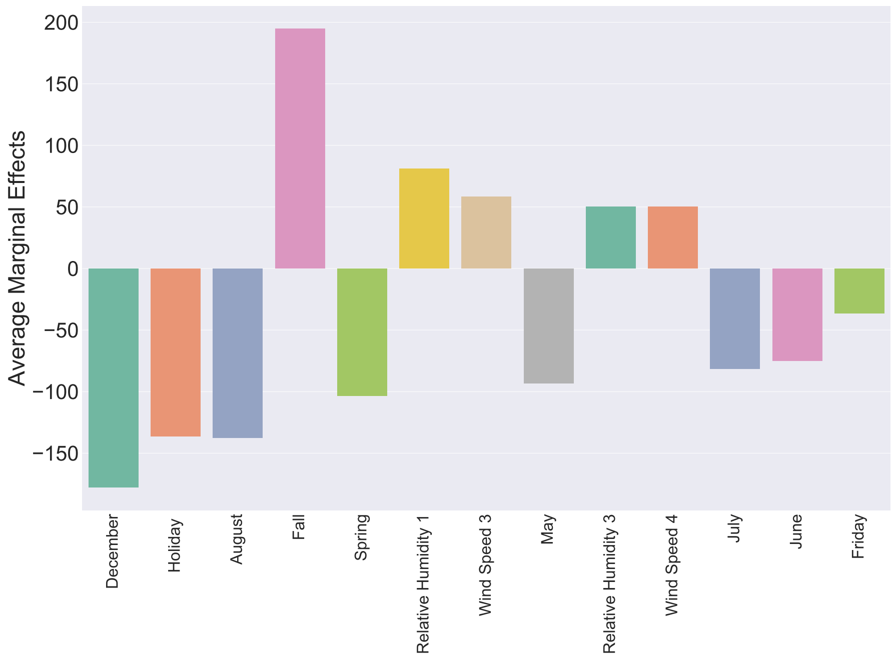


Figure 3.2: Average Marginal Effects of Statistically Significant Variables for the Best Model with No Interactions for Daily Pickups

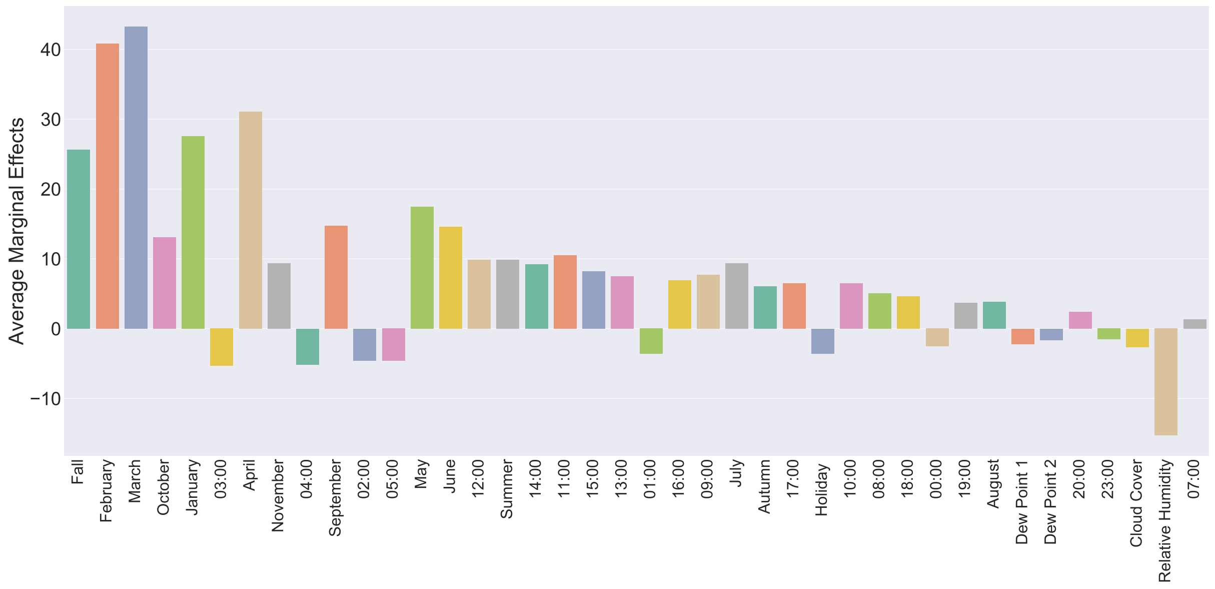


Figure 3.3: Average Marginal Effects of Statistically Significant Variables for the Best Model with No Interactions for Hourly Pickups

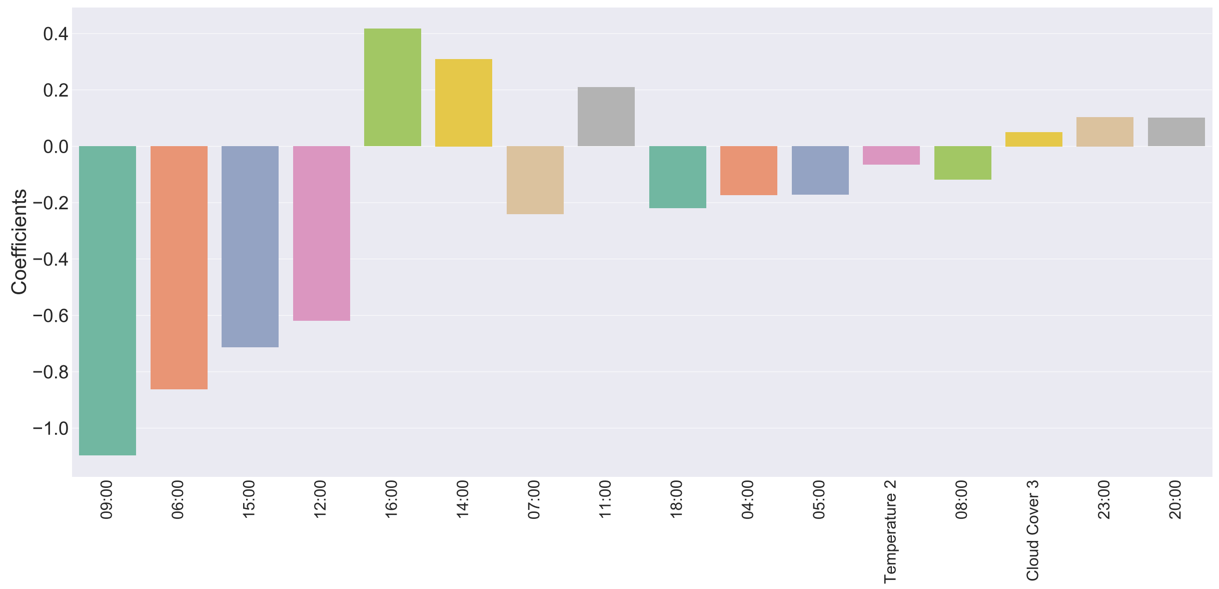


Figure 3.4: Coefficients of Statistically Significant Variables for the Best Model with No Interactions for Hourly Imbalance

a positive and a negative impact on hourly pickups respectively. It is not a surprise that both daily and hourly pickups decrease on holidays. It is interesting to note that, even though *dew point* and *wind speed* by themselves are not statistically significant, when the dew point is $16.55 - 66.0^o F$ and when wind speed is between $5.66 - 26.55$ mph they not only become statistically significant but also negatively impact hourly pickups. Further, hourly pickups decrease as the sky becomes more clouded, because it is less likely for users to commute using bikes when there is a high possibility of raining. Another interesting phenomenon occurs in the case of relative humidity. Relative humidity by itself negatively impacts hourly pickups, as it is a measure of extreme conditions. However, when relative humidity is either $0.16 - 0.62$ or $0.79 - 0.89$, pickups increase significantly. It is important to note that, we are able to identify these intervals for *dew point*, *wind speed* and *relative humidity* because of our proposed variable decomposition strategy.

Figure 3.4, visualize the coefficients of statistically significant variables for the best model with no interaction, for hourly imbalance. Figure 3.4 gives a clear indication of the time-frames of interest when imbalance is negative, i.e., 6:00 AM to 10:00 AM, 12:00 PM

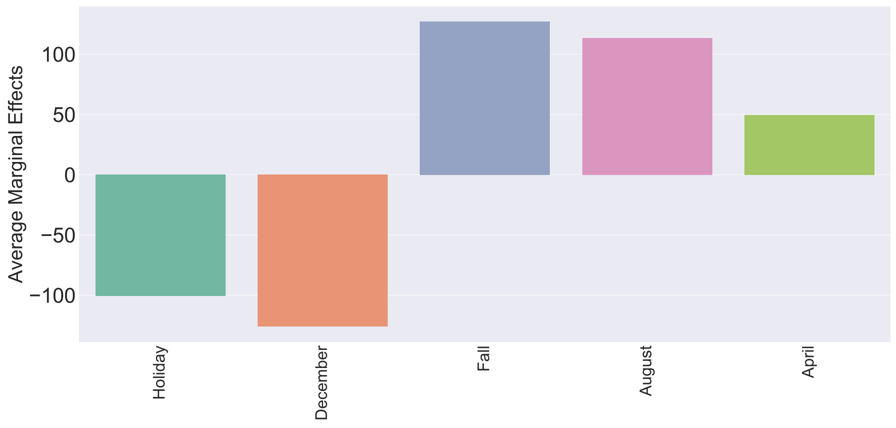


Figure 3.5: Average Marginal Effects of First Order Statistically Significant Variables for the Best Model with Interactions for Daily Pickups

to 1:00 PM and 3:00 PM to 4:00 PM. Thus, based on Figures 3.2, 3.3 and 3.4, we can provide the following three recommendations. First, (operator-based) static rebalancing and on-site maintenance operations can be conducted between 11:00 PM - 6:00 AM on a desired day. Second, dynamic rebalancing (both operator-based and user-based) if required should be held between the hours of 7:00 AM to 8:00 AM, 10:00 AM to 12:00 PM and 1:00 PM to 3:00 PM. Finally, we recommend the operator to use a user-based dynamic rebalancing / user incentives schemes in the Spring, in May, June, July, August and December, on Fridays and on holidays.

3.5.3 Models with Interactions

Figures 3.5 and 3.6, visualizes the average marginal effects of first order statistically significant variables for the best models with interactions, for both daily and hourly pickups respectively. From figures 3.2 and 3.3, we can conclude that fall season has a significant positive impact on both daily and hourly pickups. Similarly, December has a negative impact on both daily and hourly pickups. This is because many students return to their homes during this time after the semester has concluded. Thus there is a dip in the number of users. March and April as well as, October have a positive and a negative

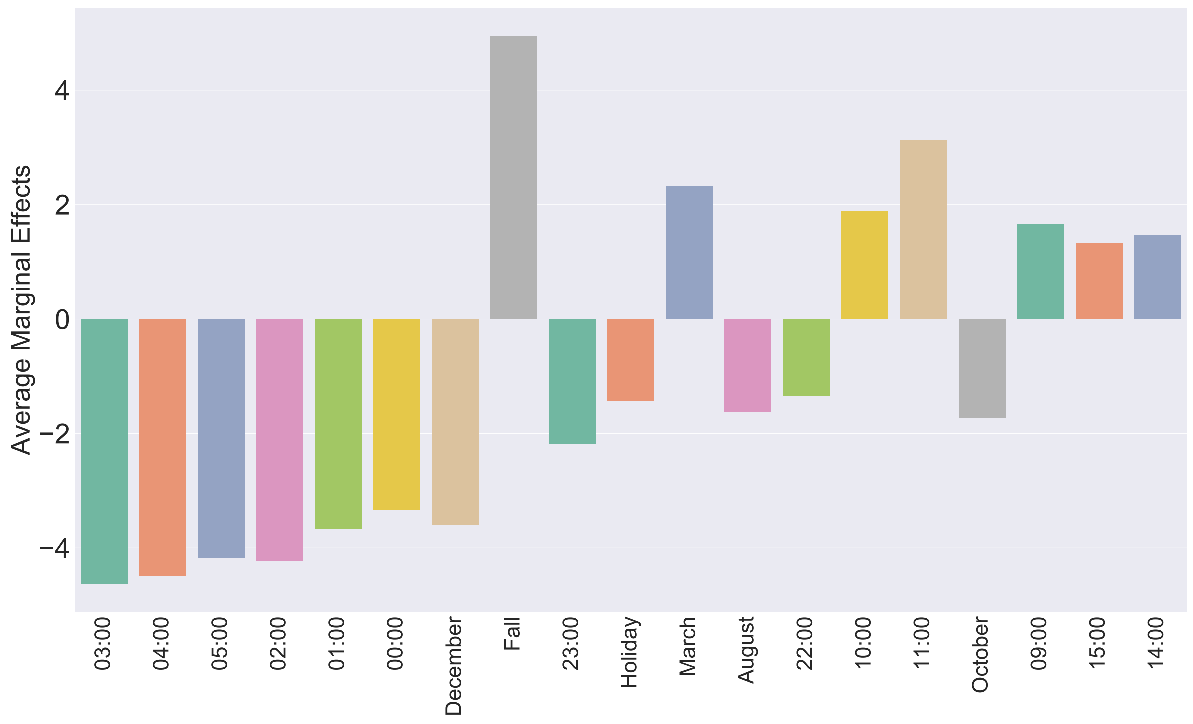


Figure 3.6: Average Marginal Effects of First Order Statistically Significant Variables for the Best Model with Interactions for Hourly Pickups

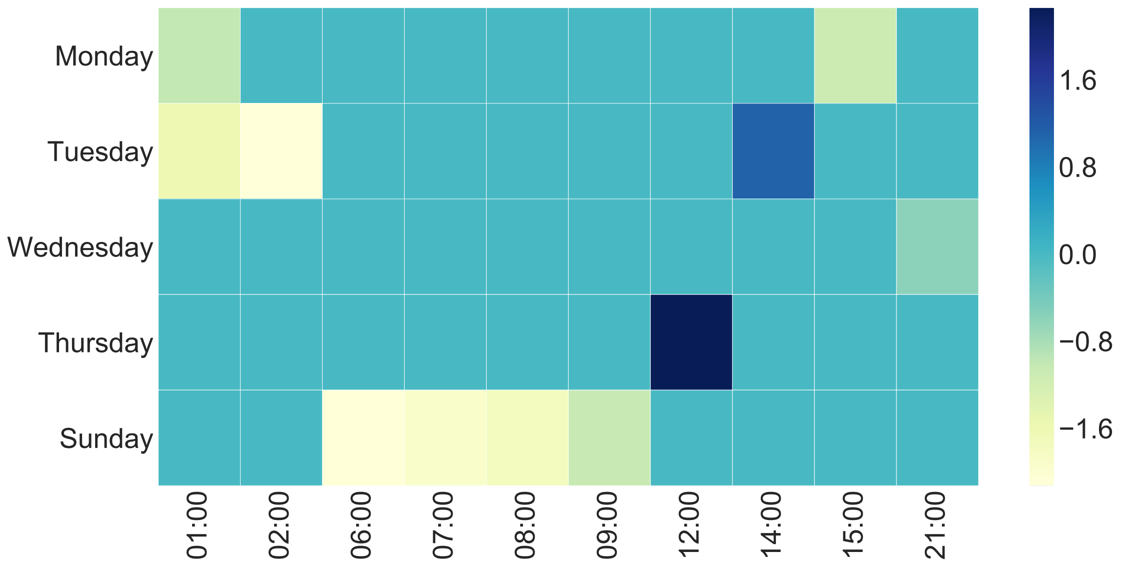


Figure 3.7: Average Marginal Effects of Second Order Statistically Significant Variables between Day, Holiday and Hour for the Best Model with Interactions for Hourly Pickups

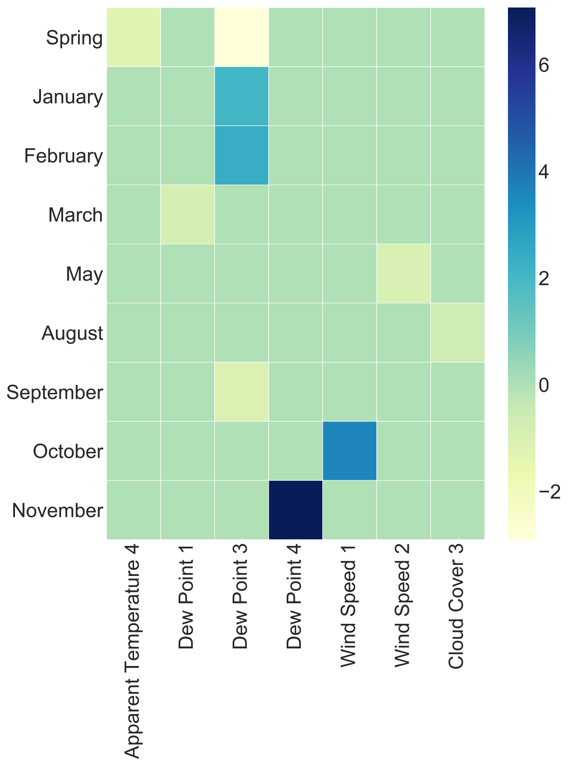
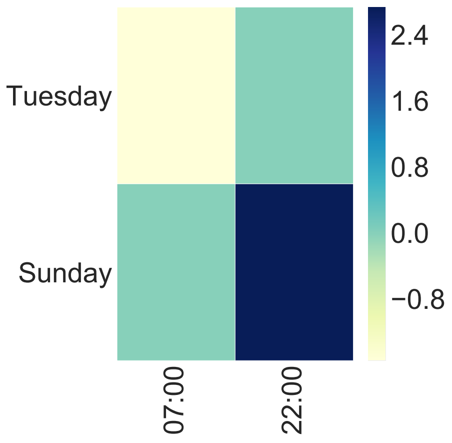
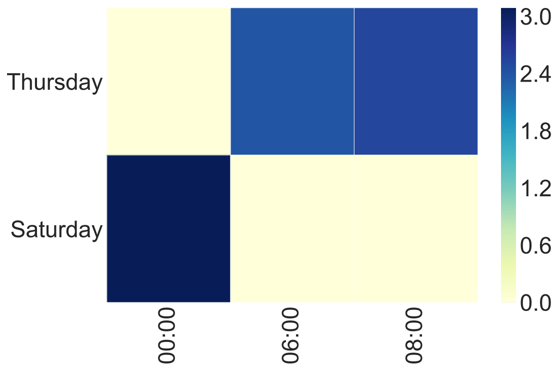


Figure 3.8: Average Marginal Effects of Second Order Statistically Significant Variables between Season, Month and Weather Variables for the Best Model with Interactions for Hourly Pickups



(a) September



(b) October

Figure 3.9: Average Marginal Effects of Third Order Statistically Significant Variables between September/October, Day, Holiday and Hour for the Best Model with Interactions for Hourly Pickups

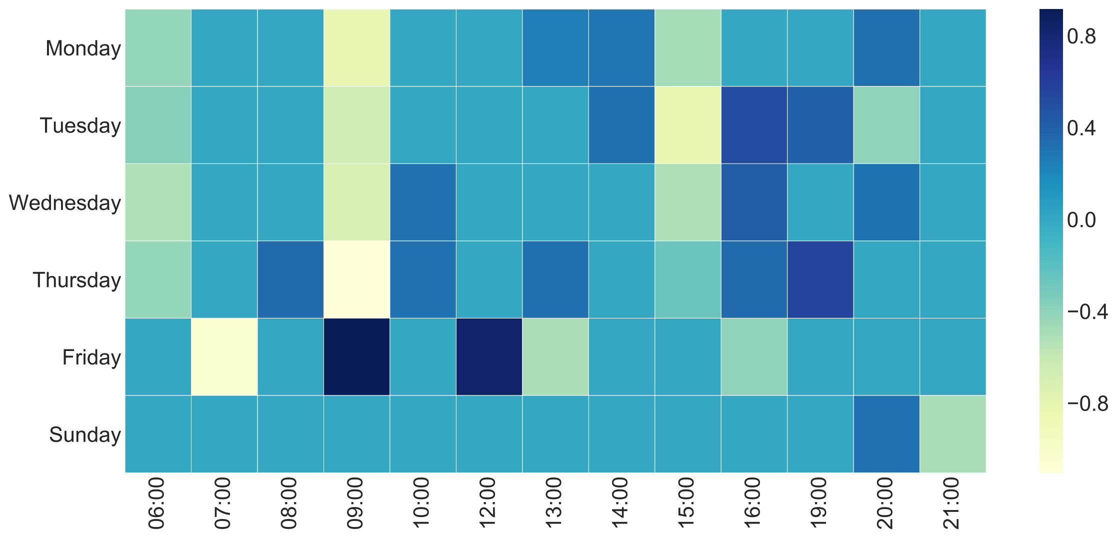


Figure 3.10: Coefficients of Second Order Statistically Significant Variables between Day, Holiday and Hour for the Best Model with Interactions for Hourly Imbalance

impact on pickups respectively. From figure 3.3, it is clear that 11:00 AM to 12:00 PM is the peak time frame, with the time frame 9:00 AM to 3:00 PM and 11:00 PM to 6:00 AM having a positive and a negative impact on hourly pickups respectively. It is not surprising that both daily and hourly pickups decrease during holidays.

Figures 3.7 and 3.8, visualize the average marginal effects of second order statistically significant variables between day, holiday and hour variables and between season, month and weather variables for the best model with interactions for hourly pickups respectively. From figure 3.7, we can make some interesting conclusions. First, there is a sudden drop in pickups on Mondays from 3:00 PM to 4:00 PM. Second, there is a sudden increase in pickups on Tuesdays from 2:00 PM to 3:00 PM. Finally, on Thursdays there is a sudden increase from 12:00 PM to 1:00 PM. Perhaps be on Thursdays the peak is from 12:00 PM to 1:00 PM instead of from 11:00 AM to 12:00 PM. From figure 3.8, we can make some interesting conclusions. When the apparent temperature is $82.495 - 107.23^o F$ during Spring, there is a decrease in hourly pickups. When the dew point is $16.55 - 58.16^o F$ during March, there is a decrease in hourly pickups. When the dew point is $66.00 - 73.08^o F$, there is a decrease in hourly pickups during Spring and during September, whereas the hourly pickups increases during the months of January and February. When the dew point is $73.08 - 82.14^{o}F$ during November, there is an increase in hourly pickups. When the wind speed is $0.00 - 3.87$ mph during October, hourly pickups increase. When the wind speed is $3.87 - 5.66$ mph during May, hourly pickups decrease. When the cloud cover is $0.1 - 0.22$ during August, hourly pickups decrease.

Figures 3.9a and 3.9b, visualize average marginal effects of third order statistically significant variables between September/October, day, holiday, and hour for the best model with interactions for hourly pickups respectively. From Figure 3.9a, we can conclude that in September, Tuesdays have a slower start compared to other months and on Sundays, there is an increase in pickups during 10:00 PM to 11:00 PM. From figure 3.9b, we can conclude that in October, Thursdays have an early start at 6:00 AM instead of at 7:00 AM, and on Saturdays there is a increase in pickups during 12:00 AM to 01:00 AM. The increase in pickups from 10:00 PM to 11:00 PM on Sundays in September and from 12:00 AM to 01:00 AM on Saturdays during October, may be because of students engaging in recreational activities during weekends in the middle of the fall semester.

Figure 3.10 visualizes the coefficients of second order statistically significant variables between day, holiday and hour for the best model with interactions for hourly imbalance. This figure provides a lot of valuable information. First, the trend of imbalance on a Friday is quite different from that on the other weekdays. Clearly, during 6:00 AM to 7:00 AM, 9:00 AM to 10:00 AM and 3:00 PM to 4:00 PM on Monday to Thursday there is negative imbalance in the system. On Friday, the negative imbalance is during 7:00 AM to 8:00 AM, 1:00 PM to 2:00 PM and 4:00 PM to 5:00 PM. This phenomenon arises due to the difference in class schedules on Friday compared to that on the other weekdays. On Sunday, there is a negative imbalance from 9:00 PM to 10:00 PM, which may be because of students engaging in recreational activities.

Based on the above inferences, we can provide the following three recommendations. First, (operator-based) static rebalancing and on-site maintenance operations can be conducted between 11:00 PM - 6:00 AM on a desired day, except for Tuesdays in September when it may be extended until 8:00 AM. Second, dynamic rebalancing (both operator-based and user-based), if required should be held from 10:00 AM to 3:00 PM on Monday through Thursday and from 9:00 AM to 1:00 PM and 2:00 PM to 4:00 PM on Friday. Third, we recommend the operator to use static rebalancing strategies in Fall, in April and August and dynamic rebalancing strategies in December and on holidays.

3.5.4 All Vantage Points

In this section, we synthesize inferences and recommendations derived from three vantage points, namely data visualization of historical data, best models with and without interactions. An inference or a recommendation is strongest if it can be validated by all of the above three methods, and weakest if only one of the above three methods validates it. For example: based on data visualization and best models with and without interactions, the best time for static rebalancing or onsite maintenance is from 1:00 AM to 7:00 AM, 11:00 PM to 6:00 AM and 11:00 PM to 6:00 AM respectively. However, if all three of these recommendations are combined, it is clear that 1:00 AM to 6:00 AM is a time frame that is valid from all of these three methods. Similar approach is followed in this section for inferences and recommendations.

Based on the above guidelines, we can draw the following conclusions about the mobility patterns of the SABB FFBS:

1. Fall has a significant positive impact on pickups, whereas, both Spring and Summer have a negative impact on pickups.

2. March and April have a positive impact, and October and December have a negative impact on pickups respectively.

3. Pickups are higher on weekdays than on weekends or holidays, reaching a peak on Tuesday, followed by Wednesday, Monday, Thursday and Friday.

4. Peak hours are from 11:00 AM to 12:00 PM (except for Thursdays when the peak is 12:00 PM to 1:00 PM), with the time frames 9:00 AM to 3:00 PM and 10:00 PM to 6:00 AM having a positive and a negative impact on pickups respectively.

5. There is a sudden decrease in pickups on Mondays from 3:00 PM to 4:00 PM and a sudden increase in pickups from 10:00 PM to 11:00 PM on Sundays in September and from 12:00 AM to 01:00 AM on Saturdays during October.

6. There is a decrease in pickups in Spring when the apparent temperature is $82.495 - 107.23^o F$.

7. In October, pickups increase when wind speed is $0.00 - 3.87$ mph, however, pickups decrease when wind speed is $3.87 - 5.66$ mph in May and between $5.66 - 26.55$ mph.

8. Pickups decrease when the dew point is $16.55 - 66.0^o F$, or $66.00 - 73.08^o F$ in Spring and September, however pickups increase when the dew point is between $66.00 - 73.08^o F$ in January and February and between $73.08 - 82.14^o F$ in November.

9. Pickups decrease with increase in cloud cover.

10. Relative humidity by itself negatively impacts pickups, however, when relative humidity is either $0.16 - 0.62$ or $0.79 - 0.89$, pickups increase significantly.

Similarly, based on the above guidelines, it is clear that during 6:00 AM to 7:00 AM, 9:00 AM to 10:00 AM and 3:00 PM to 4:00 PM on Monday to Thursday there is negative imbalance in the system. On Friday, the negative imbalance is during 7:00 AM to 8:00 AM, 1:00 PM to 2:00 PM and 4:00 PM to 5:00 PM. By combining insights and recommendations from all vantage points, we can provide the following final recommendations to the operator of the SABB FFBS. The best time for static rebalancing or on-site maintenance is between 1:00 AM and 6:00 AM, except for Tuesdays in September when it may be extended until 8:00 AM. Dynamic rebalancing (both operator-based and user-based), if required should be held from 10:00 AM to 12:00 PM and 1:00 PM to 3:00 PM on Monday through Thursday and from 9:00 AM to 1:00 PM and 2:00 PM to 4:00 PM on Friday. Static rebalancing strategies be extensively used in Fall and in April. Dynamic rebalancing strategies should be used in May, June, July and December, and on holidays.

3.6 Final Remarks

In this chapter, we propose a method to extract operational management insights from historical trip data of a shared mobility system, to help the operator make more informed decisions. A significant amount of research has been conducted on gaining various forms and types of insights with a broad range of motivation, from the historical data of the system. However, none of these studies considered interaction between independent variables or study imbalance as a dependent variable. In this chapter, we take interactions among independent variables into consideration and apply methods to remove unnecessary interactions. We also show that more insights about the mobility patterns and imbalance of the SABB program can be obtained by considering such interactions. We also propose a simple method to decompose continuous variables into binary variables which improves the base model used in the literature. Our proposed methodology gives a unique opportunity to study the system and make recommendations to the operator from various vantage points. To extend our proposed method for station-based systems, dropoffs can also be considered in conjunction to pickups.

Even though the two stage models perform better than baseline (quasi) Poisson regression models, their testing error measure is not as low as one would expect. A possible explanation for this effect is that both the two stage and the baseline models are linear models. Thus they are unable to capture possible non-linear relationships among the independent and the dependent variables. This effect is mitigated to some extent by adding up to third order interactions, as they are able to capture unobserved heterogeneity in the data. Adding fourth or even higher order interactions may improve the model, however doing so may make the model difficult to interpret. Thus, it is our belief that interactions higher than third order are unnecessary, instead nonlinear transformations and interactions may be added to determine if the performance of the models improves or not. This is a possible future research direction.

In future research, we will address how to use information from such an analysis to compute optimal inventory levels, which can then be used by the operator as inputs to their specific rebalancing strategies. Another possible research direction can be conducting this analysis for each station in case of station based bike sharing systems or each zone in case of free floating bike sharing systems.

4 Strategies to Increase Usable Bikes in Free-Floating Bike Sharing Systems

4.1 Problem Description and Related Work

Usable bikes become unusable for two major reasons: 1) from over usage by a subset of regular users and 2) from mishandling or vandalism by a subset of casual users. Once a usable bike becomes unusable, a user is unable to use it until it is repaired, decreasing his/her level of satisfaction. Now, the operator has to repair these unusable bikes either on-site or at a remote location, both of which involve routing and labor costs. These costs owing to the presence of unusable bikes can be minimized, if the operator employs strategies to prevent usable bikes from being converted to unusable bikes owing to over usage or mishandling and vandalism. In this chapter, we address several critical issues related to preventing usable bikes from becoming unusable. First, develop a method to identify and prevent over usage. Second, identifying users who are responsible for breakdowns of bikes. Finally, developing strategies that the operator of a BSS can use, once the above two informations have been obtained.

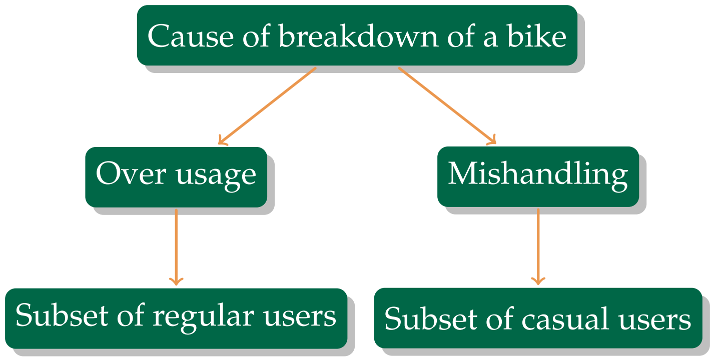


Figure 4.1: Causes of Damage of a Bike and Users Responsible for Them

There is lack of literature related to anomaly detection in bike sharing systems. To the best of our knowledge, only Delassus et al. (2016) and Kaspi et al. (2016) propose methods to detect unusable bikes in a station-based bike sharing systems. Delassus et al. (2016) use a method based on K-Means clustering to identify unusable bikes. Kaspi et al. (2016) on the other hand predicts the likelihood that a bike is unusable and the number of unusable bikes in a station, using a Bayesian model. In Kaspi et al. (2016), the authors also propose enhancements and extensions to their model, for approximating these probabilities in real time and detecting locker failures. In Kaspi et al. (2017), the authors quantify user dissatisfaction in the presence of unusable bikes as a function of the shortage of bikes and lockers in a solitary station. They propose a method to estimate this function and illustrate how it can be used in linear optimization models for operational and strategic decision making in bike sharing systems. However, none of these studies propose any data-driven method to detect over usage, identify users who are damaging bikes or how such an information can be used by the operator of the system.

In order to study over usage, we use a (quasi) Poisson regression model to model the relationship between the number of breakdowns of a bike to its total distance traveled, total duration of travel and total number of pickups. The task of identifying users who are responsible for damage is formulated as an unsupervised learning problem. We break this task down into three steps, comprising of simple intuitive rules, penalized Poisson regression and clustering. Finally, based on the above two outcomes, we provide strategies that the operator of a FFBS can use to minimize damage done to bikes in the system. In this study, we demonstrate the above mentioned methods using the Share-A-Bull BSS (SABB), an FFBS on the Tampa campus of the University of South Florida (USF). It is worth mentioning that our method is easy to implement and can be easily ported to other bike sharing systems without much changes.

The remainder of the chapter is organized as follows. Section 4.2 describes the develop a method to identify and prevent over usage. Section 4.3 describes the methodology for identifying users who are responsible for breakdowns of bikes. Section 4.4 presents strategies that the operator of a BSS can use, once the above two informations have been obtained. Finally, Section 4.5 concludes the chapter with directions for future research.

4.2 Identifying and Preventing Over Usage

There are three independent variables of interest, total distance traveled in miles (continuous variable), total duration traveled in minutes (continuous variable) and total number of trips (count variable). For simplicity in the rest of the chapter, we will refer to *total distance traveled in miles*, *total duration traveled in minutes* and *total number of trips* as *Var 1*, *Var 2* and *Var 3* respectively. Therefore, seven combination of these independent variables are possible (described in Table 4.1), each of which can be used to build a negative binomial regression model.

Table 4.1: Different Variable Sets for Estimating Negative Binomial Regression Models

Variable Set	Var 1	Var 2	Var 3
1	✓	✓	✓
2	✓	✓	
3	✓		✓
4		✓	✓
5	✓		
6		✓	
7			✓

The data used to test our proposed methods in this study, is the historical trip, data about the breakdown of bikes and other transaction data of Share-A-Bull Bikes (SABB) FFBS system at the University of South Florida, Tampa. Phase I of the program was launched in August 2015, providing 100 bikes to students, staff and faculty at no charge if the users limited their cumulative usage time to less than two hours per day. An hourly fee was imposed for the extra time beyond the daily two hour free quota. The program is expected to be integrated with parking management and other multi-modal transportation initiatives on the campus. USF researchers collaborated with the bike sharing company and developed the program in 2015. Given it is a program operated and managed internally, USF researchers had full access to the usage data, including trajectory data, of the program.

With built-in GPS and the application developed by Social Bicycles (SoBI), the trip data (trajectory of bikes) of each usage of the bikes is recorded in the operation management system. All trips have a unique ID. Further, each trip has a user ID, bike ID, starting timestamps, starting latitude, starting longitude, ending timestamps, ending latitude, ending longitude, trip duration (in minutes) and trip distance (in miles). The time frame of this study was from August 28, 2015, the launch date of the program to April 14, 2017. During this time frame, a total of $189,092$ trips were recorded. However, many of these trips were noise; hence, they had to be identified and subsequently removed before any further analysis could be conducted. Trips with the following properties were removed:

- if trip duration $\leq$ 30 seconds, in such case, the user might be checking the bike without using it.
- if trip distance $\leq$ .000621371 miles or 1 meter, in such case, the bike might be damaged after short usage and the user may not able to complete his/her trip.
- if the trip was conducted for testing the system.

After removing trips with the above mentioned properties, there was a total of $171,958$ trips. From this cleaned trip data, *Var 1*, *Var 2* and *Var 3* were computed for each bike in the system. At the time of collecting the data, the system had 99 operational bikes. The number of breakdowns for each of these bikes were also collected from the SABB FFBS management system.

For simplicity and ease of interpretability, we assume that the relationship between number of breakdowns of a bike with its total trip distance traveled, total trip duration traveled and total number of pickups is linear. Since, the number of breakdowns is a non-negative count variable, we use (quasi) Poisson regression to model the above relationship. To be more specific, negative binomial regression is used as the mean (240.39) of number of breakdowns is $<<$ than the variance (11649.91) of the number of breakdowns. For an in depth study on Poisson, quasi-Poisson or negative binomial regression models, refer to Washington et al. (2010).

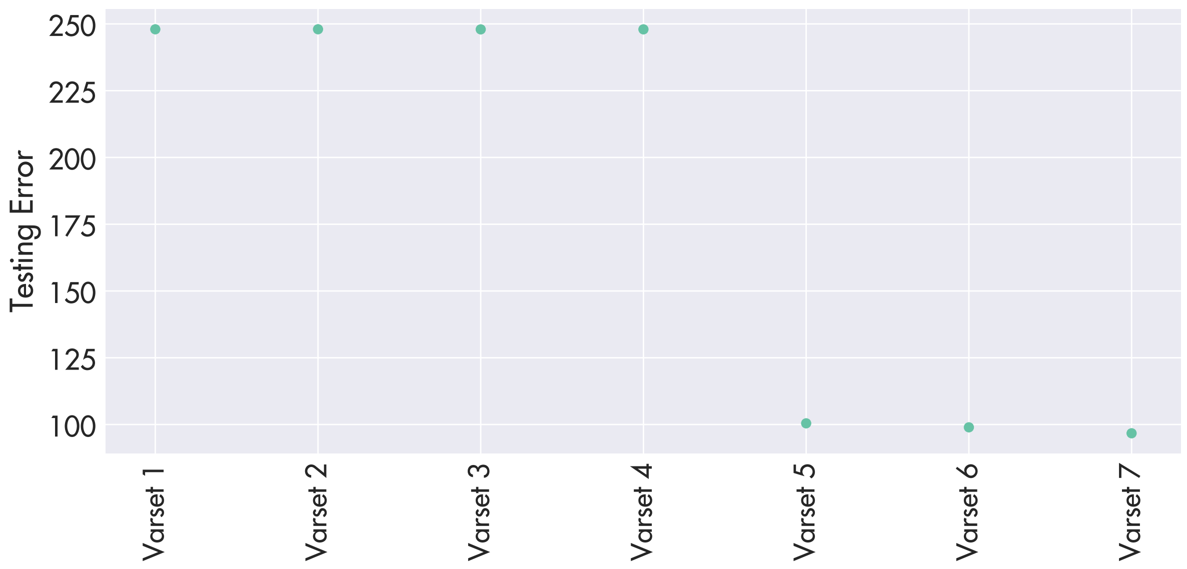


Figure 4.2: Testing Errors (RMSE)

Figures 4.2, 4.3 and 4.4 are the testing errors (RMSE), training errors (ρ^2) and inverse of average marginal effects of *Vars 1, 2 and 3* of the negative binomial regression models on the SABB FFBS dataset using different *Variable Sets (Varset)*. From Figure 4.2, it is evident that *Varsets 5, 6* and *7* perform significantly better than *Varsets 1, 2, 3* and *4*. One of the possible explanation can the high correlation between *Vars 1, 2* and *3*. However, instead of selecting any particular model among the top three models, we take advantage of all three of them.

Average marginal effect of an independent variable in a negative binomial regression model, is the amount of change in the dependent variable corresponding to 1 unit change of that particular independent variable. Hence, inverse of average marginal effect of an

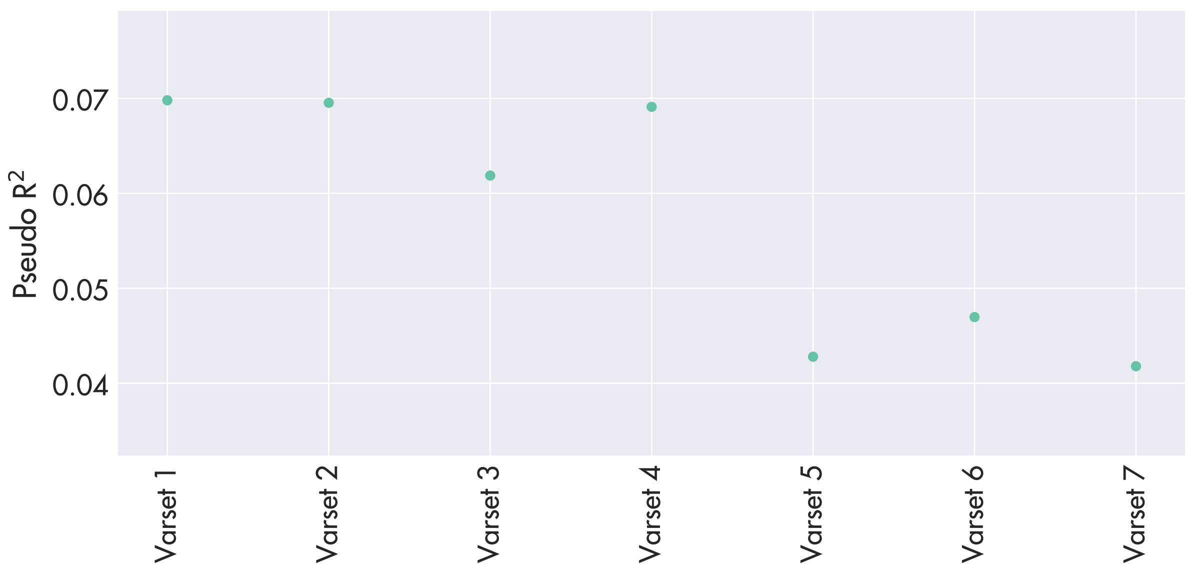


Figure 4.3: Training Errors (ρ^2)

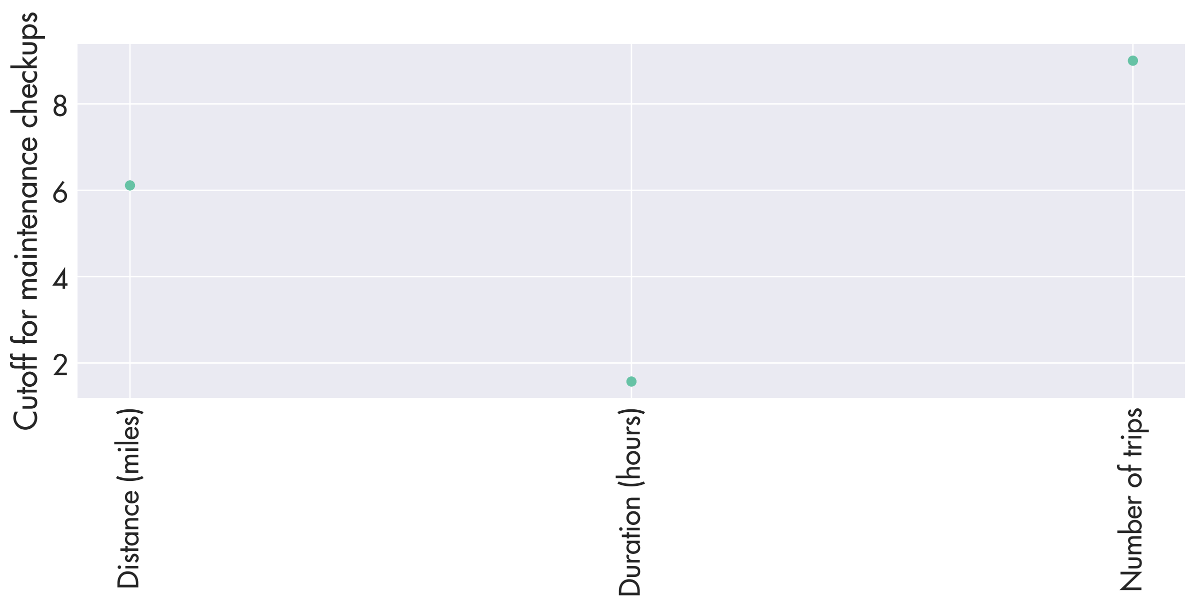


Figure 4.4: Inverse of Average Marginal Effects

independent variable, is the amount of change in that particular independent variable corresponding to 1 unit change of the dependent variable. This is very interesting, as inverse marginal effect of any the *Vars* with respect to the number of breakdowns gives us the amount of change necessary for 1 breakdown. In Figure 4.4, inverse of average marginal effects of *Vars 1, 2 and 3* are reported based on the respective negative binomial regression models using *Varsets 5, 6* and *7* respectively.

From Figure 4.4, it is evident that approximately 6 miles of distance traveled or 2 hours of usage or 9 pickups leads to 1 breakdown. Thus, a SABB bike riden for 6 or more miles, used for close to 2 hours and picked up 9 or more times, is prone to a breakdown. To prevent a potential breakdown, the system operator can schedule maintenance checks on a bike, if that particular bike is close to being riden for 6 miles, close to being used for 2 hours and close to being picked up 9 times since the last time it was reported to be broken. This will prevent damage to bikes owing to over usage. In a later section, we demonstrate how to recover the cost incurred due to maintenance checks from the subset of regular users responsible for damage.

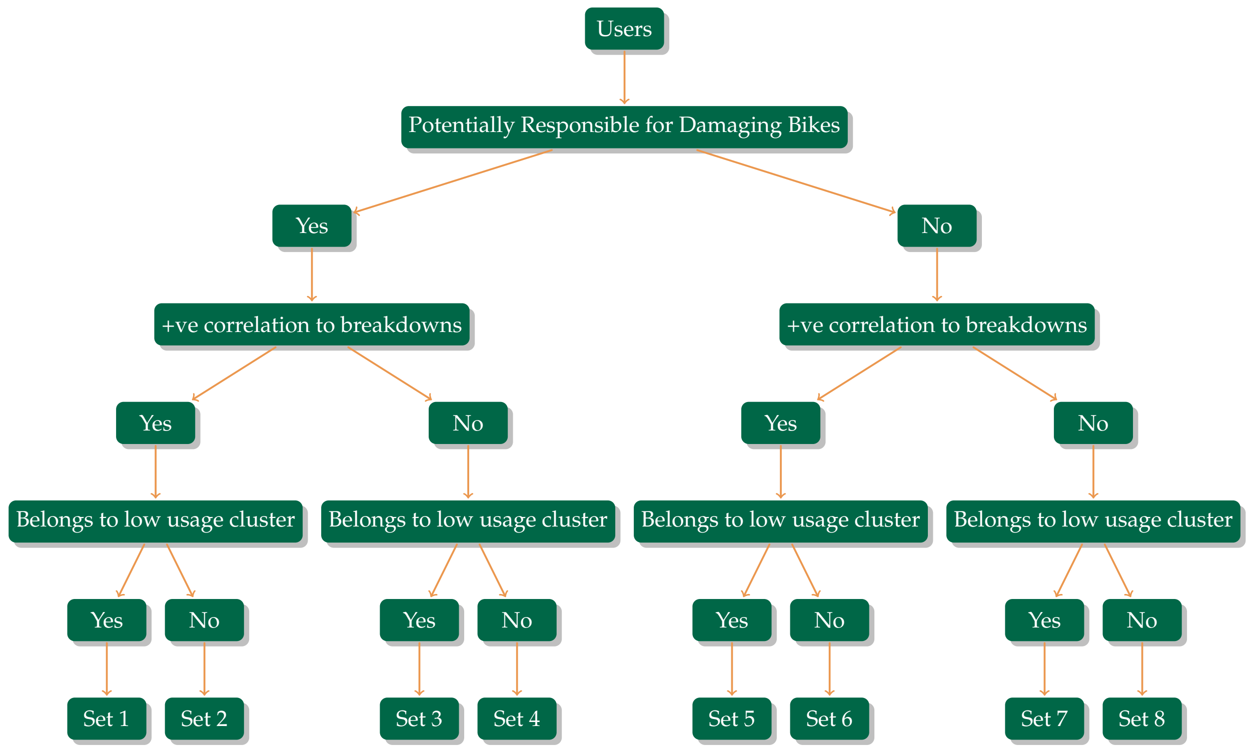
4.3 Identifying Users Responsible for Damaging Bikes

In this section, we propose a method to identify users responsible for damaging bikes in a FFBS. Our proposed method is simple and easy to implement. First, we decompose the set of all users into mutually exclusive subsets and then identify subsets of users responsible for breakdowns. Figure 4.5 outlines how the set of users is going to be decomposed into mutually exclusive subsets. The subsets of interests are:

1. Set 1: subset of casual users mishandling bikes

2. Sets 2 and 6: subset of regular users overusing bikes

The primary challenge in solving this problem is that this is an unsupervised learning problem, where no labels are available. On top of that we have a low sample ($n = 99$


Users
Potentially Responsible for Damaging Bikes
Yes
No
+ve correlation to breakdowns
+ve correlation to breakdowns
Yes
No
Yes
No
Belongs to low usage cluster
Belongs to low usage cluster
Belongs to low usage cluster
Belongs to low usage cluster
Yes
No
Yes
No
Yes
No
Yes
No
Set 1
Set 2
Set 3
Set 4
Set 5
Set 6
Set 7
Set 8
Figure 4.5: Decision Tree for Identifying Users Responsible for Damaging Bikes

bikes) high dimensional ($p \geq 6096$ users) dataset. To tackle this problem we propose a simple rule based method consisting of the following steps:

1. select a broad set of potential users who may be responsible for damage based on a metric.
2. resampling from the original dataset: number of resamples $= (\frac{100}{80} \times |\text{potential users}|) - n$
3. using regularization to select users with a +ve correlation to breakdowns of bikes
4. cluster users into two groups based on usage
5. compute the Set 1-8 by combining all the above information

This method has the following underlying assumptions:

1. A potential user mishandling bikes must have used a bike at least once. This is because, an user who is not the part of the FFBS and causes harm to the bike would be

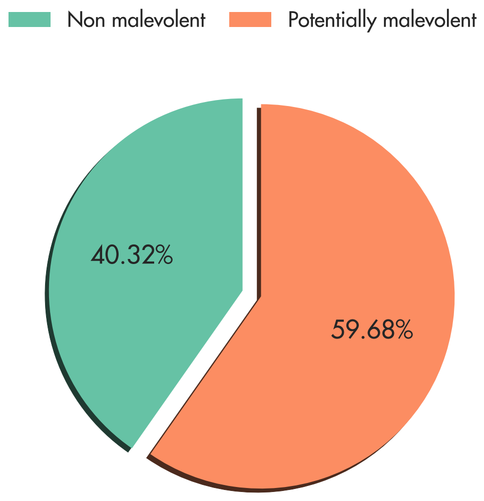


Figure 4.6: Potential Users who May Be Responsible for Damage

difficult to identify as there would be no trip data registered for any such analysis. The lack of data else wise, restricts our research to focus only on the registered users who have used the bikes.

2. The set of regular users and the set of users mishandling bikes are mutually exclusive because it is not in the interest of the regular users to mishandle bikes. We are thus assuming that regular users are behaving rationally.

4.3.1 Stage 1: Selecting a Broad Set of Potential Users who may be Responsible for Damage

In SABB FFBS, a positive charge imposed by the operator signifies that the user has violated certain rules and regulations of the SABB FFBS. Similarly, a negative charge imposed by the operator signifies that the user has acted in such a manner (parking at low supply high demand zones, etc) that improves the service level of SABB FFBS. Thus, any user with a positive charge is selected as a potential user who may be responsible for damage of bikes.

4.3.2 Stage 2: Selecting Users with a Positive Correlation to Breakdowns

In this stage, we select users whose usage have a statistically significant (positive) correlation with the breakdown of bikes. Thus, we are trying to select users whose usage increases the probability of breakdown of a bike. Like in earlier sections, the three independent variables of interest are total distance traveled in miles (continuous variable), total duration traveled in minutes (continuous variable) and total number of trips (count variable), also refered to as *Var 1*, *Var 2* and *Var 3*. The dependent variable in this case is the number of breakdowns of bikes. The three independent variables have seven possible combinations (described in Table 4.1) that can be used to build a statistical model. However, the one that performs the best outside of the training set will be selected as the final model. An user will be selected if the coefficient of any of its variables is the final model is positive. From the cleaned trip data obtained in the earlier section, we compute Var 1, Var 2, Var 3 for each user corresponding to each bike. The number of breakdowns for each bike are also provided.

The statistical model used for selecting the users is regularized Poisson regression. Two types of regularization, LASSO (Tibshirani (1996)) and ElasticNet have been considered. For details on LASSO, ElasticNet and other regularization strategies we refer the readers to James et al. (2013) and Friedman et al. (2009). We use the *glmnet* (Friedman et al., 2010) package in R to compute the regularization paths for both LASSO and ElasticNet for all the models. The parameters for both LASSO and ElasticNet are selected using cross validation.

The primary disadvantage of this method is that number of users selected $\leq$ number of observations in the training set (79), considering a 80-20 split of the training and testing dataset. To overcome this problem, we propose a simple re-sampling technique to generate requisite number of observations without modifying the population characteristics, like mean and variance.

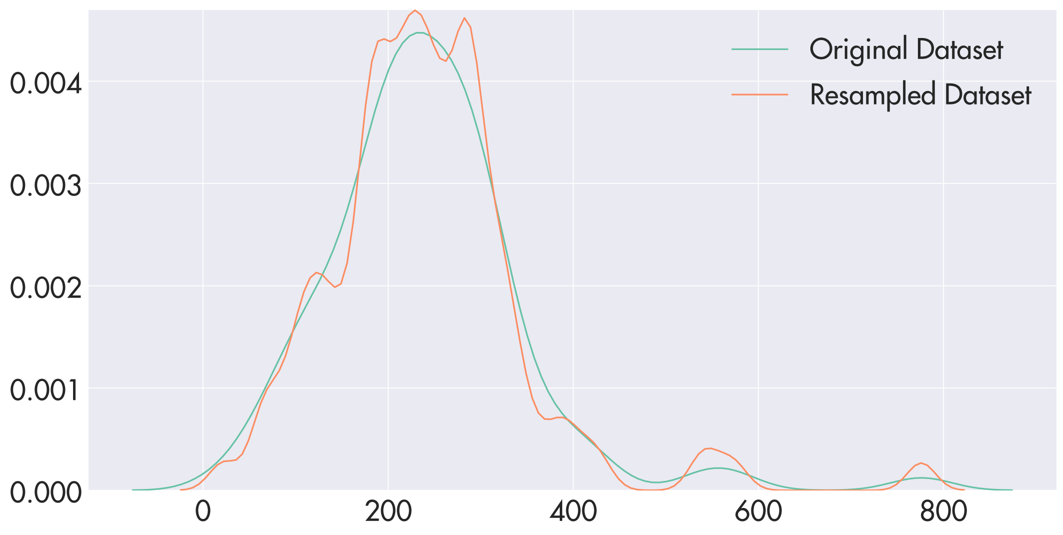


Figure 4.7: Distribution of the Number of Breakdowns in the Original and the Re-sampled Dataset

Figure 4.6 represents non malevolent and potentially malevolent users as a percentage of all users. The number of potentially malevolent users is . Thus, the number of samples that needs to be drawn with replacement from the original dataset is . The dataset that consists of the original dataset plus the additional samples drawn with replacement from the original dataset will be refered to as the re-sampled dataset. Figure 4.7 is the histogram of the number of breakdowns in the original and re-sampled dataset. From Figure 4.7, it is evident that the distribution of the number of breakdowns from the re-sampled dataset closely resembles that in the original dataset. Further, the mean and variance of the number of breakdowns in the original dataset are and and that in the re-sampled dataset are and respectively. Thus, the re-sampled dataset represents the original dataset extremely well.

As the testing errors is almost the same in all of the above models, all users are accumulated into a group. Interesting fact, if re-sampled dataset is not used no users are selected in any of these settings.

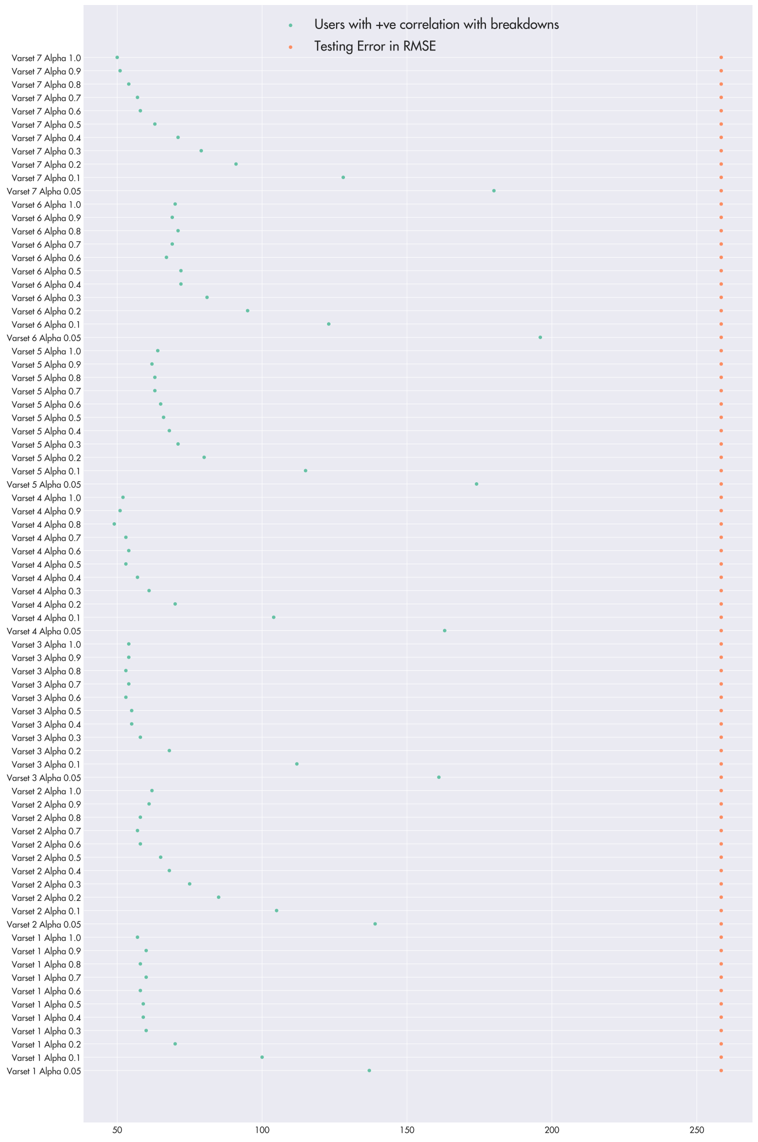


Figure 4.8: Number of Users with Positive Correlation to Breakdowns and Testing Errors for All Models

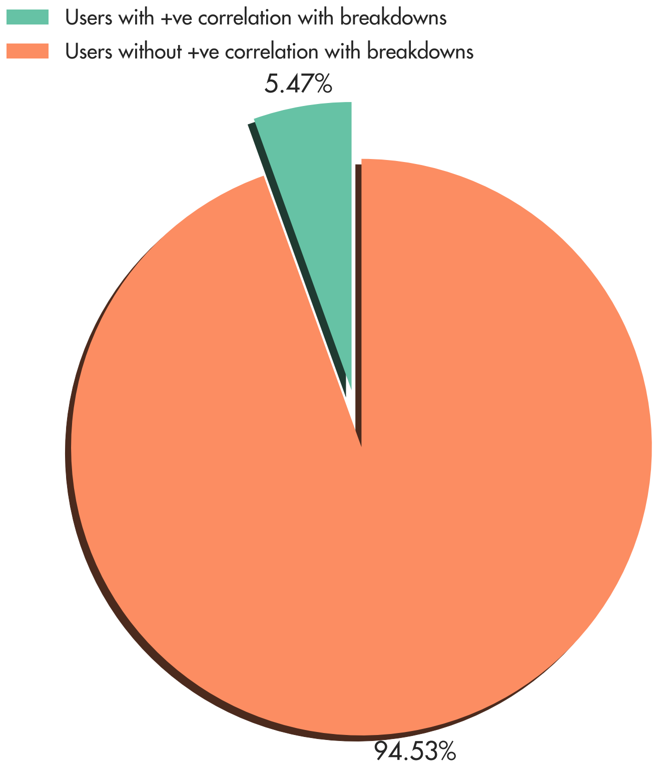


Figure 4.9: Users with and without Positive Correlation to Breakdowns

4.3.3 Stage 3: Clustering Users into Regular and Casual Users

In this stage, we cluster users into two groups, regular (high usage) and casual (low usage) users. There are three independent variables, *Var 1*, *Var 2* and *Var 3* which results in seven possible combinations of variables (Table 4.1). Therefore, distance between two users can be computed in seven possible ways for each of these seven combinations. For each of these seven combinations, first K-Means and then Hierarchical clustering is used to cluster users into two groups. Out of the two clusters, users belonging to the (high usage) cluster with a higher mean of *Var 1*, *Var 2* and *Var 3* will be refered to as the regular users. Whereas users belonging to the other (low usage) cluster will be refered to as the casual users. For details on K-means and hierarchical clustering we refer the readers to James et al. (2013) and Friedman et al. (2009). We use the *scikit-learn* (Pedregosa et al., 2011) package in Python to compute the clusters using K-means and agglomerative hierarchical clustering.

It is interesting to note that the objective function the clustering algorithms minimize is the sum of squared distances from each observation being clustered to its cluster center. However, our objective is to minimize the variance of *Var 1*, *Var 2* and *Var 3* in a cluster. Thus out of the fourteen clusters, we select the best cluster to be the one whose sum of variance of *Var 1*, *Var 2* and *Var 3* across both the clusters is the least. *Var 1*, *Var 2* and *Var 3* for each user is computed from the cleaned trip dataset of SABB FFBS.

4.4 Possible Strategies for Minimizing Damage done to Bikes

Previously, we provided certain metrics based on distance traveled, duration traveled and number of pickups that helped in identifying whether a bike requires a maintenance checkup before it can be used further. It is also important to note that this model can also be modified to predict the number of breakdowns. Now, that we have identified users who are either over using or mishandling bikes, customized strategies can be developed

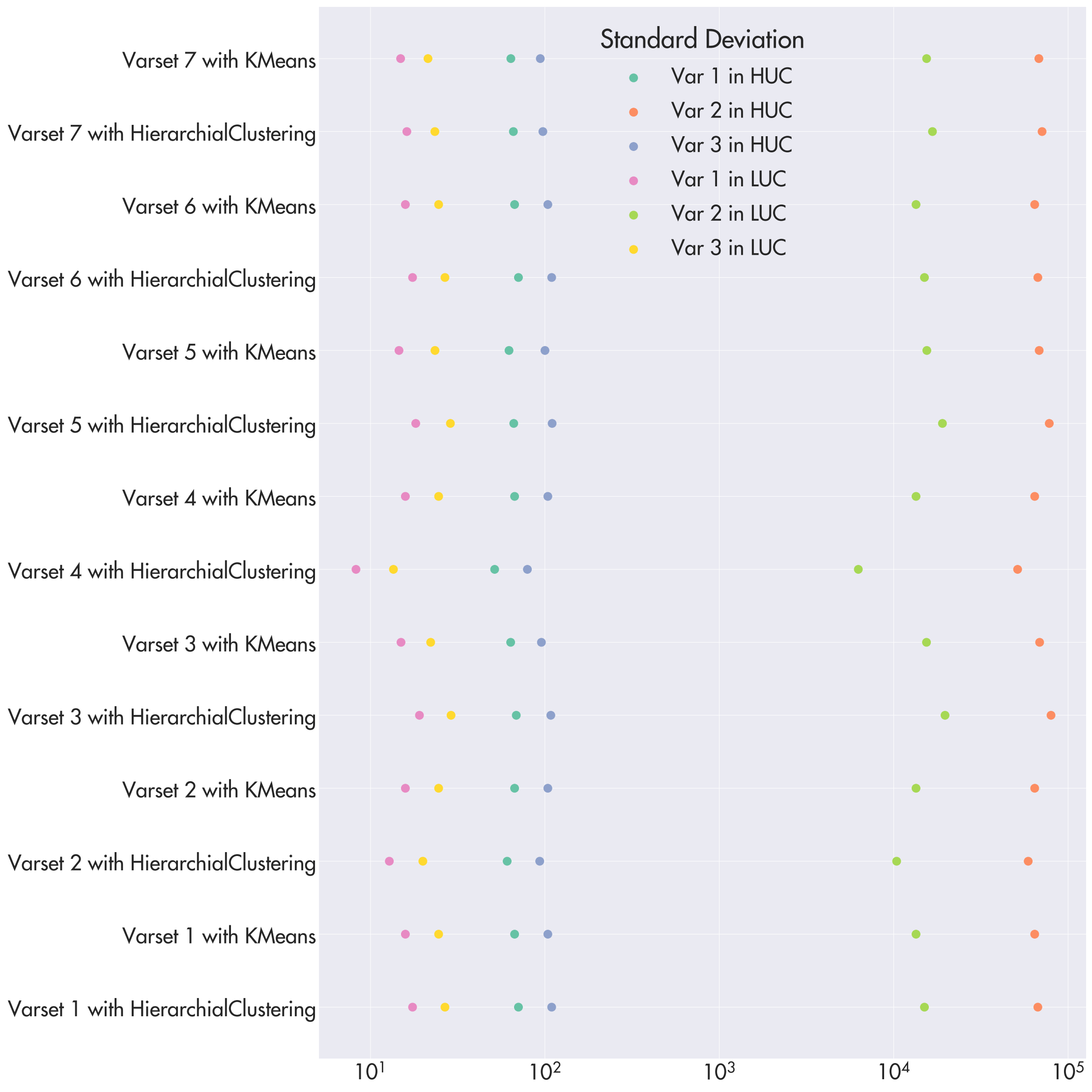


Figure 4.10: Intra Cluster Standard Deviation of the 3 Variables for Different Varsets and Clustering Method

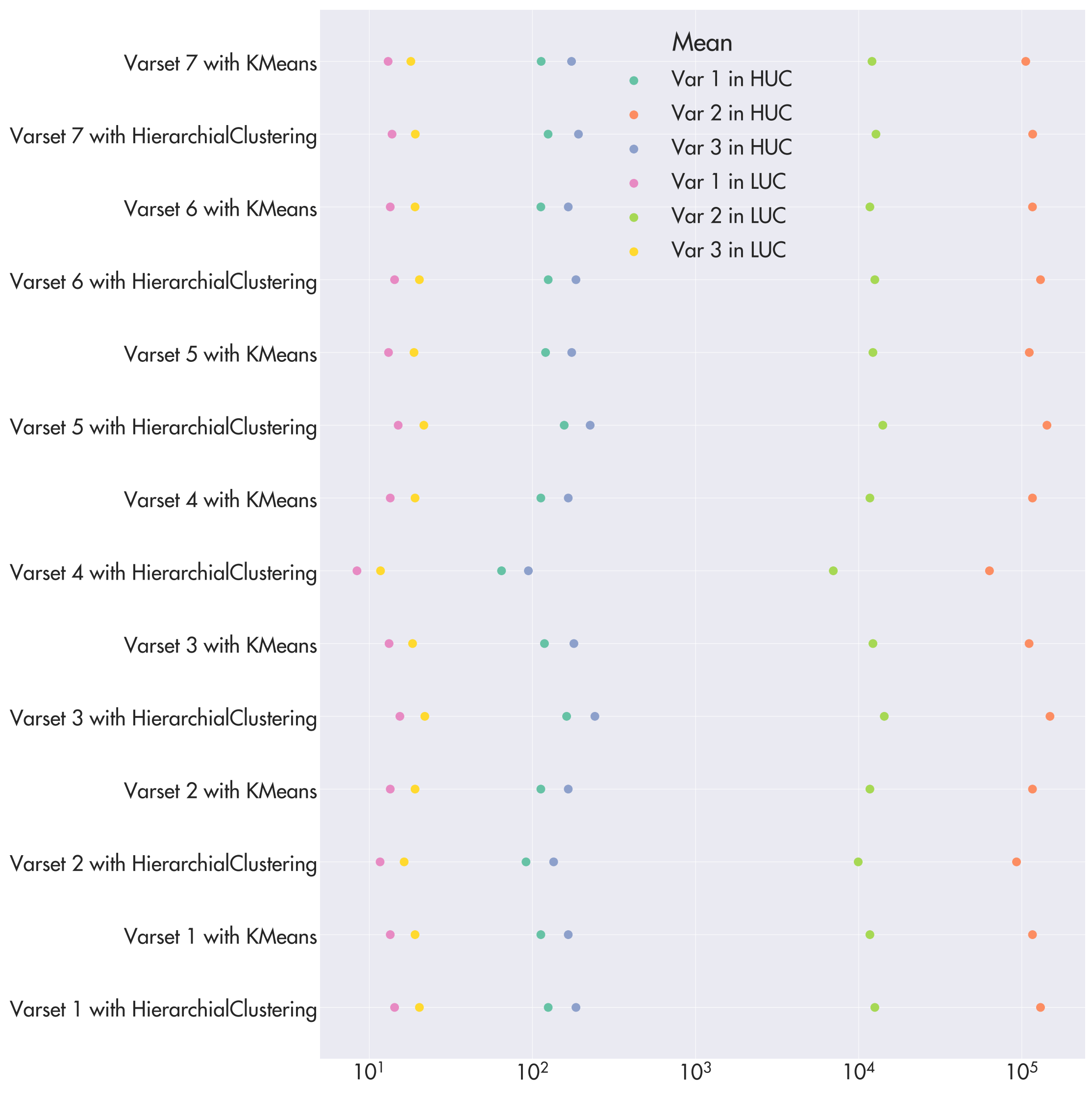


Figure 4.11: Intra Cluster Mean of the 3 Variables for Different Varsets and Clustering Method

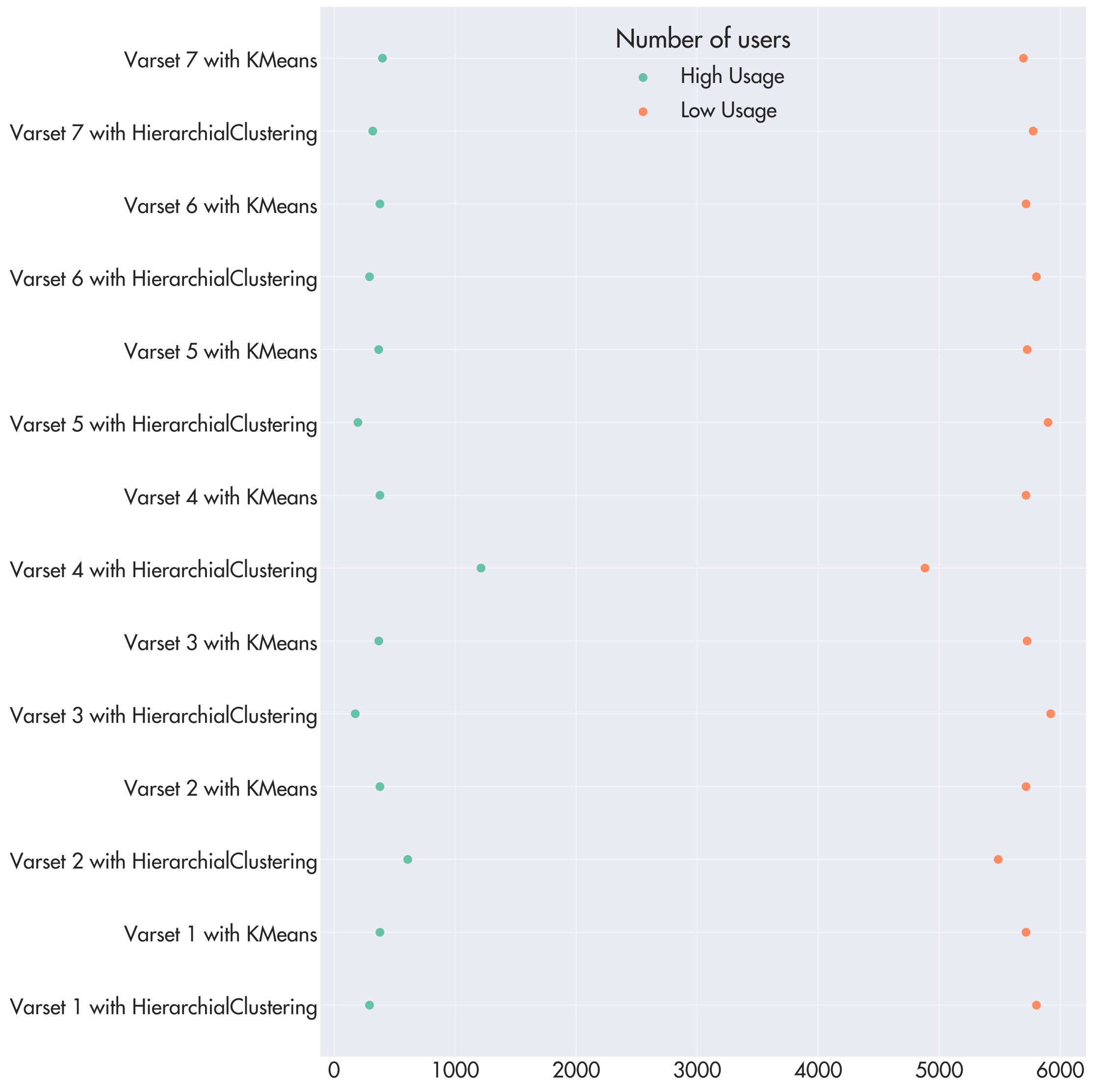


Figure 4.12: Intra Cluster Number of Users for Different Varsets and Clustering Method

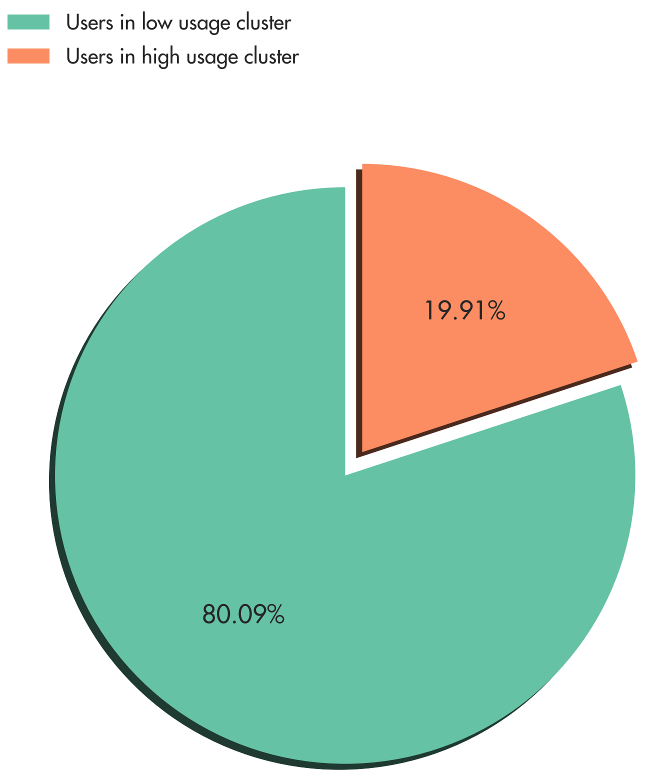


Figure 4.13: Number of Users for Selected Cluster

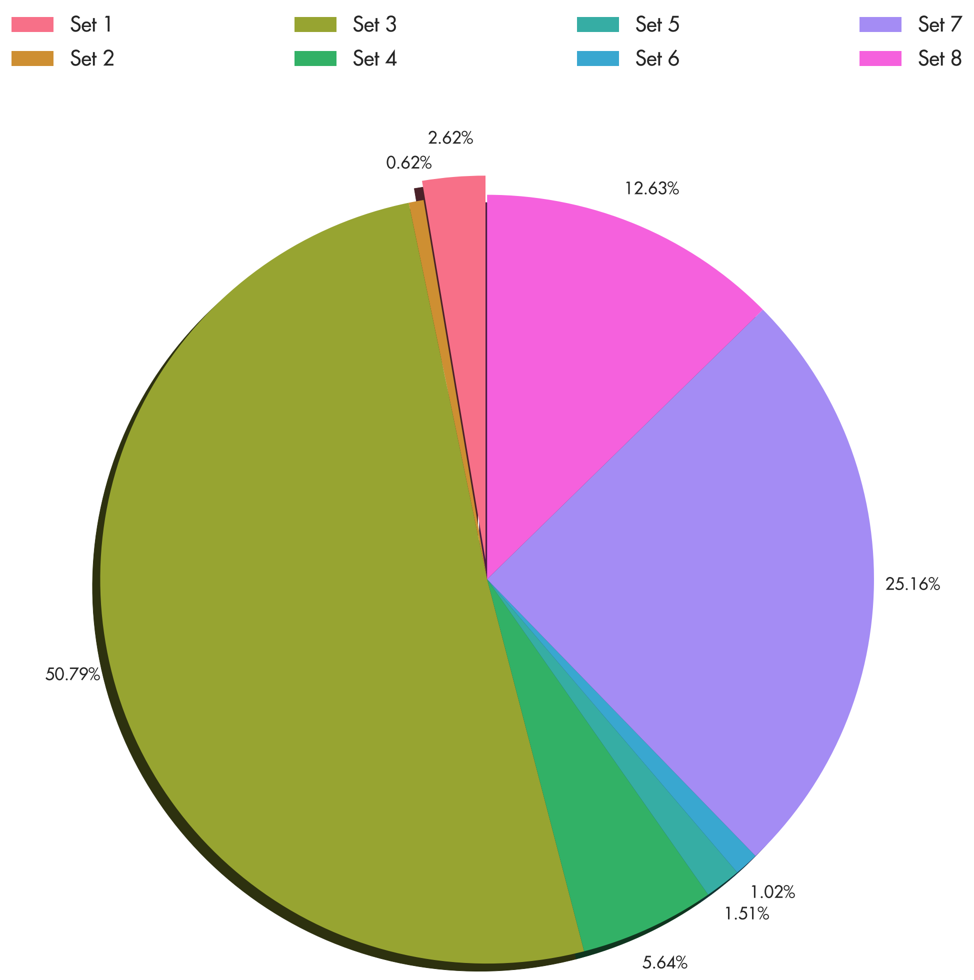


Figure 4.14: Decomposition of the Set of All Users into Mutually Exclusive Subsets

to deal with them. There are two broad strategies that the operator can use. First, prevent these identified users from using the system or imposing a penalty on their service charge. The first strategy is not applicable for the subset of regular users who are overusing the bikes as they are a major source of revenue. So, the only viable strategy is to penalize them inorder to recover the maintenance costs associated with over usage. However, for the malevolent users the operator can use either of the two strategies or a combination of both.

4.5 Final Remarks

In this chapter, we address several critical issues related to preventing usable bikes from becoming unusable. First, develop a method to identify and prevent over usage. Second, identifying users who are responsible for breakdowns of bikes. Finally, developing strategies that the operator of a BSS can use, once the above two informations have been obtained. In order to study over usage, we used a (quasi) Poisson regression model to model the relationship between the number of breakdowns of a bike to its total distance traveled, total duration of travel and total number of pickups. The task of identifying users who are responsible for damage was formulated as an unsupervised learning problem and broken into three steps, comprising of simple intuitive rules, penalized Poisson regression and clustering. Finally, based on the above two outcomes, we provided strategies that the operator of a FFBS can use to minimize damage done to bikes in the system. We also demonstrated the above mentioned methods using the Share-A-Bull BSS (SABB), an FFBS on the Tampa campus of the University of South Florida (USF). It is worth mentioning that our method is easy to implement and can be easily ported to other bike sharing systems without much changes.

5 Conclusion

In this dissertation, we propose, implement and test three methods using tools from operations research, statistical and machine learning to improve service level and decrease operating costs of free-floating bike sharing systems. In chapter 2, we propose a novel MILP for formulating SCRP in FFBS and SBBS based on spacial decomposition. The proposed formulation, can not only handle single and multiple vehicles, but also allows for multiple visits to a node by the same vehicle. However, the proposed formulation is computationally intractable even for small scale instances owing to the presence of Big M, used for subtour elimination in the constraints. It makes the linear programming relaxation of the formulation extremely weak. Another reason for the computational intractability of the formulation is the significant increase in the number of decision variables owing to spacial decomposition.

A hybrid nested large neighborhood search with variable neighborhood descent algorithm (NLNS+VND) for solving SCRP both effectively and efficiently for FFBS and SBBS is also presented. Computational experiments on 1-PDTSP instances, previously used the literature, demonstrate that NLNS+VND outperforms tabu search and is highly competitive with exact algorithms reported in the literature. A future research direction can be strengthening the linear programming relaxation of our proposed formulation by extending valid inequalities proposed in the literature for m-TSP, 1-PDTSP and Q-TSP to our proposed formulation. To deal with increase in the number of variables, strategies based on column generation can also be explored. Other interesting strategies can be using the high quality solution provided by NLNS+VND to warm start MIP solvers once some mechanism for strengthening the formulation has been implemented.

In chapter 3, we propose a method to extract operational management insights from historical trip data of a shared mobility system, to help the operator make more informed decisions. Significant amount of research has been conducted on gaining various forms and types of insights with a broad range of motivation, from the historical data of the system. However, none of these studies considered interaction between independent variables or study imbalance as a dependent variable. In this dissertation, we take interactions among independent variables into consideration and apply methods to remove unnecessary interactions. We also show that more insights about the mobility patterns and imbalance of the SABB program can be obtained by considering such interactions. We also propose a simple method to decompose continuous variables into binary variables which improves the base model used in the literature. Our proposed methodology gives an unique opportunity to study the system and make recommendations to the operator from various vantage points. To extend our proposed method for station-based systems, dropoffs can also be considered in conjunction to pickups.

Another future research direction can be, how to use information from such an analysis to compute optimal inventory levels, which can then be used by the operator as inputs to their specific rebalancing strategies. Another possible research direction can conducted this analysis for each station in case of station based bike sharing systems or each zone in case of free floating bike sharing systems

References

(2015). mfx. https://cran.r-project.org/web/packages/mfx/mfx.pdf.

(2015). pscl. http://pscl.stanford.edu/.

(2017). BusinessDays.jl. https://github.com/felipenoris/BusinessDays.jl.

(2017a). Dark Sky Api. https://darksky.net/dev/docs/forecast.

(2017b). Dark Sky Data Sources. https://darksky.net/dev/docs/sources.

(2017). Julia Stdlib - Dates and Time. https://docs.julialang.org/en/stable/stdlib/dates/#stdlib-dates-1.

Ahuja, R. K., Özlem Ergun, Orlin, J. B., and Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(13):75 – 102.

Alvarez-Valdes, R., Belenguer, J. M., Benavent, E., Bermudez, J. D., Muñoz, F., Vercher, E., and Verdejo, F. (2016). Optimizing the level of service quality of a bike-sharing system. *Omega*, 62:163 – 175.

Anily, S. and Bramel, J. (1999). Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Research Logistics (NRL)*, 46(6):654–670.

Applegate, D., Cook, W., and Rohe, A. (2003). Chained lin-kernighan for large traveling salesman problems. *INFORMS J. on Computing*, 15(1):82–92.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219.

Bezanson, J., Karpinski, S., Shah, V. B., and Edelman, A. (2012). Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*.

Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-B., and Fleury, E. (2011). Shared bicycles in a city: A signal processing and data analysis perspective. *Advances in Complex Systems*, 14(03):415–438.

Boyacı, B., Zografos, K. G., and Geroliminis, N. (2015). An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research*, 240(3):718–733.

Caulfield, B., O'Mahony, M., Brazil, W., and Weldon, P. (2017). Examining usage patterns of a bike-sharing scheme in a medium sized city. *Transportation Research Part A: Policy and Practice*, 100:152 – 161.

Chalasani, P. and Motwani, R. (1999). Approximating capacitated routing and delivery problems. *SIAM Journal on Computing*, 28(6):2133–2149.

Chemla, D., Meunier, F., and Calvo, R. W. (2013a). Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120 – 146.

Chemla, D., Meunier, F., Pradeau, T., and Calvo, R. W. (2013b). Self-service bike sharing systems: Simulation, repositioning, pricing.

Cheu, R., Xu, J., Kek, A., Lim, W., and Chen, W. (2006). Forecasting shared-use vehicle trips with neural networks and support vector machines. *Transportation Research Record: Journal of the Transportation Research Board*, (1968):40–46.

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

de Chardon, C. M. and Caruso, G. (2015). Estimating bike-share trips using station level data. *Transportation Research Part B: Methodological*, 78:260 – 279.

de Chardon, C. M., Caruso, G., and Thomas, I. (2017). Bicycle sharing system success determinants. *Transportation Research Part A: Policy and Practice*, 100:202 – 214.

Delassus, R., Giot, R., Cherrier, R., Barbieri, G., and Mélançony, G. (2016). Broken bikes detection using citibike bikeshare system open data. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7.

Dell'Amico, M., Hadjicostantinou, E., Iori, M., and Novellani, S. (2014). The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7 – 19.

DellAmico, M., Iori, M., Novellani, S., and Stützle, T. (2016). A destroy and repair algorithm for the bike sharing rebalancing problem. *Computers & Operations Research*, 71:149 – 162.

DeMaio, P. (2009). Bike-sharing: History, impacts, models of provision, and future. *Journal of Public Transportation*, 12(4):3.

Erdoan, G., Battarra, M., and Calvo, R. W. (2015). An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research*, 245(3):667 – 679.

Erdoan, G., Laporte, G., and Calvo, R. W. (2014). The static bicycle relocation problem with demand intervals. *European Journal of Operational Research*, 238(2):451 – 457.

Faghih-Imani, A., Anowar, S., Miller, E. J., and Eluru, N. (2017a). Hail a cab or ride a bike? a travel time comparison of taxi and bicycle-sharing systems in new york city. *Transportation Research Part A: Policy and Practice*, 101:11 – 21.

Faghih-Imani, A. and Eluru, N. (2016). Incorporating the impact of spatio-temporal interactions on bicycle sharing system demand: A case study of new york citibike system. *Journal of Transport Geography*, 54:218 – 227.

Faghih-Imani, A., Eluru, N., El-Geneidy, A. M., Rabbat, M., and Haq, U. (2014). How land-use and urban form impact bicycle flows: evidence from the bicycle-sharing system (bixi) in montreal. *Journal of Transport Geography*, 41:306 – 314.

Faghih-Imani, A., Hampshire, R., Marla, L., and Eluru, N. (2017b). An empirical analysis of bike sharing usage and rebalancing: Evidence from barcelona and seville. *Transportation Research Part A: Policy and Practice*, 97:177 – 191.

Fishman, E., Washington, S., Haworth, N., and Watson, A. (2015). Factors influencing bike share membership: An analysis of melbourne and brisbane. *Transportation Research Part A: Policy and Practice*, 71:17 – 30.

Forma, I. A., Raviv, T., and Tzur, M. (2015). A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological*, 71:230 – 247.

Friedman, J., Hastie, T., and Tibshirani, R. (2009). *The Elements of Statistical Learning*, volume 1. Springer.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Gebhart, K. and Noland, R. B. (2014). The impact of weather conditions on bikeshare trips in washington, dc. *Transportation*, 41(6):1205–1225.

Ghilas, V., Demir, E., and Woensel, T. V. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, 72:12 – 30.

Helsgaun, K. (2000). An effective implementation of the linkernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106 – 130.

Helsgaun, K. (2009). General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, 1(2):119–163.

Hernández-Pérez, H. and Salazar-González, J.-J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. *Discrete Applied Mathematics*, 145(1):126 – 139. Graph Optimization {IV}.

Ho, S. C. and Szeto, W. (2014). Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, 69:180 – 198.

Ho, S. C. and Szeto, W. (2017). A hybrid large neighborhood search for the static multi-vehicle bike-repositioning problem. *Transportation Research Part B: Methodological*, 95:340 – 363.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*, volume 112. Springer.

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

Kaspi, M., Raviv, T., and Tzur, M. (2016). Detection of unusable bicycles in bike-sharing systems. *Omega*, 65:10 – 16.

Kaspi, M., Raviv, T., and Tzur, M. (2017). Bike-sharing systems: User dissatisfaction in the presence of unusable bicycles. *IISE Transactions*, 49(2):144–158.

Kloimüllner, C., Papazek, P., Hu, B., and Raidl, G. R. (2014). Balancing bicycle sharing systems: An approach for the dynamic case. In Blum, C. and Ochoa, G., editors, *Evolutionary Computation in Combinatorial Optimisation: 14th European Conference, EvoCOP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers*, pages 73–84. Springer Berlin Heidelberg, Berlin, Heidelberg.

Laporte, G., Meunier, F., and Wolfler Calvo, R. (2015). Shared mobility systems. *4OR*, 13(4):341–360.

Laporte, G. and Nobert, Y. (1980). A cutting planes algorithm for the m-salesmen problem. *Journal of the Operational Research Society*, 31(11):1017–1023.

OBrien, O., Cheshire, J., and Batty, M. (2014). Mining bicycle sharing data for generating insights into sustainable transport systems. *Journal of Transport Geography*, 34:262 – 273.

Pal, A. and Zhang, Y. (2017). Free-floating bike sharing: Solving real-life large-scale static rebalancing problems. *Transportation Research Part C: Emerging Technologies*, 80:92 – 116.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pfrommer, J., Warrington, J., Schildbach, G., and Morari, M. (2014). Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1567–1578.

Rainer-Harbach, M., Papazek, P., Raidl, G. R., Hu, B., and Kloimüllner, C. (2015). Pilot, grasp, and vns approaches for the static balancing of bicycle sharing systems. *Journal of Global Optimization*, 63(3):597–629.

Raviv, T., Tzur, M., and Forma, I. A. (2013). Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229.

Regue, R. and Recker, W. (2014). Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:192 – 209.

Reiss, S. and Bogenberger, K. (2015). Gps-data analysis of munich's free-floating bike sharing system and application of an operator-based relocation strategy. In *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, ITSC '15, pages 584–589, Washington, DC, USA. IEEE Computer Society.

Schuijbroek, J., Hampshire, R., and van Hoeve, W.-J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992 – 1004.

Singla, A., Santoni, M., Bartók, G., Mukerji, P., Meenen, M., and Krause, A. (2015). Incentivizing users for balancing bike sharing systems. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 723–729. AAAI Press.

Szeto, W., Liu, Y., and Ho, S. C. (2016). Chemical reaction optimization for solving a static bike repositioning problem. *Transportation Research Part D: Transport and Environment*, 47:104 – 135.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. on Computing*, 15(4):333–346.

Wagner, S., Brandt, T., and Neumann, D. (2016). In free float: Developing business analytics support for carsharing providers. *Omega*, 59:4 – 14. Business Analytics.

Washington, S. P., Karlaftis, M. G., and Mannering, F. (2010). *Statistical and econometric methods for transportation data analysis*. CRC press.

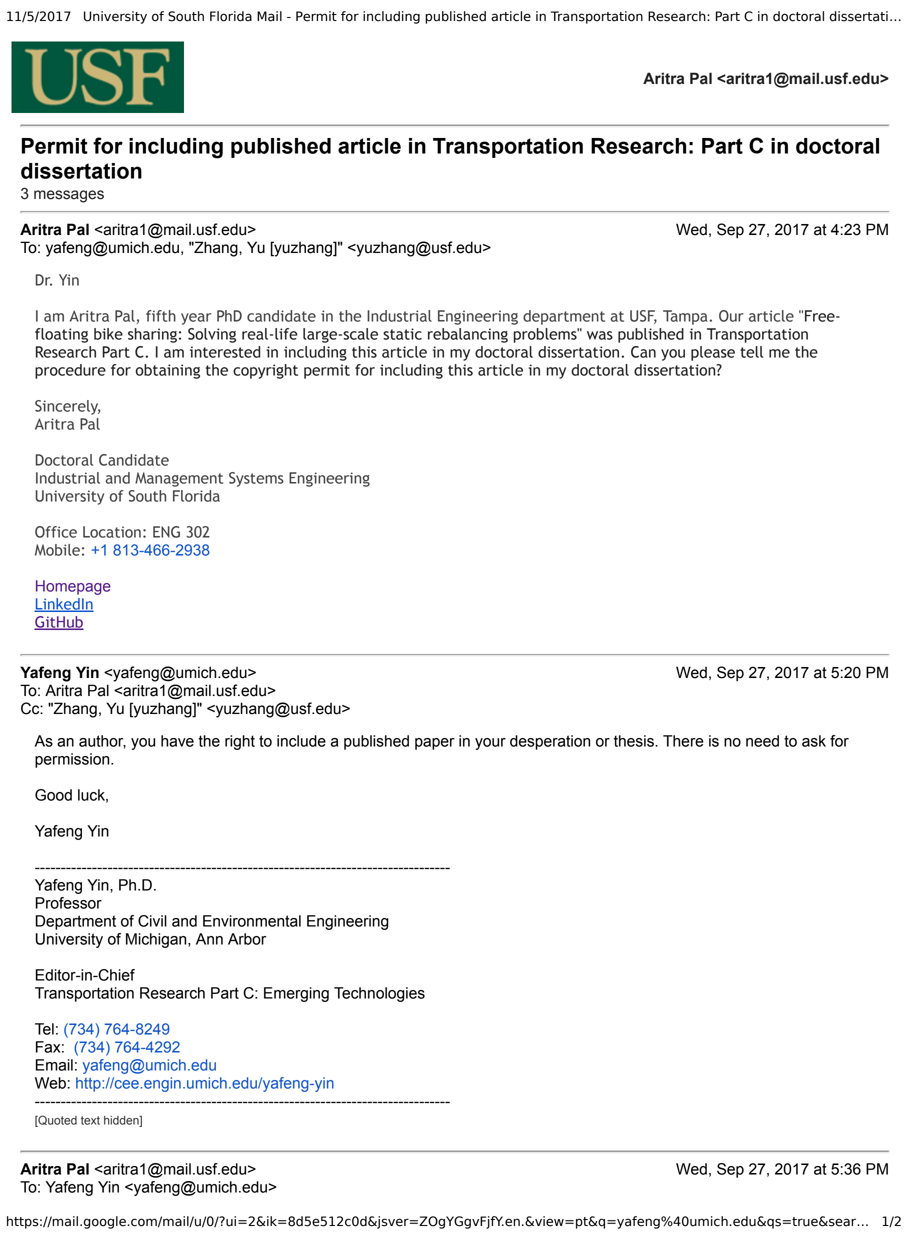
Weikl, S. and Bogenberger, K. (2013). Relocation strategies and algorithms for free-floating car sharing systems. *IEEE Intelligent Transportation Systems Magazine*, 5(4):100–111.

Zhang, Y., Thomas, T., Brussel, M. J. G., and van Maarseveen, M. F. A. M. (2016). Expanding bicycle-sharing systems: Lessons learnt from an analysis of usage. *PLOS ONE*, 11(12):1–25.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Appendix A Copyright Permissions

A.1 Reprint Permissions for Chapter 2



11/5/2017 University of South Florida Mail - Permit for including published article in Transportation Research: Part C in doctoral dissertati...

USF

Aritra Pal <aritra1@mail.usf.edu>

Permit for including published article in Transportation Research: Part C in doctoral dissertation
3 messages

Aritra Pal <aritra1@mail.usf.edu> Wed, Sep 27, 2017 at 4:23 PM
To: yafeng@umich.edu, "Zhang, Yu [yuzhang]" <yuzhang@usf.edu>

Dr. Yin

I am Aritra Pal, fifth year PhD candidate in the Industrial Engineering department at USF, Tampa. Our article "Free-floating bike sharing: Solving real-life large-scale static rebalancing problems" was published in Transportation Research Part C. I am interested in including this article in my doctoral dissertation. Can you please tell me the procedure for obtaining the copyright permit for including this article in my doctoral dissertation?

Sincerely,
Aritra Pal

Doctoral Candidate
Industrial and Management Systems Engineering
University of South Florida

Office Location: ENG 302
Mobile: +1 813-466-2938

Homepage
LinkedIn
GitHub

Yafeng Yin <yafeng@umich.edu> Wed, Sep 27, 2017 at 5:20 PM
To: Aritra Pal <aritra1@mail.usf.edu>
Cc: "Zhang, Yu [yuzhang]" <yuzhang@usf.edu>

As an author, you have the right to include a published paper in your desperation or thesis. There is no need to ask for permission.

Good luck,

Yafeng Yin

Yafeng Yin, Ph.D.
Professor
Department of Civil and Environmental Engineering
University of Michigan, Ann Arbor

Editor-in-Chief
Transportation Research Part C: Emerging Technologies

Tel: (734) 764-8249
Fax: (734) 764-4292
Email: yafeng@umich.edu
Web: http://cee.engin.umich.edu/yafeng-yin

[Quoted text hidden]

Aritra Pal <aritra1@mail.usf.edu> Wed, Sep 27, 2017 at 5:36 PM
To: Yafeng Yin <yafeng@umich.edu>

https://mail.google.com/mail/u/0/?ui=2&ik=8d5e512c0d&jsver=ZOgYGgvFjfY.en.&view=pt&q=yafeng%40umich.edu&qs=true&sear... 1/2