Resource Scheduling and Optimization in Dynamic and Complex Transportation Settings

by

Konstantina Mellou

Dipl., National Technical University of Athens (2014)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

© Massachusetts Institute of Technology 2019. All rights reserved.
Resource Scheduling and Optimization in Dynamic and Complex Transportation Settings

by

Konstantina Mellou

Submitted to the Sloan School of Management on May 17, 2019, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Operations Research

Abstract

Resource optimization has always been a challenge both in traditional fields, such as logistics, and particularly so in most emerging systems in the sharing economy. These systems are by definition founded on the sharing of resources among users, which naturally creates many coordination needs as well as challenges to ensure enough resource supply to cover customer demand. This thesis addresses these challenges in the application of vehicle sharing systems, as well as in the context of multi-operation companies that provide a wide range of services to their users.

More specifically, the first part of this thesis focuses on models and algorithms for the optimization of bike sharing systems. Shortage of bikes and docks is a common issue in bike sharing systems, and, to tackle this problem, operators use a fleet of vehicles to redistribute bikes across the network. We study multiple aspects of these operations, and develop models that can capture all user trips that are performed successfully in the system, as well as algorithms that generate complete redistribution plans for the operators to maximize the served demand, in running times that are fast enough to allow real-time information to be taken into account. Furthermore, we propose an approach for the estimation of the actual user demand which takes into account both the lost demand (users that left the system due to lack of bikes or docks) and shifted demand (users that had to walk to nearby stations to find available resources). More accurate demand representations can then be used to inform better decisions for the daily operations, as well as the long-term planning of the system.

The second part of this thesis is focused on schedule generation for resources of large companies that must support a complex set of operations. Different operation types come with a variety of constraints and requirements that need to be taken into account. Moreover, specialized employees with a variety of skills and experience levels are required, along with an heterogeneous fleet of vehicles with various properties (e.g., refrigerator vehicles). We introduce the Complex Event Scheduling Problem (CESP), which captures known problems such as pickup-and-delivery and technician scheduling as special cases. We then develop a unified optimization framework for CESP, which relies on a combination of metaheuristics (ALNS) and Linear Program-
ming. Our experiments show that our framework scales to large problem instances, and may help companies and organizations improve operation efficiency (e.g., reduce fleet size).

Thesis Supervisor: Patrick Jaillet
Title: Dugald C. Jackson Professor, Department of Electrical Engineering and Computer Science
Acknowledgments

Many people have contributed either directly or indirectly to the completion of this thesis and to making my journey the last five years an unforgettable experience. First and foremost, I would like to thank my advisor, Patrick Jaillet, for his amazing support and understanding, which began as early as the visit days through several discussions that turned out pivotal in my decision to pursue a doctoral degree in Operations Research. Throughout the years that followed, he introduced me to research problems that made me excited about research, and this thesis would not have been possible without his guidance. I learnt a lot from him and I am grateful that he gave me the opportunity to pursue my interests both within the program, as well as through internships and collaborations.

I would also like to thank the members of my thesis committee, Dimitris Bertsimas and Juan Pablo Vielma. Dimitris has been an inspiration for me as a researcher, and has had an invaluable contribution into making the ORC the unique and vibrant community that it is today. Juan Pablo has been an exemplary academic; during our first interactions through his classes, he helped me set technical foundations that influenced much of my future work, and I appreciate his advice and help ever since.

Next, I would like to thank my collaborator and mentor, Ishai Menache, who also served as part of my thesis committee. Ishai gave me the opportunity to work on projects with great impact, and I would like to thank him for believing in me since the beginning. The last part of this thesis would not have been possible without him, as well as my collaborators, Krishna Chintalapudi and Luke Marshall, who made working on this project an exciting experience. I would also like to thank Mohit Singh for our collaboration and his help in the early years of my Ph.D.

I would also like to extend my gratitude to my undergraduate advisor, Dimitris Fotakis, who was my first introduction to the research world. He inspired me to pursue a doctoral degree and grow as a researcher and a person. Many thanks go to Laura Rose, Andrew Carvalho and Rachel Cohen for their administrative assistance throughout this program; life became definitely easier thanks to them!
The past five years would not have been the same without my friends in and out of MIT. All ORC friends played a great role in creating a supportive and fun environment, and I would like to especially mention Mathieu, Ilias, Amine, Colin, Carl, Rim, and Yee Sian. From courses and qualifying exam support to movie nights at SidPac and numerous trips in the U.S. and beyond, their friendship has been a gift especially during the first year of moving to a new country. I would also like to thank Seb, Max, Maximilien, Julia, Agni, Arthur, Chong Yang, Virgile, Arthur, Nikita, Dawsen, Swati, and many others, for so many memories over all these years.

For offering me a home away from home, I would like to thank Chara, Lydia, Marinos, Themis, Katerina, Manolis, Konstantinos, Afroditi, Vasso, Christos, Kyriakos, Dimitris, Giorgos, Vasilis, and a special shout-out to Sophie for tolerating me as a roommate the past few years. I would also like to thank everyone at the Hellenic Students’ Association for all the fun experiences and events we organized adding a special note at the life at MIT. Although seven time-zones away, I am also grateful to all my friends back home, Alexandros, Marianna, Daniil, Giorgos, and Ioanna, who never stopped showing their support and keeping me a part of their life.

Finally, none of this would have been possible without my family: my parents, my sister, and my grandparents. Their encouragement and support since the early years of my life allowed me to grow as a person, discover my interests and pursue them fearlessly, while during my Ph.D. they were always there for me every step along the way, patiently supporting me to overcome all difficulties and obstacles. This thesis is dedicated to them.

This work has been supported by the the National Research Foundation of Singapore Grant No. 015824-00078.
Contents

1 Introduction .................................................. 15
  1.1 Resource Optimization in Bike Sharing Systems ............... 17
  1.2 Resource Management for Complex Event Scheduling .......... 19

I Resource Optimization in Bike Sharing Systems .................. 21

2 Bike Sharing Systems Overview .................................. 23
  2.1 Introduction .................................................. 23
  2.2 Related Work .................................................. 26
    2.2.1 Performance Analysis and Decision Support ............... 26
    2.2.2 System Design ............................................. 27
    2.2.3 User Demand .............................................. 27
    2.2.4 Redistribution ............................................. 28

3 Modeling User Trips ............................................. 33
  3.1 Model Formulation ............................................. 33
  3.2 Objective Function ............................................ 35
  3.3 Availability of Bikes and Docks ................................ 37
    3.3.1 Extension to Multiple Resource Uses per Time Period .... 38
  3.4 Discussion .................................................... 40

4 Dynamic Redistribution ......................................... 43
  4.1 Preliminaries ................................................. 43
6.1 Motivation ......................................................... 87
6.2 Problem Description ............................................. 90
   6.2.1 Main Entities .................................................. 91
   6.2.2 Schedule Characteristics ...................................... 92
   6.2.3 Challenges ..................................................... 94
6.3 Related Work .................................................... 94
   6.3.1 Vehicle Routing and Crew Scheduling ....................... 95
   6.3.2 Workforce Scheduling ......................................... 96
   6.3.3 Applications .................................................. 96
7 Schedule Generation ............................................... 101
   7.1 Adaptive Large Neighborhood Search (ALNS) ................. 101
      7.1.1 Method Description ......................................... 102
   7.2 Global Schedule Coordinator .................................... 106
      7.2.1 Coordination Requirements ................................ 106
      7.2.2 Model Description .......................................... 108
      7.2.3 Model Integration .......................................... 111
   7.3 Operators ...................................................... 113
      7.3.1 Destroy Operators ........................................... 113
      7.3.2 Repair Operators .......................................... 114
      7.3.3 Discussion .................................................. 117
   7.4 Driver Assignment .............................................. 117
      7.4.1 Full-Time Drivers .......................................... 117
      7.4.2 Employees as Drivers ....................................... 120
   7.5 Algorithmic Enhancements ...................................... 121
      7.5.1 Accelerated Insertions ..................................... 121
      7.5.2 Batch Insertions ............................................ 122
   7.6 Computational Experiments ..................................... 123
      7.6.1 Experimental Setting ....................................... 123
      7.6.2 Results ..................................................... 125
# List of Figures

2-1 Bike sharing systems in operation as in April 2016. .......................... 24

3-1 Successful trips in the system for an example of user demand. .............. 36
3-2 Results of the model for an example of user demand. ........................ 36

4-1 Geographic decomposition of stations into areas. ............................. 47
4-2 Decomposition statistics for each area. ........................................ 47
4-3 Leader selection and group formation example. ................................ 55
4-4 An example of a mini-group of stations. ....................................... 56
4-5 Redistributed resources as a function of time. ................................ 58
4-6 Partial information per group is provided to the model. ....................... 59
4-7 Example of type I (sparse) grid used in the experiments on synthetic data. ................................................. 62
4-8 Example of type II (dense) grid used in the experiments on synthetic data. ................................................. 62
4-9 Rebalancing gain for method comparisons. ..................................... 66
4-10 Rebalancing gain for multi-period redistribution. ........................... 67
4-11 Rebalancing gain with and without neutral value rebalancing actions. 68
4-12 Rebalancing gain for ideal initial state of stations without and with dynamic rebalancing. ................................. 69

5-1 Duration of outage events. ......................................................... 74
5-2 Daily average observed and estimated lost outgoing demand from May to September 2017. ................................................. 81
5-3 Daily average observed and estimated lost outgoing demand from May to September 2017 considering only the weekdays. 81
5-4 Estimated lost demand in the duration of a day. 82
5-5 Probability of walking $p$ as the allowed walking radius $l_{max}$ and the shifting distance multiplier $\alpha$ vary. 83

6-1 Operation types. 89
6-2 Representation of a customer request as a set of steps. 91

7-1 Example of step insertion and the complications that might arise to ensure the validity of the resulting schedules. 109
7-2 Histogram of the average daily number of events per start time. 124
7-3 Number of employees per skill in the dataset. 125
7-5 Average percentage of insertion attempts that corresponds to each batch size. 126
7-6 Average success rate of insertion per batch size. 126
7-4 Quality of the solution per iteration of the method. 127
List of Tables

3.1 Notation used in the formulation of the model (UT). . . . . . . . . . 34

4.1 Served users in the scenario of homogeneous rebalancing needs. . . . . 63
4.2 Served users in the scenario of heterogeneous rebalancing needs. . . . 63

5.1 Out-of-sample demand estimation MSE. . . . . . . . . . . . . . . . . 80
5.2 Consistency results for various parameter values. . . . . . . . . . . . 83

7.1 Results of daily instances. . . . . . . . . . . . . . . . . . . . . . . . . 128

A.1 Rebalancing results for method comparison. . . . . . . . . . . . . . 141
A.2 Rebalancing results for multi-period rebalancing. . . . . . . . . . . 142
A.3 Rebalancing results with and without neutral value rebalancing actions. 143
A.4 Results when dynamic rebalancing is combined with ideal static rebal-
ancing. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 143
A.5 Served users without dynamic rebalancing. . . . . . . . . . . . . . 144
Chapter 1

Introduction

Resource management has always created challenges in many transportation systems across various application domains. In particular, the limited nature of resources in most real-world settings generates the need for optimization in order to achieve efficient resource usage. This efficiency might refer to both increasing the system service levels as measured by specific metrics (such as user capacity) depending on the application, as well as lowering the operational cost which is often a major consideration.

These problems arise both in classic fields like logistics, as well as in newly emerging areas, such as transportation systems in the sharing economy. Many operations are driven by serving customer requests which is often achieved through the use of employees with appropriate expertise, as well as of a fleet of vehicles in order to transport items whenever required. Examples include package and mail delivery, technician scheduling, home health care services, and others. In all those settings, the operator needs to efficiently schedule their available resources, vehicles and employees, in order to complete all service requirements. As systems become larger and more complicated though, this task becomes more and more challenging.

This problem is further amplified in many emerging systems in the sharing economy. In particular, these systems are founded by definition on the sharing of resources among users. This might take the form of vehicle sharing (car-sharing, bike-sharing, etc.) or ride-sharing, among others. In the vehicle sharing setting, the resources (the vehicles) are placed at various locations across a metropolitan area and users can
make short-term rentals that allow them to perform one-way or round trips. In the ride-sharing or ride-hailing examples, users are offered the possibility of performing rides that they share with other users, which allows them to reduce their transportation costs. This sharing of resources creates many coordination needs, since, in these systems, it is crucial to ensure enough resource supply to cover customer demand. As a result, operators often perform resource redistribution or offer incentives to ensure adequate availability throughout the operational period.

This thesis focuses on questions both in the area of vehicle sharing systems, as well as in the scheduling of resources for companies to satisfy various types of their customers’ requests. Most of the questions studied in this work develop across three main pillars: data, scale, and operations.

Data. We undoubtedly live in the era of data. More and more people realize its power, and technological advancements have allowed operators to gather an abundance of real-time and historic data that can provide invaluable insights. As a result, this has also changed the basis of decision making, turning it to data-driven approaches and creating amazing opportunities for research in this direction. One of the main questions then is: How can we develop algorithms and models that utilize data to their full potential to inform better decisions?

Scale. In recent years, operations have become more globalized and interconnected than ever. New systems might develop across multiple regions and markets, while existing services have seen a significant increase in their user base. For example, the size of the carsharing market in North America has increased from 211,000 members in 2007 to 1.9 million members in 2017 (Shaheen et al. (2018)). How can we adapt traditional methods to scale to large system sizes or develop new approaches that can provide solutions in a timely manner?

Operations. Hubway, car2go, Amazon Prime Now; the transportation scenery around us evolves constantly and it evolves fast. New services both for passenger transport (Uber, Lyft) and product delivery (Instacart, Drizly, Postmates) have emerged in the last decade and rapidly expanded in many cities around the world or specifically in the United States. This leads to large and complex systems with
demanding constraints, whose viability depends heavily on their efficient operations planning. Products must be delivered within specific time windows, vehicle rides need to be available only minutes after their request, customer requests often arrive in an online fashion requiring updates of previous decisions, traffic conditions might derail any planned schedules, and so on. How can we develop new models and algorithms to capture the increased complexity and variety of the emerging modern services?

The remainder of this thesis is divided in two main parts. The first part focuses on optimization of resources in bike sharing systems, and the second presents methods for efficient scheduling of resources to satisfy a variety of customer requests, problem faced by many large companies that offer a wide range of services. An overview of each part follows.

1.1 Resource Optimization in Bike Sharing Systems

One of the greatest challenges that arise in vehicle sharing systems is due to spatial and temporal imbalances of user demand. In the particular example of bike sharing systems, many one-way trips, as well as large simultaneous flows of customers toward industrial areas in the morning and residential areas in the evening very often lead to a shortage of bikes and docks at the stations. Since reliable service is essential for the viability of these systems, operators use a fleet of vehicles to redistribute bikes across the network.

In this thesis, we answer questions on multiple aspects of the system’s operations by providing a model for successful user trips, a method to estimate lost and shifted user demand, as well as an approach that achieves efficient redistribution of the system in fast running times that allow the incorporation of real-time information. In particular, the contributions of this work can be summarized along three main axes:

1. We propose a linear programming model that captures the bike flows that result from all trips that are performed by users throughout the network. This model can be used instead of trip simulations to evaluate the efficiency of the system.
It does not depend on the exact arrival order of the users, which is often difficult to estimate, as simulation scenarios might, but instead on the total number of users per time period. Moreover, the fact that this is a linear program makes it possible to be used as a component of more complex models, thus incorporating user flows with other problem aspects.

2. We introduce a new mixed integer programming formulation to solve the dynamic rebalancing problem. We develop decomposition techniques based on appropriate station grouping and incorporation of partial information for each group to the global rebalancing model. This achieves smaller model sizes that still contain enough information for each station group, and we manage to produce rebalancing plans that scale to real-world instances of actual bike sharing systems. The core ideas of our approach can be extended to dockless bike-sharing (or scooter-sharing) systems by proper space discretization, as well as be applied to the solution of any pick-up and delivery vehicle routing problem.

3. We present new models for estimating the actual user demand of bike sharing systems. These include both the lost demand due to lack of bikes and docks at the stations, as well as the shifted demand which is the result of users walking to nearby stations in order to find available resources when none are available at their current location. Both of these aspects are very often not considered in bike sharing literature, but their effect on the rebalancing efficacy is crucial since redistribution plans heavily depend on reliable estimation of the demand.

This work is the focus of the next four chapters of this thesis. In particular, Chapter 2 gives an overview of bike sharing systems and the related work in this area. Chapter 3 presents the model to estimate the user trips that are performed successfully in the system, while Chapter 4 focuses on the redistribution of the system. Finally, Chapter 5 illustrates how we can estimate the actual user demand. Our methods make use of historic and real-time data provided by bike sharing systems’ operators, and computational experiments both on synthetic and real-world data are performed to evaluate the efficacy of the methods.
1.2 Resource Management for Complex Event Scheduling

This part of the thesis is motivated by multiple discussions with a large international company that provides a variety of services to individuals and corporations. Some examples include foodservice (e.g. catering events, beverage deliveries) and support services (e.g. audiovisual set-up for events), among others. Its operations are very complex and currently span across many sub-organizations that do not jointly optimize their resources.

Our goal is to develop a universal scheduler that covers this wide range of service types. However, different services come with different operations models and a variety of constraints that need to be satisfied, while specialized employees and vehicles are often required for the successful service of customer requests. This complicates the development of a scheduler, since the optimization takes place over an heterogeneous fleet of vehicles that might differ both in item and passenger capacity, as well as their properties (e.g. refrigerator vehicles), and a set of employees with varied skill sets and levels of experience. At the same time, many other aspects, such as team formation, coordination of vehicles and employees, transportation of the employees, time precedence constraints of event stages, need to be taken into account just to obtain feasible resource schedules.

Our main contributions can be summarized as follows:

1. We present the Complex Event Scheduling Problem, which includes generating schedules that can satisfy a great variety of objectives and constraints that depend on the application. Besides the company that motivated this work, this problem is relevant to many large logistics companies that need to offer a variety of services in order to satisfy their customer needs, as well as smaller companies whose services are subsets of the ones under consideration.

2. We propose a complete optimization approach to generate schedules for the resources. Our method is based on a combination of Adaptive Large Neigh-
borhood Search (Ropke and Pisinger (2006a)) with multiple other components, such as use of linear programming to maintain schedule feasibility throughout the algorithm. We evaluate our approach through computational experiments on various instances that showcase the scale and complexities that are faced in real-world settings.

These topics constitute the focus of the sixth and seventh chapters of this thesis. In particular, Chapter 6 details further the motivation of the problem and presents the Complex Event Scheduling Problem, while Chapter 7 focuses on the optimization and the experimental results.
Part I

Resource Optimization in Bike Sharing Systems
Chapter 2

Bike Sharing Systems Overview

2.1 Introduction

Early precursors of vehicle sharing systems started emerging more than half a century ago, presenting limited success at first, but slow growth, which led to the worldwide phenomenon that is observed in the present years. Millions of people have acknowledged the benefits of these systems and shown their support by substituting a smaller or larger part of their everyday commute with some form of ridesharing. The most common types of vehicles that are met worldwide are cars and bicycles, with recent advances in technology, such as electric vehicles, being gradually incorporated as well.

The idea behind vehicle sharing systems is simple; commuters are offered the benefit of a short-period vehicle rental in order to perform one-way or round trips. Vehicles can be picked-up and returned at specific locations, which are spread across the cities in order to increase the range of service. The charge depends on the duration of the trip, as well as the type of membership selected by the user. Operators usually offer monthly and annual memberships, which are ideal for regular users, as well as daily or short-period memberships designed to facilitate visitors and casual users.

The main purpose of this research is to provide the operator of a vehicle sharing system with an approach to manage major operations that are required for the smooth functioning of the system, by utilizing historic and real-time data. In this work, we assume that the system is already functional, in the sense that we do not consider
strategic decisions that need to be made while the system is being constructed. These include the locations and capacities of stations, the number of vehicles, etc., and thorough planning is required in order to ensure that the system specifications are adequate to cover most of the customer demand.

After a system launches, the operator faces many daily challenges in order to keep the service at a high level and meet the expectations of the users. Let us now focus on the particular case of the bike sharing systems, which will allow us to provide more details about specific operations. In a typical bike sharing system, users can pick up a bike from any station with available bikes and return it to any station with an empty dock. However, large flows of riders from residential to industrial and commercial areas in the morning and opposite flows in the evening, as well as many one-way trips, can render the system highly unbalanced in many instances during the day. In particular, very often there are phenomena of starvation, when customers wish to pick up a bike from a station that is empty, and congestion, when customers want to return their bikes to a station with no empty docks.

In order to deal with this problem, companies deploy a fleet of trucks which perform the rebalancing of the network: they pick up bicycles from stations with a surplus of bikes and drop them off at stations that currently require them. Efficient rebal-
ancing is of crucial importance to the operating company. If a customer repeatedly fails to find a bike or is not able to drop it off at their desired destination, they will eventually seek alternative modes of transportation, leading to significant losses for the company and endangering the viability of the system.

The goal of this work is to generate a complete plan of actions for the operating crew that will cover all daily rebalancing operations. An important factor for the success of this project is the plethora of data that is regularly gathered by the operators. Historic data, such as past trips and system statistics, will be used to accurately estimate the user demand, which can in turn be used to generate strategies that optimize the system’s performance. At the same time, real-time tracking of all components of the system allows to dynamically readjust daily plans, based on the currently realized scenarios.

The motivation and significance of this research might be obvious from the view of the operator. However, the impact of a viable vehicle sharing system is much greater than that. If users can rely on such a system covering their transportation needs, vehicle ownership statistics will be reduced, decreasing the consumption of valuable natural resources that vehicle manufacturing requires. Car sharing can solve problems that big cities often face, such as lack of parking spaces. In addition, a transition to using bike sharing systems can contribute significantly to dealing with the problem of congestion, and, at the same time, reduce harmful gas emissions in the atmosphere. A success in the operation of the existing vehicle sharing systems will likely motivate more and more cities to adopt similar transport behaviors, turning them from energy consumers and pollution generators into more eco-friendly transportation networks.

From a theoretical perspective, this topic raises many interesting research questions. The high dimensionality of the system - large number of stations, time intervals of interest, options of routing - renders it intractable for real-life instances. The answer to this type of questions is relevant to many systems, whose operation requires matching demand and supply, especially in the context of a dynamic environment. This might refer to resources that need to travel to various locations in order to serve customers requests, as well as many flexible transportation systems that have
currently emerged or been planned for the near future.

In the sections that follow, we start by presenting an overview of the literature on bike sharing systems and existing rebalancing approaches. Then, Chapter 3 focuses on modeling the user trips across the network, Chapter 4 studies the rebalancing of the system, and computational experiments follow to evaluate the efficacy of the methods. Finally, in Chapter 5, we introduce a model that accounts for lost and shifted demand given historic data of past trips.

2.2 Related Work

Research in the area of Bike Sharing Systems (BSS) is relatively recent. DeMaio (2009) presents the history of BSS: he starts from the first generation back in 1965, up to the third one in the present days, while also providing some areas for improvement that could lead to the next generation. Shaheen et al. (2010) discuss the evolution of BSS around the world, their business models, as well as their social and environmental effects. For a review of BSS, readers can also refer to Fishman (2016), which includes aspects such as growth, usage, and impact. The sections that follow focus on more specific themes of the BSS literature and their connections to our work.

2.2.1 Performance Analysis and Decision Support

This first stream of research is peripheral to our work and mainly focuses on providing insights that help decision making in BSS. Shaheen et al. (2011) conduct a survey to better understand the early adoption of BSS and users’ travel behavior. Raviv and Kolka (2013) model the user behavior in the face of lack of resources and they introduce a user dissatisfaction measure to evaluate station performance. Tao and Pender (2017) conduct an analysis on various performance measures for bike distribution across the system, while Vogel and Mattfeld (2010) study the effect of various levels of redistribution efforts on user satisfaction.
2.2.2 System Design

When a BSS is constructed or expanded, some of the questions that arise concern the number of stations that are required, as well as their optimal locations across the city. In this direction, Lin and Yang (2011) propose a model to estimate the number and locations of the stations, as well as the bike paths between them, while considering service levels and overall cost. Martinez et al. (2012) develop a MILP formulation to determine the station locations, while Nair and Miller-Hooks (2014) study the optimal system configuration in general vehicle sharing systems which include BSS. Finally, Freund et al. (2017) modify already operational systems by reallocating their docks to increase their efficiency. In our work, we assume that the system is already in operation and its configuration is set, so we are looking to optimize its everyday operations.

2.2.3 User Demand

Accurate demand modeling is essential for the efficiency of the redistribution process. Shu et al. (2013) model the number of customers traveling between each pair of stations at each time period as a Poisson process. A similar approach is employed by Nair and Miller-Hooks (2011), after they observe that in the dataset of a Singapore car-sharing system they use, both the number of vehicles leaving a station and arriving at a station at each time period can be accurately modeled with Poisson distributions. Vogel et al. (2011) cluster stations with similar behavior throughout the day and use data from realized trips aggregated per hour to generate typical customer demand. Finally, Borgnat et al. (2011) and Singhvi et al. (2015) take into account various parameters, such as information about the weather, special events in the area, taxi usage, to forecast the number of rentals across the system.

Our work differs from many of these approaches in that we do not focus on modeling demand based on observed trips or predicting future demand. Instead, we wish to reveal the actual underlying user demand of the system by taking into account trips that were not accomplished because of resource shortages. Lack of resources leads
to customers leaving the system or walking to nearby stations. Considering only the completed trips leads to a misrepresentation of the actual demand, since this user behavior can affect the resources that are required at each station. O’Mahony and Shmoys (2015) consider only the first of these two aspects (users that left the system) and obtain an estimation for this lost demand by examining the typical behavior of the stations. In our approach, we incorporate an extra dimension, the daily demand trends of each station, and show that this leads to decreased estimation error in all instances that we examined.

Finally, we provide a method to estimate the shifted demand of the system, and obtain in the end the actual user demand for each station. This shifted demand is also considered in the recent work by Goh et al. (2019). The authors propose a rank-based choice model to reveal the primary user demand of the system, which differs from our work as we follow a linear programming approach to estimate the probability of walking between stations.

2.2.4 Redistribution

A different stream of research focuses on the redistribution procedures. Some authors study the static version of the problem, where no user demand is assumed while the redistribution is in process. Others incorporate user demand in the optimization, and provide solutions for the dynamic rebalancing problem. Finally, some authors focus mostly on providing guidance on the number of resources to be picked up and dropped off, without generating decisions on the carriers’ routing. More details on each of these directions are provided below.

Redistribution without Routing

Studies in this area aim to propose the desired configuration of the system at each time period based on the anticipated demand and identify the number of vehicles that need to be carried from one station to the other in order to achieve it. Nair and Miller-Hooks (2011) present a MIP that involves joint chance constraints to
determine the least cost redistribution that will meet most of the upcoming demand in the application of car-sharing systems. Shu et al. (2013) formulate a linear program to estimate the performance of the system and the benefits of redistribution. Both of these approaches assume that redeployment of vehicles between any two stations is always possible, which might not be the case in an actual vehicle sharing system. Moreover, since they do not take into account the exact routing of the carriers, they do not provide a detailed plan for the execution of the redistribution. This contrasts with our approach which produces a complete routing and redistribution plan.

**Static Redistribution**

Another set of studies tackles the problem of static rebalancing with routing, where the goal is to find the optimal routing of the carriers that will achieve a desired configuration, considering no customer demand during the redistribution. This is the case of systems where redistribution is performed only at the end of each day, in order to prepare the system for the demand of the following day.

Schuijbroek et al. (2017) provide a MIP formulation for the problem, as well as a cluster-first route-second heuristic to reduce its running time. Raviv et al. (2013) wish to minimize user dissatisfaction in conjunction with the cost of repositioning. They introduce multiple formulations and techniques, such as arc deletion when alternative paths of same cost exist, which allow them to solve the problem in reasonable time even for larger instances. Rainer-Harbach et al. (2013) propose a variable neighborhood search approach to generate candidate routes for the carriers, while Di Gaspero et al. (2013) present a hybrid method that combines constrained programming with ant colony optimization. Chemla et al. (2013) propose an integer programming formulation and they use a branch-and-cut algorithm to solve its relaxation and tabu search to obtain an upper bound on the optimal solution of the problem. A tabu search approach is also proposed by Ho and Szeto (2014) in order to solve the static rebalancing problem and obtain good quality solutions in short running times. Dell’Amico et al. (2014) suggest multiple formulations for the problem and a branch-and-cut solving approach, while Forma et al. (2015) present a 3-step heuristic to obtain a static repo-
sitioning plan for the system.

The research described in this section differs from our approach on the assumption that the system is not being used by customers while the rebalancing takes places. In our work, we consider the system being in full operation during the redistribution procedures, and this complicates the problem further since user movements keep changing the state of the system and this needs to be taken into account.

Dynamic Redistribution

In the dynamic version of the problem, redistribution is performed multiple times during the day and demand is not neglected when rebalancing takes place. This creates extra challenges for the redistribution planning, as simultaneous user and truck movements need to be considered, and changes in the state of the system that result from user trips must be captured appropriately. Despite the difficulties, dynamic rebalancing can help the operator deal with the large fluctuations of the demand and the service needs that arise in particular during the rush hours, where most outage events usually emerge.

Ghosh et al. (2017) propose a MIP formulation for the problem, which they solve by using Lagrangian Dual Decomposition combined with an abstraction approach in the case of large instances. Contardo et al. (2012) focus on the peak hours of the day. They formulate the problem and use two decomposition methods, Dantzig-Wolfe and Benders decomposition, to obtain lower and upper bounds for medium and large instances. O’Mahony and Shmoys (2015) study the rebalancing problem both for the (static) overnight version as well as during the rush hour. They propose a matching approach that matches stations that need bikes with stations that need docks in order to transfer the resources between them. Caggiani and Ottomanelli (2013) also provide an optimization model to reduce the redistribution costs which they test on small system instances.

Our work also tackles the dynamic rebalancing of the network. However, our goal is to provide solutions that can scale to large real-world systems in running times that are fast enough to allow real-time information to be taken into account. In our
experiments, we take this type of information into account every thirty minutes of
the planning horizon, but this can take place in much shorter time intervals (espe-
cially during the rush hours) so as the most up-to-date state of the system is always
considered.
Chapter 3

Modeling User Trips

3.1 Model Formulation

Before presenting our approach for the rebalancing optimization of the network, we propose a linear program which we use in order to model the user trips that are realized throughout the system. We assume that users perform a trip only if there is an available bike at their origin station and a dock at their destination; otherwise they do not wait but leave the system instead. We do not allow demand shifting (people walking to nearby stations) in this model since we wish to use it to evaluate the efficiency of our methods and the goal is to eliminate the need for demand shifting through appropriate placement of resources.

Furthermore, we assume that each trip is completed within a single time period, which we define to be thirty minutes long. This assumption is not far from reality as 86.77% of trips in a five-month past trips dataset (May to September 2017 for Capital Bikeshare, Washington DC) that we tested indeed had duration less than thirty minutes. The model can still be generalized to longer trips by changing appropriately the indexing of the variables and constraints, which we will not attempt here for the benefit of simplicity of notation. Finally, we assume that each bike and dock is used by at most one customer per time period, but also discuss a relaxation of this assumption in Section 3.3.1.
Table 3.1: Notation used in the formulation of the model (UT).

The model notation is provided in Table 3.1 and the formulation follows.

\[(UT): \text{maximize } \text{SUCCESSFUL\_TRIPS}(f) \quad (3.1)\]

\[
\text{subject to } \quad f_{ij}^t \leq d_{ij}^t \quad \forall i, j, t \quad (3.2)
\]

\[
\sum_j f_{ij}^t \leq \text{AVAILABLE\_BIKES}(i, t) \quad \forall i, t \quad (3.3)
\]

\[
\sum_j f_{ji}^t \leq \text{AVAILABLE\_DOCKS}(i, t) \quad \forall i, t \quad (3.4)
\]

\[
b_{i}^{t+1} = b_i^t + \sum_j f_{ji}^t - \sum_j f_{ij}^t \quad \forall i, t \quad (3.5)
\]

\[
b_i^t \leq C_i \quad \forall i, t \quad (3.6)
\]

\[
f_{ij}^t \leq \frac{d_{ij}^t b_i^t}{\sum_k d_{ik}^t} \quad \forall i, j, t \quad (3.7)
\]

\[
\sum_i b_i^0 \leq B \quad (3.8)
\]

\[
f_{ij}^t, b_i^t \in \mathbb{R}_+, \quad \forall i, j, t \quad (3.9)
\]

According to (3.2), for each edge and time period the flow of bikes is bounded by the user demand. (3.3) and (3.4) require that a trip is only realized if there are available bikes at the origin station and available docks at the destination. Constraint (3.5) is a flow conservation constraint and (3.6) a capacity constraint for the bikes of each station. Constraint (3.7) is a proportionality constraint; since we do not consider the exact arrival order of users within the same time period, this fairness constraint ensures that the number of bikes traveling to each destination is proportional to the corresponding demand (idea based on Ghosh et al. (2017)). In (3.8) the number of bikes is bounded by the total bikes in the system $B$ (this can be replaced with equality.
if all bikes must be used, otherwise the solution provides the optimal number of bikes in the system). Finally, if an initial configuration of the system is provided, this is taken into account by setting appropriately the values of $b^0_i$. In the opposite case, the optimal values of $b^0_i$ are calculated by solving the above LP. Notice that this is a relaxed version of the problem, i.e. the variables are real numbers and not integers. This is further discussed at the end of the chapter.

### 3.2 Objective Function

If we maximize the number of successful trips, while respecting constraints (3.2)-(3.9), we get a good representation of the trips that were performed in the system. This is the case because each time a user is looking for a bike or a dock and they find one available, then they are going to use it. So, in reality, the number of successful trips is the largest number of trips that can be performed given demand and availability constraints, as well as the first come first served rule.

Hence, a natural option for the objective function is $\text{SUCCESSFUL\_TRIPS}(f) = \sum_{t,i,j} f^t_{ij}$. However, that would not provide the desired result, since it does not ensure that the arrival order is being respected. There are cases where the system may prevent users that arrive earlier from taking a bike if that can be later used in a better way (by serving more trips), as illustrated by the following example.

**Example 3.2.1.** Assume for simplicity that we have a BSS with three stations, one bike initially located at station 1, and three periods of demand with $d^1_{12} = 1$, $d^2_{13} = 1$, and $d^3_{32} = 1$ (recall $d^t_{ij}$ denotes how many people want to go from $i$ to $j$ at time $t$). In an actual system with such demand, the first customer would take the available bike, and customers 2 and 3 would not use the system as there are no bikes at their origin (see Figure 3-1). However, the objective of maximizing the number of people served leads to a solution where user 1 does not take the available bike, but it is instead used by customers 2 and 3 (see Figure 3-2). This way, two users travel successfully, but that does not agree with the first come first served rule.

In order to impose the rule of first come first served, we need to "prioritize"
Figure 3-1: Case I: When the first come first served principle is satisfied, user 1 takes the available bike, while users 2 and 3 do not, leading to one successful trip. (Normal edges are used for successful trips and dashed edges for the unsuccessful ones.)

Figure 3-2: Case II: Results of the model without the first come first served principle. In order to maximize the total successful trips, the solution of the model withholds the bike from user 1, in order to satisfy users 2 and 3.

Customers. This prioritization can be achieved by setting weights for each customer, which must be selected appropriately, to ensure that solutions like the one in the example will not be optimal for the model.

**Proposition 3.2.1.** The objective function $SUCCESSFUL\_TRIPS(f) = \sum_{t,i,j} 3^{T-t} f_{ij}^t$, where $T$ is the number of time periods, enforces the first come first served (FCFS) principle for the users.

**Proof.** Let the objective be of the form $SUCCESSFUL\_TRIPS(f) = \sum_{t,i,j} w^{T-t} f_{ij}^t$, where the weights $w$ need to be selected. Consider there are enough bikes to cover the demand at station $i$ at time $t$. Then, a bike may not be used only if it offers
greater payoff by remaining at the same place and being used in later time periods. Similarly, due to the interconnected nature of the resources (bikes and docks), the dock that remained empty at another station $j$ due to the bike not getting there, can be then used to satisfy future incoming demand.

Let $V_{use}$ denote the value that is added to the system if the bike is used at time period $t$ and $V_{stay}$ the value in case it remains at station $i$ during period $t$. Recall that each bike can be used only once per time period. $V_{stay}$ obtains its highest value if the specific bike at $i$ as well as the dock at $j$ are used by customers during all subsequent periods $t'$, with $t < t' \leq T$. Since each bike and dock can be used once per time period, each successful use corresponds to a flow of one unit, so:

$$V_{stay} \leq \sum_{t' = t+1}^{T} w^{T-t'} + \sum_{t' = t+1}^{T} w^{T-t'} = 2 \sum_{t' = t+1}^{T} w^{T-t'} = 2 \frac{w^{T-t} - 1}{w - 1} \quad (3.10)$$

$V_{use}$ is at least as much as the value gained by the user that rents the bike at time $t$. Of course, it can be higher if the bike is used by other people in the following time periods.

$$V_{use} \geq w^{T-t} \quad (3.11)$$

It is easy to see that for $w \geq 3$, we have:

$$V_{use} \geq w^{T-t} > 2 \frac{w^{T-t} - 1}{w - 1} \geq V_{stay} \quad (3.12)$$

$V_{stay} < V_{use}$ implies that FCFS is satisfied, since an available bike not being taken by a user cannot be part of the optimal solution. We thus select $w = 3$ for solver numerical accuracy purposes.

3.3 Availability of Bikes and Docks

If we assume that each bike and dock can be used only once per time period, then the available bikes for each period are the ones currently present at the station, and
similarly for the docks.

\[ \text{AVAILABLE BIKES}(i, t) = b^t_i \quad \forall i, t \tag{3.13} \]

\[ \text{AVAILABLE DOCKS}(i, t) = C_i - b^t_i \quad \forall i, t \tag{3.14} \]

This is what will be used for the computational experiments and the remaining sections of this paper, but since this approach might be considered conservative, we present a way we can relax this assumption by considering the expected number of arrivals until a station becomes empty or full.

### 3.3.1 Extension to Multiple Resource Uses per Time Period

Consider the number of bikes at a station \( i \) during period \( t \). Initially, it is equal to \( b^t_i \).

For each customer arrival, this number increases by one if the customer is returning a bike, and decreases by one if the customer is borrowing a bike. Given that we have \( d^t_{in,i} \) customers that want to return a bike, and \( d^t_{out,i} \) that want to take one, we can define the probabilities \( p^{t,i}_{+1} \) and \( p^{t,i}_{-1} \) for a customer returning and taking a bike respectively.

\[ p^{t,i}_{+1} = \frac{d^t_{in,i}}{d^t_{in,i} + d^t_{out,i}} \tag{3.15} \]

\[ p^{t,i}_{-1} = \frac{d^t_{out,i}}{d^t_{in,i} + d^t_{out,i}} \tag{3.16} \]

These two probabilities that sum to 1 for each \( i \) and \( t \) define a random walk. Each customer arrival is assumed to be an independent random variable that is equal to +1 (increases the bikes by 1) with probability \( p^{t,i}_{+1} \) and equal to -1 (decreases them by 1) with probability \( p^{t,i}_{-1} \). We can now use known results from the theory of random walks to estimate when the station will run out of bikes or docks for the first time.

Let \( E^{t,i}_0 \) be the expected number of customers for the station \( i \) to become empty during period \( t \), and \( E^{t,i}_C \) the expected number of customers for the station to become
full (reach full capacity \( C_i \)). Then, we have:

\[
E_{0}^{t,i} = \begin{cases} 
+\infty & \text{if } p_{-1}^{t,i} \leq p_{+1}^{t,i} \\
\frac{b_{i}^{t}}{p_{-1}^{t,i} - p_{+1}^{t,i}} & \text{if } p_{-1}^{t,i} > p_{+1}^{t,i} 
\end{cases}
\] (3.17)

and, similarly:

\[
E_{C}^{t,i} = \begin{cases} 
+\infty & \text{if } p_{+1}^{t,i} \leq p_{-1}^{t,i} \\
\frac{C_i - b_{i}^{t}}{p_{+1}^{t,i} - p_{-1}^{t,i}} & \text{if } p_{+1}^{t,i} > p_{-1}^{t,i} 
\end{cases}
\] (3.18)

By replacing \( p_{+1}^{t,i} \) and \( p_{-1}^{t,i} \) these can equivalently be written as follows.

\[
E_{0}^{t,i} = \begin{cases} 
+\infty & \text{if } d_{out,i}^{t} \leq d_{in,i}^{t} \\
\frac{d_{in,i}^{t} + d_{out,i}^{t}}{b_{i}^{t} \cdot d_{out,i}^{t} - d_{in,i}^{t}} & \text{if } d_{out,i}^{t} > d_{in,i}^{t} 
\end{cases}
\] (3.19)

and, similarly:

\[
E_{C}^{t,i} = \begin{cases} 
+\infty & \text{if } d_{in,i}^{t} \leq d_{out,i}^{t} \\
(C_i - b_{i}^{t}) \cdot \frac{d_{in,i}^{t} + d_{out,i}^{t}}{d_{in,i}^{t} - d_{out,i}^{t}} & \text{if } d_{in,i}^{t} > d_{out,i}^{t} 
\end{cases}
\] (3.20)

\( E_{0}^{t,i} \) and \( E_{C}^{t,i} \) are the expected number of customers that can be served, before a lack of resources emerges. From these, the \( p_{+1}^{t,i} \) fraction of them will be incoming and the rest outgoing. AVAILABLE_BIKES and AVAILABLE_DOCKS represent the number of outgoing and incoming users that can be served, so they can be easily computed as follows.

\[
\text{AVAILABLE\_BIKES}(i, t) = p_{-1}^{t,i} \cdot E_{0}^{t,i} = \begin{cases} 
+\infty & \text{if } d_{out,i}^{t} \leq d_{in,i}^{t} \\
\frac{d_{out,i}^{t}}{b_{i}^{t} \cdot d_{out,i}^{t} - d_{in,i}^{t}} & \text{if } d_{out,i}^{t} > d_{in,i}^{t} 
\end{cases}
\] (3.21)
AVAILABLE_DOCKS(i, t) = p_{i+1}^t \cdot E_G^t = \begin{cases} +\infty & \text{if } d_{in,i}^t \leq d_{out,i}^t \\ (C_i - b_i^t) \cdot \frac{d_{in,i}^t}{d_{in,i}^t - d_{out,i}^t} & \text{if } d_{in,i}^t > d_{out,i}^t \end{cases}

Notice that the above expressions are still linear and, thus, can be incorporated in the linear model (UT) that models the user flows. However, the objective function selected in the Section 3.2 is no longer adequate to satisfy the FCFS principle, since it assumes that each bike and dock is used once per period. Hence, a larger weight needs to be selected that takes into account how many times each bike and dock might be used per time period.

3.4 Discussion

The optimal solution of (UT) is not necessarily integral, due to proportionality constraints (3.7). However, this is not an issue as our goal is to use this model as a measure of the performance of the system. A different approach is to perform a simulation of user trips. For that, an arrival order of the customers needs to be specified, in order to determine which ones are served in case of lack of resources.

In our method, we take into account only the number of users that arrive at each station during each time period - for example each half hour - and no specific order within the period is required. The proportionality constraint (3.7) then attempts to balance the various demand scenarios that arise from different arrival orders, by upper-bounding the number of trips to each destination. The advantage of this approach is that it does not depend on the exact arrival order, which is generally harder to estimate than just the number of users per period. We can think of the solution we get as the result of running multiple simulations on different demand scenarios within each period, and taking an average of their performance. Moreover, even if a similar proportionality approach is incorporated as part of a simulation procedure, the fact that this is a linear programming model expands its capabilities, as it can be
also incorporated as part of more complicated models that combine user flows with other problem aspects.

One valid concern is the numerical issues that might arise when solvers are used to get a solution for the model. If there are many time periods per day, the objective function might obtain very large values as the objective weights grow exponentially with the number of periods. This issue is easy to tackle by solving the model in more than one stages. For example, we can solve it for the first half of the time periods, set the final system configuration as the initial state of the second run, and then solve the model separately for the second half of the periods. More stages can be used as well if required, in order to achieve a desired numerical accuracy. A solution for the model can be obtained very fast, so introducing more than one stages does not significantly change the total time that is required.
Chapter 4

Dynamic Redistribution

4.1 Preliminaries

Redistribution operations are planned based on the current configuration of the system, as well as the future needs. In this section, we provide a brief description of the main aspects that influence rebalancing decisions.

Unserved Customers

We first consider the unserved demand if no rebalancing is performed in the system. This will give an indication about the stations that have the greatest need for bikes or docks. Given the current configuration of the system, and the demand that is expected for the remainder of the day, we use the linear program (UT) of Section 3.1 to find which users successfully completed their trip. Then, for each station, we calculate the number of extra bikes or docks that are required to address the unserved outgoing demand Unserved_Demand(out, i, t) and incoming demand Unserved_Demand(in, i, t) for each station i and time period t.

Unused Resources

Since the rebalancing takes place simultaneously with the user movements, it should not interfere with their actions. In particular, bikes and docks that are used by cus-
tomers cannot participate in the rebalancing during the same time period. So, for each station $i$ and time $t$, we compute the number of unused bikes $\text{UNUSED}_\text{BIKES}(i,t)$ and docks $\text{UNUSED}_\text{DOCKS}(i,t)$ that are available for redistributing.

**Rebalancing Needs**

At this point, we have calculated the amount of customers that failed to take a trip, as well as the number of resources that are unused at their current location, and, thus, can be transferred elsewhere. We are now ready to make suggestions to the trucks regarding the number of bikes they need to pick up or drop off at each station they may visit. In particular, if we are currently planning the rebalancing to take place during period $t$, we can look at the unserved demand at time $t + 1$, that is $\text{UNSERVED}_\text{DEMAND}(\text{out},i,t+1)$ for bikes and $\text{UNSERVED}_\text{DEMAND}(\text{in},i,t+1)$ for docks, which guide the rebalancing targets for the truck. At the same time, we look at the available bikes that can be picked up/dropped off at each station ($\text{UNUSED}_\text{BIKES}$ and $\text{UNUSED}_\text{DOCKS}$ for times $t$ and $t+1$), so that the rebalancing operation is actually feasible. One drawback of this method is that it is very myopic: it only considers the following time period. So, instead, we actually consider the unserved demand and the unused resources further in the future (for example for the upcoming five or ten time periods) and determine the value of current resource needs $\text{BIKES}_\text{NEEDED}(i,t)$ and $\text{DOCKS}_\text{NEEDED}(i,t)$ for each station.

**4.2 Scalability**

In actual systems, the large number of stations often requires multiple vehicles working simultaneously to serve various parts of the network. This extends the redistribution capacity of the operator, but also creates coordination needs that increase the complexity of the problem, making it computationally intractable. Moreover, when redistribution is planned for the whole day in advance, a large number of time periods creates extra challenges for the operator. We will now discuss some methods to address these issues.
4.2.1 Geographic Segmentation

One common approach towards tackling the multi-vehicle scenario is to partition the area of service into regions with exactly one vehicle assigned to them. This is beneficial for drivers, as they can get familiar with the area more quickly, and thus they can become more efficient. It also ensures that the vehicles are adequately spread out to cover the rebalancing needs of most parts of the system.

The main challenge consists of identifying regions that achieve geographical proximity of stations and equally balanced work-load for assigned vehicles. The method that will be used is a variation of the k-median algorithm with a local search heuristic. There have been similar studies in the literature both from a theoretical viewpoint, as well as applied in the area of facility location problems. More on this topic can be found in Kanungo et al. (2004) and Arya et al. (2004).

Let $k$ denote the number of available trucks, which also determines the number of station clusters that we wish to create. $S$ is the set of all stations, $S_l$ the stations of cluster $l$, $C$ the set of centers of the clusters, and $c_l$ the station-center of cluster $l$. At any point, each station is assigned to the cluster with the closest center. The general outline of the algorithm is the following.

1. Initialization: Select $k$ stations at random out of $S$ that will serve as the initial centers $C$ for the $k$ clusters.

2. While the termination criteria have not been met, iterate:

   a. Randomly pick one center $c_l \in C$ and remove it from set of centers: $C \leftarrow C \setminus \{c_l\}$.

   b. Calculate the clustering cost when $c_l$ is replaced by any station $s \in S \setminus C$.

   c. Let $s^* \in S \setminus C$ be the station that minimizes the above cost.

   d. Update the set of centers as $C \cup \{s^*\}$. 
Clustering Cost

For each cluster, we consider the work-load of its assigned truck. This is a combination of the expected number of redistributed resources, as well as the distances that it needs to traverse. We use $w_i$ for the expected number of bikes that are required to be picked up or dropped off for station $i$, and $dist_{ij}$ for the distance between stations $i$ and $j$. Then, we define the work load of each cluster $l$ as:

$$\text{WORK\_LOAD}(l) = \sum_{i \in S_l} w_i + \alpha \sum_{i,j} \frac{dist_{ij}}{|S_l|}$$ (4.1)

Ideally, all clusters should have the same work-load, since we want to ensure that the work is well distributed among the trucks. In order to achieve that, we can consider the maximum work-load over all clusters, and aim to minimize it, reducing in that way the work of the busiest truck. The cost of clustering is in that case:

$$\text{CLUSTERING\_COST} = \max_l \text{WORK\_LOAD}(l)$$ (4.2)

Termination

At any point, the cost of the clustering solution is equal to the largest work-load among the vehicles. If this cost can be decreased by swapping a cluster center with some other station, then this replacement takes place during the iteration step of the algorithm. The cost is nonincreasing during the execution of the algorithm, and since it cannot improve indefinitely, a convergence to a local minimum is guaranteed given an adequate number of iterations. The algorithm terminates either when there has been no improvement for a set number of iterations or when a time limit, if any, has been reached. Since the execution time is very short, it can potentially be run multiple times with different random initializations and keep the best solution among them.
4.2.2 Rolling Time Horizon

If redistribution decisions need to be taken for the entire day, a large number of time periods leads to an exponential number of different scenarios. Optimizing over such a large solution space is intractable, so we propose to focus on actions for the imminent future. This choice is also reinforced by the fact that in real systems future user demand is not fully known in advance. So, planning far in the future is not always worthwhile, since the configuration of the system at that time may be very different from the one currently expected.

As a result, we use a rolling time horizon. Each time we look at the current state of the system and plan the rebalancing for the upcoming time periods. When the realization of the actual demand occurs for these periods, we consider the new state of the system, and roll the rebalancing window to solve for the next periods. Despite focusing on a few periods for the rebalancing, we still consider the demand ahead. In that way, all the information we possess about future demand is taken into account when relocating the resources.
4.3 Formulation

In this section, we introduce the Dynamic Rebalancing (DR) model. The decision variables include variables for the routing of the truck: binary variables $x_{ij}$ expressing whether the truck visits station $j$ right after station $i$, truck arrival times $t_i$ at each station $i$, station visit duration $dur_i$, as well as bikes $b_{ij}^T$ in truck upon arrival of the truck at station $i$. Moreover, the number of bikes to be picked up and dropped off from station $i$ correspond to decision variables $PU_i$ and $DO_i$ respectively. We further specify the picked up bikes by categorizing them as $PU_i^+$ for bikes that add immediate value to the rebalancing as they vacate docks for customers to use, and $PU_i^o$ for bikes that it is not necessary to be picked up from $i$, but they are in order to later be dropped off at other stations that need them. We consider the latter pick-up to be of neutral value to the rebalancing since the station where the pick-up takes place does not have an immediate gain from it. Similarly, we have $DO_i^+$ and $DO_i^o$ for the drop-offs with positive and neutral value respectively.

We wish to maximize the positive value actions, while allowing the neutral type ones to take place where needed. Weights $\gamma$ and $\delta$ in the objective are small constants in order to jointly minimize traveling time and redundant redistribution, $M$ is a large constant, $\tau_0$ is the initial time of the rebalancing period that the truck is available, $\zeta_0$ is the initial bike load of the truck, $\tau_{\text{action}}$ is the time that is required for each bike pick-up and drop-off, $\tau_{\text{station}}$ the overhead time per station visit (for parking, etc.), and $\tau_{\text{period}}$ the length of the rebalancing period. We introduce a dummy node for the truck, denoted with index 0, $S$ is the set of indices that corresponds to the station nodes and $S_0 = S \cup \{0\}$ the set that also includes the dummy node.

(DR):

$$\max \sum_{i \in S} PU_i^+ + \sum_{i \in S} DO_i^+ - \gamma \sum_{i \in S_0, j \in S} dist_{ij} x_{ij} - \delta \sum_{i \in S} (PU_i + DO_i) \quad (4.3)$$

s.t.

$$\sum_{j \in S_0} x_{ij} = 1 \quad \forall i \in S_0 \quad (4.4)$$
\[ \sum_{j \in S_0} x_{ji} = 1 \quad \forall i \in S_0 \quad (4.5) \]
\[ t_j \geq t_i + \text{dur}_i + \text{dist}_{ij} \cdot x_{ij} - M \cdot (1 - x_{ij}) \quad \forall i, j \in S, i \neq j \quad (4.6) \]
\[ t_i \geq \text{dist}_{0i} \cdot x_{0i} + \tau_0 \quad \forall i \in S \quad (4.7) \]
\[ t_i \leq \tau_{\text{period}} \quad \forall i \in S \quad (4.8) \]
\[ \text{dur}_i \geq \tau_{\text{station}} \cdot (1 - x_{ii}) + \tau_{\text{action}} \cdot PU_i + \tau_{\text{action}} \cdot DO_i \quad \forall i \in S \quad (4.9) \]
\[ b_j^T \geq b_i^T + PU_i - DO_i - M \cdot (1 - x_{ij}) \quad \forall i, j \in S, i \neq j \quad (4.10) \]
\[ b_j^T \leq b_i^T + PU_i - DO_i + M \cdot (1 - x_{ij}) \quad \forall i, j \in S, i \neq j \quad (4.11) \]
\[ b_i^T \geq \zeta_0 \cdot x_{0i} \quad \forall i \in S \quad (4.12) \]
\[ b_i^T \leq C^T - (C^T - \zeta_0) \cdot x_{0i} \quad \forall i \in S \quad (4.13) \]
\[ b_i^T \leq C^T \quad \forall i \in S \quad (4.14) \]
\[ b_i^T + PU_i - DO_i \leq C^T \quad \forall i \in S \quad (4.15) \]
\[ b_i^T + PU_i - DO_i \geq 0 \quad \forall i \in S \quad (4.16) \]
\[ PU_i \leq \text{UNUSED}_\text{BIKES}(i) \quad \forall i \in S \quad (4.17) \]
\[ PU_i^+ \leq \text{DOCKS}_\text{NEEDED}(i) \quad \forall i \in S \quad (4.18) \]
\[ PU_i = PU_i^+ + PU_i^o \quad \forall i \in S \quad (4.19) \]
\[ PU_i \leq M \cdot (1 - x_{ii}) \quad \forall i \in S \quad (4.20) \]
\[ DO_i \leq \text{UNUSED}_\text{DOCKS}(i) \quad \forall i \in S \quad (4.21) \]
\[ DO_i^+ \leq \text{BIKES}_\text{NEEDED}(i) \quad \forall i \in S \quad (4.22) \]
\[ DO_i = DO_i^+ + DO_i^o \quad \forall i \in S \quad (4.23) \]
\[ DO_i \leq M \cdot (1 - x_{ii}) \quad \forall i \in S \quad (4.24) \]
\[ x_{ij} \in \{0, 1\} \quad \forall i, j \in S_0 \quad (4.25) \]
\[ t_i, \text{dur}_i, b_i^T, PU_i, PU_i^+ , PU_i^o , DO_i, DO_i^+, DO_i^o \in \mathbb{R}_+ \quad \forall i \in S \quad (4.26) \]

Constraints (4.4) and (4.5) specify the truck routing. We impose that each node is visited exactly once, including the dummy node that corresponds to the truck. Nodes that are not visited by the truck correspond to self-loops in the solution. This idea
is inspired by Yang et al. (2004). Constraints (4.7) [for the first visit] and (4.6) [for subsequent visits] describe the arrival times of the truck at the stations which must be within the allowed horizon (4.8), and (4.9) allows for the necessary amount of time per station visit. (4.10) and (4.11) update the bikes in the truck after each visit, and (4.12) and (4.13) enforce that the truck contains its initial load of bikes at the beginning of its first visit. Constraints (4.14) ensure that the bikes in the truck do not exceed the truck capacity, while (4.15) and (4.16) are used to enforce load limits after the last truck visit. Due to (4.17) and (4.21), bikes are rebalanced only if they (or the docks) are unused at that moment to not interfere with customer flows. Each pick-up and drop-off can either be of positive or neutral value (4.19), (4.23), and they are nonzero only if the truck actually visits the station (4.20), (4.24). Finally, the positive value resources are bounded by the needed bikes and docks per station (4.18), (4.22). We can also add the following two constraints to obtain a tighter formulation:

\[
\sum_{i \in S_0 \cup S} dist_{ij} \cdot x_{ij} + \sum_{i \in S} dur_i \leq \tau_{\text{period}} \tag{4.27}
\]

\[
\sum_{i \in S} \tau_{\text{action}} \cdot PU_i + \sum_{i \in S} \tau_{\text{action}} \cdot DO_i \leq \tau_{\text{period}} \tag{4.28}
\]

Further details for model (DR) are provided in Appendix A.1.3.

### 4.3.1 Planning over Multiple Rebalancing Periods

Note that in (DR) we are planning over a single rebalancing period of length $\tau_{\text{period}}$. We will now show how Dynamic Rebalancing model (DR) can be generalized to Multiperiod Dynamic Rebalancing (MDR) to generate rebalancing plans considering jointly more than one rebalancing periods. The main ideas consist of introducing the constraints of (DR) for each rebalancing period of length $\tau_{\text{period}}$, with additional constraints to ensure feasible transition of the truck from one period to the next, as well as overall feasible bike pick-ups and drop-offs if a station is visited in more than one period.

More specifically, each rebalancing period is indexed by $r$, which takes values
between 1 and \( R_P \) (the number of rebalancing periods for which we are planning, with \( R_P \geq 2 \)). For this multi-period version of the formulation, we introduce the constraints of (DR) for each period, using the extra index \( r \) for all variables in order to specify the corresponding rebalancing period. For instance, binary variable \( x^r_{ij} \) equals 1 if the truck travels directly from \( i \) to \( j \) during rebalancing period \( r \). We also introduce new binary variables \( y^r_i \) to facilitate the transition between consecutive rebalancing periods. In particular, \( y^r_i \) equals 1 if the transition from period \( r \) to \( r + 1 \) takes place at station \( i \).

(MDR):

\[
\begin{align*}
\text{max} & \quad \sum_{i \in S, r} (PU_i^r + DO_i^r) - \gamma \sum_{i \in S_0} \sum_{j \in S, r} dist_{ij} x^r_{ij} - \delta \sum_{i \in S, r} (PU_i^r + DO_i^r) \\
\text{s.t.} & \quad \sum_{j \in S_0} x^r_{ij} + y^r_i = 1 \quad \forall i \in S, r \\
& \quad \sum_{j \in S_0} x^r_{ji} = 1 \quad \forall i \in S, r = 1 \\
& \quad \sum_{j \in S_0} x^r_{ji} + y^{r-1}_i = 1 \quad \forall i \in S, r > 1 \\
& \quad \sum_{i \in S_0} x^r_{0i} = 1, \quad \sum_{i \in S_0} x^r_{i0} = 0 \quad r = 1 \\
& \quad \sum_{i \in S_0} x^r_{0i} = 0, \quad \sum_{i \in S_0} x^r_{i0} = 1 \quad r = R_P \\
& \quad x^r_{00} = 1, \quad x^r_{0i} = 0, \quad x^r_{i0} = 0 \quad \forall i, 1 < r < R_P \\
& \quad t^r_i \geq t^r_j + \text{dur}^r_i + \text{dist}^{ij} \cdot x^r_{ij} - M \cdot (1 - x^r_{ij}) \quad \forall i, j \in S, i \neq j, r \\
& \quad t^r_i \geq \text{dist}_{0i} \cdot x^r_{0i} + \tau_0 \quad \forall i \in S, r = 1 \\
& \quad t^r_i \leq r \cdot \tau_{\text{period}} \quad \forall i \in S, r \\
& \quad \text{dur}^r_i \geq \tau_{\text{station}} \cdot (1 - x^r_{ii}) + \tau_{\text{action}} \cdot (PU_i^r + DO_i^r) \quad \forall i \in S, r \\
& \quad b^r_j \geq b^r_i + PU_i^r - DO_i^r - M \cdot (1 - x^r_{ij}) \quad \forall i, j \in S, i \neq j, r \\
& \quad b^r_j \leq b^r_i + PU_i^r - DO_i^r + M \cdot (1 - x^r_{ij}) \quad \forall i, j \in S, i \neq j, r \\
& \quad b^r_i \geq \zeta_0 \cdot x^r_{0i} \quad \forall i \in S, r = 1 \\
& \quad b^r_i \leq C_T - (C_T - \zeta_0) \cdot x^r_{0i} \quad \forall i \in S, r = 1 \\
& \quad b^r_i \leq C_T \quad \forall i \in S, r 
\end{align*}
\]
\[ b_{i}^{Tr} + PU_{i}^{r} - DO_{i}^{r} \leq C^{T} \quad \forall i \in S, r \] (4.45)
\[ b_{i}^{Tr} + PU_{i}^{r} - DO_{i}^{r} \geq 0 \quad \forall i \in S, r \] (4.46)
\[ PU_{i}^{r} \leq \text{UNUSED\_BIKES}(r, i) \quad \forall i \in S, r \] (4.47)
\[ PU_{i}^{r} \leq \text{DOCKS\_NEEDED}(r, i) \quad \forall i \in S, r \] (4.48)
\[ PU_{i}^{r} = PU_{i}^{r} + PU_{i}^{or} \quad \forall i \in S, r \] (4.49)
\[ PU_{i}^{r} \leq M \cdot (1 - x_{i}^{r}) \quad \forall i \in S, r \] (4.50)
\[ DO_{i}^{r} \leq \text{UNUSED\_DOCKS}(r, i) \quad \forall i \in S, r \] (4.51)
\[ DO_{i}^{r} \leq \text{BIKES\_NEEDED}(r, i) \quad \forall i \in S, r \] (4.52)
\[ DO_{i}^{r} = DO_{i}^{r} + DO_{i}^{or} \quad \forall i \in S, r \] (4.53)
\[ DO_{i}^{r} \leq M \cdot (1 - x_{i}^{r}) \quad \forall i \in S, r \] (4.54)
\[ t_{i}^{r} + \text{dur}_{i}^{r} \leq t_{i}^{r+1} + M \cdot (1 - y_{i}^{r}) \quad \forall i \in S, r < RP \] (4.55)
\[ t_{i}^{r} + \text{dur}_{i}^{r} \geq t_{i}^{r+1} - M \cdot (1 - y_{i}^{r}) \quad \forall i \in S, r < RP \] (4.56)
\[ b_{i}^{Tr} + PU_{i}^{r} - DO_{i}^{r} \leq b_{i}^{Tr} + M \cdot (1 - y_{i}^{r}) \quad \forall i \in S, r < RP \] (4.57)
\[ b_{i}^{Tr} + PU_{i}^{r} - DO_{i}^{r} \geq b_{i}^{Tr} + M \cdot (1 - y_{i}^{r}) \quad \forall i \in S, r < RP \] (4.58)
\[ \sum_{r'=r_1}^{r_2} PU_{i}^{r'} \leq \text{ALL\_UNUSED\_BIKES}(r_1, r_2, i) \quad \forall i \in S, r_1 < r_2 \] (4.59)
\[ \sum_{r'=r_1}^{r_2} PU_{i}^{r'} \leq \text{ALL\_DOCKS\_NEEDED}(r_1, r_2, i) \quad \forall i \in S, r_1 < r_2 \] (4.60)
\[ \sum_{r'=r_1}^{r_2} DO_{i}^{r'} \leq \text{ALL\_UNUSED\_DOCKS}(r_1, r_2, i) \quad \forall i \in S, r_1 < r_2 \] (4.61)
\[ \sum_{r'=r_1}^{r_2} DO_{i}^{r'} \leq \text{ALL\_BIKES\_NEEDED}(r_1, r_2, i) \quad \forall i \in S, r_1 < r_2 \] (4.62)
\[ \sum_{i \in S, j \in S} \text{dist}_{ij} x_{ij}^{r} + \sum_{i \in S} \text{dur}_{i}^{r} \leq \tau_{\text{period}} \quad \forall r \] (4.63)
\[ \sum_{i \in S} \tau_{\text{action}} \cdot PU_{i}^{r} + \sum_{i \in S} \tau_{\text{action}} \cdot DO_{i}^{r} \leq \tau_{\text{period}} \quad \forall r \] (4.64)
\[ x_{ij}^{r} \in \{0, 1\} \quad \forall i, j \in S_0, r \] (4.65)
\[ y_{i}^{r} \in \{0, 1\} \quad \forall i \in S, r \] (4.66)
\[ t_{i}^{r}, \text{dur}_{i}^{r}, b_{i}^{Tr}, PU_{i}^{r}, PU_{i}^{r}, PU_{i}^{or}, DO_{i}^{r}, DO_{i}^{r}, DO_{i}^{or} \in \mathbb{R}_+ \quad \forall i \in S, r \] (4.67)
Constraints \((4.30)\), \((4.31)\) and \((4.32)\) specify the truck routing between the stations: the truck can travel from station to station or transition to the next rebalancing period while remaining at the same station. We have one dummy node for the truck per period, which has an outgoing arc at period 1 \((4.33)\) that denotes the start of the route, an incoming arc at period \(RP\) \((4.34)\) denoting the end of the route, and corresponds to self-loops in all intermediate periods \((4.35)\). Constraints \((4.36)\) to \((4.54)\) and \((4.63)\) to \((4.64)\) are directly derived from (DR) by introducing one such constraint per rebalancing period \(r\). Constraints \((4.55)\) to \((4.58)\) ensure that the state of the truck (load and arrival time) remains consistent between consecutive rebalancing periods. Finally, constraints \((4.59)\) to \((4.62)\) impose limits on the resources: a bike could be available for pick-up/drop-off during more than one rebalancing periods, but if this action takes place during one of the periods, it shouldn’t take place again in any other period. We introduce such constraints for any pair \(r_1, r_2\) with \(1 \leq r_1 < r_2 \leq RP\), but since \(RP\) is small in our application this does not lead to a large number of constraints. Further details on (MDR) are provided in Appendix A.2.1.

4.4 Optimization with Station Groups

For each cluster, neighboring stations can sometimes be handled together by consecutive truck visits. The goal of this section is to group stations that are very close to each other, in order to further reduce the size of the network.

4.4.1 Leader Stations

We introduce a binary variable \(x_i\) for each station \(i\) and aim to allow only stations with \(x_i = 1\) to potentially be visited by a truck. We name these stations leader stations. The goal is to minimize the number of leader stations, minimizing in that way the visit candidates of the truck. At the same time, we need to ensure that each station has at least one leader nearby, in other words, for each station the truck can visit and
rebalance bikes in its neighborhood. Let $S_R$ denote the stations that currently need rebalancing, and $N_{il}$ the $l$-neighborhood of $i$, where distance $l$ is a parameter of the model. The formulation follows.

$$\min \sum_{i} x_i \quad (4.68)$$

s.t. $\sum_{j \in N_{il}} x_j \geq 1 \quad \forall i \quad (4.69)$

$x_i \geq x_j \quad \forall i \in S_R, j \notin S_R, j \in N_{il} \quad (4.70)$

$x_i \in \{0, 1\} \quad \forall i \quad (4.71)$

This is a variation of the set covering problem: According to constraint (4.69) each station must be covered by/be in the neighborhood of at least one leader (the station itself could be that leader). Constraint (4.70) gives priority to stations that need rebalancing: If it is possible to select a leader between a station that needs rebalancing and one that does not, then the one that faces the problem will be prioritized. As a result, a station that does not currently require rebalancing can become a leader station only if none of its neighbors are in need of rebalancing.

After the leaders have been selected, each station is assigned to its closest leader, creating in that way mini-groups of stations, each of which is represented by its leader. An illustration is provided in Figure 4-3. The capacity and the rebalancing needs of the leader are then given by the total capacity and rebalancing needs of the stations in its mini-group. The advantage of this approach is that it allows to consider only a small subset of stations, facilitating in that way the optimization. At the same time, it does not ignore the remaining stations, since the problems of all stations are taken into account.

### 4.4.2 Rebalancing Only Leader Stations

Having identified a set of leader stations, a natural next step is to solve (DR) using only the leader stations. In this case, each leader station will have the total capacity, unused resources and rebalancing needs of its mini-group. An important drawback of
this suggestion is that it implies that a visit to the leader station would momentarily solve the problem for many of the stations of its mini-group as well. This ignores the traveling time that is required to move from one station to the other, which might be close, but their distance is still not negligible. In the next section, we develop a method that can take information of each group into account in the model.

4.5 Optimization with Partial Group Information

Let us now propose an approach which will be demonstrated by considering one of the mini-groups as an example. First, we simplify our problem by assuming temporarily that all pairs of stations of the group require the same traveling time $\tau_{\text{travel}}$. We can make this assumption, because all stations are close to each other, so their traveling times do not differ significantly. An important observation is that the work performed within a mini-group depends clearly on the time the truck spends in that mini-group. A short period of time might only be enough to visit the leader station, while longer visit durations allow serving more of the stations of the group. We will express now the performance of the truck as a function of the time spent inside the mini-group, with the help of a greedy routing algorithm.
4.5.1 Greedy Routing Algorithm

Consider first only the stations that require bikes to be picked up, i.e. they need extra empty docks to satisfy their incoming demand. The truck is currently at the leader station of the group. Aiming to maximize the number of picked-up bikes, the truck starts by picking up bikes, if any, from the leader station. Then, since all traveling times between stations are assumed to be the same and equal to $\tau_{\text{travel}}$, the truck will greedily maximize its performance if it visits the station with the largest number of bikes to be picked up: by “paying” the price of $\tau_{\text{travel}}$ to reach a station, it gets the greatest possible “reward”, i.e. largest amount of bikes.

Example 4.5.1. Consider the mini-group in Figure 4-4 which consists of five stations and let station 3 be the selected leader. Assume the truck is currently at station 3, there are no constraints on its capacity, the traveling time between any pair of stations is $\tau_{\text{travel}} = 3$ minutes, the overhead time for each station visit (parking, etc.) is $\tau_{\text{station}} = 2$ minutes, and picking up each bike requires $\tau_{\text{action}} = 1$ minute. Following the greedy routing algorithm described before, we get a solution where 5 bikes are picked-up from station 3, then 4 from station 5, and finally 3 from station 1.

\[
\begin{array}{lcl}
+2 & & \\
-3 & 2 & -5 \\
1 & & 3 \\
-4 & 5 & 4 +3
\end{array}
\]

Figure 4-4: A mini-group of stations. The number by each station indicates the number of bikes that should be picked-up (negative sign) or dropped-off (positive sign).

Proposition 4.5.1. For a given rebalancing period length $T$, the greedy routing algorithm within a group of stations is optimal if we ignore the vehicle’s load and capacity, the integrality of resources and assume equal traveling times between any two stations of the group.
Proof. In this case, the routing among a group of stations can be reduced to the fractional knapsack problem. The length of the rebalancing period $T$ can be thought of as the capacity of the knapsack. Each station $i$ is an item with “value” equal to the resources $r_i$ (bikes or docks) that can be rebalanced and “weight” equal to the time required to travel to the station and perform the rebalancing $\tau_{\text{travel}} + \tau_{\text{station}} + r_i \cdot \tau_{\text{action}}$. The goal is to maximize the redistributed resources in the period length $T$. Hence, given that we allow fractional rebalancing of resources, the greedy algorithm selects the stations in decreasing order of value over weight, so the solution is optimal. ∎

In the above, we have assumed that the value (amount of redistributed resources) is being obtained equally distributed over the total time for traveling and picking up or dropping off the bikes. In reality, no value is acquired during the traveling time among stations, but this is a simplifying assumption that allows us to model redistributed resources as piecewise linear concave functions of rebalancing time, as will be demonstrated in the following section.

### 4.5.2 Group Modeling Using Piecewise Linear Functions

Proposition 4.5.2. The amount of redistributed resources in the greedy algorithm can be modeled as a piecewise linear concave function of the time the vehicle spends in the group of stations.

Proof. The greedy algorithm selects the stations in order of decreasing ratio $\frac{r_i}{\tau_i}$ of redistributed resources $r_i$ over required time $\tau_i = \tau_{\text{travel}} + \tau_{\text{station}} + r_i \cdot \tau_{\text{action}}$. Since the ratio $\eta_i = \frac{r_i}{\tau_i}$ is decreasing in the selections of the algorithm, so are the slopes of $r_i$ as a function of $\tau_i$ if we approximate them using functions $r_i(\tau_i) = \eta_i \tau_i$. Hence, the amount of redistributed resources is a concave piecewise linear function of the time the vehicle spends in the group. ∎

Example 4.5.2. Using again the example of Figure 4-4, we have three stations that need bikes to picked-up, so the piecewise linear function that corresponds to the pick-ups will consist of three pieces (for stations 3, 5 and 1 respectively). The truck is already at station 3 and we need to pick up $r_3 = 5$ bikes, so the visit will last $\tau_3 =$
Figure 4-5: Redistributed resources as a function of time. With blue color the piece-wise linear concave functions. With red the results taking into account that during traveling the number of resources remains constant (function parallel to x axis during traveling).

\[ r_3 \cdot \tau_{\text{action}} = 5 \text{ minutes.} \]

For station 5, \( r_5 = 4 \) bikes need to be picked up and the time required is \( \tau_5 = \tau_{\text{travel}} + \tau_{\text{station}} + r_5 \cdot \tau_{\text{action}} = 9 \). Finally, for station 1 and the pick-up of \( r_1 = 3 \) bikes: \( \tau_1 = \tau_{\text{travel}} + \tau_{\text{station}} + r_1 \cdot \tau_{\text{action}} = 8 \). As a result, the truck will visit station 3 for \( t \in [0, 5] \), station 5 for \( t \in [5, 14] \) and station 1 for \( t \in [14, 22] \). The corresponding line slopes are given by \( \frac{r_i}{\tau_i} \), which in this case are \( \frac{5}{5} = 1 \) for station 3, \( \frac{4}{9} = 0.444 \) for station 5, and \( \frac{3}{8} = 0.375 \) for station 1, which are decreasing and the function is concave, as it is also illustrated in Figure 4-5.

In the previous example, the focus was placed only on bike pick-ups. However, it should be obvious that the same results hold for bike drop-offs.

**Corollary 4.5.1.** The amount of redistributed resources over time for each group can be formulated using linear constraints. If \( a_p \) and \( b_p \) denote the slope and intercept vector (with elements \( a_{p,l} \) and \( b_{p,l} \) for each line segment \( l \)) that correspond to the piecewise linear concave function for the pick-ups, and respectively \( a_d \) and \( b_d \) for the drop-offs, and, finally, \( \text{durPU}_i \) is the duration of the visit in group \( i \) where the truck is picking up bikes and \( \text{durDO}_i \) is the duration of the visit where the truck is dropping off bikes, then the number of bikes \( \text{PU}_i \) that can be picked up and \( \text{DO}_i \) that can be dropped off are the largest numbers that satisfy:

\[
\text{PU}_i \leq \min_l (a_{p,l}^i \cdot \text{durPU}_i + b_{p,l}^i) \quad \forall i
\]
\[ DO_i \leq \min \left( \alpha^I_{d,l} \cdot \text{dur} \cdot DO_i + \beta^I_{d,l} \right) \quad \forall i \] (4.73)

### 4.5.3 Global Rebalancing

Having expressed the amount of redistributed resources for each group with respect to the duration of its visit, we can now introduce the rebalancing optimization using partial group information. The basis of the optimization is still model (DR) as described in Section 4.3, which is now applied only to the leader stations. The main difference is that we introduce for each group constraints (4.72) and (4.73) to model (DR), so it now performs dynamic rebalancing with partial information and is therefore denoted by (DRPI). Appendix A.1.1 presents in more details the changes in (DR) that lead to (DRPI).

![Figure 4-6: Partial group information is generated per group and provided to the model.](image)

The solution of (DRPI) produces a routing for each truck which specifies the order with which groups need to be visited. Moreover, it provides the number of bikes that need to be picked up and dropped off for each group, as well as an estimation of the time it will require. Note that picking up and dropping off the planned number of bikes is important, as these bikes might be accounted for in the resources needed...
for subsequent truck visits. Regarding the time that the actions per group actually require, there is slightly more flexibility: we have the possibility of delaying a group visit by a few minutes or performing it ahead of time. The next section concludes this method by describing how the exact routing within each group is obtained.

4.5.4 Final Retrieval of Within Group Routing

For the rebalancing within each group, the total number of bikes to be picked up and dropped off is provided by the solution of (DRPI) and now the exact routing needs to be retrieved. At this point, we remove the assumption that all pairs of stations in a group have the same distance, and consider their real distances. For the retrieval of the routing, one might use an adaptation of the greedy routing algorithm since the distances within each group are small and the routing is not very demanding in terms of time. The greedy routing algorithm selects the next station to be visited based on the largest ratio of bikes to be moved over the required time. This selection criterion can still be followed, but the load and the capacity of the truck need to be taken into account: if a truck is full, it cannot pick up more bikes, and similarly for dropping off when it is empty. Moreover, the total number of bikes to be picked up and dropped off per group is determined by the solution of (DRPI). Therefore, we adapt the selection step by considering the ratio of bikes moved over time but only for the number of bikes that is allowed based on the current load, the total capacity limits of the truck, and the (DRPI) solution. In case the final schedule includes visits that exceed the rebalancing period duration, we remove all visits with arrival time after the period end.

4.6 Computational Experiments

The results of this section are based on real-world data for Capital Bikeshare, Washington DC, for the time period May to September 2017. For each past trip, we have details about its origin, its destination, as well as its starting and ending times. We also have information about station outages, i.e. the starting and ending time, as
well as the station where it was observed, and its status - full or empty. Finally, the
design details of the system are also known: the location and the capacity for each
station, and the total number of bikes in the system. The data for past trips and
system information (Capital Bikeshare System Data (2017)), as well as the data for
outages history (Capital Bikeshare Tracker (2017)) are publicly available.

In this section, we present results of the proposed system rebalancing methods
using both synthetic and real-world datasets. Varying the structure on synthetic data
can offer good intuition on factors that influence algorithms’ performance. On the
other hand, using real-world datasets allows us to evaluate the methods in an actual
bike sharing system. For the implementations, we use Julia, a high-performance
language for numerical computing (Bezanson et al. (2017)), and JuMP, a modeling
language for mathematical optimization (Dunning et al. (2017)).

4.6.1 Experiments on Synthetic Data

All experiments are run on grid-structured instances of two types (see Figures 4-7
and 4-8). In both types, each group of stations occupies a single cell of the grid, but
Type I instances consist of a sparser grid with larger distance among groups, while
Type II instances are denser. We vary the sizes of the grid, the number of stations
per group, and the rebalancing requirement levels. In these experiments, we consider
a single rebalancing period, since it is sufficient to demonstrate the differences among
the various methods. We report the results and compare three methods: using the
MIP formulation (DR) directly on the set of stations (denoted as “S”), grouping using
only the leaders’ information in (DR) (denoted as “G/L”), and grouping with partial
information using (DRPI) (denoted as “G/PI”).

We consider instances with grid sizes $4 \times 4$ and $5 \times 5$, and each station group has
5 or 10 stations. Regarding the rebalancing needs, we consider one scenario where all
stations need the same amount of bikes to be picked up or dropped off (homogeneous
needs), and a scenario where each group has two dominant stations with much larger
needs than the rest (heterogeneous needs). In both cases, each station is being labelled
independently as pick-up station or drop-off station with probability one half.
Parameter Selection

In the experiments that follow, all stations have capacity 20 and are initially half filled, the rebalancing period length is 100, and the truck has a capacity of 25 bikes. Each side of the grid cells is 1000 meters, and in Type I (sparse) instances the vertical and horizontal distance between cells is 1000 meters, while in Type II (dense) 200 meters. We assume the overhead time for each station visit (for parking, etc.) is $\tau_{\text{station}} = 1$ minute, the time for each bike pick-up or drop-off is $\tau_{\text{action}} = \tau_{\text{pickup}} = \tau_{\text{dropoff}} = 0.5$ minute, and the average traveling time between stations of the same group is $\tau_{\text{travel}} = 1$ minute since they are close by. In the scenario with homogeneous rebalancing needs, each station needs five bikes to be picked up or dropped off, and in the one with heterogeneous needs, two stations of each group need ten bikes to be picked up or dropped off and the rest just one. In the case where the MIP is applied directly on the stations without grouping (S), we report the results (including the optimality gap) after 5 minutes and 20 minutes of running time, in order to compare the tradeoff of the solution quality and the required time with the results of our methods.

Results

Tables 4.1 and 4.2 present the number of bikes that was redistributed successfully (or equivalently the number of additional served users) in each method within the rebalancing period length. We observe that grouping can in general achieve compa-
Table 4.1: Served users in the scenario of homogeneous rebalancing needs. Each station requires the same number of bikes to be picked up or dropped off. The first column describes the grid size and the stations per group for each instance. Three methods are compared: using the (DR) formulation directly on the set of stations (denoted as “S”), grouping using only the leaders’ information in (DR) (denoted as “G/L”), and grouping with partial information with (DRPI) (denoted as “G/PI”). G/PI and G/L get within seconds solutions that are better or comparable to solutions that S obtains after 20 minutes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4 × 4, 5</td>
<td>I (sparse)</td>
<td>80</td>
<td>(43.4%)</td>
<td>85</td>
<td>(32.6%)</td>
<td>86.31</td>
<td>81.80</td>
</tr>
<tr>
<td>4 × 4, 5</td>
<td>II (dense)</td>
<td>90</td>
<td>(28.7%)</td>
<td>95</td>
<td>(21.3%)</td>
<td>91.86</td>
<td>92.37</td>
</tr>
<tr>
<td>4 × 4, 10</td>
<td>I (sparse)</td>
<td>86.09</td>
<td>(46.7%)</td>
<td>95</td>
<td>(31.5%)</td>
<td>100</td>
<td>98.27</td>
</tr>
<tr>
<td>4 × 4, 10</td>
<td>II (dense)</td>
<td>82.86</td>
<td>(56%)</td>
<td>100</td>
<td>(25.1%)</td>
<td>101.16</td>
<td>101.45</td>
</tr>
<tr>
<td>5 × 5, 5</td>
<td>I (sparse)</td>
<td>0</td>
<td>−</td>
<td>80</td>
<td>(52%)</td>
<td>87.08</td>
<td>89.11</td>
</tr>
<tr>
<td>5 × 5, 5</td>
<td>II (dense)</td>
<td>57.03</td>
<td>(117%)</td>
<td>92.37</td>
<td>(31.6%)</td>
<td>93.17</td>
<td>92.61</td>
</tr>
<tr>
<td>5 × 5, 10</td>
<td>I (sparse)</td>
<td>89.45</td>
<td>(48.1%)</td>
<td>91.22</td>
<td>(41.8%)</td>
<td>95</td>
<td>100</td>
</tr>
<tr>
<td>5 × 5, 10</td>
<td>II (dense)</td>
<td>85</td>
<td>(55.7%)</td>
<td>95</td>
<td>(36%)</td>
<td>97.69</td>
<td>95.97</td>
</tr>
</tbody>
</table>

Table 4.2: Served users in the scenario of heterogeneous rebalancing needs. Each group has two stations with much larger rebalancing needs than the rest. The first column describes the grid size and the stations per group for each instance. Three methods are compared: using the (DR) formulation directly on the set of stations (denoted as “S”), grouping using only the leaders’ information in (DR) (denoted as “G/L”), and grouping with partial information with (DRPI) (denoted as “G/PI”). Incorporating partial group information leads to G/PI significantly outperforming G/L and getting better solutions than S within seconds of running time.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4 × 4, 5</td>
<td>I (sparse)</td>
<td>86</td>
<td>(41.2%)</td>
<td>87</td>
<td>(34.4%)</td>
<td>78</td>
<td>94.45</td>
</tr>
<tr>
<td>4 × 4, 5</td>
<td>II (dense)</td>
<td>106.63</td>
<td>(18.3%)</td>
<td>108</td>
<td>(15.2%)</td>
<td>86.81</td>
<td>105.22</td>
</tr>
<tr>
<td>4 × 4, 10</td>
<td>I (sparse)</td>
<td>79</td>
<td>(71.5%)</td>
<td>84</td>
<td>(57.8%)</td>
<td>65</td>
<td>80.67</td>
</tr>
<tr>
<td>4 × 4, 10</td>
<td>II (dense)</td>
<td>0</td>
<td>−</td>
<td>91</td>
<td>(45.7%)</td>
<td>65</td>
<td>110.27</td>
</tr>
<tr>
<td>5 × 5, 5</td>
<td>I (sparse)</td>
<td>0</td>
<td>−</td>
<td>86</td>
<td>(49.4%)</td>
<td>85.18</td>
<td>101.26</td>
</tr>
<tr>
<td>5 × 5, 5</td>
<td>II (dense)</td>
<td>0</td>
<td>−</td>
<td>107.17</td>
<td>(26.6%)</td>
<td>80</td>
<td>112.68</td>
</tr>
<tr>
<td>5 × 5, 10</td>
<td>I (sparse)</td>
<td>0</td>
<td>−</td>
<td>0</td>
<td>−</td>
<td>71.71</td>
<td>100.4</td>
</tr>
<tr>
<td>5 × 5, 10</td>
<td>II (dense)</td>
<td>10</td>
<td>(1301%)</td>
<td>10</td>
<td>(1295%)</td>
<td>72.72</td>
<td>108.89</td>
</tr>
</tbody>
</table>
rable or better solutions in faster running times; in particular, G/PI provides within seconds solutions that are better or comparable to solutions obtained after 20 minutes of running the MIP in S. The difference in performance becomes more significant as instances become larger, and manifests in a larger degree in Type I (sparse) instances. In these instances, the groups of stations are not adjacent, but there is some distance among them, and the traveling times among groups are larger. This seems to complicate the solution for the MIP in S, which at times needed more than five or twenty minutes to find a feasible solution, while G/PI had already found a good quality solution within seconds.

Comparison between grouping where only leader information is taken into account (G/L) and grouping with partial group information (G/PI) shows that solutions are comparable when all stations have equal rebalancing needs. On the other hand, the difference is significant when there are stations with much larger needs (dominant stations) than the rest. G/L cannot differentiate between this and the previous case; since within group routing and time per visit are not taken into account, only the aggregate number of bikes needed influences the MIP results. However, in G/PI, the piecewise linear functions that model the redistribution operations within each group guide the MIP solution to visit mostly the dominant stations. The reason is that the value of these visits (bikes redistributed over required time) is larger than visiting the remaining stations, where the travel time is not worth the amount of bikes to be picked up or dropped off in most groups. Overall, the extra information provided in G/PI produces solutions that outperform G/L, and also compare favorably with MIP results without grouping (S) while requiring significantly less running time.

4.6.2 Experiments on Real-World Data

In this section, we will use the past trips dataset provided by Capital Bikeshare, Washington DC, to evaluate our proposed methods. In the sections that follow, we compare multiple aspects that influence rebalancing, and more detailed results can be found in Appendix A.3. Unless noted otherwise, we consider the rebalancing that takes place each day until noon, which includes the preparation for the morning rush
hour, the rush hour itself, as well as the reorganization that is required when it is over as a result of the increased user traffic.

Parameter Selection

In the following experiments, each rebalancing period has length of 30 minutes, the overhead time for each station visit is $\tau_{\text{station}} = 1$ minute and the time for each bike pick-up or drop-off is $\tau_{\text{action}} = \tau_{\text{pickup}} = \tau_{\text{dropoff}} = 0.5$ minute. In the leader selection, we set the maximum distance between stations of the same neighborhood to be 1000 meters, and the travel time between stations of the same group to be $\tau_{\text{travel}} = 1.14$ minutes, as this is the average value we obtained experimentally. In order to determine the rebalancing needs and the unused resources of each station, which then guide the actions of the trucks, we consider the user demand from the current period and up to five periods in the future, as described in Section 4.1. For these experiments there are 8 trucks, each of capacity 25 bikes, the stations start the day half full, and the MIP solver time limit for (DR) and (DRPI) is 1 minute per cluster and per time period.

Method Comparisons

Similarly to the synthetic instances, let us begin with comparing the three different methods: using (DR) on the set of stations without grouping (denoted as “S”), using (DR) to the set of leaders without incorporating partial information (denoted as “G/L”), and using (DRPI) which considers the set of leaders with partial information (denoted as “G/PI”). Results for the rebalancing gain for the three methods are presented in Figure 4-9, and more detailed results are available in Appendix A.3.

The rebalancing gain refers to the additional number of users that the system can serve compared to the scenario where no rebalancing takes place. We observe that in all instances G/L and G/PI outperform S, and, with a few exceptions, G/PI outperforms G/L. In particular, G/L achieves an 6.78% increase of the rebalancing gain compared to S (averaged over all instances), and G/PI an 11.36% increase in average. Moreover, despite obtaining better results, G/L and G/PI require 5 to 8 times less total running time compared to S.
Figure 4-9: Rebalancing gain for the (DR) formulation applied directly on the set of stations (denoted as “S”), (DR) applied on the group leaders (denoted as “G/L”), and (DRPI) on group leaders with partial information (denoted as “G/PI”), over 10 instances. G/PI obtains the largest rebalancing gain among the three methods.

Varying the Rebalancing Planning Horizon

As explained in Section 4.3.1, we can generalize (DR) and (DRPI) into multi-period versions (MDR) and (MDRPI), where we jointly plan over more than one rebalancing periods. Figure 4-10 illustrates the results of (DRPI), where we consider a single rebalancing period ($RP = 1$), and (MDRPI) that considers two rebalancing periods ($RP = 2$). Introducing a second rebalancing period in the action planning leads to a further improvement of 1.78% compared to G/PI with $RP = 1$, which brings the average improvement over S (from Figure 4-9) to 13.36%.

Benefits of Neutral Value Rebalancing Actions

In both models (DR) and (DRPI), we allow for rebalancing actions of neutral value $PU_i$ and $DO_i$, where essentially we can pick up and drop off bikes from stations that do not currently need rebalancing in order to better serve other stations that need the resources. We now evaluate the effect of these neutral value rebalancing actions, by comparing the rebalancing gain with the case where neutral value actions are not allowed. In this scenario, trucks can only transfer bikes between stations with
Figure 4-10: Rebalancing gain for G/PI when the planning horizon consists of a single rebalancing period ($RP = 1$) and (DRPI) is applied, or two ($RP = 2$) rebalancing periods, in which case its multi-period version (MDRPI) is used. The multi-period planning horizon leads to a small improvement of the average gain.

imminent rebalancing needs, and this is achieved by enforcing $PU^o_i = 0$ and $DO^o_i = 0$, $\forall i \in S$ in the model. The results are presented in Figure 4-11. Prohibiting neutral value rebalancing actions leads to an average decrease of 55.3% of the rebalancing gain compared with G/PI with $RP = 1$. The main reason is that at each cluster at any time the number of bikes that need to be picked up might differ significantly from the number of bikes that need to be dropped off. As a result, auxiliary visits to neighboring stations that do not currently need rebalancing help the truck acquire the necessary resources to satisfy more rebalancing needs.

Comparison with Static Rebalancing

We conclude the experimental analysis by also considering the case of static rebalancing, where redistribution operations take place overnight to prepare the stations for the demand of the following day. In order to determine the initial state of the stations at the beginning of the day, we use our model for user trips (UT) from Section 3.1. Without specifying the initial state for each station, (UT) provides the initial number of bikes that optimizes the objective function. We can think of this as the ideal static rebalancing that delays the most the time when the first rebalancing need
Figure 4-11: Rebalancing gain for G/PI (model (DRPI)) compared to (DRPI) without neutral value rebalancing actions. Forbidding neutral value rebalancing actions leads to a significant decrease of the rebalancing gain.

appears. We do not consider the specifics of how this static rebalancing is performed, but wish to evaluate the extent to which our dynamic rebalancing can improve further the service level of the system. Since static rebalancing benefits the morning periods (by setting each station to its ideal state) significantly more than the evening ones, in order to properly evaluate its effects, we need to consider the rebalancing gain throughout the whole day for the experiments that follow. The results are presented in Figure 4-12, which shows the rebalancing gain (over the case with no rebalancing and initially half-full stations) when there is only static rebalancing, and when dynamic rebalancing is performed as well. Performing dynamic rebalancing increases the rebalancing gain by 42.88% in average over all instances.
Figure 4-12: Rebalancing gain for ideal initial state of stations without and with dynamic rebalancing. Dynamic rebalancing improves significantly the rebalancing gain, even when the stations start the day with their ideal bike inventory.
Chapter 5

Demand Estimation

While BSS operators record a variety of data about the system such as outages and past trips data, an important component remains missing; *lost user demand*. This demand occurs whenever a user is unable to complete a desired trip due to either the absence of bikes at the origin station, or the absence of docks at the destination, and they decide to leave the system without using it. This behavior may cause inadequate planning on the system operator’s part, since this user intention is never recorded, hence leading to instances of underestimating the actual demand.

Another natural consequence of encountering empty and full stations is the *shift of demand* towards nearby locations. Overlooking this event results in overestimating the number of unserved customers, since some of them have actually been satisfied from other locations. Moreover, this phenomenon causes an artificial demand increase for stations at the receiving end of this shift. This will manifest in a greater need of resources, misleading the rebalancing operations. Therefore, we now focus on identifying the extent of this lost and relocated demand, in order to determine the actual demand of the system.

5.1 Observed Demand

The *observed demand* consists of all trips that are completed in a system. For each pair of stations \((i, j)\) and time periods \((t, t')\), let \(d_{i,j}^{t,t'}\) denote the number of users
picking up a bike from station $i$ at time $t$ and returning it to station $j$ at time $t'$. The observed demand is known to the BSS operators, since the successful trips are recorded in the system data. Using this information, we can also calculate the observed outgoing (resp. incoming) demand from station $i$ at time $t$, which is defined as the total number of users who left (resp. arrived at) station $i$ at time $t$, $d_{out,i}^t = \sum_{j,t'} d_{i,j}^{t,t'}$ (resp. $d_{in,i}^t = \sum_{j,t} d_{j,i}^{t',t}$).

5.2 Lost Demand

Lost demand refers to the users who failed to perform a trip due to lack of bikes at their origin station or lack of docks at their desired destination. For each pair of stations $(i,j)$ and periods $(t,t')$, we use $ld_{i,j}^{t,t'}$ for the number of users that failed to go from station $i$ at time $t$ to station $j$ at $t'$ due to lack of resources at $i$ or at $j$. The lost outgoing demand $ld_{out,i}^t$ and lost incoming demand $ld_{in,i}^t$ are defined similarly to the previous section, but using the lost demand instead of the observed one.

In the following analysis, we focus on the lost outgoing demand, but the same methods can be applied for the estimation of the lost incoming demand. Consider a station $i$ which is empty from time $t_s$ to time $t_e$ on a particular day. Naturally, there is no observed outgoing demand during that time. However, there is no way to discover with certainty whether this is because there are no users wanting to depart from that station or whether the lack of bikes prohibits any users to do so. It turns out that the typical behavior of the station that time of the day (average station behavior) as well as the behavior around the interval when bikes are available (daily demand trends) can offer some intuition.

5.2.1 Average Station Behavior

The usual station behavior can be used as an indicator of the number of users that wish to depart from the station. This is based on the assumption that days do not differ much from one another, as well as the fact that a station is not empty every day at the same time. For the outage interval $[t_s, t_e]$ of station $i$, consider the average
observed demand at \([t_s, t_e]\) at \(i\) on the days where \(i\) is not empty during \([t_s, t_e]\). Let that average demand be denoted by \(\text{AVG}_{ld_{out,i}}\), alternatively referred to as the average station behavior.

### 5.2.2 Daily Demand Trends

Another aspect to be considered is the daily trend of the demand. There are days that are slower, for example due to weather conditions, and others that are busier. By examining the user behavior right before and after the interval, we can get an estimate of the interpolated demand during the outage.

**Definition 5.2.1.** We define the outgoing demand rate \(r_{out,i}(t_1, t_2)\) as the average number of users per minute that want to depart from station \(i\) during the interval \([t_1, t_2]\).

\[
    r_{out,i}(t_1, t_2) = \frac{1}{t_2 - t_1} \cdot \sum_{t=t_1}^{t_2} d_{out,i}^t
\]  

(5.1)

We then proceed to provide an estimate for the demand rate \(r_{out,i}(t_s, t_e)\) for the outage interval \([t_s, t_e]\), by using the demand rate the hour right before and after this interval. In particular:

\[
    r_{out,i}(t_s, t_e) = \frac{1}{2} [r_{out,i}(t_s - 60, t_s) + r_{out,i}(t_e, t_e + 60)]
\]  

(5.2)

Assuming that there is no outage during the intervals \([t_s - 60, t_s]\) and \([t_e, t_e + 60]\), this can be calculated using (5.1). The opposite case is addressed later in this section. Then, an estimate of the lost demand based on the trends \(\text{TREND}_{ld_{out,i}}\) for the outage interval \([t_s, t_e]\) can be obtained as:

\[
    \text{TREND}_{ld_{out,i}} = r_{out,i}(t_s, t_e) \cdot (t_e - t_s)
\]  

(5.3)

**Outage duration.** One assumption of this approach is that demand does not change drastically in a very short amount of time. Therefore, a well-founded concern is whether it can work well when the outage intervals are long, for example lasting a
few hours. Computational experiments based on real data reveal that this is not an issue since the vast majority of outages last less than thirty minutes, as is illustrated in Figure 5-1, while many of the longer lasting outages take place during the night, where demand is already negligible. Note here that empty station intervals during low operation periods such as nighttime are not considered indicators of lost demand and are not taken into account.

**Artificial outage endings.** Consider a user returning their bike at an empty station that is very soon picked up by another customer. This action will cause the outage interval to be split in two parts, when essentially it is the same event. Thus, before estimating the lost demand, we merge outage intervals that are a few minutes apart. Any recorded users between the initial intervals are then deducted from the estimated lost demand of the merged interval.

**Lack of information around the outage.** We discussed how the outage demand rate can be calculated based on the demand rates of the hours preceding and following the event. In case that we do not have information about the complete hour due to another outage, we can compute the demand rate based on a shorter time interval or look slightly further in the future or past, where information becomes available.
5.2.3 Lost Demand Estimation

The lost demand estimate is calculated as a convex combination of the estimates obtained by the average station behavior and the daily demand trends.

\[ ld_{out,i}^t = \lambda \cdot \text{AVG}_{ld_{out,i}} + (1 - \lambda) \cdot \text{TREND}_{ld_{out,i}} \]  \hspace{1cm} (5.4)

A data-driven approach is followed in order to select the value of \( \lambda \) that minimizes the MSE of the estimated demand. More details and results are presented in the Section 5.4 as part of the computational experiments.

The main characteristic of this approach is that we take into account both the average demand of the station as well as the specific demand trends of the day that encountered the outage. Since not all days are the same, by using information from the particular day, as well as focusing on the time periods that are closer to the outage interval, we can better estimate the system behavior during the outage, as is also illustrated in experiments on real-world datasets in Section 5.4.

This method implicitly considers various factors that influence biking. For instance, estimated lost demand in rainy days will probably be lower, since the traffic volume is generally reduced during these days. On the other hand, for outages that emerge in the middle of a rush hour, if the preceding observed demand is large, this will be captured in the estimation of the lost demand as well. Having calculated the number of unserved customers for each outage event, we assume that their arrival times are uniform over the outage interval and generate the lost demand trips.

5.3 Shifted Demand

The next step is to identify the extent of relocated demand in the system resulting from station outages. Let \( E_t \) denote the set of stations \( i \) that are empty at time \( t \). For each station \( i \), we define its \( l \)-neighborhood \( N_{i,l} \) as the set of stations whose distance from \( i \) is at most \( l \):

\[ N_{i,l} = \{ j | \text{distance}(i, j) \leq l \} \]  \hspace{1cm} (5.5)
and its active $l$-neighborhood at time $t$ as the subset of $N_{i,t}$ that have the resources to serve relocated demand from station $i$ at time $t$ (i.e. are not empty in the case of outgoing demand):

$$N_{i,t}^l = \{ j | j \in N_{i,t}, j \notin E_i \}$$  \hspace{1cm} (5.6)

**Shifting probability**

We introduce a model for the probability of a user shifting to a different location. We assume that there is some limit $l_{\text{max}}$ on the distance that users are willing to walk. So, if there is no station with available bikes close enough, users just leave the system without using it. In case such a station exists, users select to walk with probability $p$. More specifically, for station $i$ and time $t$, the probability of shifting is given by:

$$P_t^i(\text{user shifts}) = \begin{cases} 
0 & \text{if } N_{i,t,l_{\text{max}}} = \emptyset \\
p & \text{otherwise}
\end{cases}$$  \hspace{1cm} (5.7)

**Shifting coefficients**

If $|N_{i,t,l_{\text{max}}}^t| > 1$, the user has more than one alternative stations to walk to. In that case, they select each station with probability decreasing with respect to its distance. In particular, out of the customers that have selected to walk from station $i$ at time $t$, the probability of them going to station $j$ is given by:

$$q_{ij}^t = P_t^i(\text{goes to } j | \text{user shifts}) = \frac{1}{Z_i^t \cdot \text{distance}(i,j)}, \quad \forall i, t, j \in N_{i,t,l_{\text{max}}}$$  \hspace{1cm} (5.8)

where $Z_i^t = \sum_{j \in N_{i,t,l_{\text{max}}}^t} 1/\text{distance}(i, j)$ is a scaling factor, so that the coefficients sum to 1.

**Shifting distance multiplier**

In this model, we assume that users can walk to any station with available resources in the $l_{\text{max}}$ radius. However, when some of the candidate stations are really far compared to the others, a rational user will not consider them when selecting a station to walk
to. In order to include this behavior in the model, assume the closest non-empty station to station \(i\) at time \(t\) is at distance \(r_{min,i}^t\). A user will then walk only to stations up to distance \(\alpha \cdot r_{min,i}^t\), where \(\alpha \geq 1\) is a parameter of the model called \textit{shifting distance multiplier}. If \(\alpha = 1\), a user can only go to their closest available station, while if \(\alpha = \infty\) a user can go to any station of the network within the allowed walking limit \(l_{max}\). The integration of the shifting distance multiplier to the model we described so far requires simply the replacement of \(l_{max}\) with \(\min(l_{max}, \alpha \cdot r_{min,i}^t)\) in \(N_{i,t,l_{max}}\) in equations (5.7) and (5.8).

In this model, users only walk to stations that currently have resources available. This assumes that users have a prior knowledge of the state of each station, which is true in most real-world systems since they offer real-time tracking of the stations that can inform users of the availability of bikes at each one of them.

In order to compute the value of \(p\), we formulate the linear program \textit{Shifted Demand} (SD). For each station \(i\) that is not empty at time \(t\) and has demand \(d_{out,i}^t > 0\), we introduce a variable \(x_i^t\) representing the actual outgoing demand that was originally intended for that station, i.e. is not the result of customer movements from nearby empty stations. We also define for each station the average value of \(x_i^t\) over the same time of all remaining days where \(i\) is not empty, denoted by \(\bar{x}_i(t)\). The model formulation follows.

\[
\text{(SD)} : \quad \min \sum_{i,t} |x_i^t - \bar{x}_i(t)| \tag{5.9}
\]

s.t. \[
d_{out,i}^t = x_i^t + \sum_{j : i \in N_{j,t,l_{max}}} p \cdot q_{ji}^t \cdot l_{out,j}^t \quad \forall i, t : d_{out,i}^t > 0 \tag{5.10}
\]

\[
\bar{x}_i(t) = \text{avg}(x_i^t) \quad \forall i, t \tag{5.11}
\]

\[
0 \leq p \leq 1 \tag{5.12}
\]

According to constraint (5.10), the observed demand \(d_{out,i}^t\) at each station consists of the demand \(x_i^t\) that was actually intended for that station combined with part of the demand for the nearby empty stations. The latter is the lost demand for
these empty stations, which is denoted as $ld_{out,j}^t$ for station $j$, and can be computed according to the previous Section 5.2. Out of this lost demand, the probability of users going to station $i$ is given by the probability $p$ of people choosing to walk (instead of abandoning the system), multiplied by the probability $q_{ji}^t$ that they select the specific station $i$ to walk to.

Constraint (5.11) connects $\bar{x}_i(t)$ and $x_i^t$, but for simplicity of notation not all details are presented in the model: Each time $t$ is a timestamp, consisting of a day and a time period, and $\bar{x}_i(t)$ is the average of $x_i^t$ that refer to the same time period over all days for which station $i$ is not empty. The objective function takes $\bar{x}_i(t)$ as a reference for each station $i$ and time $t$, and considers the absolute differences of each $x_i^t$ with the corresponding $\bar{x}_i(t)$. The goal is to minimize the sum of these absolute differences. Solving (SD) provides a value for $p$ which is the estimated probability that people walked to nearby stations. We now need to readjust the demand of the stations that received this relocated demand, since it was originally directed to other locations. For each station $i$ with an observed demand $d_{out,i}^t$, the actual demand is given by $\max(x_i^t, 0)$.

The fact that $d_{out,i}^t$ includes some of the users for other stations may have influenced the demand trends and the average behavior based on which the lost demand was calculated in Section 5.2. Since $d_{out,i}^t$ might be larger than the actual demand for that station, this might lead to a slight overestimation of the lost demand. For that reason, one might consider running another iteration of lost demand estimation, as described in Section 5.2, considering now the actual demand $\max(x_i^t, 0)$, for each station $i$ and time $t$.

5.4 Computational Experiments

The results of this section are based on real-world data for Capital Bikeshare, Washington DC, for the time period May to September 2017. For each past trip, we have details about its origin, its destination, as well as its starting and ending times. We also have information about station outages, i.e. the starting and ending time, as
well as the station where it was observed, and its status - full or empty. Finally, the
design details of the system are also known: the location and the capacity for each
station, and the total number of bikes in the system. The data for past trips and
system information (Capital Bikeshare System Data (2017)), as well as the data for
outages history (Capital Bikeshare Tracker (2017)) are publicly available.

5.4.1 Lost Demand

In order to evaluate the methods suggested in Section 5.2, we identify all periods for
which each station is not empty, and for each such period we assume that the demand
is unknown and estimate it as the convex combination of the demand obtained by
the average station behavior and the daily trends.

\[ ld_{out,i} = \lambda \cdot \text{AVG}_{-} ld_{out,i} + (1 - \lambda) \cdot \text{TREND}_{-} ld_{out,i} \] (5.13)

Since the station is not actually empty at that time, we know the exact demand that
was observed. So we can compare our estimation results with the observed demand
of the day using the mean square error (MSE) as the evaluation metric.

Computation of parameter \( \lambda \)

The optimal value of the parameter \( \lambda \) that minimizes the MSE can be computed in
closed form. Note that \( d_{out,i} \) appears instead of \( ld_{out,i} \) below, since in this setting only
stations with known demand participate.

\[ \lambda = \frac{\sum_{i,t} (d_{out,i} - \text{TREND}_{-} ld_{out,i}) \cdot (\text{AVG}_{-} ld_{out,i} - \text{TREND}_{-} ld_{out,i})}{\sum_{i,t} (\text{AVG}_{-} ld_{out,i} - \text{TREND}_{-} ld_{out,i})^2} \] (5.14)

The optimal value of \( \lambda \) is calculated using half of the dataset as the training set,
and then the MSE with the selected \( \lambda \) is evaluated on the remaining dataset. We ran
100 experiments with different random splits for the training and the testing set, and
the values are consistent. The average optimal \( \lambda \) is 75.56% with a standard deviation
of 1.45%.
Table 5.1: Out-of-sample demand estimation MSE during rush hours and for the whole day. The minimum trips column shows the least number of daily trips for a station to be considered, which allows to evaluate separately the results for the busiest stations of the system. The combination of the average demand and the daily trends outperforms considering just the average which is a common approach.

The average MSE results on the testing sets across all experiments are presented in Table 5.1. In this table, weekends and vacations are excluded and only the 80% busiest days have been considered to avoid very slow days and outliers where external factors may lower bike usage, so as to have more accurate comparisons with O’Mahony (2015), who remove days with rain and snow from their estimation based on average station behavior. We show the results for the whole day as well as for the morning and the evening rush hour. Similarly, we present separately results for busy stations (the ones that have a minimum of 100 or 50 trips per day). In all instances, using a combination of the average station behavior and the daily trends outperforms taking simply the average demand.

Having evaluated the methods on intervals where the demand is known, we will now illustrate some of the results on lost demand estimation. The data has already been preprocessed to identify intervals of lost demand, and the size of this lost demand is being estimated according to Section 5.2. In the following Figures 5-2 and 5-3, we present the results on the lost demand for each of the five months, May to September 2017, and compare it with the corresponding observed demand for the same period.

Figure 5-4 shows how the lost demand is distributed in the duration of the day. We can observe that there are two peaks both in the lost outgoing and the lost incoming
demand. Assuming the morning rush hour is from 7:30am to 9:30am and the evening from 5pm to 7pm, we can see that these peaks emerge towards the end and after each rush hour of the day. This is not an unexpected behavior, since the large masses of customers that move during the rush hours often leave big parts of the network unbalanced, and many outages may follow raising the amount of lost demand.

5.4.2 Shifting Demand

Having calculated the lost demand, both the observed and the lost demand are aggregated in periods of thirty minutes, and we use the model of Section 5.3 to calculate the probability $p$ with which users choose to walk if there is a station with available bikes nearby.

The maximum distance $l_{max}$ that people are willing to walk is a parameter of the model. We will now experiment with various values of $l_{max}$ and show how that influences the value of $p$. We assume that if people have more than one station they can walk to in the $l_{max}$-radius, then the probability of going to each station is inversely proportional to its distance, as explained in Section 5.3, making it more likely to visit
closer stations than further ones. We also experiment with various values for the shifting distance multiplier \( \alpha \). The results are illustrated in Figure 5-5.

In order to evaluate the results of the model, we want to check its consistency across the network. So, we randomly split the set of 485 stations into two separate groups \( A \) and \( B \) of approximately the same size. The model first runs on the stations of group \( A \) and provides a walking probability value \( p_A \), and then runs on group \( B \) and calculates the value of \( p_B \). Every time we run the model on a set of stations, the shifted demand that originates from stations of the other group is still taken into account. The reason for this is that at the end it was served by the stations under consideration, so it consists part of their observed demand. The difference of the values across the two groups is then given by \( |p_A - p_B| \).

Since a unique split is not sufficient to evaluate the performance of the model, we run 100 iterations, each with a different random split of the stations into the two groups. Let \( A^it \) and \( B^it \) denote the two groups that were obtained during iteration \( it \). We then use \( p_A^it \) (resp. \( p_B^it \)) for the value of the probability of walking that was
Figure 5-5: Probability of walking $p$ as the allowed walking radius $l_{\text{max}}$ and the shifting distance multiplier $\alpha$ vary.

$$
\begin{array}{c|ccc}
 l_{\text{max}} & \alpha & 1 & 1.2 & 1.5 \\
\hline
400m & 0.0228 & 0.0272 & 0.0319 \\
500m & 0.0181 & 0.0235 & 0.0249 \\
600m & 0.0165 & 0.0204 & 0.0236 \\
\end{array}
$$

Table 5.2: Consistency results for various parameter values. These values for the mean absolute difference $dp$ were obtained for 100 iterations and various values of maximum radius $l_{\text{max}}$ and multiplier $\alpha$.

obtained by running the model on group $A^i$ (resp. $B^i$). We can now compute $dp$, which is the average absolute difference of the values $p_A^i$ and $p_B^i$ across the iterations. For a number of $I$ iterations, we have:

$$
dp = \frac{1}{I} \sum_{i=1}^{I} |p_A^i - p_B^i| \quad (5.15)
$$

Table 5.2 summarizes the results for 400m, 500m and 600m of maximum walking distance $l_{\text{max}}$, and values 1, 1.2 and 1.5 for shifting distance multiplier $\alpha$. All combinations for $l_{\text{max}}$ and $\alpha$ that we tested lead to a small average difference between the $p$ values of the station groups. Since the number of iterations is large, this suggests that in the average case the $p$ values of the groups do not differ significantly. Similar values of $p$ indicate that this method is consistent throughout the network.

Having evaluated the consistency of the model, at this point we need to make a selection regarding the value of $l_{\text{max}}$ and $\alpha$ that will be used to calculate the shifted
demand. We cannot base our selection solely on the objective value of (SD), since large values for $l_{\text{max}}$ are not representative of the human behavior in the real world: someone would rarely walk one kilometer in order to find an available bike. For this reason, we decided to use $l_{\text{max}} = 500m$, which is a reasonable choice for a walking distance. Similarly, we selected $\alpha = 1.5$, which assumes people might go to other stations besides their closest one as long as they are not very far from it. Indeed, often the closest station might be on the opposite direction of the desired destination, so someone might decide to walk slightly further but simultaneously cover part of the distance towards their final destination. These values for $l_{\text{max}}$ and $\alpha$ correspond to a walking probability $p = 43.59\%$. 
Part II

Resource Management for Complex Event Scheduling
Chapter 6

Problem Overview

6.1 Motivation

Operations scheduling has been the focus of many applications, starting from military operations and now ranging from technician scheduling, to medical personnel shifts and airline planning, among others. Automating and optimizing these procedures is crucial both for attracting and retaining customers, and for maintaining low cost for the operators.

This work is motivated by multiple discussions with a large foodservice and support services company that operates in many countries around the world and employs thousands of employees. Its customers include many corporations that require large-scale services to cover the needs across their campuses. Some of the services that are offered include organization of catering events, with sizes ranging from small group meetings to company-wide social events, beverage deliveries so that company’s kitchens have adequate stock for the employees, water refills to ensure campus’ dispensers remain operating throughout the day, and audiovisual set-up of various company events.

Currently, these services fall under different sub-organizations that work independently with separate resources (vehicles), while data collected through vehicle tracking revealed that they remain idle for significant portions of their shift. This underutilization of resources showcases the need for efficient scheduling solutions that
are also able to optimize resource usage over a variety of service types. However, different services come with different operation models, and, therefore, a variety of requirements and constraints that need to be taken into account.

Furthermore, in order to be able to serve this wide range of request types, there must be a great variety of available resources. In particular, employees must have very different skill sets; for instance, in the catering example, there might be caterers with various levels of experience, bartenders, and others, whose combined expertise is necessary for the service of an event. Moreover, the vehicle fleet can consist of vehicles with different capacities and properties, e.g., refrigerator vehicles. In order to serve a customer request, all necessary resources must be present at the request location, resulting in demanding coordination needs which are further intensified by the great flexibility of the resources. All these challenges complicate the development of a unique scheduler that can cover the needs of the whole organization: be able to handle all service scenarios while also providing efficient solutions for each of them.

This problem is not unique to this particular company. Many large logistics companies offer a diverse set of services in order to cover many of their customers’ needs and remain competitive in the market. For instance, some services might mostly require extensive vehicle usage (transport of equipment, item deliveries), while others might need employees with specialized skills (technician repairs), or a combination of the two.

The goal of this work is to develop a universal scheduler that covers a wide range of operations. Each operation type can be considered as a sequence of steps that are required in order to complete a customer request. Each of these steps might then require a particular set of vehicles or employees with certain characteristics. The generality of this description forms an umbrella that includes various operation types: item pick-up and delivery, performance of various services at customer locations, scheduling of events, and others. A diagram of operation types (and the steps they consist of) is provided in Figure 6-1.

For simplicity of description and notation, the main running example throughout this work is going to focus on the organization of catering events. This belongs to the
Figure 6-1: Operation types.

fourth type of Figure 6-1, event scheduling, and the reason we selected this particular operation is that it is among the most complicated ones: At first, food items and appropriate equipment need to be brought to the event location, which might require the use of specialized vehicles (e.g., refrigerator vehicles) and employees with the respective driving licenses. Then, everything must be prepared before the event starts (set-up) by personnel with proper domain knowledge and experience. During the event itself, multiple employees might be present with appropriate expertise (servers, bartenders), and after it is over, everything must be cleaned-up and then returned to the depot. Each of these steps might be served by different employees, or the same if they have the required skills and their schedule allows it. The setting can be complicated further if we allow for an arbitrary number of steps for each event, as is demonstrated in the fifth category of Figure 6-1.

Deriving a scheduler that can provide good quality solutions over this wide range of scenarios is a very challenging problem from a theoretical perspective. It combines an heterogeneous fleet of vehicles, routing with time windows, scheduling of employees with various types and levels of expertise, precedence constraints on the various stages
of each event, variety of skills needed for each stage, as well as need for coordination due to the great flexibility of resources.

In this work, we propose a complete optimization approach to generate good quality schedules for all scenarios presented in Figure 6-1. This solver, although primarily inspired by large companies with complex and diverse operations, can, of course, be used by any company that offers services that belong to any of the categories under consideration. In order to be able to adapt to various business needs, our method allows the incorporation of various types of constraints and objectives, depending on the application.

Finally, our solver allows both short-term and long-term planning. In the short-term, this method can be used to automate the scheduling procedure for day-to-day operations of the companies. In particular, a use case might be to plan the schedules for the following day in order to serve the maximum number of customer requests, while satisfying all business constraints and minimizing the cost that corresponds to the use of resources. However, with appropriately selected objectives and constraints, decisions that guide long-term planning can also be provided. This is inspired by our discussions with the foodservice and support services company that would like to estimate the number of extra vehicles to acquire over the upcoming months. By minimizing the number of vehicles or performing sensitivity analysis while optimizing over typical demand, these questions can be answered while also providing schedules that can showcase the effects these long-term decisions will have on the operations.

### 6.2 Problem Description

In this section, we introduce the *Complex Event Scheduling Problem (CESP)*. We present the details of the problem, its main constraints and objectives, as well as some of the challenges that arise in the development of efficient solution approaches.
6.2.1 Main Entities

The main components of this problem consist of the customer requests and the operator’s resources which include vehicles and employees.

Events/orders.jobs. These terms are used throughout this work interchangeably to represent the customer requests. Each request might consist of multiple steps and each of them might come with a location for the service to take place, a time window during which the service must start, service duration, items that need to be carried, and skills that the assigned employees should have. We assume that the information for each request is provided by the customer and is known; we don’t consider any uncertainty associated with it.

Figure 6-2: Representation of a customer request as a set of steps. Each step comes with information and requirements, such as location, time windows, skills and vehicle capacity required for its service. Steps of an event might have various forms of interactions; for example, a step might need to be completed before another step can start (precedence), or some steps might be coupled, i.e. they must be assigned to the same resources. An example for the latter case includes loading and unloading of items. These are two separate steps as they do not need to be consecutive in a schedule, and being separate allows for other steps to be inserted between them, however they need to be performed by the same resources (e.g., vehicles).

Vehicles. The operator has a fleet of vehicles in order to transport people and items to the event locations. This fleet is heterogeneous, as it may contain various types
of vehicles. For instance, in the catering example, vehicles range from small vans to large box-trucks, as well as refrigerator vehicles, etc. Each vehicle comes with a capacity both for the items it can carry, as well as the number of people it might transport: most trucks have only one to two passenger seats, but there are passenger vans that can carry up to 12 people.

**Employees.** Each employee might have a specific shift for which they are available to work, as well as specific geographic areas in which they can operate. Moreover, each person has a set of characteristics and skills. For example, using again the catering setting, an employee might have a varying level of catering experience (caterer, senior caterer, lead, area manager) or no catering expertise, so they focus on the clean-up and transportation of items where this expertise is not a prerequisite. Furthermore, people might have bartending licenses so they can serve as bartenders during the events, specialized driving licenses that allow them to drive larger vehicles (such as large box-trucks), or special access to restricted buildings that require stricter security checks.

### 6.2.2 Schedule Characteristics

**Objectives**

The scheduling objectives might vary according to the needs and goals of the operators. Some possible objectives include maximizing the number of successfully served customer requests, minimizing the number of vehicles used, their cost, the travel time or idle time of vehicles and employees, and others. Combinations of the previous objectives might also be desired, and this is possible by introducing appropriate weights based on their priority.

**Constraints**

A schedule needs to satisfy various constraints in order to be feasible and serve orders successfully.
**Event requirements.** As already mentioned, each customer request might consist of multiple steps, each of which having its own requirements. In order to serve an order successfully, all step requirements must be satisfied. This leads to constraints where time windows need to be respected, vehicles with sufficient capacity must be available to transfer items to the event location, and employees with appropriate skills and expertise must be available at the event location for the required duration of each step. If the requirements of a step cannot be covered successfully, then the customer order cannot be served by the operator.

**Step precedence.** Steps within an event often come with specific ordering. For instance, the set-up of an event must be completed before the event can start, or items must be delivered to the event location for the set-up to begin, etc. These step precedence constraints must be respected for an event to be served successfully.

**Coupled steps.** Some steps are coupled in the sense that they need to be performed by the same resources, and this constraint needs to be taken into account for a solution to be considered valid. An example is presented in Figure 6-2.

**Vehicle capacities.** At any point, the items and people that are assigned to each vehicle, must satisfy its capacity limits, which might vary from vehicle to vehicle as we assume an heterogeneous fleet.

**Traveling.** Enough time must be allocated for vehicles and employees to travel between locations. Furthermore, at any point each vehicle is able to travel only if an employee with the appropriate driving license is assigned as its driver.

**Resource coordination.** An important aspect of this problem is the flexibility of the resources and the resulting need for their coordination. For example, for the set-up of an event to start, it is not enough for the vehicle with the items to arrive at the event location; the person that will unload them or prepare them for the event must also be present. Otherwise, the vehicle might not be able to continue its trip while awaiting for the employee to arrive. In particular, all resources - vehicles and employees - that are assigned to a specific step of an event must be available for that step to be served successfully.
6.2.3 Challenges

The main challenges of this problem arise from the combination of multiple aspects that need to be concurrently considered within the optimization. First, there is the vehicle routing component, with time windows, capacity constraints and an heterogeneous fleet of vehicles, which forms a challenging problem on its own. This is further combined with the event scheduling aspect of the problem, where step precedence constraints and possible trip duration limits must be respected.

However, the greatest challenge arises when these are combined with employee scheduling, i.e. assignment of employees to events and derivation of their schedules. Each employee might have a different set of skills and expertise which determine their ability to serve a customer request. Large events might require the formation of teams of employees to obtain a combined skill set that can cover all requirements. Depending on the application, the flexibility in employee transportation might provide them with many options to travel from one location to the other: drive a vehicle that might also transfer the event items, join it as a passenger, be picked up and dropped off by a different vehicle given that its schedule allows it, or travel with a passenger van. In all cases, coordination is necessary among all resources for each event step: all assigned vehicles and employees must be present for the execution of the step to begin.

All these aspects under consideration create a complex setting and demonstrate the need to develop efficient optimization solutions that take all requirements into account and can generate good quality schedules in a timely manner.

6.3 Related Work

The core of our problem consists of both a vehicle routing and an employee scheduling component, which are combined with various constraints and requirements. In this section, we present a brief overview of the literature for these problems, as well as focus on certain applications that can either be considered special cases of our problem or are closely related to it.
6.3.1 Vehicle Routing and Crew Scheduling

The Vehicle Routing Problem (VRP) and its variants have been extensively studied in the literature. Solomon and Desrosiers (1988) provide a survey on a range of VRP classes with time windows. Laporte (1992) gives an overview of exact and approximate algorithms that have been proposed, while Laporte (2009) summarizes the main developments that have taken place over the fifty year horizon that followed the introduction of the problem in Dantzig and Ramser (1959). A more recent survey by Lahyani et al. (2015) presents a taxonomy of Rich Vehicle Routing Problems (RVRP), which are problem classes that also incorporate many characteristics, such as constraints and preferences, inspired by real-world applications. This viewpoint also aligns with the ultimate goal of our work to achieve applicability of the solution on real-life problems.

Drexl (2012) surveys VRP with Multiple Synchronization Constraints, where two or more vehicles need to be synchronized in order to perform a task. Some reasons that might give rise to the need for synchronization are the transshipment of load between vehicles, the existence of non-autonomous vehicles, or the requirement of multiple vehicles at the same location in order to perform a task. The latter is highly relevant to our problem as well, since large events might require the presence of multiple vehicles at the event location. Bredström and Rönnqvist (2008) propose a model that considers dependencies between vehicles and temporal constraints between customer visits (an analogous to the precedence constraints in our problem), which appear in various applications, such as home care staff scheduling and harvest planning in forest management.

Multiple studies consider the VRP in conjunction with crew scheduling, as very often crew is required to staff the vehicle movements. Some examples include crew scheduling for mass transit and the airline industry. The main difficulties of these problems arise by the additional constraints that need to be satisfied for the schedules of the crew, such as breaks, respect of regulations, etc. A survey that presents the crew scheduling problem and its combination with vehicle routing can be found at
Raff (1983). Desaulniers et al. (1998) propose a unified framework for combined vehicle routing and crew scheduling problems, while Zäpfel and Bögl (2008) present a metaheuristic approach for the solution of the vehicle routing and personnel planning problem for postal companies that also allows for outsourcing options of drivers and vehicle tours if needed. Goel and Irnich (2016) propose an exact method for the combined vehicle routing and truck driver scheduling problem. The problems in this line of work appear as an aspect of our setting, as we need to ensure that all vehicles are assigned drivers appropriately in the generated schedules.

6.3.2 Workforce Scheduling

In this stream of research, the focus is particularly placed on the scheduling of workforce (personnel) in order to perform tasks at various locations. The types of problems that are closer to our work, include tasks at different locations that require traveling of the employees, which introduces a routing component to the problem. Employees might have a diverse set of skills and various modes of transportation available for their travel between customer locations.

A survey on workforce scheduling problems can be found at Castillo-Salazar et al. (2016). Castillo et al. (2009) incorporate multiple criteria and objectives in the generation and characterization of solutions, and Baker (1976) presents some problems with cyclic demand for staff. Ernst et al. (2004) provide a survey around the problem types and solution methods that have been proposed for the staff scheduling and rostering problem. In the following section, we focus on specific applications that require scheduling of personnel, as well as vehicle routing.

6.3.3 Applications

There are various applications that require a combination of vehicle routing and employee scheduling. Some examples include crew scheduling for urban mass transit, scheduling of personnel for home health care, technician scheduling, and others. An overview of these applications is provided below. Depending on the specific con-
Many of them can be viewed as special cases of our problem.

**Urban Mass Transit Crew Scheduling**

In urban mass transit crew scheduling, the goal is to schedule drivers in order to cover the requirements of all active vehicles. Vehicle schedules are planned such that they include relief points, where a change of driver might occur. Parts of the vehicle schedule between two relief points define a task and need to be assigned to the same driver. A series of tasks forms a duty for a particular person, given that it also satisfies all necessary constraints and regulations. This work is closer to the driver assignment that is required in our problem.

In order to solve this problem, Haase et al. (2001) propose an exact approach based on column generation combined with branch-and-bound. A column generation approach is also suggested by Desrochers and Soumis (1989) who decompose the problem into two parts: a set covering problem and a subproblem for the generation of new workdays to improve the current solution of the set covering problem. Ball et al. (1983) solve the problem by developing a matching-based heuristic. For more details on the problem description and solution approaches, readers are referred to Carraraesi and Gallo (1984) and Freling et al. (2003).

**Technician Scheduling**

Field service technician scheduling is a problem faced by many service providers. In particular, customer requests of various types (repair, maintenance, etc.) require skilled employees (technicians) who need to travel to the customer location within specific time windows in order to perform a service. The employees might have different skill sets and specific geographic areas where they can travel, while customer requests might also have a priority level depending on their urgency.

For the development of methods that can scale to real-world instances, the main focus has been placed on heuristics and metaheuristics. More specifically, Xu and Chiu (2001) develop heuristics for the problem, including a greedy heuristic, a local search heuristic, and a greedy randomized adaptive search procedure (GRASP). Hashimoto
et al. (2011) also develop an approach based on GRASP in order to schedule technicians for a telecommunication company. Cordeau et al. (2010) solve the same problem using an Adaptive Large Neighborhood Search (ALNS) approach. The same metaheuristic framework appears in our approach as well, but since our setting is much more complicated, more aspects need to be incorporated in order to obtain feasible and good quality solutions. Tang et al. (2007) consider a maintenance scheduling problem, which they formulate as a multiple tour maximum collection problem and use a tabu search based heuristic to solve it, while Kovacs et al. (2012) develop another approach based on ALNS.

The technician scheduling problem can be incorporated under our problem description, as our setting includes scheduling of specialized employees to serve customer requests at various locations. Also, note that while Hashimoto et al. (2011), Cordeau et al. (2010), and Kovacs et al. (2012) take into account the possibility of team formation, they assume that teams are formed in the beginning of the day and stay together for the whole duration of the day. This contrasts with our approach where employees can belong to multiple teams in the same day, as there is flexibility based on the current needs and resource availability.

Home Health Care

The Home Health Care Scheduling problem focuses on generating shifts and schedules for nurses to visit client locations. These visits might determine specific time windows where they must take place or require specialized personnel skills for their successful completion. Routing is an important aspect of the problem, as nurses might need to visit more than one clients on a given day, and depending on the study more than one modes of transportation might be available to the nurses. Synchronization and time constraints might also be considered in some scenarios where more than one employees are required at a client location. A review on these problems can be found at Fikar and Hirsch (2017), however, an important difference from our setting is that the transportation of items and equipment and the corresponding capacitated vehicle routing problem are not taken into account.
A range of solution approaches (exact, approximation, heuristic, metaheuristics) have been suggested for variants of the problem. Akjiratikarl et al. (2007) develop a Particle Swarm Optimization technique to schedule home health care personnel with homogeneous skill sets. Eveborn et al. (2006) suggest a heuristic approach based on repeated matching. Mankowska et al. (2014) give a mathematical model formulation for the problem and solve it using Adaptive Variable Neighborhood Search, while Fikar and Hirsch (2015) propose a two-stage matheuristic based on creating promising walking routes and optimizing the system using a tabu search framework. Bredström and Rönqvist (2008) propose a Mixed Integer Programming formulation in order to obtain exact solutions, as well as a matheuristic for the problem. An exact approach is developed by Rasmussen et al. (2012) based on a branch-and-price algorithm.

Airline Scheduling

The airline industry is faced with various planning problems, such as flight schedule generation, fleet assignment, maintenance planning, and crew scheduling. Crews might have varied expertise and they need to be assigned to flights in order to cover all their requirements, while also respecting various constraints for the employees. More details on this problem, both on the crew scheduling component, as well as some of its remaining aspects, can be found at Barnhart et al. (2003), Gopalakrishnan and Johnson (2005), and Vance et al. (1997).

Other Applications

The applications presented in this section are not an exhaustive list of problems that are relevant to our setting. Another example includes the routing and rostering of security personnel, where security tasks at various locations must be assigned to the employees, who need to travel to the appropriate locations to fulfill them (Misir et al. (2011)). This is a simpler problem since it does not require the transportation of items or equipment to the task location.

In a different setting, Valls et al. (2009) consider the Skilled Workforce Project Scheduling Problem that arises in many company Service Centers. In particular,
tasks and projects need to be assigned to employees with homogeneous or heterogeneous skill sets. This problem consists mainly of a project scheduling and a resource allocation component. A routing component is not present, therefore, this problem can be viewed mostly as an employee scheduling problem.

Finally, let us conclude this section with a mention to the well-studied pick-up and delivery problem. Parragh et al. (2008), Berbeglia et al. (2007) and Savelsbergh and Sol (1995) present problem types and solution methods that can be found in the literature. Depending on the assumptions, many versions of this problem can viewed as special cases of our setting; in particular, they can be incorporated under the first operation type present in Figure 6-1.
Chapter 7

Schedule Generation

This chapter describes our proposed optimization framework for the generation of the schedules for both vehicles and employees in order to satisfy the customer requests. Given the great complexity of our problem, exact approaches are not able to produce good solutions in a timely manner, so we turn to methods that also make use of heuristics and meta-heuristics to efficiently search the solution space. We begin with Sections 7.1 and 7.2 that present two main components of our scheduler: the first is a general framework called Adaptive Large Neighborhood Search (ALNS) suggested by Ropke and Pisinger (2006a), and the second is a linear program which we call Global Schedule Coordinator and is crucial for maintaining the feasibility of schedules throughout the execution of the algorithm. Sections 7.3 and 7.5 present the details of our approach, and Section 7.6 concludes with the evaluation of the proposed methods.

7.1 Adaptive Large Neighborhood Search (ALNS)

Adaptive Large Neighborhood Search (ALNS) is a general heuristic-based optimization framework proposed by Ropke and Pisinger (2006a) for the solution of the Pickup and Delivery Problem with Time Windows. In subsequent work (Ropke and Pisinger (2006b), Pisinger and Ropke (2007)), they have also applied this method to multiple variants of the Vehicle Routing Problem (VRP), such as the Capacitated Vehicle Routing Problem, the Multi-Depot Vehicle Routing Problem, and the Vehicle Rout-
ing Problem with Backhauls, showing its robustness and its ability to discover good quality solution.

The problem we are studying in this work is much more complex than the previous VRP variants due to the coordination of both employees and vehicles and the possible team formation that is required for the service of customer requests, among others. As a result, ALNS cannot be trivially extended to our setting, but more components are required for feasible schedules to be produced. In this section, we give an overview of this method, which will be used as a general framework of our solution, and the way to adapt it to our problem along with all other characteristics that are required to make it work are presented in the sections that follow.

7.1.1 Method Description

ALNS is an extension of the Large Neighborhood Search heuristic proposed by Shaw (1997), and is composed of a set of heuristics that are being used depending on their historic performance during the execution of the algorithm. In particular, starting with an initial feasible solution, ALNS performs a number of iterations in each of which it first “destroys” the solution by removing some of the requests from its schedules, and then “repairs” it by trying to reinsert the unserved requests to the partial schedules. The destroy and repair steps of the algorithm take place through the use of heuristics, which from now on are referred to as destroy operators and repair operators respectively.

The main steps of the method in each iteration, starting from a current feasible solution, are the following:

1. Select a destroy operator D and a repair operator R depending on the values of some selection weights.
2. Using D, remove a number of requests from the schedules of the current solution. The resulting schedules are called partial schedules.
3. Using R, try to insert any unserved requests (including the ones that were just
(removed) to the partial schedules obtained in Step 2 to obtain a new feasible solution.

4. If the new solution is better than the current one in terms of the objective function, accept the new solution which will serve as the start of the next iteration. If the new solution is worse than the current one, accept it with certain probability which depends on the difference in their objective function values.

5. Update the selection weights for the operators.

6. If the new solution is the best discovered so far, update the global best (so far) solution.

The main idea of this method is to explore various neighborhoods of the solution space using the destroy and repair operators in each iteration. These operators are heuristics, which often also contain an element of randomness, that allow the method to diversify or improve the current solution. These operators should preferably be fast enough in order to be able to complete many iterations of the method in short solution times. Most of the heuristics are not based on complex criteria, and this might lead to bad quality solutions in some of the iterations. However, even in this case, this can be beneficial for the final outcome of the algorithm, as these solutions can contribute to the diversification of the search, which might eventually lead to better solutions overall.

**Destroy Operators**

Destroy operators are heuristics that select a specific set of requests that are served in the current solution and remove them from the existing schedules. The selection of requests to be removed can vary a lot depending on the heuristic; for instance, one destroy operator might randomly remove requests, while another might be based on more sophisticated criteria. In Section 7.3, we detail all the destroy operators that we use for the solution of our problem.
Repair Operators

Repair operators start from a partial solution and try to insert any unserved requests to the schedules. Repair operators barely create a solution from scratch. Various criteria can be used in order to select where to insert each request. For instance, requests can be inserted greedily to the schedule position that currently seems to improve the objective function the most. The repair operators that we developed for our problem are described in Section 7.3.

Initial Solution

The algorithm assumes that upon the start of the first iteration there is an initial feasible solution available. There are multiple ways in which this initial solution can be constructed. One approach includes using one of the repair operators: considering empty schedules for each resource, and each request as unserved, sequentially insert all requests to the schedules, until all requests are inserted or it is not possible to insert more requests.

Adaptive Operator Selection

Each operator $i$ has a selection weight $w_i$ associated with it that guides its selection probability in each iteration. Let $R$ denote the set of all repair operators and $D$ the set of all destroy operators. Then, each operator is selected with probability proportional to its weight:

$$p_r = \frac{w_r}{\sum_{r' \in R} w_{r'}} \quad \forall r \in R \tag{7.1}$$

$$p_d = \frac{w_d}{\sum_{d' \in D} w_{d'}} \quad \forall d \in D \tag{7.2}$$

In our implementation, instead of maintaining separate weights for repair and destroy operators, we consider the operators in pairs, as both types contribute in the generation of each solution. In particular, we have weights $w_{dr}$ for each combination of destroy operator $d \in D$ and repair operator $r \in R$. The pair of operators to be
used in a specific iteration is selected based on probabilities:

\[ p_{dr} = \frac{w_{dr}}{\sum_{d' \in D, r' \in R} w_{d'r'}} \quad \forall d \in D, r \in R \tag{7.3} \]

Weights do not remain constant during the execution of the algorithm; they are updated based on the quality of the solutions that were generated through their use. We can select to update them in every iteration or every time a specific number of iterations has passed. Superscript \( t \) refers to the current values and \( t + 1 \) to the updated ones.

\[ w_{dr}^{t+1} = \gamma \cdot \bar{w}^t + (1 - \gamma) \cdot w_{dr}^t \tag{7.4} \]

Two terms determine the updated values of the weights: its previous values \( w_{dr}^t \) and the quality of the solutions they produced \( \bar{w}^t \), which can take different values based on the outcome of the iteration (new best solution found, new solution was accepted, new solution was rejected). Details on the particular values for each scenario are provided in the Computational Experiments Section 7.6, along with the value for the parameter \( \gamma \). At each weight update round, only the weights of operator pairs that were used during this round are updated. In our experiments, we select to update the weights after each ALNS iteration.

Acceptance Criteria

At the end of each iteration, we need to decide whether to accept or reject the new solution that was produced in the current iteration. If the new solution is accepted, then it serves as the starting solution of the following iteration. If it is rejected, then the previous starting solution is used for the next iteration as well. If \( s' \) is the new solution, \( s \) the previous solution, \( f(s') \) and \( f(s) \) their respective objective function values, and assuming a maximization problem, then:

- \( s' \) is always accepted if it is better than \( s \),

- \( s' \) is accepted with probability \( e^{(f(s') - f(s))/T} \) if it is worse than \( s \), where \( T > 0 \)
is called temperature and is a parameter of the model.

The temperature $T$ is updated in each iteration through a multiplication with a positive constant $\alpha < 1$. This makes the acceptance of worse solutions more likely in the beginning of the algorithm in order to explore more the solution space. As the number of iterations increases, the search is focused on the most promising areas, and worse solutions are accepted with lower probability. This simulated annealing inspired criterion can help the method escape local optima.

**Termination**

The method terminates after a prespecified number of iterations or when a time limit has been reached, if there are strict time constraints.

### 7.2 Global Schedule Coordinator

The Global Schedule Coordinator (GSC) ensures that the feasibility of all schedules is maintained for the whole duration of the algorithm’s execution. GSC does not decide which resource will be assigned to which request or in what order the requests will be served. Instead, given the schedules of the resources, it attempts to see if there are possible start times that allow all planned services to be performed, taking into account time windows, coordination and other constraints.

#### 7.2.1 Coordination Requirements

As previously mentioned, each customer request consists of a set of steps, each of which might require more than one resources in order to be served successfully. These resources might either be a team of employees, whose combined skill set is necessary for the execution of the step, or a combination of vehicles and employees.

It is worth noting that when a step needs more than one resources in order to be served successfully, we assume that its service can only begin when all required resources have arrived. In case one of the resources, vehicle or employee, arrives
earlier, they need to wait until the remaining resources are present, as well as until the start time of the step’s time window has been reached.

This creates cross-dependencies among the resources’ schedules: it is not sufficient for a resource to arrive on the location on time, perform the service, and continue traveling for the next location. It might have to wait for other resources to arrive, and this might delay the time it will be available to serve the upcoming steps in its schedule. Therefore, to check the validity/feasibility of a schedule, we need to consider it in conjunction with all other resources’ schedules and take these cross-dependencies into account.

This problem does not arise in the work of Ropke and Pisinger (2006). In their settings, each customer request is served by a unique vehicle, so all schedules are independent. As a result, when a request is removed or inserted, that only alters the schedule of a single resource. In that case, it is easy to check whether the new schedule is valid: we just need to ensure that the vehicle can serve its assigned requests during their time windows, while also allotting sufficient time for the service duration and the traveling among locations.

In our problem though, schedule feasibility checks cannot take place in the same way. This is illustrated through an example in Figures 7-1a through 7-1c. In particular, a step insertion in one resource’s schedule (Figure 7-1a) might be possible but might cause the remaining steps of the schedule to be executed earlier or later than previously planned in order to accommodate the new step (Figure 7-1b). However, if any of these adjusted steps require more than one resources, then the start time needs to be adapted in all other resources’ schedules as well, which might naturally lead to more steps needing to be adjusted in other schedules, etc.

Essentially, one step insertion in one schedule might create a domino effect of changes on virtually all the resources. Therefore, checking whether this step insertion is feasible is not a simple task; one could potentially check all resource schedules multiple times readjusting a large number of steps’ start times in order to determine if the insertion can be accommodated. To this effect, we propose to use instead the Global Schedule Coordinator (GSC), a linear program that allows for easy feasibility
checks as well as determination of all steps’ execution start time.

### 7.2.2 Model Description

GSC is a linear program with variables the start times of each step and constraints that ensure all time requirements are taken into account for each schedule.

**Variables**

More specifically, if \( S \) denotes the set of all request steps, for each step \( i \in S \), we have a variable \( t_i \in \mathbb{R}_+ \) in the model to denote the time the service at step \( i \) begins.

**Constraints**

We have three main types of constraints.

*Time windows.* Let \( T_i^s \) and \( T_i^e \) denote the start time and end time of the allowed time window for each step \( i \in S \). Assuming this time window refers to the acceptable times the service may begin, we have the following constraints:

\[
T_i^s \leq t_i \leq T_i^e, \quad \forall i \in S
\]  

(7.5)

*Precedence between steps of the same event.* Steps that correspond to the same customer request might have a specific ordering in which they need to be executed. We can introduce a constraint for each such pair of steps \( i \) and \( j \). Assume we can only start \( j \) if \( i \) is complete. This is the most common type of constraint met in practice, but more might be supported if they can be expressed as linear constraints of the variables \( t_i \) and \( t_j \). Let \( \text{dur}(i) \) denote the service duration of step \( i \). Then, we have:

\[
t_i + \text{dur}(i) \leq t_j
\]  

(7.6)

*Precedence between schedules’ steps.* For each resource schedule, the service of each step might begin only once the previous step is over, as we assume no parallelization of the service. If \( i \) and \( i + 1 \) are two consecutive steps in the schedule of the same
(a) This figure depicts the current schedules for a vehicle and an employee. The schedules include steps A to H, and we would like to insert step I at the vehicle’s schedule at the position seen above.

(b) Step I can be inserted at the position under consideration, but would lead to a delay of the subsequent step B of the vehicle’s schedule. However, B is also served by the employee, so in order to achieve I’s insertion we need to ensure that the employee can accommodate B’s new time.

(c) Even if the employee’s schedule can serve B at its new time, feasibility checking is not necessarily complete. Assume step F requires the presence of another vehicle. In that case, that vehicle’s schedule must be checked for F’s new time, and, in that way, changes might easily be propagated to a large number of resource schedules.

Figure 7-1: Example of step insertion and the complications that might arise to ensure the validity of the resulting schedules.
resource, and $dist(i, j)$ is the distance between their locations, we must have:

$$ t_i + dur(i) + dist(i, j) \leq t_j \quad (7.7) $$

**Other Constraints.** The previous constraints express requirements that are relevant to most applications. However, more specific application-based constraints can be incorporated as well. For instance, in the food services domain, regulations might set strict time limits on the duration that food items can spend in the vehicles. If $t_i$ is the variable that corresponds to loading the items, $t_j$ to unloading them, and $\tau_{max}$ is the maximum allowed duration in between, then we can incorporate this to the GSC by introducing constraints of the type

$$ t_j - t_i \leq \tau_{max} \quad (7.8) $$

Similarly, we can incorporate constraints for the maximum time a vehicle can be away from the base location, shift limits for employees, and others.

**Objective Function**

In this particular setting, it is sufficient to find a feasible solution for the linear program, so the objective function could just be set to zero. However, a better option is try to maximize the start time of each request while minimizing its end time. This would lead to more compact schedules, without having as much idle time which would occupy resources unnecessarily. In particular, if $t_{first}^r$ and $t_{last}^r$ are the variables that correspond to the service start of the first and the last step of request $r$, then the objective function is as follows:

$$ \min_r \sum (t_{last}^r - t_{first}^r) \quad (7.9) $$
7.2.3 Model Integration

GSC can be integrated with ALNS in a single optimization framework, which will form the basis for our scheduling approach. In particular, both ALNS and GSC need to be initialized at the start of the algorithm. Then, ALNS performs a number of iterations as explained in Section 7.1. GSC needs to be updated during each iteration accordingly, such that at any moment it is fully compatible with the current ALNS solution and includes all required precedence constraints for the current schedules. We will now discuss in more details the initialization of GSC, as well as the actions that are required in each destroy and repair step of the ALNS.

Initialization

At the beginning of the algorithm, before starting the ALNS, we initialize the GSC by adding all variables (variables $t_i$ for each step $i$), as described in the previous section. We also add the time window constraints (7.5), and precedence constraints between steps of the same event that have explicit ordering requirements (7.6). These constraints are independent of the scheduling; they are originated from event specific requirements and must be satisfied throughout the algorithm to ensure the validity of the schedules.

Model Updates

ALNS performs a sequence of destroy and repair actions in each iteration, where steps are selected to be removed or inserted to particular schedules. The details on the operators’ actions are presented in the following Section 7.3. In order for the GSC to always be up to date with the current ALNS solution, changes to the model are required every time an operator removes or inserts a step to a schedule.

**Insertion.** Assume a step is added at the $i^{th}$ position of a resource schedule. Let that step be denoted with $s_i$. If $s_i$ is not the last step of the schedule, then upon inserting $s_i$, we need to ensure that there is enough time to complete $s_i$ and travel to $s_{i+1}$ on time. Note that time window constraints for all steps have been included.
in the model upon initialization. Therefore, we just need to add the precedence constraint (7.7) for steps \( s_i \) and \( s_{i+1} \). Similarly, if \( s_i \) is not the first step of the schedule, a precedence constraint (7.7) needs to be added for \( s_{i-1} \) and \( s_i \). Finally, if \( s_i \) is inserted in the middle of the schedule, the previous precedence constraint between \( s_{i-1} \) and \( s_{i+1} \) needs to be removed, since it is no longer necessary. Removing redundant constraints at each schedule change is important in order to make it easier to insert and remove steps in future iterations, as well as to keep the model at the smallest possible size, as the number of iterations can be very large and redundant constraints might become problematic.

Removal. Similarly, for each step removal, assuming \( s_i \) denotes the step to be removed, if \( s_i \) is not the last step of the schedule, we remove the precedence constraint between \( s_i \) and \( s_{i+1} \), and if it is not the first step, we remove the precedence constraint for \( s_{i-1} \) and \( s_i \). Finally, if \( s_i \) used to be in the middle of the schedule, we need to add the precedence constraint for \( s_{i-1} \) and \( s_{i+1} \) to ensure the resulting schedule remains feasible.

Optimization

After each step removal, we know that schedules remain feasible: if the resources could do the steps they were assigned to before the removal, they are still able to do the ones that remain after the removal. This is not true for step insertions: a step that is inserted in a schedule might lead to infeasible schedules where either precedence or coordination constraints cannot be satisfied. To this end, upon each step insertion and the appropriate update of the model, the model needs to be reoptimized.

The results of the model indicate whether the step insertion is possible. If the model is feasible, then it produces a solution where all variables are set to times that satisfy all schedules’ constraints and can be used by the planner for the execution of the schedules. If, however, the model is infeasible, then the insertion of the step cannot take place. In this case, the insertion needs to be reverted; the step needs to be removed from any schedule it was just added in order to return to a feasible solution before continuing with the next insertion attempt.
Note that each update of the model adds or removes only a small number of constraints. As a result, given that the solution that was obtained in the last optimization of the model is available, it can be used as a warm start which can make the model reoptimization very quick. In Section 7.5, we show how we can further reduce the time spent on the model reoptimization.

7.3 Operators

This section focuses on a more detailed presentation of the operators. We start with the destroy operators, which select requests or requests' steps to be removed from the resources’ schedules, and then continue with the repair operators, whose goal is to insert unassigned requests/steps to partial resource schedules in order to obtain better solutions overall.

7.3.1 Destroy Operators

Every destroy operator differs in the way it selects requests or steps to be removed. However, once the steps to be removed have been determined, the removal mechanism is the same: these steps need to be deleted from all resources’ schedules that are currently involved, and for each such deletion, we need to update the GSC appropriately following the procedure described in Section 7.2.

Random Request Removal

This operator selects a random number between 1 and an upper bound $R_{\text{max}}$ of requests to be removed. The upper bound cannot be too large as, in this case, a very big part of the solution would be rearranged in each ALNS iteration, which would slow it down considerably. On the other hand, it cannot be too small, as this would not allow enough flexibility in the search for better solutions. The Computational Experiments Section 7.6 presents our specific parameter selection for our problem instances.
Once a random number within this range has been selected, we choose uniformly the corresponding amount of requests to be removed. We only consider the ones that are currently served by resources, and do not take into account any unassigned requests. Finally, for each request, we remove all of its steps from all schedules (both vehicles and employees) where they participate.

**Vehicle Schedule Removal**

This operator is particularly useful when we are interested in minimizing the number of used vehicles. Instead of removing requests and steps at random, it selects some vehicles and clears their entire schedule by removing all its steps. The number of vehicles for which the schedules are cleared is again selected at random within a range from 1 to $V_{\text{max}}$. To identify the specific vehicles to be removed, we then follow an approach where the less busy vehicles are selected. The reasoning behind this decision is to remove steps that could be more probable to accommodate using less vehicles. This seems more likely if fewer steps are removed at each time and we then manage to incorporate them to the schedules of the remaining resources.

### 7.3.2 Repair Operators

The repair operators go through all the incomplete events and try to assign the appropriate resources to their steps so that they are served successfully. Each operator differs in the way it selects vehicles or employees and the position in their schedule where the steps are added. We assume that the events are processed in a random order. Of course, different selection processes might be reasonable and perform well too.

**Earliest Schedule Insertion**

This operator selects the vehicles and employees for each step in a random order among the ones that are compatible with serving the step. Once a resource has been selected, the operator goes through the resource’s schedule and attempts to insert the
step at the earliest valid position. Time windows can help avoid checking many invalid insertions: a step cannot be inserted before a step with an earliest time window.

This procedure is followed first for the vehicle schedules until the step has been assigned successfully and all its required items can be transported to the step location. For each assignment, we need to take into account the capacity of the vehicle and its current load in order to determine its validity. Once the vehicle requirements have been covered, we need to also assign employees with the appropriate skills to serve the current step. In a similar way, we can select randomly among the employees that have the required expertise, and insert the step in the earliest possible position of their schedule. This process is repeated until all step requirements have been covered, and it might lead to team formation if more than one employees need to be present. Note that we do not mention driver assignment here, as the next section 7.4 focuses on this exact topic.

This repair operator is the simplest one, but its power is in its simplicity: it can perform a large number of step insertions very quickly. Depending on the objective and the stage of the algorithm, this characteristic might be very desirable, as schedules are formed and requests are allocated in short running times. However, given that the insertion criterion is that simple, it might also lead to solutions of lower quality. Therefore, the operators we present next will focus on obtaining better quality solutions at the expense of increased running time.

**Optimal Resource Reusability Insertion**

This operator focuses on maximizing the load of already active resources, before assigning any steps to resources with empty schedules. This is motivated by the common objective of minimizing the number of used vehicles. The active vehicles are examined in random order and the current step is inserted in the earliest valid position.

For the employee assignment, we still use the Earliest Schedule Insertion presented above, as the Optimal Resource Reusability approach might not be very useful for the employees’ schedules. In particular, we often want to have balanced workload among
the employees, or the working days/shifts of the employees might have already been
determined, therefore, inactive employees do not offer any gain to the employer. If
this is not the case, this approach can be easily extended to the employee selection
as well.

Note that prioritizing already active resources might be beneficial for the objec-
tive, especially if it includes minimizing the number of vehicles. A downside, however,
includes creating dense schedules that reduces the possibility of successful upcoming
insertions, as well as increases the running time. In particular, because active sched-
ules have larger workload, there is less flexibility as well as less available time for new
insertions. As a result, a step insertion might require multiple attempts in many possi-
able schedules and many possible positions per schedule, until the GSC determines
that it can take place successfully.

**Greedy Insertion**

This operator selects the resources and the exact position in their schedule based on
their effects on the optimization objective. In particular, it attempts to insert the
step at all valid positions of the schedules of all compatible resources, in order to
greedily select the one that leads to the best objective value.

This approach might be beneficial when the objective includes, for instance, mini-
mizing the traveling time of the vehicles. However, it can become very time-consuming
as the size the instances increases, as it requires going through all resource schedules
and all schedule positions. To this end, for our experiments, we decide to apply this
technique mostly in the vehicle schedules, and instead of examining all schedules to
determine the best, we focus only on a subset. In particular, we select $V_{\text{subset}}$ vehicles
and insert the step under consideration in the schedule and position that benefits the
objective value the most. $V_{\text{subset}}$ is a parameter of the model: smaller values examine
fewer resource schedules, while larger values lead to slower running times.
7.3.3 Discussion

The operators that we presented are just a small subset of the possible algorithms that someone might consider. Note that we can always develop more complicated approaches with the hope of obtaining better solutions. However, given the complexity of our problem, these methods usually lead to much longer running times per execution, so we focused mostly on simpler approaches.

7.4 Driver Assignment

Each vehicle that is being used to serve customer requests needs to be assigned a driver in all its trips. In some applications, there might be employees whose tasks exclusively involve driving, while in others, employees might have several roles, for example, caterers might drive the vehicle with the items to the event location before starting their catering role for the event. For this reason, we are going to present two separate approaches that consider both of these scenarios. One of them is based on an integer programming formulation to perform the assignment of drivers to vehicle trips that have already been generated through our optimization approach. The other shows how we can incorporate the driver assignment within the general optimization framework.

7.4.1 Full-Time Drivers

Assume the company decides to hire people exclusively designated as drivers. From an optimization perspective, this simplifies the problem as the same person cannot both drive and serve events, so the solution space is smaller, since we cannot have schedules combining both sorts of actions. However, this might be inefficient from the planner’s perspective in some cases. For instance, assume there is a small catering event with short duration that needs one caterer. In this case, we could have had this caterer drive to the event location, have the vehicle idle for the short duration of the event, and then drive back. With the assumption that caterers cannot serve as
drivers, we are forced to occupy two employees for this event.

**Loop**

Before presenting the optimization model, we need to introduce one of its building components. A *loop* is a sequence of steps in a vehicle schedule which starts and ends at the base location (depot), such that if a step is part of the loop, then all its coupled steps are also part of the same loop. Loops are the shortest such sequences, in the sense that a loop cannot contain another loop.

We can now split each vehicle schedule into a sequence of loops. We can insert a base visit between any consecutive steps of a vehicle schedule in case there is long idle time between these two steps and given that this new visit can serve as the end of the loop. The purpose of this action is to avoid having long idle times for the drivers.

**Model**

We will introduce an integer programming formulation that assigns drivers to each step. Steps that are part of the same loop must use the same driver, as driver changes are allowed only when the vehicle is at the base location and not in the process of serving some request. As a result, we can instead assign drivers to loops.

Let $L$ denote the set of all loops as derived from the vehicle schedules. Some loops might be incompatible with each other, in the sense that they cannot be assigned to the same driver. This might be the case for example if they are overlapping. Thus, for each loop $l \in L$, let $I_l$ be the set of loops that are incompatible with $l$. Finally, let $D$ denote the set of available drivers, and $D_l$ the set of drivers that can serve loop $l$, i.e. they have a driving license appropriate for the vehicle that performs the loop. We assume a large enough number of available drivers, so that there is always a feasible solution where all loops are assigned a driver, and then we aim to minimize the drivers needed.

In particular, let $x_{ld} \in \{0,1\}$ denote a binary variable which equals 1 if loop $l$ is assigned to driver $d$, and 0 otherwise. Also, let $y_d \in \{0,1\}$ denote a binary variable that indicates whether driver $d$ is working: if $y_d = 1$, then $d$ is assigned to at least
one loop, otherwise \( y_d = 0 \) and driver \( d \) is not working in that particular instance.

The formulation follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{d \in D} y_d \\
\text{subject to} & \quad y_d \geq x_{ld} \quad \forall l \in L, d \in D_l \\
& \quad \sum_{d \in D_l} x_{ld} = 1 \quad \forall l \in L \\
& \quad x_{ld} + x_{l'd} \leq 1 \quad \forall l, l' \in I_l, d \in D_l \cap D_{l'} \\
& \quad x_{ld} \in \{0, 1\} \quad \forall l \in L, d \in D_l \\
& \quad y_d \in \{0, 1\} \quad d \in D
\end{align*}
\]

In the above model, the objective function minimizes the total number of active drivers needed. Constraints (7.11) ensure that only active drivers can be assigned to loops, while constraints (7.12) indicate that all loops must be assigned to a driver in order to be performed successfully. Constraints (7.13) take into account loops’ incompatibilities, which don’t allow them to be served by the same driver. An obvious example includes loops that take place simultaneously, but more complicated constraints can be taken into account as well. For instance, we can incorporate driver shifts. Assume each driver works for up to eight consecutive hours. If the start time and end time of the shift is fixed, then we can remove driver \( d \) from \( D_l \) or set to zero all variables \( x_{ld} \), for all loops \( l \) that are not fully overlapping with the driver \( d \)'s shift. If, on the other hand, the start and end time of the shifts are flexible, then we can define as incompatible any steps \( l \) and \( l' \) that cannot be part of the same shift, i.e. serving both of them would require a shift longer than eight hours.

**Discussion**

Regarding drivers’ shifts, note that since vehicles’ schedules are determined independently from the drivers, we might have loops with duration longer than the shift length. This can be addressed by incorporated extra constraints in the vehicle sched-
ule generation. In particular, in the Global Schedule Coordinator, we can add constraints such that all loops at any time have length less than the shift length. In case an insertion leads to longer loops, then it will be rejected by the GSC.

The approach presented in this section works well in the instances that we tested, whose size ranged to about 200 customer events and up to 40 vehicles. However, it might require long solution times if instances get larger, so it might not be appropriate in these cases. For this reason, we also present in the next section how the driver assignment can be incorporated in the general optimization framework. We focus on the case where employees serve as drivers, but the same approach can also be applied if there are separate people exclusively responsible for driving.

### 7.4.2 Employees as Drivers

The assignment of drivers can take place as part of the general schedule optimization. This is especially required in the case where employees have a joint role of driving and serving customer requests, but can be applied when drivers are hired separately as well. In this approach, for each step that requires items to be transported by a vehicle, adding the step to a vehicle’s schedule is not enough. Each vehicle must also be assigned a driver for the trip to be performed successfully. As a result, each insertion to a vehicle’s schedule is followed by a driver search.

More specifically, after a step has been successfully inserted to a vehicle’s schedule, we first need to identify if the step (or steps in case they are coupled) was inserted as part of an existing loop or if a new loop was created. To this end, we split the vehicle schedule into loops and identify the loop $l$ that contains the current step(s). If $l$ was an existing loop, it might contain more steps that have already been assigned a driver. In this case, there is a unique choice for the driver of the new step(s), since each loop can only be assigned a unique driver. The loop’s driver is then the candidate driver of the current step(s). When the current step(s) form a new loop that has no assigned driver, then essentially any person with the appropriate driving license for the vehicle could serve as a driver. In this case, we obtain a list of candidate drivers.

Having identified the candidate drivers, we attempt to insert the step(s) into their
schedule, until an insertion has been successful or the whole list of drivers has been examined. Each insertion is approved or rejected by the GSC. Time windows and the remaining steps of the loop \( l \) (if any) can guide the appropriate order of the current step(s) in the driver’s schedule in order to avoid redundant insertion attempts.

If a driver is found successfully, then the transportation of items can take place as both vehicle and driver are available. However, in the case where a driver cannot be found, then the step(s) need to be removed from the vehicle’s schedule as well, as the transportation cannot take place. In this scenario, the customer request is considered not served. As a result, if any of its remaining steps were already assigned to resources, they need to be removed as well, as there is no gain for partial service and their removal could free up time for the resources.

### 7.5 Algorithmic Enhancements

This section presents some methods that can be used to accelerate the running time of the algorithm.

#### 7.5.1 Accelerated Insertions

In Section 7.2.3, we describe how the linear program GSC can be integrated with the ALNS framework. In particular, the GSC is being updated with all insertions and removals that are attempted by the algorithm. In the case of insertions, this needs to be done after every possible insertion in order to check whether adding a particular step to a schedule would disrupt the feasibility of the solution. Given that the previous GSC solution is available, and each insertion requires a small number of changes to the model, the reoptimization can take place fairly quickly. However, we can accelerate this process by using a mechanism of *accelerated insertions*.

More specifically, before escalating the changes to the GSC, we first check the individual schedules where the steps are being inserted. This entails adding the steps under consideration to some candidate schedules and ensuring that these particular schedules remain feasible. This can be performed by confirming that the time windows
of all steps of the schedules can be respected, while allowing for the necessary service
duration and traveling time. We also check that all precedence constraints between
steps of the same event are respected as well. If this is not possible, the insertion is
being rejected. Otherwise, it is being characterized as a pending insertion, as it also
requires confirmation by the GSC in order to be accepted. This simple mechanism
corresponds to solving a relaxation of the problem where the coordination of resources
is not a requirement, and will allow us to perform batch insertions as explained in
the following section.

7.5.2 Batch Insertions

The idea behind batch insertions is to use the GSC in order to confirm multiple
insertions with a single reoptimization of the model. We start by performing a number
of accelerated insertions, as described in the previous section. The specific number of
insertions to be attempted is determined by the current batch size.

Once we have identified a set of insertions that have reached their pending state,
i.e., they have been approved for the relaxed version of the problem without the
coordination needs, we can now update the GSC by performing all changes that
are required for these insertions. The GSC reoptimization is then performed, and if
an optimal solution is obtained, all pending insertions are added successfully to the
solution. In case a feasible solution does not exist, all pending insertions are rejected.

The problem with this approach is that when a rejection occurs, we do not know
which insertion(s) caused the infeasibility of the solution. As a result, none of the
insertions can be accepted by the GSC at this point without another update and
reoptimization of the model taking place. Therefore, this approach can only be useful
if the batch size is selected carefully to ensure that the gain we obtain by reducing
the GSC reoptimization attempts through batch insertions compensates for the delays
caused by the rejections.
**Batch size selection**

It is obvious that the value of the batch size is very important for the success of this approach. If the batch size is large, more insertions can be achieved with a single GSC call, but the risk for rejections is very high. On the other hand, if the batch size is small, we lower the risk of rejection, but the gain of doing batch insertions is not that significant anymore, as we do not reduce by much the reoptimization attempts.

The optimal batch size depends also on the current schedules of the resources: if the schedules are sparse, i.e., each resource has fewer steps to perform, there is greater flexibility in the insertions, so the probability of them being successful is higher. This contrasts with the case where resources are already busy, so inserting many new steps successfully is less likely.

For this reason, we propose an adaptive batch size selection. In particular, in each iteration, we start with a large value for the batch size. Every time the GSC reoptimization is not successful, we reduce the batch size in half. This value is maintained until the next unsuccessful GSC call. This approach allows us to adjust the batch size based on the feedback we receive on its current value. Furthermore, it follows naturally the state of the schedules: in the beginning of the iteration when partial schedules contain less steps, we start with a large batch size which might be beneficial, but as more steps are inserted and schedules become busier, we reduce the batch size to make it more likely that insertions are successful.

**7.6 Computational Experiments**

**7.6.1 Experimental Setting**

To evaluate our methods, we focused on a variety of catering instances that were provided by the foodservice and support services company which motivated this work. Similar results can be obtained through the use of synthetic datasets as well.

*Orders*. Each instance corresponds to the events of a particular day and consists of up to 200 caterer requests. Figure 7-2 shows the distribution of the start times
of the events. Depending on the type of the event, various types of items might be required; for instance, bakery items, hot food, beverages, etc. Since not all unit items require the same capacity in the vehicles, a preprocessing step is performed to transform all item capacities into the same unit, which also facilitates the description of vehicles. In particular, hot food items take double the space as the rest of the item types, so each hot food item needs two capacity units of storage in the vehicles. The description of each event includes all the steps that are required for its successful organization, as well as the needed expertise of employees that should be assigned to each of them.

![Event Start Times](image)

**Figure 7-2:** Histogram of the average daily number of events per start time. There are three main start zones: morning 7am-9am (mostly breakfast events), mid-day 11:30am-12pm (lunch), and afternoon 2pm-3pm.

**Vehicles.** There are a total of 37 vehicles available. Of these, 30 are smaller vans, each of capacity 100 units of items, 5 are sprinters of capacity 500, and 2 are large boxtrucks with 1500 capacity.

**Employees.** We consider a variety of experience/skills for the employees. Each person might be a caterer, a senior caterer, a caterer lead, or an area manager. Furthermore, some people might only be eligible to perform event clean-up since they may not have the appropriate catering expertise to prepare an event. Each person might or might not have a driving license, which according to its type could allow them to drive the vans or the larger trucks. There is a total of 59 employees.
Parameter Selection

This section describes the parameter values that were selected for the experiments. The initial temperature parameter is set to 10% the objective value of the initial solution of the algorithm that is obtained before the iterations start. The temperature is multiplied by $\alpha = 0.999$ in each iteration. Regarding the adaptive weights, they are all initialized as 1, and at each new solution we reward the generation of a new best solution with 5, a better solution than the current with 2, an accepted solution with 1, and a rejected with 0. The value of $\gamma$ for the update of the weights is set to 0.5. In each destroy stage of the algorithm, we allow up to 20% of orders or vehicle schedules to be removed, and $V_{subsel}$ for the greedy insertion is set to 5. Finally, the batch size in each iteration is initialized to 8, as experiments showed that larger values were not beneficial. The results are reported for 5000 iterations, which for our instances require around 5 minutes to run.

7.6.2 Results

This section presents the results of our proposed method. The objective function in the experiments that follow has three components prioritized with appropriate weights: first maximize the number of served events with weight 100, then minimize the number of vehicles with weight 1, and, finally, minimize vehicles’ traveling time.
with weight 0.00001.

**Evolution of the Solution Quality**

Figure 7-4 illustrates how the quality of the solution changes during the execution of the algorithm. Notice that in the beginning when the temperature is larger then there is more experimentation as more solutions are accepted. As the iterations increase and the temperature gets smaller, fewer solutions are accepted, and the method focuses the search on more promising solutions.

**Batch Size**

Section 7.5 presents an accelerated method to perform step insertions which allows for the insertions to take place in batches. The size of the batch is adjusted dynamically during the execution of the algorithm. We start each iteration of the method with a batch size of 8, and, in each infeasible GSC optimization, we reduce it in half with a minimum value of 1. Figures 7-5 and 7-6 illustrate the insertion attempts and the success rate for each batch size.

![Average Percentage of Attempts](image1.png)

![Average Insertion Success](image2.png)

**Figure 7-5:** Average percentage of insertion attempts that corresponds to each batch size (8, 4, 2, and 1).

**Figure 7-6:** Average success rate of insertion for each batch size (8, 4, 2, and 1).
Figure 7-4: Quality of the solution per iteration of the method. We are reporting the current number of served events (in blue), the number of events served in the best solution so far (in orange), the current number of vehicles used (in grey), and the number of used vehicles in the best so far solution (in yellow).
Vehicle Fleet

Table 7.1 presents a breakdown of the objectives for a small number of instances. An observation that is consistent in all of our experiments is that the available fleet is much larger than required for the service of the customer requests. This might bring extra costs to the operator for maintaining and parking the vehicles. Therefore, using this optimization approach for the generation of the schedules, the operator can determine the appropriate fleet size that is required and reduce their operational costs accordingly. Furthermore, this approach can contribute to the system planning phase, by generating recommendations for the proper fleet size to the operators.

7.6.3 Discussion

One benefit of this approach is that it can be easily parallelized. In particular, the optimization can take place simultaneously on multiple threads, and this can significantly further speed up the running time. Moreover, this allows for random restarts of the algorithm which might be able to explore larger part of the solution space.

Section 7.4 covered the assignment of drivers to the vehicles that are required for the steps’ service. However, we have not discussed the transportation of the employees when they are not driving. In our application, the operator also owns passenger vans, so employees can use them for their transportation to and from the event locations. In a different application, employees might also use their personal vehicles to travel from location to location or might travel using the vehicles that carry the items. The latter case can be handled similarly to the driver assignment. In particular, it is not enough to assign an employee to a particular step; the transportation of the person
needs to be considered as well. This can be done similarly to the vehicle assignment for items, but we now need to consider the passenger capacity of the vehicles instead.
Chapter 8

Conclusion

8.1 Resource Optimization in Bike Sharing Systems

In the first part of this thesis, we focused on dynamic redistribution and demand estimation with an application in bike sharing systems. Since dynamic redistribution takes place simultaneously with user trips, we proposed a linear programming formulation that captures the user trips that are performed successfully in the system. With a novel mixed integer programming formulation and the development of a decomposition and optimization algorithm with partial group information, we were able to solve large real-world instances of the dynamic rebalancing problem in running times that were fast enough to allow real-time information to be taken into account. Evaluation of this approach on both synthetic and real-world data revealed the benefits of this method as well as the value of incorporating partial group information, especially when the stations have heterogeneous rebalancing needs.

Next, we also studied the demand estimation problem in vehicle sharing systems, since accurate demand estimation is crucial for the optimization of the system. We proposed a data-driven method for the estimation of lost user demand and showed it achieved lower errors with experiments on data from Capital Bikeshare, the bike-sharing system of Washington DC. We addressed the issue of demand relocation to neighboring stations (shifted demand) caused by lack of resources, and developed a linear programming formulation to estimate the probability with which people choose
to walk to nearby stations. Computational experiments validate this approach by showing its consistency across the network, as well as demonstrate the effects that the selected parameter values have on the walking probability.

Our proposed algorithms are not specific to bike sharing systems and can be generalized to other application domains. More specifically, the optimization with partial group information can be applied to the solution of any pick-up and delivery vehicle routing problem. Furthermore, its fast running times extent its applicability to online versions of these problems as well: with our method, online requests and real-time information can be easily taken into account with quick re-optimization of relevant aspects of the current plans. Evaluating the benefits of incorporating partial group information in more general pick-up and delivery applications is of particular interest and the focus of our future work in this area.

Some future directions in this field also include extending the methods to new models and systems that are currently emerging in the sharing economy. Some examples include dockless bike sharing systems, scooter-sharing, and electric vehicles. Approaches might include proper space discretization in order to consider small geographic regions in the place of bike stations, which will allow the application of methods similar to this work, but the development of other appropriately tailored techniques is also of interest. Finally, the incorporation of additional constraints and considerations, such as maintenance or uncertainty on various aspects of the operations, is also highly relevant to real-world systems and worth exploring efficient solutions that can scale well in practice.

\section*{8.2 Resource Management for Complex Event Scheduling}

The second part of this thesis focused on the scheduling of events of various types. In particular, we introduced the Complex Event Scheduling Problem (CESP), which combines vehicles and employees to serve various customer requests. More specif-
ically, each request might have a wide range of constraints, items that need to be transported to the event location, and skill requirements for the employees. The vehicle fleet is heterogeneous, and employees might have diverse skill sets. We proposed a representation of customer requests as a set of steps, each of which can then be served by different resources, while also allowing constraints between the steps, both temporal and coupling, in case steps need to be performed in a particular order or by the same resources. This representation provides great flexibility and is able to capture a large variety of operation types, such as pick-up and delivery, services provided at customer locations, and catering events.

We then developed a unique scheduler that can handle jointly all these operation types, which is a desirable result for many large companies that need to optimize their resource usage across multiple sub-organizations. Our approach is based on a combination of two main building blocks. The first is an adaptation of a metaheuristic framework (Adaptive Large Neighborhood Search) that considers a relaxed version of the problem focusing mostly on its combinatorial aspects without any coordination constraints. This is then combined with a linear programming formulation (Global Schedule Coordinator) that produces the schedule details and ensures their feasibility is maintained throughout the algorithm by checking that all constraints (e.g., time constraints, coordination) can be satisfied.

This hybrid approach manages to deal with the complexities of this problem, and along with some additional methods we developed to accelerate the total running time, is able to generate good quality solutions in a timely manner for real-world instances. Our algorithm can be used both to optimize the daily operations, such as ensure service coverage while minimizing the total cost, as well as help in strategic decisions, such as in the determination of the fleet size that is required for the typical customer demand.

Future directions that are of interest in this work include the incorporation of multiple transportation modes for the employees. In our setting, passenger vans were available to transfer the employees between event locations, but other operators might prefer to allow walking of the employees for close distances, the use of personal
vehicles, or other modes of transportation. Another interesting direction emerges from the vast amount of data that is recorded through vehicle tracking. This data can offer valuable insights on aspects such as travel time, and also allow for real-time adjustment of the plans to handle delays and disruptions.

8.3 Concluding Remarks

This thesis has focused on the development of models and algorithms for several complex transportation and logistics applications. On the modeling aspect, one of our main considerations has been to incorporate many real-life constraints and objectives in order to capture appropriately the complexities of real-world applications. As systems get more complicated, being able to account for many of their operational aspects becomes a challenging but important task.

Applicability to real-life systems has also been the main driving force behind the development of the algorithms presented throughout this work. In particular, our focus has been placed on designing solution methods that scale to the sizes of real-world systems by being able to generate good quality solutions in short running times. The timeliness of the solution is very important for many problem settings, especially for the ones that are dynamic in nature, such as the dynamic redistribution of bikes sharing systems presented in the first part of this work.

These approaches turn out increasingly useful, especially as the operations become more interconnected and systems get larger. Operators can benefit both in improving the quality of their services, by serving more customers and making their operations more sustainable, as well as in reducing their running costs. This is particularly crucial when the smooth functioning of the systems might have greater societal impact, as is the case in many systems in the sharing economy. Success of these systems can lead to their adoption in more cities, improving their quality of life and impacting favorably the environment.

Future steps in this area can focus on exploring further aspects that are faced by both the operators and the users of these systems in practice, as well as place
more emphasis on the data that can allow for more efficient operations; for instance, real-time data can help in the prevention and quick recovery from disruptions and plan deviations. Research progress in these directions will be especially useful for newer types of systems, as is the case for many systems in the shared mobility, where planning procedures are currently in an earlier phase and there is great margin for optimization and impact.
Appendix A

Appendices for Chapter 4

A.1 Omitted Details for Models (DR) and (DRPI)

A.1.1 Obtaining (DRPI) from (DR)

In this section, we present the changes that are required to obtain (DRPI) from (DR). We first rewrite constraint (4.9) of (DR), which is repeated here for convenience:

\[ \text{dur}_i \geq \tau_{\text{station}} \cdot (1 - x_{ii}) + \tau_{\text{action}} \cdot PU_i + \tau_{\text{action}} \cdot DO_i \quad \forall i \in S \]  \hspace{1cm} (A.1)

by using auxiliary variables \( \text{dur} PU^+_i \), \( \text{dur} PU^o_i \), \( \text{dur} DO^+_i \), \( \text{dur} DO^o_i \in \mathbb{R}^+ \), \( \forall i \), which denote the duration of the visit at \( i \) spent picking up bikes with positive rebalancing values and neutral rebalancing values, and similarly dropping off bikes with positive and neutral rebalancing values. The above constraints can then be written equivalently as:

\[ \text{dur}_i \geq \tau_{\text{station}} \cdot (1 - x_{ii}) + \text{dur} PU^+_i + \text{dur} PU^o_i + \text{dur} DO^+_i + \text{dur} DO^o_i \quad \forall i \in S \]  \hspace{1cm} (A.2)

\[ \text{dur} PU^+_i = \tau_{\text{action}} \cdot PU^+_i \quad \forall i \in S \]  \hspace{1cm} (A.3)

\[ \text{dur} PU^o_i = \tau_{\text{action}} \cdot PU^o_i \quad \forall i \in S \]  \hspace{1cm} (A.4)

\[ \text{dur} DO^+_i = \tau_{\text{action}} \cdot DO^+_i \quad \forall i \in S \]  \hspace{1cm} (A.5)

\[ \text{dur} DO^o_i = \tau_{\text{action}} \cdot DO^o_i \quad \forall i \in S \]  \hspace{1cm} (A.6)
Now, we are ready to introduce partial information for the actions of each group, using the results of Corollary 4.5.1. We keep constraints (A.2), but replace (A.3)-(A.6) with (A.7)-(A.10).

\[ P_U^+ \leq \min_i (\alpha_{p,i}^+ \cdot \text{dur} P_U^+ + \beta_{p,i}^+) \quad \forall i \in S \]  
\[ P_U^o \leq \min_i (\alpha_{p,i}^o \cdot \text{dur} P_U^o + \beta_{p,i}^o) \quad \forall i \in S \]  
\[ D_O^+ \leq \min_i (\alpha_{d,i}^+ \cdot \text{dur} D_O^+ + \beta_{d,i}^+) \quad \forall i \in S \]  
\[ D_O^o \leq \min_i (\alpha_{d,i}^o \cdot \text{dur} D_O^o + \beta_{d,i}^o) \quad \forall i \in S \]

In the following section, we describe how we can obtain the coefficients \( \alpha \) and \( \beta \) that are used in (A.7)-(A.10).

### A.1.2 Additional Details for Coefficients \( \alpha \) and \( \beta \) for (DRPI)

As illustrated in constraints (A.7)-(A.10), for each group \( i \), we need four sets of coefficients: \((\alpha_{p,i}^+, \beta_{p,i}^+)\) for \( P_U^+ \), \((\alpha_{p,i}^o, \beta_{p,i}^o)\) for \( P_U^o \), \((\alpha_{d,i}^+, \beta_{d,i}^+)\) for \( D_O^+ \), and \((\alpha_{d,i}^o, \beta_{d,i}^o)\) for \( D_O^o \). Sections 4.5.1 and 4.5.2 outline how these coefficients can be obtained based on the pick-up and drop-off needs of the stations.

Indeed, pick-up and drop-off needs are used to obtain the coefficients for the positive value rebalancing actions \( P_U^+ \) and \( D_O^+ \). In the case of neutral value rebalancing actions, we need to consider the remaining resources for each station, i.e. resources that are both not used by customers and not offering positive rebalancing value. Then, a process similar to Example 4.5.2 of Section 4.5.2 can be followed to obtain the line slopes and intercepts.

Since a station might be able to offer resources of both positive and neutral rebalancing value, in order to avoid counting the travel time \( \tau_{\text{travel}} \) twice, for these stations we take \( \tau_{\text{travel}} \) into account only in the line derivation for the positive actions and we skip it for the neutral ones, i.e. the slope is equal to 1 for these stations in the latter case.
A.1.3 Overhead Time per Visit $\tau_{\text{station}}$

In both (DR) and (DRPI), we assume that each visit requires an overhead time $\tau_{\text{station}}$ that corresponds to parking, etc. This parameter appears in constraint (4.9) or equivalently constraint (A.2), and this overhead time is being enforced on every truck visit. However, there are cases where the truck starts its route being already at the station that constitutes its first visit. In this case, we wish to avoid having to pay the $\tau_{\text{station}}$ value, and a way to do that is using the following constraints instead for the station $i$ that is the initial truck location $i_0$:

\[
\text{dur}_i \geq \tau_{\text{station}} \cdot (1 - x_{ii} - x_{0i}) + \tau_{\text{action}} \cdot PU_i + \tau_{\text{action}} \cdot DO_i \quad \text{if } i = i_0 \quad (A.11)
\]

\[
\text{dur}_i \geq \tau_{\text{station}}(1 - x_{ii} - x_{0i}) + \text{dur} PU_i^r + \text{dur} PU_i^o + \text{dur} DO_i^r + \text{dur} DO_i^o \quad \text{if } i = i_0
\]

(A.12)

Variables $x_{0i}$ are used for the dummy node that corresponds to the truck: $x_{0i} = 1$, if the truck travels from its initial location directly to $i$, and 0 otherwise. The use of $x_{0i}$ in the above constraints results in not paying $\tau_{\text{station}}$ in the cases where the truck is already at the location of its first visit.

A.2 Planning over Multiple Rebalancing Periods

A.2.1 Overhead Time per Visit $\tau_{\text{station}}$

Similar to Appendix A.1.3, we can make sure we do not consider the overhead time $\tau_{\text{station}}$ multiple times for the same visit, by further specifying (4.39) to take into account the truck transitioning to the next period, since it remains at the same location:

\[
\text{dur}_i^r \geq \tau_{\text{station}} \cdot (1 - x_{ii}^r - y_{i}^{r-1}) + \tau_{\text{action}} \cdot PU_i^r + \tau_{\text{action}} \cdot DO_i^r \quad \forall i \in S, r > 1 \quad (A.13)
\]
as well as the station $i$ that constitutes the initial location $i_0$ of the truck during the first rebalancing period:

$$
dur_i^r \geq \tau_{station} \cdot (1 - x_{ri}^r - x_{0i}^r) + \tau_{action} \cdot PU_i^r + \tau_{action} \cdot DO_i^r \quad \text{if } i = i_0, r = 1 \tag{A.14}
$$

For $r = 1$ and $i \neq i_0$, (4.39) remains as is.

### A.2.2 (MDR) with Partial Information

The incorporation of partial information to (MDR) can take place in a similar way as in Appendix A.1.1, and this generates the Multiperiod Dynamic Rebalancing with Partial Information model (MDRPI).

### A.3 Computational Results

This section includes detailed statistics of the solutions in the Computational Experiments Section 4.6.2. Unless noted otherwise, they refer to the rebalancing actions that take place until noon in each instance. The rebalancing gain is the extra number of customers that can be served compared to the case where no rebalancing is performed and the stations start the day half full. The travel distance is in kilometers, the travel and service times in minutes, and the running time in seconds. All quantities are aggregated for all trucks and time periods.
<table>
<thead>
<tr>
<th>Inst.</th>
<th>Method</th>
<th>Served users</th>
<th>Rebalancing gain</th>
<th>Bikes rebalanced</th>
<th>Travel distance</th>
<th>Travel time</th>
<th>Service time</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>3207.48</td>
<td>281.89</td>
<td>326.29</td>
<td>263.96</td>
<td>791.88</td>
<td>532.29</td>
<td>3550.88</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3214.63</td>
<td>289.04</td>
<td>377.78</td>
<td>297.37</td>
<td>892.11</td>
<td>606.79</td>
<td>548.83</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3222.22</td>
<td>296.62</td>
<td>363.20</td>
<td>286.17</td>
<td>858.51</td>
<td>572.20</td>
<td>561.01</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>3527.41</td>
<td>326.09</td>
<td>426.67</td>
<td>275.68</td>
<td>827.05</td>
<td>648.67</td>
<td>4195.29</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3534.31</td>
<td>333.00</td>
<td>485.98</td>
<td>309.07</td>
<td>927.21</td>
<td>764.98</td>
<td>752.01</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3572.71</td>
<td>371.40</td>
<td>479.71</td>
<td>320.06</td>
<td>960.18</td>
<td>741.72</td>
<td>722.60</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>3637.30</td>
<td>354.91</td>
<td>446.95</td>
<td>323.98</td>
<td>971.94</td>
<td>675.95</td>
<td>4223.83</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3693.63</td>
<td>411.24</td>
<td>498.66</td>
<td>347.47</td>
<td>1042.41</td>
<td>778.66</td>
<td>543.12</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3699.15</td>
<td>416.76</td>
<td>487.59</td>
<td>356.84</td>
<td>1070.51</td>
<td>751.59</td>
<td>617.78</td>
</tr>
<tr>
<td>4</td>
<td>S</td>
<td>3467.68</td>
<td>339.37</td>
<td>475.45</td>
<td>276.75</td>
<td>830.26</td>
<td>711.45</td>
<td>3571.24</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3504.70</td>
<td>376.39</td>
<td>546.24</td>
<td>289.54</td>
<td>868.63</td>
<td>828.24</td>
<td>534.52</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3508.42</td>
<td>380.11</td>
<td>533.85</td>
<td>297.38</td>
<td>892.13</td>
<td>799.86</td>
<td>588.18</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>3123.00</td>
<td>257.61</td>
<td>332.24</td>
<td>262.39</td>
<td>787.16</td>
<td>543.24</td>
<td>3589.69</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3133.53</td>
<td>268.14</td>
<td>379.32</td>
<td>278.71</td>
<td>836.12</td>
<td>615.32</td>
<td>542.66</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3162.12</td>
<td>296.73</td>
<td>376.89</td>
<td>289.88</td>
<td>869.65</td>
<td>616.90</td>
<td>589.37</td>
</tr>
<tr>
<td>6</td>
<td>S</td>
<td>3328.63</td>
<td>361.15</td>
<td>437.51</td>
<td>277.07</td>
<td>831.21</td>
<td>659.51</td>
<td>3533.19</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3338.14</td>
<td>370.65</td>
<td>510.62</td>
<td>304.90</td>
<td>914.69</td>
<td>786.62</td>
<td>560.72</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3358.92</td>
<td>391.43</td>
<td>515.45</td>
<td>298.99</td>
<td>896.98</td>
<td>778.46</td>
<td>609.03</td>
</tr>
<tr>
<td>7</td>
<td>S</td>
<td>3406.20</td>
<td>371.73</td>
<td>440.75</td>
<td>291.81</td>
<td>875.42</td>
<td>663.75</td>
<td>3903.83</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3408.13</td>
<td>373.67</td>
<td>498.18</td>
<td>320.32</td>
<td>960.97</td>
<td>762.18</td>
<td>547.78</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3442.52</td>
<td>408.05</td>
<td>494.34</td>
<td>323.42</td>
<td>970.26</td>
<td>749.34</td>
<td>602.76</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>3114.34</td>
<td>281.96</td>
<td>384.51</td>
<td>241.41</td>
<td>724.23</td>
<td>603.51</td>
<td>3465.21</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3127.75</td>
<td>295.37</td>
<td>444.36</td>
<td>256.81</td>
<td>770.44</td>
<td>712.36</td>
<td>482.50</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3143.35</td>
<td>310.96</td>
<td>423.33</td>
<td>264.32</td>
<td>792.96</td>
<td>659.34</td>
<td>535.93</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>3167.63</td>
<td>270.72</td>
<td>339.25</td>
<td>262.23</td>
<td>786.68</td>
<td>550.25</td>
<td>4111.02</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3208.47</td>
<td>311.57</td>
<td>411.87</td>
<td>287.54</td>
<td>862.61</td>
<td>649.87</td>
<td>532.29</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3203.84</td>
<td>306.94</td>
<td>396.49</td>
<td>298.57</td>
<td>895.71</td>
<td>640.49</td>
<td>591.51</td>
</tr>
<tr>
<td>10</td>
<td>S</td>
<td>3565.88</td>
<td>352.60</td>
<td>466.47</td>
<td>326.99</td>
<td>980.96</td>
<td>717.47</td>
<td>3998.96</td>
</tr>
<tr>
<td></td>
<td>G/L</td>
<td>3598.69</td>
<td>385.41</td>
<td>501.53</td>
<td>340.61</td>
<td>1021.82</td>
<td>801.53</td>
<td>553.99</td>
</tr>
<tr>
<td></td>
<td>G/PI</td>
<td>3594.39</td>
<td>381.12</td>
<td>509.23</td>
<td>334.67</td>
<td>1004.01</td>
<td>776.23</td>
<td>669.61</td>
</tr>
</tbody>
</table>

Table A.1: Rebalancing results for method comparison: S [(DR) on the set of stations], G/L [(DR) on the set of leaders], G/PI [(DRPI)].
Table A.2: Rebalancing results for single-period rebalancing (DRPI) \((RP = 1)\) and multi-period rebalancing (MDRPI) with \(RP = 2\).
<table>
<thead>
<tr>
<th>Instance</th>
<th>Neutral actions</th>
<th>Served users</th>
<th>Rebalancing gain</th>
<th>Bikes rebalanced distance</th>
<th>Travel time</th>
<th>Service time</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True</td>
<td>3222.22</td>
<td>296.62</td>
<td>363.20</td>
<td>286.17</td>
<td>858.51</td>
<td>572.20</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3029.70</td>
<td>104.11</td>
<td>98.14</td>
<td>187.21</td>
<td>561.63</td>
<td>198.14</td>
</tr>
<tr>
<td>2</td>
<td>True</td>
<td>3572.71</td>
<td>371.40</td>
<td>479.71</td>
<td>320.06</td>
<td>960.18</td>
<td>741.72</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3386.46</td>
<td>185.15</td>
<td>160.93</td>
<td>271.20</td>
<td>813.59</td>
<td>309.94</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
<td>3699.15</td>
<td>416.76</td>
<td>487.59</td>
<td>356.84</td>
<td>1070.51</td>
<td>751.59</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3484.33</td>
<td>201.94</td>
<td>148.10</td>
<td>268.87</td>
<td>806.61</td>
<td>285.10</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
<td>3508.42</td>
<td>380.11</td>
<td>533.85</td>
<td>297.38</td>
<td>892.13</td>
<td>799.86</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3301.77</td>
<td>173.46</td>
<td>168.96</td>
<td>212.69</td>
<td>638.08</td>
<td>301.96</td>
</tr>
<tr>
<td>5</td>
<td>True</td>
<td>3162.12</td>
<td>296.73</td>
<td>376.89</td>
<td>289.88</td>
<td>869.65</td>
<td>616.90</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>2983.64</td>
<td>118.25</td>
<td>90.76</td>
<td>171.82</td>
<td>515.46</td>
<td>173.77</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>3358.92</td>
<td>391.43</td>
<td>515.45</td>
<td>298.99</td>
<td>896.98</td>
<td>778.46</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3159.55</td>
<td>192.06</td>
<td>166.19</td>
<td>244.33</td>
<td>732.98</td>
<td>304.20</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>3442.52</td>
<td>408.05</td>
<td>494.34</td>
<td>323.42</td>
<td>970.26</td>
<td>749.34</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3254.62</td>
<td>220.16</td>
<td>178.90</td>
<td>234.12</td>
<td>702.36</td>
<td>305.91</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>3143.35</td>
<td>310.96</td>
<td>423.33</td>
<td>264.32</td>
<td>792.96</td>
<td>659.34</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>2955.20</td>
<td>122.81</td>
<td>111.28</td>
<td>156.89</td>
<td>470.67</td>
<td>216.28</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>3203.84</td>
<td>306.94</td>
<td>396.49</td>
<td>298.57</td>
<td>895.71</td>
<td>640.49</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3007.05</td>
<td>110.14</td>
<td>89.08</td>
<td>180.56</td>
<td>541.68</td>
<td>180.08</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>3594.39</td>
<td>381.12</td>
<td>509.23</td>
<td>334.67</td>
<td>1004.01</td>
<td>776.23</td>
</tr>
<tr>
<td></td>
<td>False</td>
<td>3402.69</td>
<td>189.41</td>
<td>177.70</td>
<td>237.45</td>
<td>712.34</td>
<td>321.71</td>
</tr>
</tbody>
</table>

Table A.3: Rebalancing results comparing the benefit of neutral value rebalancing actions.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Served users</th>
<th>Rebalancing gain</th>
<th>Bikes rebalanced distance</th>
<th>Travel time</th>
<th>Service time</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10749.45</td>
<td>1375.52</td>
<td>455.87</td>
<td>445.89</td>
<td>1337.68</td>
<td>834.87</td>
</tr>
<tr>
<td>2</td>
<td>12858.70</td>
<td>1929.27</td>
<td>681.68</td>
<td>592.85</td>
<td>1778.54</td>
<td>1183.69</td>
</tr>
<tr>
<td>3</td>
<td>12532.99</td>
<td>1925.33</td>
<td>691.02</td>
<td>667.61</td>
<td>2002.83</td>
<td>1250.02</td>
</tr>
<tr>
<td>4</td>
<td>11453.28</td>
<td>1872.85</td>
<td>651.61</td>
<td>588.83</td>
<td>1766.48</td>
<td>1130.61</td>
</tr>
<tr>
<td>5</td>
<td>11115.80</td>
<td>1389.29</td>
<td>453.77</td>
<td>457.79</td>
<td>1373.36</td>
<td>841.77</td>
</tr>
<tr>
<td>6</td>
<td>11986.09</td>
<td>1892.76</td>
<td>600.09</td>
<td>623.04</td>
<td>1869.12</td>
<td>1137.09</td>
</tr>
<tr>
<td>7</td>
<td>13033.88</td>
<td>1995.86</td>
<td>727.58</td>
<td>569.19</td>
<td>1707.58</td>
<td>1224.58</td>
</tr>
<tr>
<td>8</td>
<td>7099.40</td>
<td>1158.66</td>
<td>297.04</td>
<td>308.61</td>
<td>925.82</td>
<td>578.04</td>
</tr>
<tr>
<td>9</td>
<td>12100.82</td>
<td>1602.95</td>
<td>646.40</td>
<td>566.54</td>
<td>1699.62</td>
<td>1145.40</td>
</tr>
<tr>
<td>10</td>
<td>13142.20</td>
<td>2072.26</td>
<td>760.87</td>
<td>566.25</td>
<td>1698.76</td>
<td>1269.87</td>
</tr>
</tbody>
</table>

Table A.4: Results when dynamic rebalancing is combined with ideal static rebalancing. These results refer to the duration of the whole day.
Table A.5: Served users without dynamic rebalancing. Number of served customers for half day (until noon) and the whole day when stations’ initial state is half-full or at its ideal level (obtained by (UT) and representing the results of an ideal static redistribution in the system).

<table>
<thead>
<tr>
<th>Instance</th>
<th>Half day</th>
<th>Whole day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Half-full</td>
<td>Ideal</td>
</tr>
<tr>
<td>1</td>
<td>2925.59</td>
<td>3466.75</td>
</tr>
<tr>
<td>2</td>
<td>3201.31</td>
<td>4057.04</td>
</tr>
<tr>
<td>3</td>
<td>3282.39</td>
<td>4169.91</td>
</tr>
<tr>
<td>4</td>
<td>3128.31</td>
<td>3994.23</td>
</tr>
<tr>
<td>5</td>
<td>2865.38</td>
<td>3415.82</td>
</tr>
<tr>
<td>6</td>
<td>2967.49</td>
<td>3822.17</td>
</tr>
<tr>
<td>7</td>
<td>3034.47</td>
<td>3867.97</td>
</tr>
<tr>
<td>8</td>
<td>2832.38</td>
<td>3458.28</td>
</tr>
<tr>
<td>9</td>
<td>2896.90</td>
<td>3499.44</td>
</tr>
<tr>
<td>10</td>
<td>3213.28</td>
<td>4084.85</td>
</tr>
</tbody>
</table>
Bibliography


O’Mahony, E. (2015), ‘Smarter tools for (citi) bike sharing’.


The Bike-sharing World Map (2016), https://www.google.com/maps/d/u/0/viewer?ll=43.580391\&zC-42.890625\&spn=143.80149\%2C154.6875\&hl=en\&msa=0\&z=1\&source=embed\&ie=UTF8\&om=1\&mid=zGPtSU9zZvZw.kmqv\_ul1MfKI. Accessed: April 2016.


Vogel, P. and Mattfeld, D. C. (2010), Modeling of repositioning activities in bike-sharing systems, in ‘World Conference on Transport Research (WCTR)’.
