NTNU
Norwegian University of
Science and Technology

Sigrid Fosen, Øyvor Salvesen Haldorsen

2020

Master's thesis

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Economics and Management
Department of Industrial Economics and Technology
Management

Sigrid Fosen
Øyvor Salvesen Haldorsen

Rebalancing and Charging Strategies in Electric Bike Sharing Systems

A Two-Stage Stochastic Programming Column Generation Heuristic

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Rebalancing and Charging Strategies in Electric Bike Sharing Systems

A Two-Stage Stochastic Programming Column Generation Heuristic

Sigrid Fosen
Øyvor Salvesen Haldorsen

Industrial Economics and Technology Management
Submission date: June 2020
Supervisor: Kjetil Fagerholt
Co-supervisor: Henrik Andersson

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Abstract

This thesis explores the utility of combining battery swaps and bike relocation efforts by means of service vehicles in an electric bike sharing system (BSS). In an electric BSS, electric bikes are available for short-term rent between stations located in a defined geographical area. With time, customer trips cause potential problems in a BSS related to e.g. flat batteries or accumulation of bikes in some parts of the system. This prompts the need for effective charging solutions and bike relocation strategies to maintain an efficient BSS that displays high levels of customer satisfaction.

There are several possible approaches to maintain an efficient electric BSS; this thesis focuses on how a charged and balanced system may be achieved using battery swaps in combination with relocation of bikes between charging and non-charging stations. The problem is defined as the Dynamic Stochastic Bicycle Rebalancing and Charging Problem (DSBRCP). In previous work conducted by Fosen and Haldorsen (2019) on a simpler version of the problem, we know that exact solution methods are incapable of solving this problem for fully sized systems within reasonable time. Therefore, this thesis introduces and implements a two-stage stochastic programming column generation heuristic capable of solving this problem for real-world instances. This heuristic aims to be of value as a decision making tool when organizing electric BSSs.

The DSBRCP heuristic is composed of three main steps; 1) initializing routes and patterns, 2) solving subproblems, and 3) solving a master problem. The first step is implemented heuristically, while the second and third step find optimal solutions. In the first step, a set of routes and pattern combinations are generated using a branching algorithm. For each of the initialized route and pattern combinations, a score for the combination is found by solving a subproblem in step two. Together, the first and second step construct a column that is passed to the master problem. The master problem combines these columns to find the optimal first stage decisions for each vehicle.

An overview of previously published literature related to the topic is included in this thesis. To the author's knowledge, the DSBRCP has not been studied before. However, it has significant overlaps with the more explored *bicycle rebalancing problem*, which solves a less complex rebalancing problem in traditional BSSs, where charging decisions are excluded. The main distinctions between the present problem and the rebalancing problem are the introduction of depot, charging stations, and battery swap activities. Thus, our contribution is to solve the extended rebalancing problem that also incorporates charging decisions by means of battery swaps and charging stations.

Data from the BSS in Oslo, operated by Urban Infrastructure Partners (UIP), is used to create a simulation framework reflecting a real-world test environment for the two-stage stochastic programming column generation heuristic. Using this framework, a computational study is conducted to assess solution time and discover the optimal parameter configurations. From the study, we observe that the solution time is highly dependent on the required number of subproblems to solve, which is driven by the number of service vehicles, the number of subproblem scenarios, and the branching factor when initializing routes.

The simulation framework is also used to provide managerial insights and evaluate the performance of service vehicles guided by heuristic decisions. To evaluate this, the number of violations occurring during a day when employing the heuristic is compared to employing alternative strategies on the same day. Two alternative strategies are introduced: a strategy with no operations and a strategy inspired by UIP's current operations. From the analyses, we observe that the DSBRCP heuristic is able to prevent 50% more violations in a day than the UIP-inspired strategy when employing five service vehicles. The improvement corresponds to a 23.3% reduction in daily violations when compared to a system with no service vehicle operations.

Sammendrag

Denne oppgaven undersøker hvordan man kan maksimere kundetilfredshet i et elektrisk bysykkelsystem ved bruk av servicebiler som utfører batteribytter og rebalansering av sykler. I et elektrisk bysykkelsystem tilbys elektriske sykler for korttidsleie mellom stasjoner innenfor et bestemt geografisk område. Gjennomførte turer kan potensielt forårsake problemer i et bysykkelsystem, for eksempel ved at stasjoner tømmes for sykler, eller at stasjoner fylles opp og hindrer andre kunder fra å parkere. For å redusere forekomsten av slike tilfeller, benyttes gjerne et sett av servicebiler som kan reallokere sykler mellom stasjoner i løpet av dagen. Behovet for operasjonell beslutningsstøtte er større i et elektrisk bysykkelsystem sammenlignet med et tradisjonelt bysykkelsystem, da beslutningene også skal sørge for at syklene er ladet til enhver tid.

Oppgaven fokuserer på hvordan et ladet og balansert system kan oppnås ved bruk av batteribytter i kombinasjon med rebalansering av sykler mellom ladestasjoner og stasjoner uten lademulighet. Problemet som løses er definert som *the Dynamic Stochastic Bicycle Rebalancing and Charging Problem (DSBRCP)*. Tidligere arbeid utført av Fosen og Haldorsen (2019) viser at eksakte løsningsmetoder ikke skalerer godt nok for å kunne løse problemet for realistiske størrelser av bysykkelsystemer innen rimelig tid. Basert på dette introduserer og implementerer vi en to-stegs stokastisk kolonnegenereringsheuristikk, som gjør det mulig å løse DSBRCP for bysykkelsystemet i Oslo på under ett minutt. Heuristikken skal fungere som et beslutningsverktøy for operasjonelle beslutninger i elektriske bysykkelsystemer.

DSBRCP-heuristikken består av tre steg: (1) Initialisere ruter og *patterns*, (2) Løse subproblem og (3) Løse masterproblemet. Første steg er implementert heuristisk, mens andre og tredje trinn løses til optimalitet. I første steg genereres et sett av rute- og pattern-kombinasjoner ved bruk av en forgreningsalgoritme. Hver av de initialiserte kombinasjonene får en poengsum ved å løse subproblemet for et bestemt scenario av kundeankomster i steg to. Kombinasjonen av rute, pattern og poengscore fra subproblemet utgjør en kolonne. Masterproblemet i trinn tre kombinerer kolonnene for å finne de optimale beslutningene for hver servicebil, koordinert på tvers av alle biler.

Som inspirasjon til utvikling av heuristikken, presenteres en oversikt over tidligere publisert litteratur relatert til emnet. Forfatterne er ikke kjent med at DSBRCP er studert tidligere. Problemet overlapper imidlertid markant med det mye utforskede rebalanseringsproblemet i tradisjonelle bysykkelsystemer. De viktigste forskjellene mellom rebalanseringsproblemet i tradisjonelle bysykkelsystemer og DSBRCP er innføringen av depot, ladestasjoner og batteribytteaktiviteter – og implikasjonene dette får for de operasjonelle beslutningene i sin helhet. Vårt bidrag med denne masteroppgaven er dermed å utvide rebalanseringsproblemet til å også inkludere ladebeslutninger.

Data fra bysykkelsystemet i Oslo, som opereres av Urban Infrastructure Partners (UIP), er brukt til å lage et simuleringsrammeverk. Simuleringen skal etterligne realistiske kundeinteraksjoner i et bysykkelsystem i løpet av en dag. Dette brukes som et hjelpemiddel for å bedømme den langsiktige kvaliteten på operasjonelle beslutninger som gjennomføres i systemet. Dette gjøres ved å måle antall kundeforespørsler som ikke kan oppfylles i løpet av en dag. Et eksempel på et slikt tilfelle er en kunde som ønsker å starte en tur fra en stasjon uten sykler. Simuleringsrammeverket gir også mulighet for å vurdere alternative lade- og rebalanseringsstrategier mot hverandre. I tillegg til den to-stegs stokastisk kolonnegenereringsheuristikken (DSBRCP heuristikk), introduseres to alternative heuristikker; en strategi uten bruk av servicebiler, og en strategi inspirert av UIPs nåværende tilnærming til rebalansering. Analyser viser at DSBRCP-heuristikken er i stand til å forhindre 50% flere tilfeller av uoppfylte kundeforespørsler målt mot den UIP-inspirerte strategien når fem bilder opererer i systemet. Forbedringen tilsvarer en reduksjon på 23.3% fra et system uten drift av servicebiler.

Simuleringsrammeverket brukes også til å gi innsikt i strategiske og taktiske beslutninger som må fattes i et bysykkelsystem. Dette gjelder for eksempel bestemmelse av antall servicebiler som benyttes i systemet til enhver tid, og effekten av ladestasjoner i systemet. Målet er at denne innsikten skal kunne anvendes i bysykkelsystemer generelt.

Preface

This thesis concludes our Master of Science at the department of Industrial Economy and Technology Management at the Norwegian University of Science and Technology. It is written during the spring semester of 2020. The thesis builds on the work conducted during our specialization project within TIØ4500 Managerial Economics and Operations Research in the fall of 2019.

A special thanks to our supervisors Professor Kjetil Fagerholt and Professor Henrik Andersson for their valuable guidance throughout the project. We would also like to thank Urban Infrastructure Partners (UIP) and Hans Martin Espegren for providing necessary data and useful industry knowledge.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The world faces an increasing rate of urbanization, causing frequent traffic congestions and high emission quantities in the bigger cities (Rao, 2012). As a response to these problems, bicycling as a mode of transport has rapidly become very popular. Usage of bicycles has also been incorporated into the sharing economy, leading to a growing numbers of bike sharing systems (BSS) worldwide. Over time, these BSSs have introduced new features, including the introduction of electric bikes. Electric BSSs provide benefits such as shorter travelling times and less physical requirements for the customers, but also introduce challenges for the operator related to battery charging.

The concept of a bike sharing system (BSS) is as follows: the user picks up an available bike at a start station, and rides it to the desired end station where the bicycle is locked. With increasing digitization and technological advances, the operational model for emerging BSSs evolves constantly. The focus of this thesis is on electric, station based systems. This architecture poses two potential challenges for the user; (i) No available bikes with sufficiently charged batteries at the start station, and (ii) No available locks at the end station. This thesis refers to situation (i) as *starvation*, and (ii) as *congestion*. These situations prompt the need for a balanced system, where the available charged bikes and locks reflect the demand. The problem of finding a good relocation strategy of bikes in a BSS is commonly referred to as the *rebalancing strategy problem*. In this thesis, we address the problem of meeting customer demand by securing a charged fleet of bicycles through swapping of batteries and relocation of bicycles, which can be seen as an extension of the rebalancing problem. This problem is referred to as the *Dynamic Stochastic Bicycle Rebalancing and Charging Problem* (DSBRCP).

This thesis is written in collaboration with Urban Infrastructure Partners (UIP). UIP currently runs pilot projects with small-scale electric BSSs in Norway, and provides software assistance to electric bike sharing systems in other European cities. The company has provided the data for the present research, where Oslo Bike sharing system is used as an example case. Although these bicycles are traditional bicycles without batteries, the demand patterns are assumed to be similar to that of electric bikes.

As the electric BSSs become more common and larger in size, the need for an advanced decision support system with optimized set-ups for bicycle charging and relocation has emerged. Only addressing the rebalance problem could prove to be efficient in a BSS consisting of numerous charging stations. However, charging solutions at stations, require steady supply of electricity, which limits the possible station locations (Shuguang et al., 2014). Introducing rebalancing vehicles with possibility for battery swapping eliminates this problem. This approach also allows for more flexibility, preparing the ground for emerging inventions within BSSs like free floating systems.

The objectives of this thesis are to 1) Conduct a literature survey on relevant optimization problems and solution methods addressing the bicycle rebalancing problem and other shared mobility charging problems, 2) Develop a solution algorithm that solves the DSBRCP for real-sized systems that may be integrated as part of operational decision making tools in electric BSSs, 3) Create a simulation environment that can assess the performance of different solution strategies to the DSBRCP using real world data, and 4) Perform analyses providing managerial insight relevant for decision-making at UIP. To our knowledge, existing literature and decision making tools addressing rebalancing in a BSS, do not incorporate charging strategies. Thus, the suggested solution of the DSBRCP presented in this thesis, will contribute by providing a flexible decision tool based on optimization of rebalancing and charging efforts together.

This thesis starts with an introductory Background Chapter, elaborating on the concept and the development of BSSs from its introduction in 1965 until today. This is followed by a literature survey in Chapter 3. The survey presents different models for bike sharing problems, and aims at synthesizing the most important aspects to take into account when modeling the DSBRCP. Much focus is directed towards literature on the bike rebalancing problem, as these problems share central aspects and modeling components. Further, a thorough problem description is given in Chapter 4.

Chapter 5 presents a two-stage stochastic programming column generation heuristic as a solution algorithm to the DSBRCP. It also includes reasoning behind the chosen solution approach and important concepts and simplifying assumptions. Following, is a detailed description of the simulation framework in Chapter 6. Chapter 7 explains how the test instances are developed based on data collected from UIP. This collection is important in order to derive meaningful results that can be evaluated intuitively and computationally.

A computational study for parameter tuning is provided in Chapter 8, where all input parameters are set. The following chapter, Chapter 9, elaborates on operational insights based on observations from results retrieved using the established configuration. Further, Chapter 10 provides concluding remarks for the thesis, before possibilities of future research are introduced in Chapter 11.

Chapter 2

Background

This chapter provides an overview of the bike sharing concept and how it has evolved since it was first introduced. Section 2.2 provides an overview of the concept today, followed by a study of the history of BSS in Section 2.2. Further, Section 2.3 take a close look at how bike sharing systems are operated around the world. Finally, future challenges of BSS are addressed in section 2.5.

2.1 Bike Sharing Concept

Modern BSSs provide short-term rental of bikes within a defined geographic area. Typically, stations are spread out in strategic locations, offering bikes and locks for customers to use. Bike rental is usually offered through a mobile application, requiring users to create an account with identification and payment details. The application enable users to get a real-time overview of available bikes and stations for use in the area, while also providing the operators with valuable data. In order to rent a bike, the user must follow the interaction scheme shown in Figure 2.1. The dark blue colored steps are only performed once, while the lighter blue colored steps must be conducted each time the user rents a bicycle.

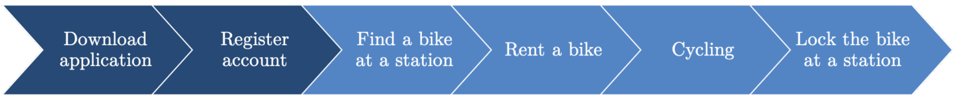


Figure 2.1: Scheme of the user interaction in a station based bike-sharing system

Public bikes are often used as a supplement to traditional public transportation, offering first- and last-mile transportation opportunities. Therefore, stations are usually placed nearby residential areas or in connection with public transport hubs. The movements of users between stations are unpredictable. It is therefore common that the operators use service vehicles to redistribute bikes in the system to avoid full or empty stations. As the demand numbers are uncertain, two important challenges for an operator are the decisions of (i) The location of stations, and (ii) The redistribution strategy of bicycles in the system. The latter is crucial to avoid full or empty stations, implying that customers cannot leave or retrieve a bike at their preferred station.

BSSs are associated with social, environmental, and health benefits, including emission and congestion reductions, more flexible mobility, consumer financial savings, and positive public health impact (Midgley, 2011). These benefits make a continued upturn of emerging BSSs likely, especially in light of the increased focus on environmental issues and global warming.

From an economic perspective, a BSS is seldom self-funding and often relies on government subsidies and/or advertisement on the bikes (Zhang, 2015). The investment, maintenance, and operational costs of a BSS are significant and highly depend on the fleet size, service area, number of stations and technology used (DeMaio, 2008). For most operators, rebalancing represent the most substantial cost.

The growth of bike sharing systems with electric bikes has been significant over the past couple of years. Electric bikes provide an even higher level of service compared to BSSs with traditional bikes, while still maintaining a low impact on the environment. With the extra boost supplied by a battery, customers are able to travel longer distances with higher incline and less fatigue and sweat compared to traditional bikes. Electric bikes are also regulated similar to traditional bikes in most jurisdictions and often outperforms other motorized transport modes, including bus transit, on travel time and energy efficiency on shorter distances. However, e-bikes are significantly more expensive than non-electric bikes. By spreading the higher cost of e-bikes over a community of users over time, e-bike sharing has a higher potential of overcoming the price barrier created by the battery technology premium than the private market (Shuguang et al., 2014). Even though providing additional benefits, a new challenge arises in electric BSSs, namely how to maintain sufficient battery levels on the bikes in the system.

2.2 Bike Sharing Evolution

BSSs have seen great improvements since the first system was introduced in 1965. This section examines the evolution of system features in a historical perspective, as well as providing an in-depth description of the most recently introduced technology.

2.2.1 History and Generations

The first generation BSS was initiated by the Dutch Luud Schimmelpennink in 1965. The original system aimed at reducing air pollution and traffic congestion in the city, and was quite primitively shaped: Schimmelpennink painted bikes white and placed them across the city for people to use. This system enabled customers to pick up a bike and ride it to their destination, leaving it available for others. Unfortunately, Schimmelpennink's *White Plan* system failed due to theft and vandalism of the bikes (Cripps, 2013).

The first large-scale second generation program was established in Copenhagen in 1995, called *Bycyklen*. The system incorporated measures to conserve the bicycles, learning from Schimmelpennink's first failure. In addition, it had physical stations and coin deposits where customers paid a small deposit when renting a bike. It was mandatory to return the bike at a station after use. This second generation BSS also included advertisement on the bikes. However, this system did not see the desired effect on theft and vandalism, as the users remained anonymous (DeMaio, 2008).

The animosity problem was addressed by the third generation bike sharing program, introduced by *Bikeabout* at Portsmouth University in England in 1996 (DeMaio, 2008). These systems are still used at a large scale today. The biggest improvement from second generation systems is that users must register in order to use the bikes. Customers can thereby be identified, creating a positive effect on the vandalism and theft problems. The required registration also gives the operators valuable information on riding patterns and customer groups. The first large-scale system to succeed using third generation technology was the Vélib' in Paris, which is still employed today.

2.2.2 Modern Technology

Although third generation systems are still widely used, newer systems involve significant improvements. *BIXI*, launched in Montréal in 2009, marks the beginning of a fourth generation bike sharing system. This system characterized the fourth generation by building upon the third generation and emphasize significant changes to two of the main components in bike sharing systems: (1) Stations and (2) Bikes and their accessibility.

Regarding stations, fourth generation systems introduce mobile stations. By including mobile stations, the operator has the possibility of dynamically relocating stations depending on the demand patterns. Modular stations can for example benefit from employing solar power electricity. This feature also rises attraction to a new group of bike sharing systems; electric bikes. As a further improvement on stations, fourth generation systems exploit existing transportation networks to find strategic locations for stations. Such networks could for instance be bus/train networks, car pooling networks or taxi stations. By exploiting existing networks, bike sharing systems strengthen its purpose to function as a first- and last mile mode of transport.

In addition, the new system incorporates technology enhancements by making bikes accessible through a real-time mobile application. This feature is beneficial for both the user and the operator. It makes the look after bikes more convenient for the user, while also providing the operator with valuable data about demand patterns and bike movement. The gathered data has among others been used to predict demand patterns and bike movements in order to optimize tactical and operational strategic decisions (Chen et al., 2018).

The use of modern technology in bike sharing systems was further extended by Chinese operators who invented and popularized dockless systems in Beijing in 2015 (Zhang, 2015). Such systems omit docking stations and opt for flexible stations where users employ mobile phone technology and street furniture for bicycle pick up and drop off (Shaheen et al., 2010). Dockless systems are attractive to users because they offer cycling options without the need to find a destination dock. The users are usually restricted to park within a predefined geographic area, a characteristic which gives birth to the alias *geo-fenced systems*. However, these systems have encountered similar problems as Schimmelpennink did with his ”White plan” in Amsterdam; vandalism and theft (Chen et al., 2018). Hybrid systems which provide a combination of docking stations and geo-fencing technology have also been proposed by Bieliński and Ważna (2019).

As a part of the fourth generation of BSS, electric bicycle systems emerged. Electric bikes, particularly pedal assist bikes, supplement the rider’s effort with electromechanical power from a battery that is carried on the bicycle. This effectively increases the range of the bike and reduces fatigue barriers, particularly in hilly terrain. These benefits make electric bikes more attractive to casual riders, and increases the usage of bicycles for longer distance traveling compared to traditional BSS (Cherry et al., 2010).

In September 2011, the University of Tennessee-Knoxville (UTK) installed the first e-bike sharing station as a technical and operational research pilot test (Shuguang et al., 2014). This BSS was used internally on a vast and hilly campus to transport students efficiently between buildings. There were two stand-alone stations with ten bikes and a capacity to charge and distribute 15 batteries. Users pick up a battery with more than 70% charge using a magnetic card when checking in to the station, and place the battery back at the charging dock when checking out. Experience from this study highlights the system’s sensitivity to trip rates, trip length, and activity duration. The system is also dependent on access to reliable energy, which limits the selection of station sites.

Today, the number of electric BSSs are increasing rapidly around the world. The innovations in the operations of the systems are also many. In April 2019, one of the largest electric BSSs was launched in Gdansk, Poland (Modijevsky, 2019). In addition to the regular pick-up and delivering of bikes at stations spread across the city, bicycles can also be returned anywhere within a given area for an additional charge, albeit requiring the bikes to be left in a public place, not bothering other activities.

As seen, bike sharing systems have improved significantly from the original system in Amsterdam, and is currently developed to better exploit the digital revolution. Table 2.1 summarizes the generation evolution from 1965 until present.

Table 2.1: Overview of the bike sharing system generations

1st generation	2nd generation	3rd generation	4th generation
Free of charge	Small deposits	Subscription	Subscription
Specific color	Specific color	Custom design	Custom design
Stationless	Stations	Stations	Stations and geofencing
No membership	No membership	Membership	Membership
Regular bike	Regular bike	Regular bike	Regular or electric bike
			Real-time mobile application

2.3 Bike Sharing Systems Worldwide

The Hangzhou Public Bicycle system was the first BSS in China, funded and operated by the local government. Since launched in 2008, the system has earned a global reputation for being one of the most successful bike sharing programs worldwide. By May 2019, this system had the largest docked bike fleet in the world, comprised of 86,000 bikes and accounting for 473,000 daily rides (Jiang and Jamba, 2019). The bikes are free of charge for up to an hour, promoting environmental friendly travels and facilitating new commute patterns. Due to high ridership, popular docking stations congest quickly, leaving many riders unable to return their bikes at wanted location. Aiming at solving parts of the issue, the city introduced smart lock technology in March 2018, allowing users to park bikes in dedicated areas next to docks when stations are congested. This technology has proven to be very successful for the customers, and since bikes can now be parked near full docks, operators don't need to spend as much time and money relocating bikes (Jiang and Jamba, 2019).

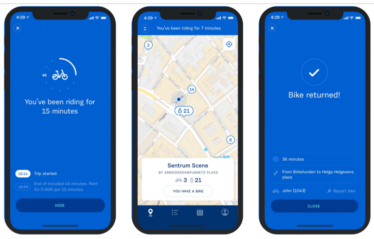
Through 2018 and 2019, there has been an explosion of electric BSSs worldwide. One of the major operators is the San Fransisco based company Jump, launching its first service in Washington D.C. in 2017. Jump's systems employ geo-fencing technology. Each Jump bike has a swappable 250-watt electric battery which powers the front wheel, providing extra pedaling assistance (Ceres, 2018). The well-known transportation company Uber acquired Jump in April 2018, and expansion into European markets began in June the same year. Jump has also joined the electric scooter market, offering shared scooter services in many cities. As of December 2019, the company offers electric bikes in more than 20 cities worldwide (Jump, 2019).

Inspired by the Spainish electric BSSs ”dBizi” in San Sebastian and ”BiciMad” in Madrid, ”MEVO” was launched in Poland in March 2019. The system covers fourteen polish muncipalities with characteristic light-blue colored bikes, illustrated in Figure 2.2. MEVO is entirely comprised of electric bikes, 4080 in total. With 660 stations, this makes the system to the largest electric BSS in Europe (Mevo, 2019). One might only rent a bike if the battery level is at least 20%. Some of the stations are charging stations, but the system also use service personnel to swap batteries when needed. Further, the system allows subscribers to rent and return the bike without additional fees. Due to the use of GPS-tracking and geo-fencing technology of fourth generation systems, there are also possibilities of returning the bikes beyond the parking points. In such cases, an additional fee is charged (Baltic Sea Region Electric, 2018).

Figure 2.2: Characteristic light blue colored bikes in Europe's largest electric BSS; MEVO in Poland. Source: (Mevo, 2019)

2.4 Urban Infrastructure Partners

The case study presented in this thesis builds on data from two systems to which Urban Infrastructure Partners (UIP) provides operational software; the BSSs in Oslo and Edinburgh. The system in Oslo has regular bicycles, while the system in Edinburgh is among the first electric system in the west European region. UIP owns, operates and funds BSSs, and is the largest BSS operator in Norway. The municipalities offer available space for public advertising and UIP is responsible for building, operating and maintaining the systems. UIP's main sources of financing are subscriptions, advertisements and sponsorships. UIP does also operate a BSS in Edinburgh, and plan on further international expansion in the near future.


4:29
You've been riding for 15 minutes
16:14 Trip started.
16:59 End of included 45 minutes. Rent for 5 NOK per 15 minutes.

You've been riding for 7 minutes
Sentrum Scene
BY ARBEIDERSAMFUNNETS PLASS
3 21

Bike returned!
36 minutes
From Birkelunden to Helga Helgesens plass
John (1043)

Figure 2.3: Mobile application interface of the Oslo City Bike system. Source: (Oslo Bysykkel, 2020a)

Figure 2.4: Bicyles locked at a station in the Oslo City Bike system. Source: (Oslo Bysykkel, 2020a)

UIP provides separate mobile applications for each of its systems. The applications offer users real-time information on the location of available bikes and locks, illustrated in Figure 2.3. In addition, the applications provide assistance for unlocking a bicycle, and information about the duration of an ongoing trip. Bicycles can also be unlocked through the application or by using a panel at the station. Figure 2.4 illustrates Oslo City bikes' characteristic blue bikes, locked at one of the stations. Upon unlocked, the bicycles are available for use for 45 minutes, with the option of extending the rental time for a fixed fee per additional time unit. UIP offers two subscription schemes; a season pass and day passes.

In the next couple of years, UIP will enter the electric micro-mobility market, employing several electric BSSs. At present, they are commissioned to finance and operate such systems in Stockholm and Nice.

2.5 Challenges of Electric BSSs

As demand for bike sharing increases around the world, new challenges arise for service providers. Making bikes available when and where people need them is vital, but not easy. There are numerous challenges related to the planning and operation of an electric BSS. The vast majority of these challenges are also seen in traditional bike sharing systems, but electric systems do naturally also include charging strategy as an important decision.

Many companies have encountered difficulties when determining the size and locations of a system. To limit the risk of failure, a lot of cities have budgeted for undersized systems. However, this has been shown only to increase the chance of failure (Hughes, 2017). Underestimating the capacity requirements of a system make the rebalancing problem more challenging. This problem is currently the biggest problem for BSSs, and aims at minimizing starvation and congestion for a system as a whole. These problems have historically been solved by service vehicle operations as illustrated in Figure 2.5. Recently, researchers have examined self-balancing systems, where users are incentivized to travel to specific stations to avoid congestion and starvation (Pfrommer et al., 2013).

When solving the battery swap and rebalancing problems with service vehicles, an operating cost is incurred. Yet, there is no direct revenue associated with solving these problems. The decisions must therefore induce an increased customer utility resulting in additional customer retrievals to create profit. Associated with these problems is also the challenge of predicting demand for bicycles and locks at each station at a particular time. This stands out as a difficult task for the bike sharing companies, as they have no way of finding the real demand as lost demand is not recorded. In addition, a lot of external factors may affect the demand patterns, such as weather, time of day, nearby events and traffic (Chen et al., 2018).

The current solutions addressing the charging problem involve charging at stations through the docks, through vending machines or through battery swapping conducted by the operator. The first solution require steady supply of electricity, which limits the possible station locations. It is also not as flexible, making it hard to combine with geo-fencing technology or a free-floating operating model. The second solution also require locations with steady supply of the electricity for the vending machines, but it allows for more flexibility in the system architecture (Shuguang et al., 2014). Battery swapping provides a very flexible solution. However, it is difficult to efficiently implement, and the operational costs are high.

2.5.1 Other Decisions

There are also other challenges related to the infrastructure and operations of electric BSSs. Such challenges include maintenance, theft and vandalism, and the technological interface towards customers (Midgley, 2011). These challenges are all important for the success of a system, especially with respect to customer satisfaction. However, they are not addressed as much in the literature as strategic and rebalancing challenges.

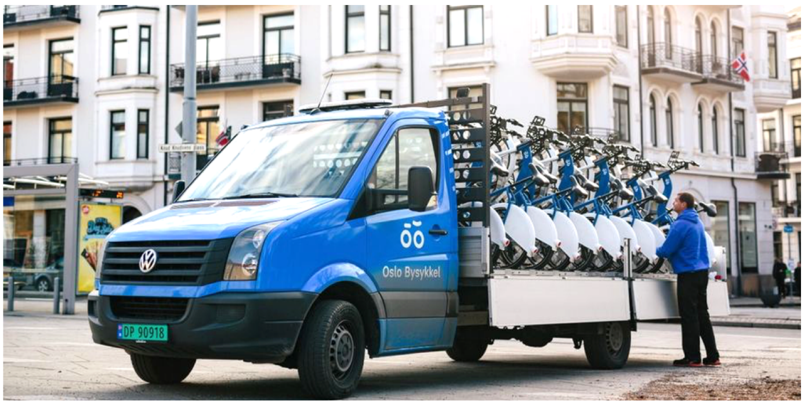


Figure 2.5: Service vehicle with bicycles loaded, serving UIP's largest BSS; Oslo City Bike. Source: (Oslo Bysykkel, 2020b)

Chapter 3

Literature Survey

This chapter aims at giving a broader understanding of existing literature relevant for our problem and the motivation behind our work. NTNU's online library, *Oria*, is the main source of materials for this chapter. As a student at the university, this site provides free access to research papers. The work done by Hagen and Gleditsch (2017) and Fosen and Haldorsen (2019) has also served as a foundation for this study.

Section 3.1 elaborates on problems and solution methods at different planning levels in a bike sharing system (BSS). Rebalancing of bikes is a central part of the studied problem in this report. Thus, work on various strategies to solve this problem is presented in Section 3.2. Following is a study of column generation heuristics applied on similar problems in Section 3.3. Further, Section 3.4 compares studies related to electric mobility systems and rebalancing of BSSs. Lastly, findings from the literature survey are summarized together with our motivation for this report in Section 3.5.

3.1 Planning Levels in Bike Sharing Systems

There are three independent planning levels in a BSS that influence the total system performance: the strategic level, the tactical level, and the operational level (Vogel, 2016). Each level address different challenges and decision points within the system. The strategic level plans how the system should be designed. These decisions are mainly related to the business model or the infrastructure, i. e. pricing, marketing, station locations, bike types, number of bikes, number of locks. The tactical level include decisions like deciding the optimal inventory levels at stations and method for detecting broken bicycles and locks. Unchangeable strategic level input informs these decisions. The operational level plans the everyday operations. Fixed strategic and tactical level input is used to decide how to relocate bikes between stations and swap batteries efficiently. Overall, the planning decisions are interdependent. Reasonable sizing of stations and suitable inventory levels at stations may reduce relocation effort, whereas high relocation effort may compensate insufficient sizing and inventory levels. Hence, distinct optimization of the planning levels may lead to suboptimal decisions (Vogel, 2016).

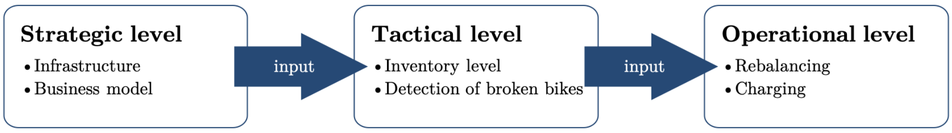


Figure 3.1: The planning levels of a bike sharing system

Figure 3.1 shows the planning levels of a bike sharing system and their interdependence. The strategic level decisions inform the tactical level decisions, which again informs the operational level decisions.

3.1.1 Strategic Level

The strategic decisions are often made up front and require capital investments. Once the decisions are made they will typically impact a BSS for its lifetime as they are difficult and costly to change (Vogel, 2016).

In strategic planning, the main decisions are related to system capacities and station locations. Information on typical system behavior is critical when deciding this. It is important to analyze the customer demand, the geographical area, and the interplay with other transportation modes like public transit and private cars. Cluster analysis based on historical data from a BSS or demographic data is a common approach to do this, where clusters in the data reveal useful insights. Based on this insight, decisions on station locations and docking capacities can be made. Liu et al. (2016) propose a multi-objective model under one hybrid mode performing conventional bus scheduling in the context of a bike sharing system. Vogel (2016) determines temporal demand “activity clusters”, describing typical departing and arrival activities at stations in the course of the day. Borgnat et al. (2011) characterizes interrelated stations by cluster analysis of bike flows between stations. Wang et al. (2015) apply linear regression to model the correlation of bike activity at stations and external factors like demography and transportation infrastructure. Faghih-Imani et al. (2017) models the influence of weather, bike infrastructure, land-use and environmental attributes on bike rental and return rates.

An important problem on the strategic level is to decide on the optimal number of bikes to offer in the system. Fricker and Gast (2014) try to make a policy for this by quantifying the

the influence of changing the station capacities and compute the optimal number of bikes in the system by minimizing the number of stations where violations occur regularly. Their concluding policy is to offer half as many bicycles, plus potentially a few more depending on the system parameters, as there are parking slots. Chen et al. (2019) study the same problem as well as the station location problem in an electric BSS formulated as a bi-level programming model. They propose a system where the generalized trip cost is measured as the sum of the time spent at stations and the travel time.

3.1.2 Tactical level

Strategic level decisions work as input to the tactical level, and are seen as unchangeable. At the tactical planning level, the aim is to efficiently use the limited resources in order to yield a high expected service level for characteristic user demand patterns, e.g. for a working day in a given season (Neumann Saavedra et al., 2016). The main decision is to determine the ideal bike fill levels at stations at given times. Ideal bike fill levels should compensate for varying bike demand in the course of day and stimulate the number of successful customer rentals. High fill levels increase the probability of successful rentals and decrease the probability of successful returns at particular stations (Vogel et al., 2017). Due to demand uncertainty in the bike usage, thorough demand and user pattern forecasts are necessary to support the decisions at this level.

Vogel (2016) sets the ideal state through use of data mining techniques and knowledge to get insight in the bike activity pattern of the users. Neumann Saavedra et al. (2016) assumes for his Service Network Design that that the user demand exhibits similar patterns each day. The suitable fill level at the end of the time horizon is then limited by a fixed absolute value for the mismatch between the initial and final fill level with the estimated demand pattern. The closer this value is to zero, the more redistribution effort is required. A third approach by Schuijbroek et al. (2013), model stochastic demand using a queue system and define service level requirements at each station. The output of the model is intervals for optimal state to obtain the desired service levels. This inventory model is also solved together with a routing problem, combining two problems that are usually solved separately. Finally, Vogel et al. (2017) proposes a MIP formulation which determines optimal target fill levels at stations. In the course of the day, demand scenarios in this formulation are fulfilled according to a predefined service level. The objective is to obtain fill levels at minimal expected costs of system operation. Lastly, Espegren and Kristianslund (2015) argue that the ideal state should be determined by setting the probability of congestion equal to the probability of starvation for the planning horizon.

3.1.3 Operational Level

Operational level relates to the everyday operation. Strategic and tactical decisions work as input, and are seen as unchangeable. The everyday operations in an electric BSS consist of three main aspects: rebalancing of bikes, charging and swapping of batteries, and efficient route planning for the service vehicles.

Rebalancing of bikes

The natural imbalance and stochasticity of customer arrivals lead operators to develop redistribution strategies in order to ensure a sufficiently high user satisfaction. The overall goal of rebalancing efforts is to minimize the arrival of unsatisfied customers who find their station empty or full (Legros, 2019). This problem is elaborated further in Section 3.2.

Charging of Bicycle Fleet

The problem of charging the batteries in a BSS can either be handled by battery swaps through service vehicles, or charging systems at stations. The station implementation of the battery charging system can either require the users to remove the battery from the bicycle for charging or connect the charging system to the bicycle without removing the battery. Charging the battery on the bike simplifies the bicycle check-out process and hardware, but has the disadvantages of taking the bicycle out of service while charging and increasing the probability of theft. Charging the battery after removing it from the bicycle requires the ability to dispense and return the batteries but allows all available bicycles to remain in service while the battery is recharged if there are more batteries than bicycles in the system (Cherry et al., 2010).

The battery swap approach uses service vehicles to swap flat batteries at stations. Uber's first generation of Jump bikes had to be collected and charged at depots by employees. However, the new generation has swappable battery packs to keep more e-bikes on the street and increase availability for riders (Toll, 2018).

Efficient Route Planning for Service Vehicles

Both aspects mentioned above are commonly handled through use of service vehicles. These service vehicles need efficient route planning, which is a problem falling into the combinatorial optimization and integer programming problem category known as Vehicle Routing Problems. Vehicle routing problems concern the challenge of selecting a set of routes for a fleet of vehicles to serve the demands of a set of customers. Almost invariably, the vehicles have limitations on the amount of goods they can carry, and the primary goal of the decision-maker is most often to minimize the total transportation cost (Oyola and Arntzen, 2016).

3.2 The Bike Rebalancing Problem

There are many studies with approaches to efficient rebalancing of bikes in a BSS. Pal and Zhang (2017) divides the strategies to this problem into two: user based strategies and operator based strategies. User based strategies focus on incentives for self-rebalancing like dynamic pricing or reward systems. Operator based strategies use vehicles to manually rebalance the system. If the manual rebalancing is done when the user intervention is negligible, it is known as static rebalancing, whereas if it is done when there is significant user intervention, the rebalancing is known as dynamic rebalancing. Further, the dynamic problems can be viewed as stochastic or dynamic depending on the representation of customer demand. An overview of the rebalance problem taxonomy can be seen in Table 3.1.

Table 3.1: The taxonomy of rebalancing problems

		Customer arrival classification	
		Deterministic	**Stochastic**
System development	**Static**	Static and deterministic	-
	Dynamic	Dynamic and deterministic	Dynamic and stochastic

The static rebalancing problem is well-studied. In practice, this is equivalent to the problem solved when operators perform rebalancing operations during the night when the system is out of service. Common for many approaches is the attempt of minimizing the gaps between the target balancing levels and the state of stations after performing the balancing operation in addition to the total operational time and the number of operations (Kadri et al., 2016). Rainer-Harbach et al. (2013) proposes a Greedy construction heuristic combined with a variable neighborhood search approach that takes the loading and unloading times into account. Pal and Zhang (2017) attempts to solve this problem by construction a Mixed Integer Linear Program where the objective is to minimize the make-span of the rebalancing fleet. Chemla et al. (2013) use a branch-and-cut algorithm for solving a relaxation of the problem. An upper bound of the optimal solution of the problem is obtained by a tabu search, which is based on some theoretical properties of the solution.

The dynamic rebalancing problem is not as well studied as the static problem but it is subject to increased attention in the recent years. Hagen and Gleditsch (2017) present a deterministic approach. They approximate the problem into a set of smaller deterministic subproblems where the customer demand is assumed known. In Appendix A, we have formulated an extended version of this deterministic problem mathematically as an initial solution approach to the DSBRCP.

The dynamic stochastic version of the bike rebalancing problem can be viewed as our problem only in a traditional BSS setting, i. e. without electric bikes. In contrast to pure dynamic and deterministic problems, there is a strong incentive to exploit and integrate all available information on foreseen future events in the solution process, which make the solution methodologies considerably more intricate (Toth and Vigo, 2014). Warrington and Ruchti (2019) adapts a two-stage stochastic approximation scheme from the generic SPAR algorithm, in which rebalancing plans are optimized against a value function representing the expected cost (in terms of fulfilled and unfulfilled customer demand) of the future evolution of the system. The true value function is approximated by a separable function of contributions due to the rebalancing actions carried out at each station and each time step of the planning horizon.

Legros (2019) assumes that the demand for travelling between stations follows a time-dependent Poisson distribution. Using a dynamic programming approach, he computes the relative value function of the system, the average cost, and the optimal state, to discover an optimal rebalancing strategy. This study and most other studies use an exact target inventory for each station to make rebalancing decisions. An alternative approach is introduced by Schuijbroek et al. (2013), who present a clustering approach that simultaneously considers the service level feasibility and approximate routing costs during rebalancing. They argue that inventory flexibility is needed as routing costs to attain exact target inventories will always be larger than or equal to the costs strictly necessary to maintain appropriate service levels.

3.3 Heuristic Column Generation

The DSBRCP is solved using a two-stage column generation heuristic. A standard column generation algorithm decomposes a problem into a master problem and an associated pricing problem. First, a set of possible solutions are generated and passed on to the master problem as *columns*. The algorithm then iterates between solving a restricted master problem (RMP), where the optimal solution is formed from a subset of possible columns, and solving a pricing problem, where columns that may improve the solution are identified. For a column generation to be considered an exact solution algorithm, both the master problem and the pricing problem must be solved to optimality, ensuring that unexplored columns that may improve the solution do not exist. As it is difficult to develop a procedure to find exact solutions for complex problems in a pricing problem, it is often reasonable to incorporate heuristic approximations when applying a column generation algorithm.

Hagen and Gleditsch (2017) present a column generation heuristic to solve the bike rebalancing problem. It consists of an initialization procedure, a master problem, and a pricing problem. Three variations of the master problem are proposed, where each differs in the amount of predetermined information. In version 1, the loading quantities are determined in the master problem; in version 2, the loading quantities are predetermined in the initialization and evaluated again in the master problem; and in version 3, the loading quantities are entirely predetermined. The initialization process consists of a branching algorithm that generates many initial columns or routes per service vehicle, while the master problem optimizes the combination of columns by allocating one column to each service vehicle. A heuristic pricing problem is developed with the goal of generating new and better columns. To diversify the generated routes, a clustering model for stations is applied. The presented DSBRCP heuristic in this thesis, shares many characteristics with their implementation. However, a major difference is our incorporation of subproblems to assign values to the columns in the master problem.

As further literature employing column generation algorithms on BSSs proved scarce, a further survey were conducted on literature where it is applied to related problems. Guedes and Borenstein (2015) describe a heuristic framework based on column generation to solve instances of the multiple-depot vehicle-type scheduling problem with a good compromise between efficiency and quality of the solution. They emphasize the importance of good initial solutions to obtain this. Further, Venkateshan and Mathur (2011) allow initial routes to be infeasible when solving a pick-up and delivery problem through a routine that reduces the combinatorial explosion in the number of routes generated. However, these solutions are associated with a high cost in the master problem. This is comparable with our approach, where the infeasible patterns are restrained in the master problem.

3.4 Comparison of Relevant Studies

In this section we compare studies that specifically focus on rebalancing of a BSS or maintaining an electric mobility system. The main objective of doing this is to discover aspects in the literature where there are room for improvements, which rationalizes our motivation for this thesis. Table 3.2 provides an overview of the selected studies that are compared according to suitable specifications. Some of the specifications are elaborated in the next subsections. A summary of the comparisons can be found in Table 3.3. Articles

1 to 6 address BSSs, and articles 7 to 9 are related to car sharing systems.

Table 3.2: Studies used for comparison

#	**Article**
1	Han et al. (2018)
2	Legros (2019)
3	Espegren and Kristianslund (2015)
4	Hagen and Gleditsch (2017)
5	Vogel (2016)
6	Shuguang et al. (2014)
7	Nourinejad and Roorda (2014)
8	Warrington and Ruchti (2019)
9	Barth and Todd (1999)

3.4.1 Objective Function

The objective function reveals the overall goals of a model and the importance of different aspects. When optimizing the operations of a BSS, there are two main objectives that should be balanced; minimizing costs and maximizing customer utility. Some of the studies focus on cost metrics like minimizing travel distance or maximizing system profit. This applies to Han et al. (2018), Vogel (2016), Nourinejad and Roorda (2014), Warrington and Ruchti (2019), and Shuguang et al. (2014). The rest of the studies attempt to maximize customer satisfaction or minimizing customer inconvenience, e.g. total waiting time.

3.4.2 Demand

The demand prediction in the articles are either based on historical data, assumptions of the underlying demand distribution, or a combination of the two. Vogel (2016) bases the demand prediction solely on historical numbers, Shuguang et al. (2014) and Legros (2019) assume that the demand follow a Poisson distribution, while Barth and Todd (1999) combines the two, predicting the demand based on historical data and probabilities conditional on geographic criteria.

3.4.3 Type of System

The systems used as foundation for the articles can be categorized in two main groups: BSSs and car sharing systems. The first group of systems consists of traditional BSSs or electric BSSs with varying operating models. All of them are station based, but vary in size and geographical dispersion. Shuguang et al. (2014) demonstrate an example of an electric BSS where vending machines are used to charge the batteries. The users collect a battery from a vending machine located at the station together with a bike when starting a trip. At the desired destination, the battery is detached from the bike again, and placed for charging in another vending machine.

The second group use car sharing systems as basis for their work. The type of problems encountered in a car sharing system have many common traits that are applicable to our problem as well. There is a need for relocation, battery charging, demand predictions, and parking areas.

3.4.4 Main Decisions

The decisions studied in this literature survey are mainly related to routing, rebalancing, and system variables. The system variables are the primary focus for Shuguang et al. (2014) and Barth and Todd (1999), where the effective size of the system and the asset specifications are scrutinized.

3.5 Conclusion and Motivation of the Thesis

Increasing focus from the academic community on bike sharing systems has evolved in the last decade. In light of the growing societal demand for robust micro-mobility services, in which complex problems are encountered, this is not surprising. Strategic, tactical, and operational problems are addressed in the literature. Different approaches have been presented for modeling and solving problems within shared vehicle services. Increased focus have been directed towards exploration of operator-based rebalancing during the day, as the DSBRCP can be seen as an extension to this problem.

The literature on electric BSSs is still limited, lacking some essential aspects that we want to examine in this thesis. The focus of current literature is primarily on the relocation problem, not the problem of having a charged fleet. As presented in this chapter and Chapter 2, the studied approaches of charging bicycles in BSSs utilize charging stations and vending machines. Mitigating the problem through battery swaps by means of service vehicles has not been studied yet as far as we know. However, this approach is used in many systems, usually in addition to charging stations. For instance, Uber's Jump (2019) system operate in this manner.

As the electric BSSs become more common and larger in size, there is an urgent need for advanced decision support that can guide optimal set-ups for charging solutions. Charging solutions at stations, either through a vending machine or the dock itself, require steady supply of electricity, which limits the possible station locations (Shuguang et al., 2014). Solving the problem through battery swapping with service vehicles eliminates this problem. This approach also allows for more flexibility, facilitating the development of free floating systems. Most BSSs already employ service vehicles to rebalance bikes. Thus, it is reasonable to extend the rebalancing problem to also include charging decisions. Combining both aspects in operational decision making is believed to increase the solution quality and capture potential value related to decreased operational costs. Solutions addressing this for BSSs have to our knowledge not been published in the literature.

Inspired by the work done within rebalancing and car relocation, we have developed a two-stage stochastic programming column generation heuristic to address the combined problem of rebalancing and charging encountered in electric BSSs. The main focus is directed towards the user benefit, aiming to maximize the number of successful trips in a system. This is primarily based on the work done by Hagen and Gleditsch (2017) and Legros (2019).

Table 3.3: Overview of comparisons between selected studies

Specification	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Taxonomy	Static and deterministic	Dynamic and stochastic	Static and deterministic	Dynamic and stochastic	Dynamic and stochastic	Static and deterministic	Dynamic and stochastic	Dynamic and stochastic	Static and deterministic
Problem focus	Split delivery vehicle routing problem	Rebalancing of bicycles between stations	Ideal state for stations	Rebalancing of bicycles between stations	Service Network Design of a BSS	Sensitivity analysis of system characteristics	Dynamic model for vehicle relocation	Dynamic rebalancing in shared mobility systems	Optimal relocation of shared vehicles
Demand	Clustered demand based on historical data	Assumed Poisson distributed	Based on historical data	Based on historical data	Based on historical data	Assumed Poisson distributed	Based on historical data	Monte Carlo simulation with 100 random demand scenarios	Geographical trip analyses
Objective function	Minimize total travel distance	Minimize arrival of unsatisfied customers	Minimize deviations from optimal state	Minimize violations and deviations	Minimize total cost for relocation services	Minimize cost and maximize system reliability	Maximize system profit	Minimize general cost metric	Minimize customer waiting time
Size of solvable instances	812 stations	30 stations	13 stations	10 stations	400 stations	2 stations	6 stations	225 stations	-
Modelling method	Genetic Algorithm	MDP with relative value function	Mixed Integer Programme	MDP, simulation	Mixed Integer Programme	Sensitivity analysis	Discrete event simulation	Two-stage stochastic approximation	Simulation
Main decisions	Routing and relocation	Routing and relocation	Routing and relocation	Routing and relocation	Relocation of bikes	System variables	Relocation	Routing schedules	Effective fleet size
Multiple vehicles	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Depot oriented	No	No	No	No	No	No	Yes	No	No
Type of system	Traditional BSS	Traditional BSS	Traditional BSS	Traditional BSS	Traditional BSS	Electric BSS	Car sharing system	Shared mobility systems	Electric car sharing system
Operational costs	Neglected	Neglected	Neglected	Neglected	Variable	Variable	Variable	Neglected	Variable
Study area	San Fransisco	NA	Oslo	Oslo	Wien	Tennessee	Toronto	Philadelphia	California

Chapter 4

The Dynamic Stochastic Bike Rebalancing and Charging Problem

This chapter presents the *Dynamic Stochastic Bicycle Rebalancing and Charging Problem* (DSBRCP) encountered in BSSs. The problem is a combination of the Dynamic Stochastic Bike Battery Swap Routing Problem presented in Fosen and Haldorsen (2019), and the more well-known *Bike Rebalancing Problem*, described in Chapter 3. This chapter provides a holistic overview of the problem in Section 4.1, followed by a detailed elaboration on the problem with a toy-sized example in Section 4.2.

4.1 Description of the Real-World Problem

The studied electric BSS consists of a given number of stations and identical electric bicycles. The batteries on the bicycles are chargeable and possible to detach. Stations can be either charging stations, where batteries are charged while attached to a parked bicycle, or non-charging stations, where charging is obtained through battery swaps. Bicycle sharing is offered between two stations as long as there is an available bike with sufficiently charged battery at the start station, and an available lock at the end station.

Bicycles that are parked at a station with battery level below a certain threshold are referred to as *flat* bicycles. These bicycles are not available for customers to use until they are recharged. Bicycles with battery level above the threshold are referred to as *charged* bicycles. The batteries discharge as a function of the bicycle usage.

A typical trip for a user between a starting point and end destination is illustrated in Figure

4.1. The customer walks to a pick-up station, where he/she picks up a charged bicycle. Then, the customer bikes to a end station, where the bicycle is parked. Ultimately, the customer walks from the end station to the final destination. A trip creates demand for an available charged bicycle at the pick-up station and demand for an available lock at the end station.

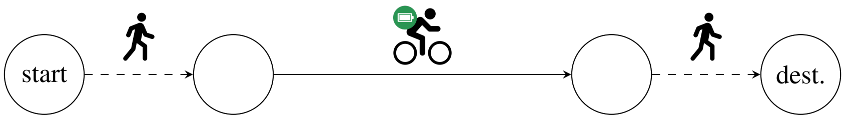


Figure 4.1: Illustration of a customer trip from start to destination

Customer utility is important for a BSS operator in order to maintain a satisfied customer base and attract new customers. There are two situations that can occur in the system with negative impact on the customer utility:

1. *Starved station*: There are no available charged bikes at a station when a customer arrives with the intention of using a bicycle
2. *Congested station*: There are no available locks when a customer wishes to deliver a bicycle upon arrival at a station

Each of the two cases produce an incident of the collective term *violation*. We refer to case 1, where a customer cannot find an available charged bicycle at the desired pick-up station, as *starvation violation*. Case 2, where the customer cannot find an available lock at the desired end station, is referred to as *congestion violation*. Violations are undesirable occurrences in the system.

To maintain an efficient BSS, the system employs a number of service vehicles that swap batteries on flat bicycles at station visits and redistribute bicycles in the system. The route of a vehicle starts at a predefined starting station. At each station visit, including at the start station, the service vehicle may load or unload bicycles, and perform battery swaps on flat bicycles located at the station. There is a service time associated with each bicycle handling and each battery swap. As part of the route, the service vehicles also have the opportunity to visit a depot to restock the vehicle with charged batteries, and drop off flat batteries to recharge.

In summary, the service vehicle personnel must make the following four decisions upon arrival at a station:

1. The number of charged bicycles to load or unload

2. The number of flat bicycles to load or unload
3. The next station to visit
4. The number of batteries to swap

The DSBRCP aims at finding a decision policy for the service vehicles that maximizes customer utility. Assuming that customer utility is solely based on the number of occurred violations, customer utility can be formulated as a linear function where the derivative of utility is negative with respect to violations. Hence, maximizing customer utility is equivalent to maximizing the number of violations prevented by the service vehicle operations, as compared to a system where no service vehicle operates.

A model solely determined by the number of prevented violations may however prove too shortsighted to perform well in a full day simulation of the BSS. To adjust for use of shorter planning horizons in the DSBRCP, a station has a predefined desired number of charged bicycles at each hour in a day. This number is referred to as the station's *ideal state* and aims to plan for future customer arrivals. As part of the service vehicle's decision policy, distributing the bikes as close to the stations' ideal state as possible is targeted. The policy also includes a reward associated with unloading of flat bikes at charging stations.

4.2 Modelling DSBRCP as a Markov Decision Process

A mathematical model of the real-world problem should reflect the problem's main characteristics. The real-world problem is *dynamic* as the personnel operating the service vehicles must make sequential decisions during operation, and information considering changes in the system's bicycle distribution becomes available during operation. Furthermore, the problem is *stochastic* as the future demand for bicycles and available locks are assumed to be random variables with known probability distributions. The future distribution of bicycles is partly dependent on stochastic customer demand variables and partly on controlled bicycle rebalancing and battery swap decisions. Thus, the problem can be modelled as a *Markov Decision Process* (MDP). An MDP is a discrete-time stochastic control process, serving as a framework for modeling decision making in situations where outcomes are both random and controlled by a decision maker (Mes and Rivera, 2017).

An MDP consists of a set of possible states $\mathcal{S}$ for the system, together with a set of decisions $\mathcal{X}$. The probability distribution for the outcome of the stochastic variable is assumed known. This probability distribution may depend on the current state and the decision that is chosen from $\mathcal{X}$. It is however not affected by the sequence of events that has preceded the state, nor previous decisions. The problem is to be solved over a set of times t in the time horizon $\overline{T}$. Bellman (1957) formulates an MDP as shown in Formula (4.1).

$$s_t \xrightarrow{x_t} s_t^x \xrightarrow{\omega} s_{t+1} \tag{4.1}$$

The transition process starts in state $s_t \in \mathcal{S}_t$, where $\mathcal{S}_t$ is the set of possible system states at time t. In the DSBRCP, state s_t contains information about the charge levels and number of bikes at stations, as well as the next station visit with associated arrival time for each of the service vehicles. Based on this information and the expected future demand distribution, a decision $x_t \in \mathcal{X}_{s_t}$ has to be made. $\mathcal{X}_{s_t}$ is the set of all feasible decisions in state s_t. In our problem, this decision includes (1) how many batteries to swap at the current destination, (2) how many charged bicycles to load/unload, (3) how many flat bicycles to load/unload, and (4) which station to visit next for each service vehicle. The decision leads to a deterministic post-decision state s_t^x. The realization of the stochastic variable is now revealed, referred to as the exogenous information, ω. In the DSBRCP, this variable contains the realized demand of bicycles and available locks. The process now transitions into the next state s_{t+1}, which gives the decision maker a corresponding reward or penalty depending on the expected future value of being in state s_{t+1}.

A reasonable decision point in the DSBRCP is the time a service vehicle arrives at a station. The state information at that time and the expected future demand distribution are used to make a decision. While driving to the next station, the outcome of the stochastic variable is revealed. Arriving at the next station, a new state is observed and used to find the next decision.

The number of violations in the system is affected by the stochastic customer demand, and can therefore *not* be directly used in the objective function. A good approximation is to minimize the expected number of violations through an optimal policy $\pi \in \Pi$, which as further described maximizes customer utility given certain assumptions. A policy π maps a decision x_t to every state s_t over the horizon T, where Π is the set of potential policies. The overall goal is to map the decisions that minimize the expected number of violations.

It can however be extremely difficult to determine such an optimal policy $\pi \in \Pi$, as both the state space $\mathcal{S}_t$, the decision space $\mathcal{X}_t$ and the outcome space Ω_t are victims of the curse of dimensionality. There exists a state for every possible station distribution of charged bikes and flat bikes, and a stochastic variable for every possible realization of customer demand. Additionally, the time horizon can be very long, making it even more difficult to find an optimal policy.

This thesis studies an approximation method to determine a service vehicle decision policy that maximizes customer utility. The method is elaborated in Chapter 5, and a study of the performance of the method, and its implications for managerial decision making, is described in Chapter 9.

4.3 Example of DSBRCP

Figure 4.2 shows an example of a transition from state s_t to the next state s_{t+1} via the deterministic substate s_t^x. In this example, there are three stations and one service vehicle. Station 2 is a charging station, while the two other stations are traditional non-charging stations. The small circles indicate capacity. The service vehicle has capacity of four batteries, illustrated as blue circles. A blue filled circle indicates an available charged battery. Further, the vehicle has capacity of six bicycles, indicated by gray circles. Each bicycle slot can be occupied by either a charged or a flat bicycle. A green filled circle indicates that the slot is occupied by a charged bicycle, while a red filled circle indicates a flat bicycle. An empty circle indicates that the slot is idle. The stations have six bicycle locks each. Similar to the coloring of bicycle slots on the vehicle, green fill indicates a charged bicycle, red fill indicates a flat bicycle and an empty circle represents an idle lock.

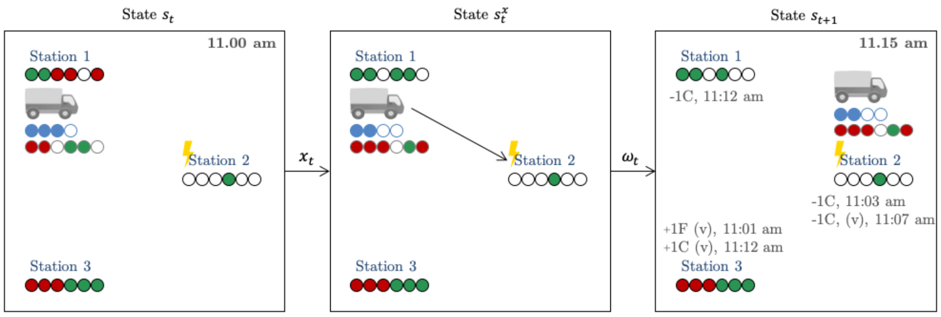


Figure 4.2: Example of a Markov Decision Process with initial state s_t, decision x_t, post-decision state s_t^x, exogenous information ω_t and transition to the next state s_{t+1}.

In state s_t, station 1 contains two charged bicycles and three flat bicycles. The time is 11:00am. The service vehicle is located at station A with an inventory of three charged batteries, two flat bicycles and two charged bicycles. The decision x_t is made based on the current state of the system, in combination with an expected demand distribution for each station over the next time period. In this example, x_t brings the following decisions:

1. **Unload one charged bicycle** from the vehicle
2. **Load two flat bicycles** to the vehicle
3. Replace **one battery** on a flat bicycle
4. Next station visited is **station 2**

By the time the service vehicle arrives at station 2, the stochastic information ω_t is revealed. This information is highlighted with gray text in the rightmost square in Figure 4.2. As observed, one charged bicycle is retrieved from station 1 within the time period, at time 11:12 am. At station 2, one charged bicycle is desired at 11:03 am. Another customer wants to start a trip at 11:07 am, but there are no available bicycles at the station. Hence, a starvation violation occurs. Further, two bicycles arrives station 3 during the time period, one flat at 11:01 am and one charged at 11:12 am. As the station is full, none of them can be parked, and hence both leads to a congestion violation. The example illustrates a total of three violations; one starvation violation at station 2 and two congestion violations at station 3.

Chapter 5

Two-Stage Stochastic Programming Column Generation Heuristic

Since the DSBRCP needs to be solved in real time, short computational times are required. In this chapter, we present the background for choice of solution method in Section 5.1, before the model assumptions are explained in Section 5.2. An overview of the chosen solution method, a two-stage stochastic programming column generation heuristic algorithm, is given in Section 5.3. The algorithm consists of three parts; the master problem, the initialization of routes and patterns, and the subproblem. The master problem is detailed in Section 5.4, followed by an explanation of the initialization in Section 5.5. Lastly, Section 5.6 presents the subproblem of the algorithm.

5.1 Solution Approach

A common technique to address the previously described dimensionality challenge of a Markov Decision Process (MDP), is to break the MDP into smaller subproblems that collectively make up the solution for the MDP (Mes and Rivera, 2017). In the DSBRCP, a possible approach to generate subproblems, and thus simplify the problem, is to split up the full problem into problems using smaller time intervals. A shorter planning period contributes to a significant reduction in problem complexity. It is hence of interest to examine whether a series of problems with shorter planning horizons could still produce high quality long-term solutions.

We refer to a problem with shortened time horizon as a *MDP subproblem*. Within a MDP subproblem, an important modeling choice is to determine how the demand uncertainty should be handled. With short time horizons, it may be assumed that the demand rates are known constants, or they could be modelled as stochastic variables. This choice is essentially choosing between a deterministic or stochastic model.

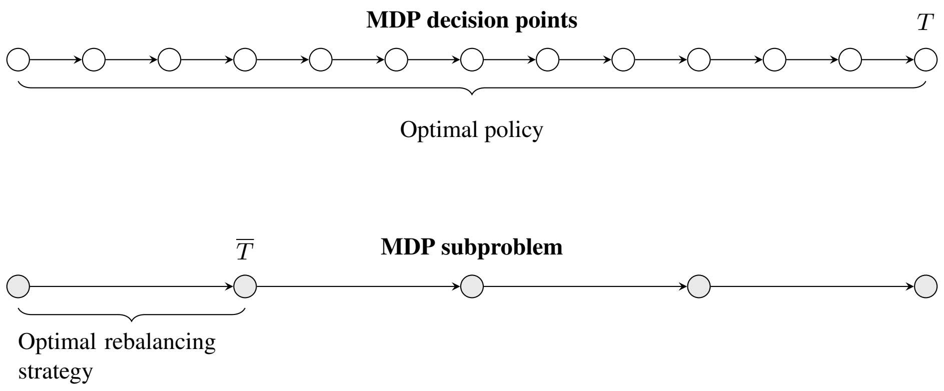


Figure 5.1: Illustration of how a Markov Decision Process is approximated by multiple subproblems.

Figure 5.1 shows the MDP formulated problem broken down into a series of MDP subproblems. Each MDP subproblem gives a strategy for the shortened time horizon. The combined solutions of the MDP subproblems make up the policy for the full time horizon. The circles indicate decision points. While a decision in the MDP provides an optimal policy for each state, the MDP subproblem provides a rebalancing and battery swap strategy for the entire shortened time horizon $\overline{T}$. The shortened time horizon makes the subproblem a lot easier to solve, but it only considers the near future. However, future implications of decisions may be addressed through approximations presented in Section 5.6.

A deterministic version of the MDP subproblem, where the demand rates are assumed to be known constants, is found in Appendix A. Fosen and Haldorsen (2019) address a deterministic version of the MDP for a similar, yet simplified, problem of the DSBRCP with only battery swap decisions. Their study concludes that the deterministic problem is not scalable with respect to the number of stations in the system. The largest test instance addressed in the report consisted of only 20 stations, and could not be solved to optimality within an hour. Hence, we conclude that this approach would not be applicable in real-world operations, as even the simplified version of the problem does not scale.

Due to the scalability challenges encountered by Fosen and Haldorsen (2019) and the general inefficiency of exact solution methods, we have investigated quicker solution approaches that also address the stochastic nature of the problem. In the following section, we present a stochastic column generation heuristic for the MDP subproblem, with the intention of solving realistically sized problem instances quickly and satisfactorily.

5.2 Assumptions

Routes

A service vehicle can visit multiple stations during a given time horizon, producing a vehicle route. To reduce the dimensionality of the problem, a station can only be visited once in the same time horizon. This is with exception of the depot, which can be visited once by each vehicle to restore the battery inventory to its full capacity. The operational time at a station visit is assumed to be constant when generating possible routes.

Customer arrivals

The number of customer arrivals within a time horizon are modelled using three parameters per station, one for each type of customer that may interact with a station:

A. *Incoming flat* customers, who want to park a flat bike at the station

B. *Incoming charged* customers, who want to park a charged bike at the station

C. *Outgoing charged* customers, who desire to use a charged bike from the station

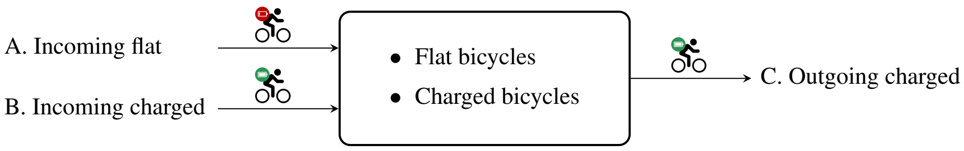


Figure 5.2: Illustration of the three types of customer interactions in an electric BSS

Figure 5.2 illustrates how the three customer arrival parameters relate to a station. Starting from the left, there are two types of incoming customers; type A and type B listed above. As long as there are available parking slots, arrivals of these customer types affect the station's bicycle inventory. Type A only affects the station inventory of flat bikes while type B only affects the inventory of charged bikes. These inventory levels are tracked separately in the station inventory, shown in the central box. Further, the station faces a demand incurred by customers who desire a charged bike, customer type C. Naturally, interactions with these customers affect the station inventory of charged bikes, but only when these are available at the station as flat bikes are not available for use. The number of trips initiated at a station is the minimum of available charged bikes and the number of arriving customers of type C. A defined sequence of customer arrivals of each type occurring within a time horizon, is referred to as a *customer arrival scenario.*

Violations

As discussed in Chapter 4, violations represent customer dissatisfaction. Violations can either stem from no available charged bikes at a station when a customer desires one, or no available locks when a customer wants to park a bike. The former is referred to as starvation violation, while the latter is referred to as congestion violation. The violations occurring in a system with no service vehicle operations in a given customer arrival scenario is referred to as *base violations*.

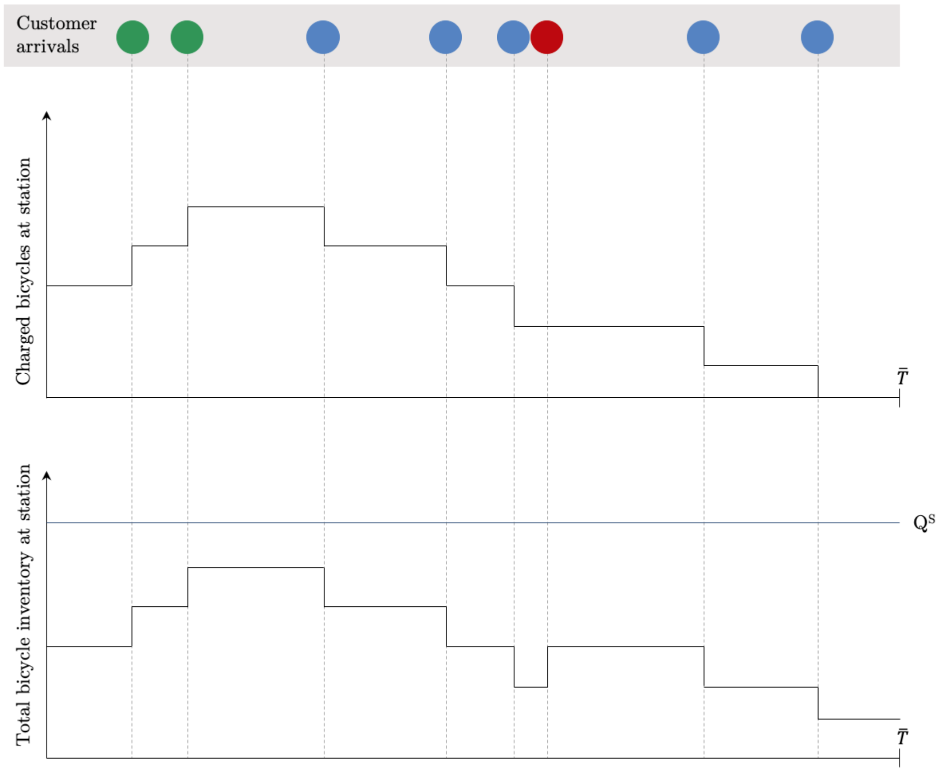


Figure 5.3: Example of a *best case* sequence of customer arrivals, given the numbers of each customer type arriving before $\overline{T}$. Red circles represent incoming flat customers (type A), green circles represent incoming charged customers (type B), and blue circles represent outgoing charged customers (type C).

The number of base violations at a station varies depending on the sequence of customer arrivals generated through a Poisson arrival process. In a given customer arrival scenario, base violations could either be calculated through simulation or estimated using an expected value. To simplify the model, we apply the expected value, defining *expected base violations* as the average number of violations across a *best case* and *worst case* sequencing of customer arrivals. Figures 5.3 and 5.4 illustrate the best case and worst case customer arrival scenarios for a station, where only the order of customer arrivals differs. In this example, the station has three charged bicycles and no flat bikes at the start point. The grey bar at the top displays the customer arrival scenario, where the x-axis position of a colored circle indicates the time of arrival, and the color of the circle specifies the customer type. Red circles represent incoming flat customers (type A), green circles represent incoming charged customers (type B), and blue circles represent outgoing charged customers (type C). Observe that eight customers interact with the station before the time horizon $\overline{T}$; five customers who desire a charged bike, two customers wishing to park a charged bike and one customer wishing to park a flat bicycle.

The best case scenario in Figure 5.3 shows a situation where all customer requests are met. This is however not the case in Figure 5.4, where two starvation violations occur, illustrated with shaded circle fill. At these points in time, no charged bikes are available at the station. The customer demand for a charged bike can hence not be offered. Similarly, a congestion violation would occur if a lock is requested at a point in time where the total bike inventory at a station is equal to the station's capacity level. Note that there can be multiple configurations of the customer arrival sequence leading to best and worst case scenarios in terms of number of violations, but the number of worst and best case violations are always uniquely defined. The calculation of worst and best case violations assumes that once a station is full, there are no changes in station inventory before the station is visited. Any occurrence of starvation and congestion is penalized equally.

Deviation

Deviation at a station is defined as the absolute difference between the station's *ideal state* at the time horizon $\overline{T}$ and the expected inventory of charged bikes at $\overline{T}$ if the station is not visited. The ideal state, O_i, is estimated by UIP, and defined for each station every hour. A station's ideal state at a given time aims at representing the optimal charged bike inventory level at the station. UIP's estimates are based on the work of Espegren and Kristianslund (2015). Their approach is to calculate the number of bicycles that minimizes the probability of violations occurring in the next three hours considering predicted demand. This occurs at the point where he probability of congestion is equal to the probability of starvation, elaborated in the work of (Espegren and Kristianslund, 2015). Typically, stations with expected high charged bike demand in the near future will have a high ideal state, whereas stations with high expected demand for parking slots in the near future will have a low ideal state.

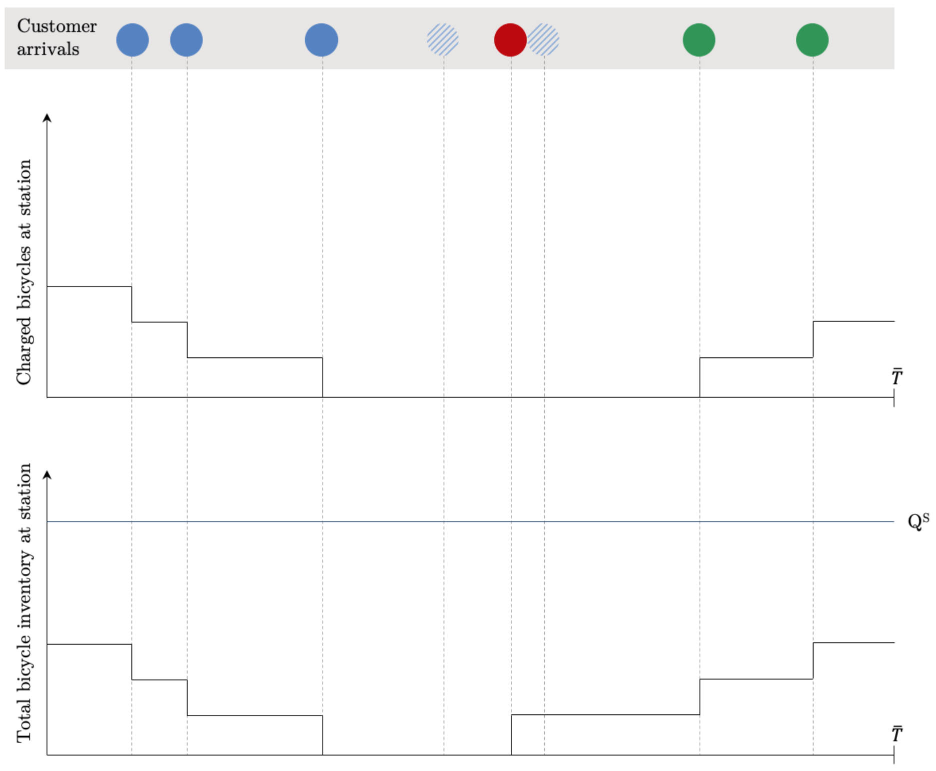


Figure 5.4: Example of a *worst case* sequence of customer arrivals, given the numbers of each customer type arriving before $\overline{T}$. Red circles represent incoming flat customers (type A), green circles represent incoming charged customers (type B), and blue circles represent outgoing charged customers (type C).

Similar to the previously described calculation of base violations, *base deviations* at a station can be calculated. Base deviation is defined as the expected number of deviations measured at the time horizon when no service vehicles are operating in the system. Analogously to the expectation of base violations, the expected value of base deviation is the average between the worst case and best case sequence of customer arrivals. These values are calculated for each station and used in the subproblem to measure the number of prevented deviations stemming from the service vehicle operations.

Batteries and charging stations

The battery level of a bike will not change unless the battery is swapped or charged at a charging station. Charging stations are assumed to not require battery swaps or unloading of charged bikes. The effect of charging at a charging station is not seen until the next time horizon. A battery swap is assumed to result in a fully charged electric bike, and the flat batteries need to be recharged at the depot before they can be put to use again.

Other considerations in a BSS

The costs associated with utilizing service vehicles, i.e. labour and fuel, are neglected. This is based on the assumption that the service vehicles work constantly, and not only upon request. Considerations around the size of the labour force, the number of stations and their locations are strategic and tactical questions that we consider fixed.

5.3 Overview of the Two-Stage Stochastic Programming Column Generation Heuristic

In this section, an overview of the two-stage stochastic programming column generation heuristic is provided. In Subsection 5.3.1, we elaborate on its two-stage stochastic property. Further, Subsection 5.3.2 presents the column generation algorithm used.

5.3.1 Two-Stage Stochastic Problems

As discussed in the problem description in Chapter 4, information about future customer arrivals is unknown when a service vehicle makes decisions upon a station arrival. In order to capture this uncertainty in the problem, stochastic programming can be applied. Stochastic programming involves defining a finite set of future scenarios, and use these to find a solution that values flexibility. A deterministic solution does not value flexibility, and could hence be suboptimal.

Two-stage stochastic programming separate between decisions that must be made here and now, first stage decisions, and second stage decisions that can be made at a later stage when more information about the future is revealed. The basic idea of two-stage stochastic programming is that (optimal) first stage decisions should be based on data available at the time the decisions are made, and not depend on future observations. The first stage decisions should however be flexible in terms of adapting to the stochastic information that is revealed at a later stage.

The DSBRCP is solved upon every station arrival, where the following six decisions are made:

1. The number of charged bikes to unload from the service vehicle
2. The number of charged bikes to load onto the service vehicle

3. The number of flat bikes to unload from the service vehicle
4. The number of flat bikes to load onto the service vehicle
5. The number of batteries to swap
6. The station to visit next

The above decisions represent the *first stage decision* in a two stage formulation. The stochastic information regarding customer arrivals is represented in a scenario. As the first stage decisions must be taken without information about the future, they are solely based on the current information and thus scenario independent. The solutions of the DSBRCP are such scenario independent first stage decisions. Upon arrival at the next station, new information about the system state is revealed, which is used to make new first stage decisions. As the problem is solved upon every station arrival, only the first stage decisions are used for operational decision support.

5.3.2 Column Generation Algorithms

Column generation algorithms have been specially designed for solving mathematical programs with a huge number of variables. Unfortunately, this method suffers from slow convergence, something that limits its efficiency and usability. To attempt to overcome these limits, Moungla et al. (2010) propose a method that combines exact optimization and heuristics, which is used as inspiration when constructing the heuristic detailed below.

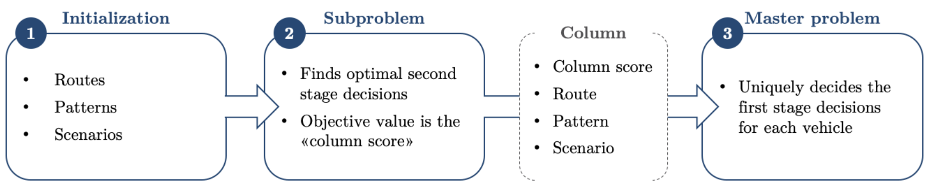

1
Initialization
Routes
Patterns
Scenarios
2
Subproblem
Finds optimal second stage decisions
Objective value is the «column score»
Column
Column score
Route
Pattern
Scenario
3
Master problem
Uniquely decides the first stage decisions for each vehicle

Figure 5.5: Overview of the decisions and flows in the column generation heuristic

Figure 5.5 shows the three steps in the column generation heuristic. The first step is implemented heuristically, while the second and third step find optimal solutions. In the first step, a set of routes and pattern combinations are generated using a branching algorithm. A *route* defines a sequence of station visits and the vehicle's corresponding arrival time at each station. A *pattern* contains parameters for the rebalancing and swapping decisions performed at the starting station in the route. Each generated pattern represents a possible first stage decision, i.e. rebalancing and charging decisions at the station where the service vehicle is currently located.

In step two, a subproblem is solved for every combination of route, pattern, customer arrival scenario, and vehicle. The objective value of the optimal solution serves as a score for this combination. Together, the route, pattern, customer arrival scenario, and their respective subproblem score, compose a *column* that is passed to the third step of the heuristic, the master problem. The master problem, finds the optimal combination of the set of columns for the full fleet of service vehicles. This step is solved by stochastic programming, and the solution method is elaborated in Section 5.4. As the master problem can only choose from the predefined set of routes for each service vehicle, this is in fact a *Restricted Master Problem (RMP)*. However, we refer to the problem as the *Master Problem (MP)* in this thesis.

The master problem solves the problem for the set of proposed columns. Thus, it is crucial with efficient initialization of route and pattern combinations as these define the solution space. Much attention is therefore directed towards the initialization step in this thesis.

5.4 Master Problem

The problem consists of two sets of nodes. The first set, $\mathcal{S}^+$, contains all stations in the BSS, including the depot. The second set, $\mathcal{S}$, contains the subset of stations where bike rebalancing and battery swaps can be performed, i.e. not including the depot. Both sets of stations are indexed by i. Further, the set $\mathcal{V}$ consists of service vehicles, indexed by v. Each vehicle v has an associated set of possible routes, $\mathcal{R}_v$, and an associated set of possible patterns, $\mathcal{P}_v$. Routes are indexed by r and patterns are indexed by p. Lastly, the set $\mathcal{F}$ defines the set of possible customer arrival scenarios, indexed by s.

There are five types of quantity parameters describing the first stage decision performed by vehicle v driving route r with pattern p. Q^{FCL}_{vrp} and Q^{CCL}_{vrp} denote the quantities of flat and charged bikes loaded onto the vehicle, respectively. Similarly, Q^{FCU}_{vrp} and Q^{CCU}_{vrp} denote the unloaded quantities of flat and charged bikes. Q^{B}_{vrp} stores the number of battery swaps performed by service vehicle v at the start station of route r with pattern p. Note that all routes considered by a vehicle have the same starting station, as the routes are chosen from a vehicle indexed set.

A service vehicle has an inventory consisting of L^{BV}_v batteries, L^{FV}_v flat bikes and L^{CV}_v charged bikes at its origin station. The capacity for bikes at each service vehicle is Q^{CV}_v. Further, the starting station for vehicle v has an inventory of L^{FS}_v flat bikes and L^{CS}_v upon arrival of vehicle v. Each station has an associated capacity Q^{S}_v.

Recall that a subproblem, step two of the presented heuristic, is solved for a given combination of route, pattern and scenario. The objective value of the subproblem solution for route r, pattern p and scenario s is used in the master problem as parameter R_{rps}. This value aims at expressing the benefit of choosing route r and pattern p in scenario s. Each scenario s has probability P_s of occuring. The binary matrix A_{irv} indicates the first station i for a vehicle v in a specific route r. The matrix is used to determine the next station decision in a given scenario.

The model uses the variables λ_{vrps} to find the weighted combination of columns that results in the highest total column score, while satisfying the problem constraints. The quantity variables q_v^B, q_v^{FCL}, q_v^{CCL}, q_v^{FCU} and q_v^{CCU} ensure that the first stage quantity decisions are uniquely defined for each vehicle. Further, the first stage next station decisions are defined by the binary variable x_{ivs} for each vehicle in each scenario. Non-anticipativity constraints capture the operational decision x_{iv}, making the decision consistent between scenarios. The full notation used in the master problem is summarized in Table 5.1.

Table 5.1: Notation used for modelling the master problem

Sets	
$\mathcal{S}^+$	Set of stations, including the depot
$\mathcal{S}$	Set of stations
$\mathcal{V}$	Set of vehicles
$\mathcal{R}_v$	Set of possible routes for vehicle v
$\mathcal{P}_v$	Set of possible quantity patterns for vehicle v
$\mathcal{F}$	Set of possible customer arrival scenarios
Indices	
i	Stations $i \in \mathcal{S}^+$
v	Vehicle $v \in \mathcal{V}$
r	Route $r \in \mathcal{R}_v$
p	Pattern $p \in \mathcal{P}_v$
s	Scenario $s \in \mathcal{F}$

Parameters

A_{ivr}	1 if station i is the start station in route r for vehicle v, otherwise 0
R_{rps}	Objective value of subproblem for route r with pattern p in scenario s
P_s	Probability of scenario s
Q_{vrp}^{FCL}	Number of flat bikes loaded to service vehicle v at the start station for route r with pattern p
Q_{vrp}^{CCL}	Number of charged bikes loaded to service vehicle v at the start station for route r with pattern p
Q_{vrp}^{FCU}	Number of flat bikes unloaded from service vehicle v at the start station for route r with pattern p
Q_{vrp}^{CCU}	Number of charged bikes unloaded to service vehicle v at the start station for route r with pattern p
Q_{vrp}^{B}	Number of battery swaps performed by service vehicle v at the start station for route r with pattern p
Q_v^{CV}	Capacity for bikes at service vehicle v
Q_v^{S}	Station capacity at the origin station for service vehicle v
L_v^{FV}	Load of flat bikes at service vehicle v at its origin station
L_v^{CV}	Load of charged bikes at service vehicle v at its origin station
L_v^{BV}	Load of batteries at service vehicle v at its origin station
L_v^{FS}	Load of flat bikes at its origin station for service vehicle v
L_v^{CS}	Load of charged bikes at its origin station for service vehicle v

Variables

λ_{vrps}	Weighting variable of the allocation of vehicle v to route r with pattern p in scenario s
x_{ivs}	1 if vehicle v travels directly to station i in scenario s, 0 otherwise
x_{iv}	Non-anticipativity variable for x_{ivs}, representing the first stage decision of which station to visit next
q_v^{FCL}	First stage decision of flat bikes loaded to vehicle v
q_v^{CCL}	First stage decision of charged bikes loaded to vehicle v

q_v^{FCU}	First stage decision of flat bikes unloaded from vehicle v
q_v^{CCU}	First stage decision of charged bikes unloaded from vehicle v
q_v^{B}	First stage decision battery swaps for vehicle v

Objective

The master problem objective finds the combination of columns that has the highest sum of rewards, equal to subproblem objective values. Each scenario is multiplied with the probability that the scenario occurs.

$$\max \sum_{s \in \mathcal{F}} P_s \sum_{r \in \mathcal{R}_v} \sum_{p \in \mathcal{P}_v} \sum_{v \in \mathcal{V}} R_{rps} \lambda_{rpvs} \tag{5.1}$$

Constraints

The master problem constraints secure that the first stage decisions are linear combinations of the weighted columns. The decisions must be valid in terms of the current vehicles and stations inventories. Constraints (5.2) - (5.6) capture the quantity variables for each vehicle as a linear function of the pattern quantity parameters in the weighted columns. Further, constraints (5.7) secure that the first stage loading decision is valid for the number of available bike slots at the vehicle. Constraints (5.8) work similarly for the first stage unloading decisions with respect to available locks at the station. The number of battery swaps must be lower than the available flat bikes after rebalancing decisions and lower than the number of charged batteries at the vehicle, secured by (5.9) and (5.10). Constraints (5.11) and (5.12) restrict the number of flat and charged bikes loaded from a station, respectively. The number of unloaded bikes at a station are restricted by Constraints (5.13) and (5.14) in the same manner. Constraints (5.15) capture the next station to visit for a specified vehicle in a specified scenario. These constraints also secure that all columns with nonzero λ-values in a given scenario must have the same first station. Each vehicle can decide on maximum one station to visit next, secured by (5.16). Further, each station, apart from the depot, may be chosen as the next destination by at most one vehicle. Constraints (5.17) secure this. Constraints (5.18) are non-anticipativity constraints, and ensure that the first stage next station decision is the same for all scenarios. As an extention to this, Constraints (5.19) ensure that each vehicle chooses a station to visit next. Lastly, constraints (5.20) - (5.22) restrict the feasible values for the decision variables.

$$\sum_{r \in \mathcal{R}_v} \sum_{p \in \mathcal{P}_v} Q_{rpv}^{FCL} \lambda_{rpvs} = q_v^{FCL} \quad v \in \mathcal{V}, s \in \mathcal{F} \tag{5.2}$$

$$\sum_{r\in\mathcal{R}_v}\sum_{p\in\mathcal{P}_v} Q_{rpv}^{CCL}\lambda_{rpvs} = q_v^{CCL} \quad v\in\mathcal{V}, s\in\mathcal{F} \tag{5.3}$$

$$\sum_{r\in\mathcal{R}_v}\sum_{p\in\mathcal{P}_v} Q_{rpv}^{FCU}\lambda_{rpvs} = q_v^{FCU} \quad v\in\mathcal{V}, s\in\mathcal{F} \tag{5.4}$$

$$\sum_{r\in\mathcal{R}_v}\sum_{p\in\mathcal{P}_v} Q_{rpv}^{CCU}\lambda_{rpvs} = q_v^{CCU} \quad v\in\mathcal{V}, s\in\mathcal{F} \tag{5.5}$$

$$\sum_{r\in\mathcal{R}_v}\sum_{p\in\mathcal{P}_v} Q_{rpv}^{B}\lambda_{rpvs} = q_v^{B} \quad v\in\mathcal{V}, s\in\mathcal{F} \tag{5.6}$$

$$q_v^{CCL} + q_v^{FCL} \leq Q_v^{CV} - L_v^{FV} - L_v^{CV} + q_v^{CCU} + q_v^{FCU} \quad v\in\mathcal{V} \tag{5.7}$$

$$q_v^{CCU} + q_v^{FCU} \leq Q_v^{S} - L_v^{CS} - L_v^{FS} + q_v^{CCL} + q_v^{FCL} \quad v\in\mathcal{V} \tag{5.8}$$

$$q_v^{B} \leq L_v^{FS} + q_v^{FCU} - q_v^{FCL} \quad v\in\mathcal{V} \tag{5.9}$$

$$q_v^{B} \leq L_v^{BV} \quad v\in\mathcal{V} \tag{5.10}$$

$$q_v^{FCL} \leq L_v^{FS} \quad v\in\mathcal{V} \tag{5.11}$$

$$q_v^{CCL} \leq L_v^{CS} \quad v\in\mathcal{V} \tag{5.12}$$

$$q_v^{FCU} \leq L_v^{FV} \quad v\in\mathcal{V} \tag{5.13}$$

$$q_v^{CCU} \leq L_v^{CV} \quad v\in\mathcal{V} \tag{5.14}$$

$$\sum_{r\in\mathcal{R}_v}\sum_{p\in\mathcal{P}_v} A_{irv}\lambda_{rpvs} = x_{ivs} \quad i\in\mathcal{S}^+, v\in\mathcal{V}, s\in\mathcal{F} \tag{5.15}$$

$$\sum_{i\in\mathcal{S}^+} x_{ivs} \leq 1 \quad v\in\mathcal{V}, s\in\mathcal{F} \tag{5.16}$$

$$\sum_{v\in\mathcal{V}} x_{ivs} \leq 1 \quad i\in\mathcal{S}, s\in\mathcal{F} \tag{5.17}$$

$$x_{iv} = x_{ivs} \quad i \in \mathcal{S}^+, v \in \mathcal{V}, s \in \mathcal{F} \tag{5.18}$$

$$\sum_{i \in \mathcal{S}^+} x_{iv} = 1 \quad v \in \mathcal{V} \tag{5.19}$$

$$0 \leq \lambda_{rpvs} \leq 1 \quad r \in \mathcal{R}_v, p \in \mathcal{P}_v, v \in \mathcal{V}, s \in \mathcal{F} \tag{5.20}$$

$$x_{ivs} \in \{0, 1\} \quad i \in \mathcal{S}^+, v \in \mathcal{V}, s \in \mathcal{F} \tag{5.21}$$

$$q_v^{FCL}, q_v^{CCL}, q_v^{FCU}, q_v^{CCU}, q_v^{B} \geq 0, integer \quad v \in \mathcal{V} \tag{5.22}$$

5.5 Initialization of Columns

The first step of the heuristic, the *initialization*, initializes the columns that are later used in the subproblem. A column is a combination of a certain route, pattern and customer arrival scenario. In this section, we describe how these initial combinations are generated using a branching algorithm and calculations. To ensure high quality solutions, we attempt to create sufficiently good initial solutions by taking advantage of the problem characteristics.

5.5.1 Overview of Branching Algorithm

For every service vehicle, Algorithm 1 is run to create possible routes. From the starting station, routes are created by adding stations with the highest criticality score to routes under construction. The branching factor B determines the number of different stations that are used to extend a route in every step. This branching factor is dynamic, and hence decreased as the routes are extended, resulting in more exploration of possible stations to visit early in the route. When the duration of the routes are greater than the time horizon, the route is completed. The arrival time associated with each station visit is estimated based on driving times provided by Google Maps and average time spent on a station by operators in Urban Infrastructure Partners.

When the routes are extended in line 8, a station with high enough criticality score to be included among the B top candidates after a filtering process is added to a route. The length of the route is then updated with the driving time to the station plus a predefined estimated time for the operations at the current station.

Algorithm 1: Branching Algorithm

```
Data: $S$:= Starting station, $\overline{T}$:= Time horizon, $B$:= Branching factor
Result: $\mathcal{R}$: set of finished created routes
1  $\mathcal{C}$: List of routes under construction $\leftarrow S$ ;
2  while $\mathcal{C}$ non-empty do
3      for each route r in $\mathcal{C}$ do
4          if duration of r $< \overline{T}$ then
5              Calculate criticality score for all stations;
6              Apply filtering;
7              Select the B stations with highest criticality score ;
8              $\mathcal{C} \leftarrow$ Add new routes to $\mathcal{C}$ by extending $r$ with the selected stations
9          end
10         else
11             Add $r$ to $\mathcal{R}$
12         end
13         Remove $r$ from $\mathcal{C}$
14     end
15 end
```

5.5.2 Calculating Criticality Scores

Each station in the BSS is given a criticality score to promote better initialization of routes. The criticality score aims to reflect the importance of visiting a station. For example, it is assumed to be important to visit a station with short expected time to violation, and less important to visit a station with low net demand. After calculating the criticality scores, the list of stations is sorted by decreasing criticality score and filtered. The route under construction is then extended into B new routes with a station from the B stations with highest criticality scores.

When calculating the criticality score, we define the vehicle's current position as the location of the station that was last added to the route. The criticality score of a station has four components:

1. T_i^S, driving time to station i from the vehicle's current position

2. t_i, expected time to violation, counted from arrival at the route's start station

3. n_i, net demand rate, measured in absolute value

4. d_i, deviation from ideal state at $\overline{T}$ if station i is not visited

1. Driving time

The first component of a station's criticality score favor stations that are located closer, represented by the driving time to station i from the station last added to the route.

2. Time to violation

The second component of the criticality score expresses the expected time to violation, defined in minutes. Lower expected time to violation makes the station visit more critical. If the station is already congested or starved, the expected time to violation is 0. Calculation of starvation considers charged bikes only, whereas congestion is dependent on both flat bikes and charged bikes. Hence, the computation of expected time to next violation is separated into (i) Computing expected time to next starvation and (ii) Computing expected time to next congestion. We can the express t_i as the minimum of the expected time to starvation and the expected time to congestion, more formally expressed by Equation (5.23).

$$t_i = \min\{t_i^S, t_i^C\} \tag{5.23}$$

The charged and flat bike inventories at station i are reflected in L_i^{CS} and L_i^{FS}, respectively. These parameters are defined when arriving at the start station of the route under construction. Further, the expected rates of customer arrivals within a time horizon, defined by customers per minute, is stored in parameters I_i^{IFR}, I_i^{ICR}, and I_i^{OCR}. These rates denote the three customer types as described in Section 5.2; incoming flat, incoming charged, and outgoing charged. The station's capacity is Q_i^S, and the expected time to starvation, t_i^S is calculated using Equation (5.24). Only non-negative, finite values of t_i^S are of interest.

$$t_i^S = \frac{L_i^{CS}}{I_i^{OCR} - I_i^{ICR}} \tag{5.24}$$

Equation (5.25) expresses the expected time to congestion at station i, t_i^C. Only non-negative, finite values of t_i^C are of interest.

$$t_i^C = \frac{Q_i^S - L_i^{CS} - L_i^{FS}}{I_i^{IFR} + I_i^{ICR} - I_i^{OCR}} \tag{5.25}$$

3. Net demand

The third criticality score component ensures higher criticality score for stations with high absolute net demand rate. The absolute value of the net demand is used as means to detect both high congestion rates and high starvation rates. The net demand rate at station i is defined by Equation (5.26). Note that the demand rates at a station may vary by time of day.

$$n_i = \mid I_i^{OCR} - I_i^{ICR} \mid \tag{5.26}$$

4. Deviation

Finally, deviation is included as a component in the criticality score. The calculation of deviation is dependent on whether the net demand of charged bikes, $I_i^{OCR} - I_i^{ICR}$, is positive or negative. If the net demand is positive, the inventory at $\overline{T}$, s_i^C is calculated as expressed in Equation (5.27). Otherwise, if the station has negative net demand, s_i^C is captured by Equation (5.28).

$$s_i^C = \max\{L_i^{CS} - (I_i^{OCR} - I_i^{ICR})\overline{T}, 0\} \tag{5.27}$$

$$s_i^C = L_i^{CS} + \min\{(I_i^{ICR} - I_i^{OCR})\overline{T}, (I_i^{ICR} - I_i^{OCR})t_i^C\} \tag{5.28}$$

Note that Equation (5.28) assumes that once a station is full, the inventory remains the same. In other words, it is not possible to retrieve charged bikes from a congested station. Equation (5.29) captures the deviation d_i, which is used in the criticality score.

$$d_i = \mid s_i^C - O_i \mid \tag{5.29}$$

Final expression of the criticality score

Combining all four components, the criticality score of station i, where $i \neq b$ (depot), is expressed by Equation (5.30). W^S, W^T, W^N and W^D are the weight constants for criticality score component 1, 2, 3 and 4, respectively.

$$W^S T_i^S - W^T t_i + W^N n_i + W^D d_i \tag{5.30}$$

The criticality score of the depot is calculated separately from other stations. Naturally, a depot visit is more critical when the vehicle inventory of charged batteries is low. However, it may also be effective to visit the depot if the vehicle is located at a station nearby. To ensure that the depot is considered as first stage decision when advantageous, the depot is given an artificially high criticality score when the battery inventory at a service vehicle is below a certain threshold. We assume a reasonable threshold value of five batteries.

5.5.3 Filtering

Although the criticality scores capture the importance of a station visit, the effect of a visit may vary depending on the current load of the service vehicle. For instance, there is little reason to visit a starving station if the service vehicle contains neither charged bikes nor batteries. Following from such empirical reasoning, we define a set of rules to filter out stations that it are unreasonable to visit. We refer to this process as a *filtering process*. Note that the filtering process is only applied at the root node in the branching algorithm, Algorithm 1. This is because we do not have the same inventory information at later branching steps due to rebalancing and charging decisions not yet performed. The rules can be divided into two categories: rules applicable when considering visits to stations in need of charged bikes, and rules applicable when considering visits to stations with risk of congestions occurring.

Category 1: rules when considering visits to stations in need of charged bikes.

- The service vehicle should have more than one charged bike or one battery
- If there are no charged bikes at the vehicle, there must be more than one flat bike at the station available for battery swaps

Category 2: rules when considering visits to stations with risk of congestions occurring.

- The service vehicle should have available parking space
- If the current station is also congested, the station should not be visited

Combined, criticality scores and filtering attempts to determine the most important stations to visit.

5.5.4 Estimating Patterns for the Subproblem

The starting station with associated bicycle inventory levels is known for the service vehicle when generating patterns. Information about the current inventory levels of the vehicle is also known. Based on this, upper and lower limits for battery swaps and loading variables are calculated. One combination of these limits forms an extreme pattern. The upper and lower limits for the decision variables represent a maximization and minimization of possible operational activity. The lower limit is zero for all five decision variables. The following rules are used to decide on the upper limits for the five decision variables:

1. **Battery swaps**: Maximum of available batteries on vehicle, and total number of flat bikes on station and vehicle
2. **Flat bike unload**: Maximum of flat bikes on vehicle and available locks on station

3. **Charged bike unload**: Minimum of the number of charged bikes to unload to reach ideal state, the charged bike inventory on vehicle, and available locks on station

4. **Flat bike load**: Maximum of flat bikes on station and available slots on vehicle

5. **Charged bike load**: Minimum of charged bikes on station to be removed to reach ideal state, and available slots on vehicle

The five quantity decision variables gives a total of $2^5 = 32$ possible extreme patterns, using the upper and lower limits. However, we restrict *charged bike unload* and *charged bike load* to be greater than zero at the same time, and *flat bike unload* and *flat bike load* to be greater than zero at the same time. Additionally, we are able to predict whether a station needs more or less bikes of each type based on the current inventory and expected customer arrivals. Based on this, it is unreasonable to generate patterns that deviate from the assumed need at the station. For example, we do not generate a pattern that unloads charged bikes from a starved station. Since the starting station is the same in all routes, a vehicle's pattern apply to all of the generated routes for the vehicle. These assumptions reduce the number of possible extreme patterns to twelve.

From optimization theory, we know that the optimal decision for each vehicle must be a fractional combination of extreme patterns. As the subproblem gives scores to completing entire patterns, it is necessary to also generate patterns representing fractions of the extreme patterns. In theory, the number of fractional patterns should approach infinity to ensure that the optimal solution is found. This would however substantially increase the solution time and complexity of the master problem, as the number of columns is drastically increased by introducing more patterns.

As an attempt to approximate the optimal solution, we create four versions of each original extreme pattern described above. Each of the four versions corresponds to the original pattern multiplied by a constant in the set $\{0.25, 0.5, 0.75, 1\}$. Four versions of each original extreme pattern gives a total of $4 \times 12 = 48$ possible patterns. However, the pattern with all lower limits, i.e. all zeros, generate four duplicate patterns in the set. Hence, we may subtract three of the patterns in the set. This results in a final number of $48 - 3 = 45$ possible patterns generated for a vehicle.

5.6 Subproblem

The subproblem is solved for a specified route, customer arrival scenario, and pattern. The purpose of the subproblem is to *score* this combination by finding the optimal second stage decisions. The optimal rebalancing and swapping strategy for the stations visited after the starting station in the route is found in the subproblem. The objective value serves as a score, and is included in the columns of the master problem to find optimal first stage decisions.

5.6.1 Notation

This section describes the sets, indices, parameters and variables that are used in the mathematical model. A summary is found at the end of the section.

As opposed to using the full set of stations $\mathcal{S}^+$, we let $\mathcal{K}$ be the set of indexes describing the route. The set $\mathcal{K}$ is indexed $k = 0, ..., m$. Index $k = 0$ refers to the starting station of the vehicle, and $k = m$ refers to the last station that is visited in the route, represented by $k = 4$ in Figure 5.6. The depot station is denoted by index b. Further, let $i(k)$ represent the actual station index in the full station set that is visited at index k. Observe that many of the parameters and variables only depend on the visit index.

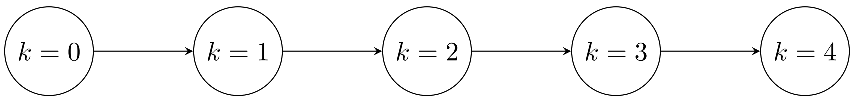


Figure 5.6: Illustration of the index set $\mathcal{K}$ for a route with five station visits.

At the beginning of visit $k = 1$, L^{BV}, L^{CV}, and L^{FV} represent the vehicle inventories of batteries, charged bikes, and flat bikes, respectively. These are adjusted for the decisions made contained in pattern at the starting station $k = 0$. We also adjust the bike capacity at the service vehicle based on the pattern decisions. The adjusted capacity Q^{CV} reflects the original vehicle capacity, with the net load of bikes from the pattern added. The adjustment is done to retain legal node balances at later visits. Q^{BV} represents the maximum storage capacity for batteries at the vehicle, and Q_i^S the number of locks at station i.

Station i has an has an associated vehicle arrival time T_i based on estimates in the route generation. The inventory of charged bikes at station i when service starts is L_i^{CS}, and L_i^{FS} for flat bikes. The vehicle inventory upon beginning of visit k is represented by variables l_k^{BV}, l_k^{CV} and l_k^{FV} for batteries, charged bikes and flat bikes, respectively. Recall that the subproblem finds the optimal second stage bike rebalancing and battery swap decisions. These decisions are captured for each station visit $k = 1, ..., m$. Battery swap decisions are stored in variables q_k^B, load variables are denoted by q_k^{FCL} and q_k^{CCL} for flat and charged bikes, and unloaded bike quantities are captured by q_k^{FCU} and q_k^{CCU} for flat and charged bikes, respectively.

To capture starvations and congestions that can be prevented by service vehicle operations, we need information about the customer arrivals happening from T_i to $\overline{T}$. Based on a specific customer arrival scenario, the number of customers arriving in the interval $[T_i, \overline{T}]$ is predetermined and contained in the parameters I^{IC}, I^{IF}, and I^{OC}. The superscript of the parameters indicates customer types; incoming charged (IC), incoming flat (IF) and outgoing charged (OC). The number of arrivals for each customer type is used to calculate violations occurring in the system. Similar to the base violation calculation in Section 5.2, the actual number of occurred violations after a station visit is captured in both a worst case and a best case customer arrival sequence. In addition, we separate between starvation violations and congestion violations. Thus, there are four variables representing the occurring violations. $\underline{v}_k^S$ and $\overline{v}_k^S$ denote the best case and worst case of starvations, while $\underline{v}_k^C$ and $\overline{v}_k^C$ capture the best case and worst case of congestions for the node with index k. The estimated number of occurred violations of each type is an average of the best and worst case results. The estimated total violation is a sum of estimated starvations and estimated congestions.

One of the subproblem objectives aim at maximizing the number of violations prevented by the service vehicle operations. This subobjective is weighted by W^V in the objective function. Base violations, introduced in Section 5.2, are used to find the number of prevented violations. The improvement is defined as a sum of the base violations at all visited stations in the interval $[T_i, \overline{T}]$ less the estimated number of occurred violations after service vehicle charging and rebalancing operations. Any violations occurring outside this time interval or at stations not visited cannot be prevented. Base violations for station i is defined in the parameters V_i^B. Violations occurring at the start station after service vehicle operations are captured in V^O. This number may be predetermined as the service vehicle operations at this station are solely based on a specified pattern.

Further, the subproblem objective aims at maximizing the prevented deviation measured at the time horizon $\overline{T}$. Deviation at a station is defined as the absolute difference between the number of charged bikes at the station at the time horizon, s_k^C, and the ideal state O_i at the corresponding station of node k. Calculation of prevented deviation is done similar to the calculation of prevented violations, using a base value as reference. The base deviation at station i, D_i^B, represents the deviation at the time horizon at a station if no service vehicle operations were performed. D^O denotes the deviation at the time horizon for the starting station. For all other stations, the deviation at the time horizon is captured by the variable d_k. Prevented deviations are weighed by W^D in the subproblem objective function.

Reward is given for unloading flat bikes at charging stations, i.e. stations in the set $\mathcal{S}^E$. The total amount of reward from second stage decisions in a route is captured in the variable r^F. Reward at the starting station, determined based on the pattern, is given in parameter R^O. The total reward is weighted by W^R in the objective function.

Lastly, the subproblem separates between the effect of decisions performed *now* (first stage decisions, defined by the pattern) and the effect of actions performed *later* (second stage decisions, captured by the subproblem model). Recall that only first stage decisions give operational guidance. It is hence of interest to prioritize resources preventing violations in this decision, as saving resources to prevent violations later is subject to more uncertainty. The effect of first stage decisions is weighted by W^N, while the effect of later operations is weighted by W^L.

Sets

$\mathcal{S}^E$	Set of charging stations
$\mathcal{K}$	Index set describing the route, $\mathcal{K} = \{0, 1, 2, ..., m\}$

Indices

k	Index $k \in \mathcal{K}$
$i(k)$	The station i corresponding to visit number k in the route
b	Depot station

Parameters

Q^{BV}	Storage capacity of batteries at the service vehicle
Q^{CV}	Storage capacity of bikes at the service vehicle, pattern adjusted
Q_i^S	Docking capacity at station i
L^{BV}	Load of batteries at the service vehicle at the beginning of the visit $k = 1$
L^{CV}	Load of charged bikes at the service vehicle at the beginning of the visit $k = 1$
L^{FV}	Load of flat bikes at the service vehicle at the beginning of the visit $k = 1$
L_i^{CS}	Number of bikes with sufficient battery at station i when service starts
L_i^{FS}	Number of flat bikes at station i when service starts
I_i^{OC}	Estimated number of customers demanding a charged bike from station i in the time interval $[T_i, \overline{T}]$

I_i^{IC}	Estimated number of customers wishing to park a charged bike at station i in the time interval $[T_i, \overline{T}]$
I_i^{IF}	Estimated number of customers wishing to park a flat bike at station i in the time interval $[T_i, \overline{T}]$
V_i^B	Total violations at station i in the time interval $[T_i, \overline{T}]$ if no service vehicles are in operation, referred to as *base violations*
V^O	Total violations at the origin station, $k = 0$, after visit
D_i^B	Deviation at station i at $\overline{T}$ if no service vehicles are in operation, referred to as *base deviation*
D^O	Deviation at the origin station at time $\overline{T}$
O_i	Ideal state at station i at time $\overline{T}$
R^O	Reward at starting station
W^N	Weight of the effect of the first stage decisions in the objective function
W^L	Weight of the effect of the second stage decisions in the objective function
W^V	Weight of violations in the objective function
W^D	Weight of deviation in the objective function
W^R	Weight of reward in the objective function

Variables

q_k^B	Number of battery swaps performed at node k
q_k^{CCU}	Number of charged bikes unloaded at node k
q_k^{FCU}	Number of flat bikes unloaded at node k
q_k^{CCL}	Number of charged bikes loaded at node k
q_k^{FCL}	Number of flat bikes loaded at node k
l_k^{BV}	Number of batteries in the vehicle at the beginning of a visit at node k
l_k^{CV}	Number of charged bikes in the vehicle at the beginning of a visit at node k
l_k^{FV}	Number of flat bikes in the vehicle at the beginning of a visit at at node k

$\underline{v}_k^S$	Starvation violations at station i in the time interval $[T_i, \overline{T}]$, based on best case customer arrival sequence
$\overline{v}_k^S$	Starvation violations at station i in the time interval $[T_i, \overline{T}]$, based on worst case customer arrival sequence
$\underline{v}_k^C$	Congestion violations at station i in the time interval $[T_i, \overline{T}]$, based on best case customer arrival sequence
$\overline{v}_k^C$	Congestion violations at station i in the time interval $[T_i, \overline{T}]$, based on worst case customer arrival sequence
s_k^C	Number of charged bikes at node k at time $\overline{T}$
d_k	Deviation from optimal state at node k at time $\overline{T}$
r^F	Number of flat bikes moved to charging stations, imposing a reward

5.6.2 Constraints

Vehicle loading constraints

Constraints (5.31) restrict swapping of more batteries than available at the service vehicle upon station arrival. Constraints (5.32) ensure that the total number of bikes loaded onto the vehicle do not exceed the available parking capacity at the vehicle. The battery inventory balance for a vehicle after a depot visit is handled in Constraints (5.33), and for all other stations in Constraints (5.34). The vehicle inventory balance for bikes are handled in Constraints (5.35) and (5.36). Lastly, Constraints (5.37) set the available battery load for vehicles on the first station visit after the start station. Constraints (5.38) and (5.39) do the same for charged and flat bikes, respectively.

$$q_k^B \leq l_k^{BV} \quad k = 1, ..., m \tag{5.31}$$

$$q_k^{CCL} + q_k^{FCL} \leq Q^{CV} - l_k^{CV} - l_k^{FV} + q_k^{CCU} + q_k^{FCU} \quad k = 1, ..., m \tag{5.32}$$

$$l_{k+1}^{BV} - Q^{BV} = 0 \quad k = 1, ..., m-1 \mid i(k) = b \tag{5.33}$$

$$l_{k+1}^{BV} - l_k^{BV} + q_k^B = 0 \quad k = 1, ..., m-1 \mid i(k) \neq b \tag{5.34}$$

$$l_{k+1}^{CV} - l_k^{CV} + q_k^{CCU} - q_k^{CCL} = 0 \quad k = 1, ..., m-1 \tag{5.35}$$

$$l_{k+1}^{FV} - l_k^{FV} + q_k^{FCU} - q_k^{FCL} = 0 \quad k = 1, ..., m-1 \tag{5.36}$$

$$l_1^{BV} = L^{BV} \tag{5.37}$$

$$l_1^{CV} = L^{CV} \tag{5.38}$$

$$l_1^{FV} = L^{FV} \tag{5.39}$$

Station loading constraints

It should not be possible to swap more batteries than there are flat bikes at a station after rebalancing actions, handled in Constraints (5.40). Constraints (5.41) and (5.42) ensure that the number of bikes loaded onto the vehicle does not exceed the station inventory of flat or charged bikes, respectively. The total number of unloaded bikes is restricted by the number of available locks at a station, secured in (5.43).

In our model, q_k^{FCU} is forced to 0 for nodes that correspond to a station in the set $\mathcal{S}^N$, dismissing unloading of flat bikes to non-charging stations. Further, q_k^{FCL} and q_k^B are set to 0 for all nodes corresponding to charging stations, implying that it is not possible to load flat bikes nor swap batteries at charging stations. Lastly, all bike rebalancing variables are set to 0 at the depot, eliminating rebalancing actions. These assumptions are included to remove solutions including meaningless operational effort in the model.

$$q_k^B \leq L_{i(k)}^{FS} + q_k^{FCU} - q_k^{FCL} \quad k = 1, ..., m \mid i(k) \neq b \tag{5.40}$$

$$q_k^{FCL} \leq L_{i(k)}^{FS} \quad k = 1, ..., m \mid i(k) \neq b \tag{5.41}$$

$$q_k^{CCL} \leq L_{i(k)}^{CS} \quad k = 1, ..., m \mid i(k) \neq b \tag{5.42}$$

$$q_k^{FCU} + q_k^{CCU} - q_k^{FCL} - q_k^{CCL} \leq Q_{i(k)}^S - L_{i(k)}^{CS} - L_{i(k)}^{FS} \quad k = 1, ..., m \mid i(k) \neq b \tag{5.43}$$

Non-negativity constraints

Constraints (5.44) - (5.45) are non-negativity constraints for all decision variables. Note from these that the subproblem is an LP problem.

$$q_k^B \geq 0 \quad k = 1, ..., m \mid i \neq b \tag{5.44}$$

$$q_k^{CCU}, q_k^{FCU}, q_k^{CCL}, q_k^{FCL}, l_k^{BV}, l_k^{CV}, l_k^{FV} \geq 0 \quad k = 1, ..., m \tag{5.45}$$

5.6.3 Objective Function of the Subproblem

The objective function has three subobjectives: 1) Maximize total prevented base violations, 2) Maximize total prevented deviation and 3) Maximize reward from unloading flat bikes at charging stations. We use the weighted sum method to treat the multi-objective optimization problem (MOP) as a single-objective optimization problem (SOP). The weight W^V is used for the first subproblem, W^D for the second, and W^R for the third. For each of the weight components emphasizing first stage and second stage decisions, now and later, the objective is structured as follows:

$$W^V \cdot \text{[Total prevented violations]} + W^D \cdot \text{[Total prevented deviation]} + W^R \cdot \text{[Total reward]}$$

Subobjective 1: Violations

The purpose of the subproblem is to score a combination of (route, pattern, scenario) by finding the optimal second stage decisions. Aiming at maximizing customer utility, one of the subproblem objectives is to prevent violations. As service vehicle operations is the only aid to prevent violations, this objective may be expressed as a maximization of *prevented* violations achieved by service vehicle operations. Through this formulation, the objective value is unaffected by violations at stations not visited. This property reduces bias between routes in the master problem.

The number of prevented violations is found by subtracting the violations that occurred at each station i within $[T_i, \overline{T}]$ *with* service vehicle operations, from the base violations V_i^B. However, capturing the number of occurred violations is challenging. This is because the violations occurring are dependent on the sequence of the customer arrivals, which is unknown. Thus, to estimate the number of violations, we calculate the number of violations occurring with the least favorable customer sequencing, worst case violations, and with the most favorable customer sequencing, best case violations, for a scenario. We refer to this average as the expected number of violations.

Constraints (5.46) capture the best case starvation violations at a station after it is visited. Similarly, Constraints (5.47) capture the best case congestion violations in the same interval. Further, the worst case violations are calculated by Constraints (5.48) and (5.49) for starvation and congestion violations, respectively.

$$L^{CS}_{i(k)} + q^{B}_{k} + q^{CCU}_{k} - q^{CCL}_{k} + I^{IC}_{i(k)} - I^{OC}_{i(k)} + \underline{v}^{S}_{k} \geq 0 \quad k = 1, ..., m \tag{5.46}$$

$$L^{CS}_{i(k)} + q^{CCU}_{k} - q^{CCL}_{k} + L^{FS}_{i(k)} + q^{FCU}_{k} - q^{FCL}_{k} + I^{IC}_{i(k)} + I^{IF}_{i(k)} - I^{OC}_{i(k)} + \underline{v}^{S}_{k} - \underline{v}^{C}_{k} \leq Q^{S}_{i(k)} \quad k = 1, ..., m \tag{5.47}$$

$$L^{CS}_{i(k)} + q^{B}_{k} + q^{CCU}_{k} - q^{CCL}_{k} - I^{OC}_{i(k)} + \overline{v}^{S}_{k} \geq 0 \quad k = 1, ..., m \tag{5.48}$$

$$L^{CS}_{i(k)} + q^{CCU}_{k} - q^{CCL}_{k} + L^{FS}_{i(k)} + q^{FCU}_{k} - q^{FCL}_{k} + I^{IC}_{i(k)} + I^{IF}_{i(k)} - \overline{v}^{C}_{k} \leq Q^{S}_{i(k)} \quad k = 1, ..., m \tag{5.49}$$

The sum of prevented violations at a station in the $\mathcal{K}$-sequence where $k \neq 0$, is formulated in (5.50).

$$V^{B}_{k} - \frac{1}{2}\left(\underline{v}^{S}_{k} + \overline{v}^{S}_{k} + \underline{v}^{C}_{k} + \overline{v}^{C}_{k}\right) \tag{5.50}$$

Note that we subtract the expected number of violations after a visit from the base case violations at a station. Hence, the objective value reflects the expected number of violations that was prevented by the station visit.

Subobjective 2: Deviations

The objective function also attempts to minimize the total deviation of the system at the end of each time horizon. Recall that the deviation is defined as the absolute value between the ideal state, O_i, and current station inventory of charged bikes, s^{C}_{k}, both measured at $\overline{T}$. Similar to the approximation of violations, this objective may also be formulated as a maximization of the prevented deviation as compared to the base deviation D^{B}_{i} at each station. Base deviation is, analogously to base violation, defined as the observed deviation if no service vehicles were operating in the system.

The number of charged bikes at station k at the end of the time horizon is estimated in variable s^{C}_{k} by Constraints (5.51) and (5.52). From these variables and the ideal states, the estimated deviation at each station in the route is captured by the combination of Constraints (5.53) and (5.54).

$$s_k^C \geq L_{i(k)}^{CS} + q_k^{CCU} - q_k^{CCL} + q_k^B + I_{i(k)}^{IC} - I_{i(k)}^{OC} + \frac{1}{2}\left(\overline{v}_k^S + \underline{v}_k^S - \overline{v}_k^C + \underline{v}_k^C\right) \quad k = 1, ..., m \tag{5.51}$$

$$s_k^C \leq L_{i(k)}^{CS} + q_k^{CCU} - q_k^{CCL} + q_k^B + I_{i(k)}^{IC} - I_{i(k)}^{OC} + \frac{1}{2}\left(\overline{v}_k^S + \underline{v}_k^S - \overline{v}_k^C + \underline{v}_k^C\right) \quad k = 1, ..., m \tag{5.52}$$

$$d_k \geq O_{i(k)} - s_k^C \quad k = 1, ..., m \tag{5.53}$$

$$d_k \geq s_k^C - O_{i(k)} \quad k = 1, ..., m \tag{5.54}$$

The number of prevented deviations as a result of service vehicle operations at stations $k = 1, ..., m$ may further be expressed by Equation (5.55).

$$\sum_{k=1}^{m} D_k^B - d_k \tag{5.55}$$

For station $k = 0$, where the service vehicle operations are determined based on the pattern, Equation (5.56) yields the estimated prevented deviation.

$$D_0^B - D^O \tag{5.56}$$

Subobjective 3: Reward

We want to reward moving flat bikes to charging stations. This is both to reduce the need for battery swaps in the system and to exploit the charging capacity at the stations by favoring flat bikes over charged bikes when unloading bikes at charging stations. The total number of flat bicycles unloaded at a charging station is captured by Constraint (5.57).

$$r^F \leq \sum_{i(k) \in S^E} q_k^{FCU} \tag{5.57}$$

Final expression for the objective function

The objective function of the subproblem distinguishes between the effect of the first stage decisions and the second stage decisions. The effect of first stage decisions is weighted by W^N, while the effect of second stage decisions is weighed by W^L. Within each component, we may formulate the objective as a combination of the three subobjectives, each weighted by its respective weight. The final objective function is formulated in (5.58).

$$\max \quad W^N \Big[W^V \left(V_0^B - V^O \right) + W^D \left(D_0^B - D^O \right) + W^R R^O \Big]$$
$$+ W^L \sum_{k=1}^{m} \Big[W^V \left(V_k^B - \frac{1}{2} \left(\underline{v}_k^S + \overline{v}_k^S + \underline{v}_k^C + \overline{v}_k^C \right) \right) + W^D \left(D_k^B - d_k \right) + W^R r^F \Big] \tag{5.58}$$

Chapter 6

Simulation Framework

The heuristic solution approach described in this thesis provides decision support based on assumptions about the system evolution within a time horizon of 25 minutes. In an attempt to better evaluate the longer term solution quality of these decisions, a simulation framework is set up, imitating the evolution of a real-world BSS under different customer arrival scenarios. This chapter elaborates on the set-up of this simulator.

6.1 Overview of Simulator

In the simulator, we evaluate the number of violations occurring over what can be seen as a series of customer arrival scenarios, referred to as a *day*. Specifically, a day represents 16 hours of simulation. Note however that the term customer arrival scenarios refer to the predicted arrivals used in the subproblems, whereas the simulated arrivals actually affect the system evolution. The performance of the rebalance and battery swap strategies are evaluated based on the recorded number of violations occurring in a day of simulation.

The simulator used is programmed in Python and event triggered. An event can be triggered by a customer arrival at a station, denoted as *bicycle trigger*, or a service vehicle arrival at a station, denoted as *vehicle trigger*, illustrated in Figure 6.1. A bicycle trigger results in a created trip or completed trip in the simulation environment. A vehicle trigger prompts solving of the DSBRCP using parameter data from a snapshot of the simulated system. When solving this problem, the other vehicles in the system are assumed to also have arrived at the station next up in their respective routes. This is demonstrated in Figure 6.2, where Vehicle 1 is assumed to have arrived at Station 5 when solving the DSBRCP triggered by Vehicle 2's arrival at Station 7. However, only the strategy obtained for Vehicle 2 is realized in the simulation.

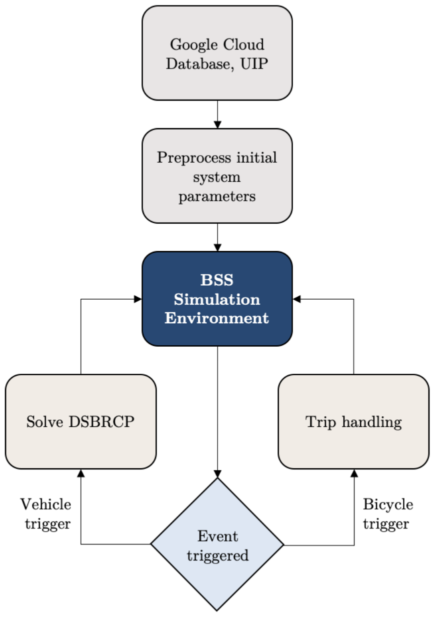

Google Cloud Database, UIP
Preprocess initial system parameters
BSS Simulation Environment
Solve DSBRCP
Trip handling
Vehicle trigger
Event triggered
Bicycle trigger

Figure 6.1: Flowchart illustrating the high-level simulation interactions in the Python simulator

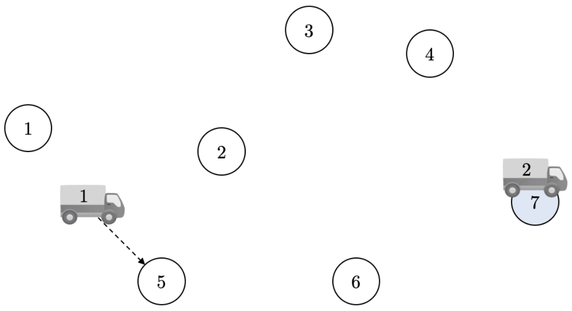


Figure 6.2: Demonstration of vehicle location assumption used to solve the DSBRCP when a vehicle arrives at a station. Vehicle 2 has arrived at station 7, triggering solving of the DSBRCP. Vehicle 1 is assumed to have arrived at its next destination, station 5.

When the simulation starts at t^{start}, the system is set up with predefined inputs describing the initial state of the system for the specified starting time. When a service vehicle arrives at a station, a snapshot of the current system inventories are used to solve the DSBRCP. The solution of the problem contains the service vehicle's route and rebalancing and battery swap strategy, and is used to update the system before continuing the simulation. Bike trips are based on poisson arrivals of customers and handled in sequence between vehicle arrivals. Requests to initiate trips at stations with no available charged bikes result in starvation violations. Requests to park bikes at full stations result in congestion violations. Every time a vehicle arrives at a new station, the DSBRCP is re-solved with the new stochastic system information revealed upon arrival. This is done in an iterative manner until the duration of the simulation is up at t^{end}, shown in Figure 6.3. The current simulation time is tracked using a variable $t^{current}$. When t^{end} is reached, the total number of violations are calculated and evaluated. Note that it is essential that the configurations are evaluated on several days of simulation as each day only represents one possible outcome of customer arrivals.

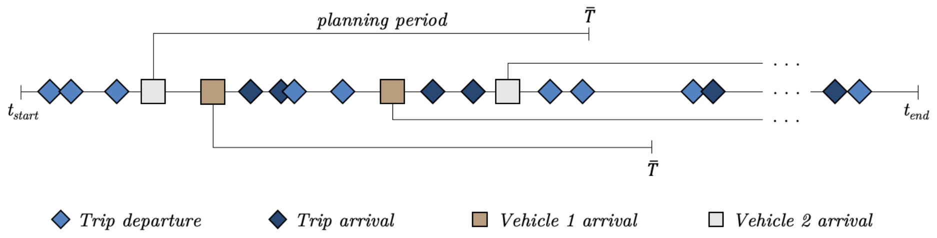


Figure 6.3: Sequential handling of events for one scenario of customer arrivals. Subproblems are solved every time a vehicle arrives at a station. The time horizon represents the planning period used for customer arrival prediction when solving the DSBRCP

6.2 Modelling Customer Arrivals Through a Poisson Process for a Station in the Subproblem

A Poisson Process is a model for a series of discrete events where the average time between events is known, but the exact timing of events is random. The arrival of an event is independent of the event before, meaning that the waiting time between events is memoryless (Gallager, 1997). A parameter λ, typically referred to as rate or intensity, is the expected number of customers arriving per minute in a given time interval. The Poisson distribution is a discrete distribution for the probability of x events happening in a time interval defined on non-negative integers as follows:

$$Poisson(\lambda t) \sim P(X = x) = \frac{e^{-\lambda t}(\lambda t)^x}{x!}, \quad x = 0, 1, 2, ...$$

where x is the number of customer arrivals, λ is the intensity rate, and t is time steps. A Poisson process can be either homogeneous with constant intensity rate, commonly referred to as a counting process, or non-homogeneous with varying intensity rate. Since the time horizon is short, we use a homogeneous Poisson process to model the customer arrivals within each time horizon (Drazek, 2013).

The arrival of customers are modelled as separate and independent Poisson processes for the three groups of customers. For each time horizon, the intensity rates, denoted λ_1, λ_2, λ_3, are calculated based on historical data from the stations. The homogeneous Poisson point process adheres to its own form of the law of large numbers. More specifically, with probability one:

$$\lim_{t \to \infty} \frac{X(t)}{t} = \lambda$$

This implies that the intensity rate λ approaches the expected number of arrivals occurred per unit of time. This property is used to estimate the intensity rates for each station.

Algorithm 2 simulates the customer arrivals with a Poisson process for a customer type at a station. Every minute, the number of customer arrivals are drawn from the Poisson distribution adding the time of arrival for each customer to a customer arrival list.

Algorithm 2: Poisson arrivals for a station

```
  Data: $\lambda$:= arrival rate at station, $t$:= time steps in minutes
  Result: Customer arrivals:= list of times for customer arrivals
1 for i in range t do
2     time step arrivals = $X \sim Poisson(\lambda)$ ;
3     for j in range time step arrivals do
4         add i to Customer arrivals
5     end
6 end
```

6.3 Generating Customer Trips

Bicycle trip events are generated and handled during simulation, mirroring real world trips performed by customers. All intended trips are generated at the beginning of the simulation period with a dedicated start station, end station, starting time, estimated end time, and whether the bike needs charging before starting a new trip, illustrated in Figure 6.4. The trips starting from each station with associated starting time is generated using Algorithm 2 within a specified simulation interval. A random draw from a probability distribution reflecting the typical customer movements from a station, determines the end station of a trip. Following from this, the end time is set to the starting time plus the transportation time between the start and end stations. A binomial draw is performed to decide if the bike needs charging after completing the trip. On average, a bike needs to be charged after every 20th completed trip.

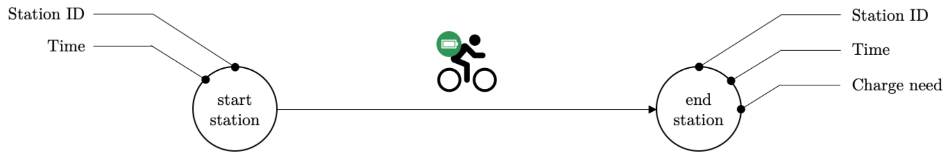


Figure 6.4: A trip object and its associated data fields

The simulator handles events in sequence, beginning with the earliest occurring event. Two stacks are used to keep track of events: a departure stack sorted by ascending starting time and an arrival stack sorted by ascending end time. Vehicle arrivals are only present in the latter stack. An event is handled by removing it from the stack and performing a set of actions. The actions performed are shown in a flow diagram in Figure 6.5 for each of the three event types. The stacks are resorted every time a new event is added to it, and the triggered event is the earliest occurring event looking at the two stacks combined.

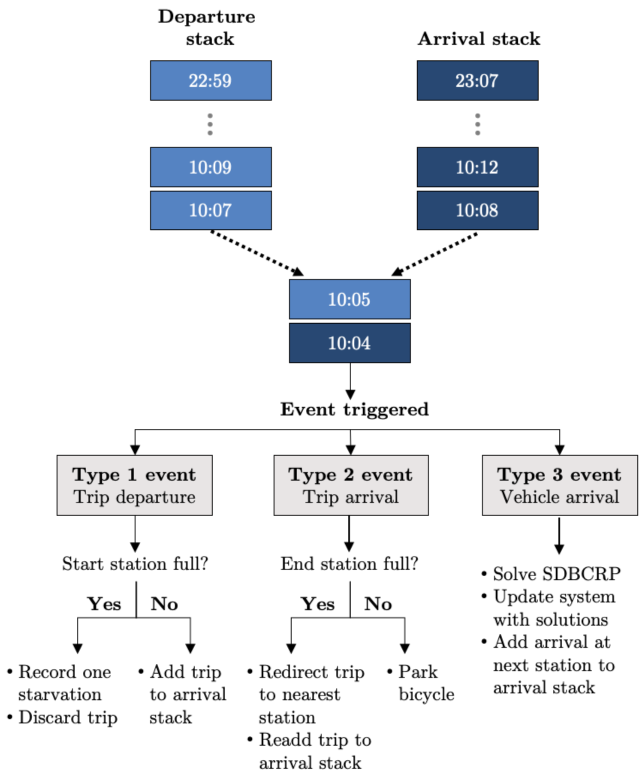

Departure stack
Arrival stack
22:59
23:07
10:09
10:12
10:07
10:08
10:05
10:04
Event triggered
Type 1 event Trip departure
Type 2 event Trip arrival
Type 3 event Vehicle arrival
Start station full?
End station full?
Yes
No
Yes
No
• Record one starvation • Discard trip
• Add trip to arrival stack
• Redirect trip to nearest station • Readd trip to arrival stack
• Park bicycle
• Solve SDBCRP • Update system with solutions • Add arrival at next station to arrival stack

Figure 6.5: Illustration of the performed simulator actions when an event is triggered

Chapter 7

Implementation and Generation of Test Instances

The column generation heuristic presented in Chapter 5 has been implemented in a PyCharm IDE with use of the Gurobi Python Interface. Subsections 7.1 and 7.2 present the key input data used and the test instances. Our test instances are based on UIP's largest BSS, the bike sharing system in Oslo, as well as their newly introduced electric BSS in Edinburgh. The BSS in Oslo serves as a main source of data, while the electric BSS in Edinburgh provides data regarding battery levels and charging. UIP has provided access to their databases with necessary data for the implementation, which was pulled using SQL-queries through a client from BigQuery, a Google Cloud Platform. Specifications of the computer and software used to run the model are listed in Table 7.1.

Table 7.1: Details of computer, solver, programming language, and IDE used in the computational studies

Processor	2x 3.4GHz Intel E5-2643v3 – 6 core
RAM	512.0 GB
Operating System	Linux 3.10.0
Gurobi License ID	375868
Gurobi version	Gurobi Optimizer version 9.0.2
Python version	Python 3.7.4
PyCharm version	PyCharm 2019.2.4 (Professional Edition)

7.1 Key Input Data

Key input data is used as basis for the parameters in the heuristic and simulator. Figure 7.1 shows how the collected data serves as foundation for the system set-up. Parameters not discussed in this chapter, e.g. component weights in the subproblem objective and the criticality score, are calibrated through testing in Chapter 8.

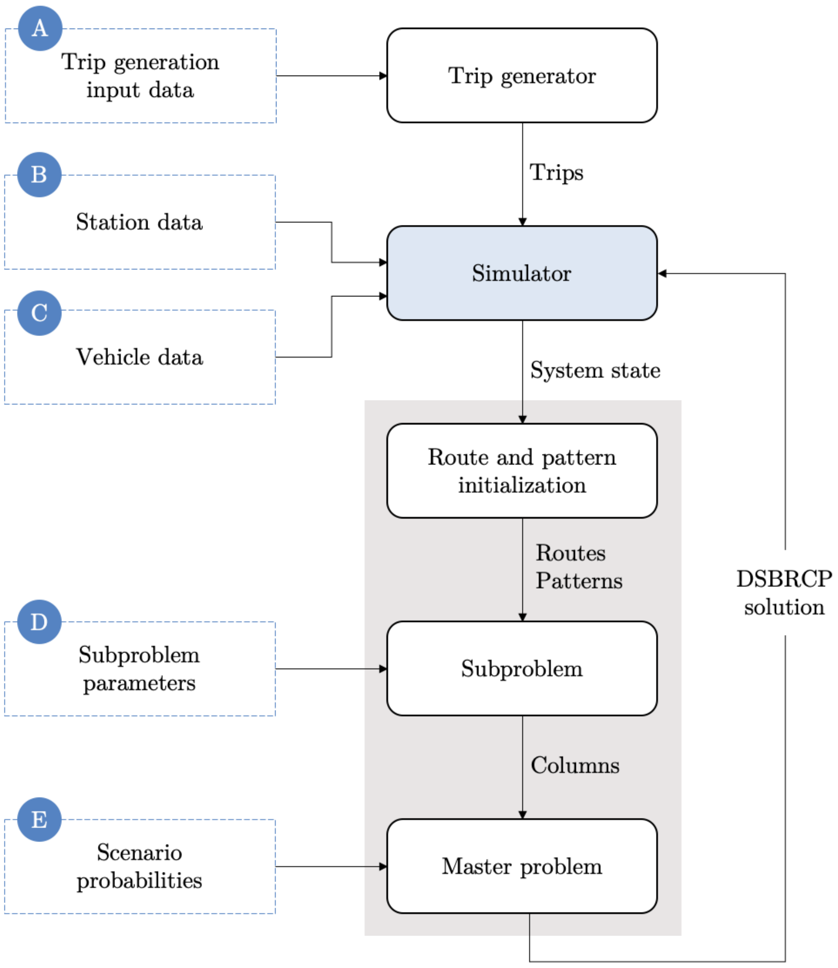


Figure 7.1: Flow of key input data used to generate parameters for the two-stage stochastic programming column generation heuristic in a simulation environment

The input data presented in Fig 7.1 can be split into five categories: (A) Trip generation input data, (B) Station data, (C) Vehicle data, (D) Subproblem parameters and (E) Scenario probabilities. Each category is detailed in the following subsections. Do also note that the grey area in Figure 7.1 represents the three stages of the DSBRCP heuristic presented in Chapter 5.

A. Trip Generation Input Data

Bicycle travel time

The driving time between two stations mulitiplied by a factor of 1.3 is used as an estimate for the associated bicycle travel time. This factor is assumed reasonable based on a quick analysis of the relationship between bicycle travel times from UIP's electric BSS in Edinburgh with associated driving times.

Outgoing Charged Customer arrivals

There are many factors that affect the arrival of customers at a station at a given time, i. e. time of day, time of week, time of year, weather, empty stations nearby, events occurring in near proximity. Many of these factors are difficult to predict and would require real-time input from external data sources to model well. In this thesis, we only consider UIPs demand estimates based on previous history of customer trips to generate outgoing charged customer arrivals. With the present bicycle fleet and stations, only half of the demand faced by customers wishing to initiate trips is typically met in the UIP in Oslo.

For every hour in a day, UIP has estimated an expected number of customers wishing to initiate a trip at every station in the Oslo BSS. This number is used as outgoing charged customer intensity rate in a Poisson arrival process to generate trips in the simulator, which is further elaborated in the following chapter.

Fraction of charged and flat bikes

In Edinburgh, a fully charged battery is flat after an average of 10 trips. As the average length of a trip in Oslo is half the average trip length in Edinburgh, we assume that 20 trips can be completed with one battery charge in the Oslo BSS. With this assumption, there is in general a $\frac{1}{20} = 5\%$ probability of a bicycle needing charge after a trip, i. e. arriving as a flat bicycle. This chance is used to perform a binomial draw to determine a bicycle's need of charging after a completed trip. For charging stations, all incoming bicycles are assumed to be charged bicycles after being parked for 30 minutes.

Next station probabilities

At every station i, there is a distribution describing the end station probabilities for initiated trips. This means, when initiating a trip from station i, the trip has a defined probability of ending at station j, denoted P_{ij}. As all trips must have an end station, these probabilities sum to one for all stations i in the BSS as follows:

$$\sum_{j \in \mathcal{S}} P_{ij} = 1 \quad i \in \mathcal{S}$$

Figure 7.2 shows the next station probabilities at station 2 in a fictive BSS with five stations. Observe that there is a 0.5 chance of a trip ending at station 4 when starting at station 2, while the probability of ending at station 3 is only 0.1. Do also note that the probabilities sum to one. A higher next station probability implies a greater flow of bicycles, illustrated by the thickness of the arrows. When generating a trip, the end station for a trip is drawn from the next station probability distribution at the start station. In the real world, this distribution will vary with time. However, we base this on estimates from UIP who simplifies it to be constant every day.

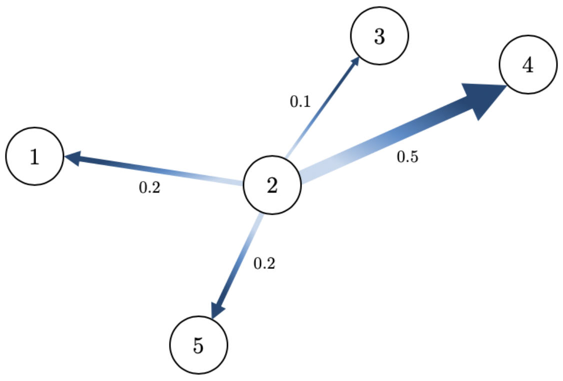


Figure 7.2: Illustration of the next station probabilities when starting a trip at station 2.

B. Station data

The set of stations used in the computational study is a subset of the stations used in UIP's BSS in Oslo. The subset consist of 200 stations in Oslo with activity on October 10th 2019. The stations are either charging stations or non-charging stations and have an associated parking capacity and geographical location. The latitude and longitude of these locations are used to estimate the driving time between stations through an API provided by Google Maps, returning the average driving times between two geographic coordinates. The depot location is set to Ullevålsallèen in Oslo, shown on the map in Figure 7.3. We assume that 10% of the stations are charging stations, and these are spread out evenly in the system.

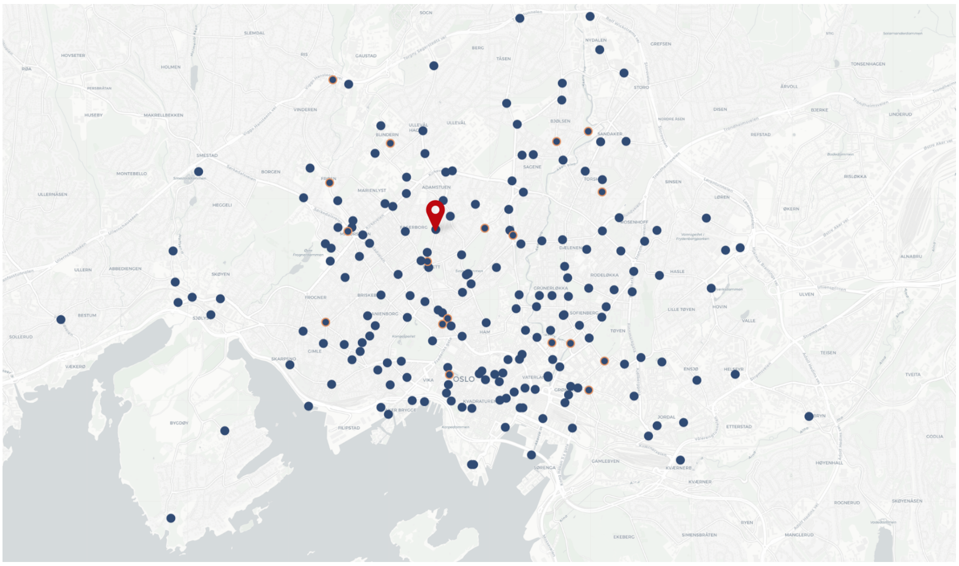


Figure 7.3: Map showing the location of the included test stations in the BSS in Oslo. The depot used in the model is marked with a red needle. Charging stations are indicated by an orange outline. Source: (Oslo Bysykkel, 2020b)

Inventory levels

The inventory level at each station varies with time. When initializing the system, the inventory levels are set to the actual number of bicycles at each station on October 10th 2019 at a chosen starting time. All bicycles are assumed to be fully charged when simulation starts. In an evoked simulation, the DSBRCP uses station inventory levels collected from a snapshot of the current state of the environment.

Ideal state

The ideal state for every station is estimated per hour by UIP for October 10th 2019. Their approach is to calculate the number of bicycles that minimizes the probability of violations occurring in the next three hours considering predicted demand. This is when the probability of congestion is equal to the probability of starvation, elaborated in the work of Espegren and Kristianslund (2015).

C. Service Vehicle Data

Number of vehicles and their capacities

The number of service vehicles and their capacities are based on information collected from UIP and fleets used in other similar systems around the world. The model can handle both homogeneous and heterogeneous fleets. Currently, the BSS in Oslo operates five service vehicles with capacity for 20 bikes. The battery capacity at each vehicle is assumed to be 40. At the start of a simulation, the initial vehicle load of bicycles is assumed to be no bicycles and 40 charged batteries. This is to keep the number of bicycles in the system constant regardless of the number of vehicles in operation.

Handling time and parking time

Based on observations of UIP's daily operations conducted by Hagen and Gleditsch (2017), the parking time and unit battery swap time are assumed constant with a value of 2 minutes and 0.5 minutes, respectively, to simplify the model. The battery swap unit handling time is assumed comparable to the bike balance unit handling time. In reality, these numbers will vary mainly based on the handling speed of the worker and the distance between the station and the parked service vehicle. The above mentioned times are used to estimate an average time spent at a station based on typical operational activity. When generating routes, this time is added at every station visit.

D. Subproblem parameters

Base violations and base deviations

Based on the customer arrival scenario generated through a Poisson arrival process, the total congestion, starvation and deviation when no service vehicles are operating is calculated as follows for each station, using notation from Chapter 5:

a. Starvations $= abs\left(min\left(0, L^{CS} + I^{IC} - I^{OC}\right)\right)$

b. Congestions $= max\left(0, L^{CS} + L^{FS} + I^{IC} + I^{IF} - min\left(L^{CS} + I^{IC}, I^{OC}\right) - Q^{S}\right)$

c. Deviations $= abs\left(L^{CS} + I^{IC} - I^{OC} + Starvations - Congestions - O\right)$

The sum of starvations and congestions is passed on as total base violations along with base deviations for each station in a generated route to the subproblem.

Customer arrivals in subproblem scenarios

The following three intensity rate estimates are needed to generate customer arrival scenarios in the subproblems for every station in a route:

1. λ_i^{IC} = the average number of incoming charged bicycles for a station i
2. λ_i^{IF} = the average number of incoming flat bicycles for a station i
3. λ_i^{OC} = the average number of outgoing charged bicycles for a station i

When a subproblem is solved, the rates from the appropriate hour of the day is used. The outgoing rate λ_i^{OC} for a station i is equal to the number of outgoing charged customer arrivals per minute, derived from the hourly rate elaborated in part A. An incoming bicycle rate for a station j is calculated as follows:

$$\lambda_j^{IC} = \sum_{i \in \mathcal{S}} \lambda_i^{OC} \times P_{ij} \times P_{initiated} \times R_C \quad j \in \mathcal{S}$$

$$\lambda_j^{IF} = \sum_{i \in \mathcal{S}} \lambda_i^{OC} \times P_{ij} \times P_{initiated} \times R_F \quad j \in \mathcal{S}$$

where P_{ij} is the next station probability of traveling to station j from station i and $P_{initiated}$ is the general chance of a trip being initiated. Further, R_C and R_F are the typical fraction of charged bicycles and flat bicycles arriving at station, respectively. Following from the discussion regarding the probability of a flat bicycle arriving in part A, these fractions are set to 0.95 and 0.05. Note that all three customer arrival rates represent *requests* for a bicycle or a lock, i. e. predictions in the subproblems, and not confirmed events. During simulation, trips are generated solely based on λ_i^{OC} and next station probabilities.

Figure 7.4 illustrates how a subproblem scenario is generated. The combined sequence of arrivals over a single time horizon for the three customer types, is referred to as a customer arrival scenario.

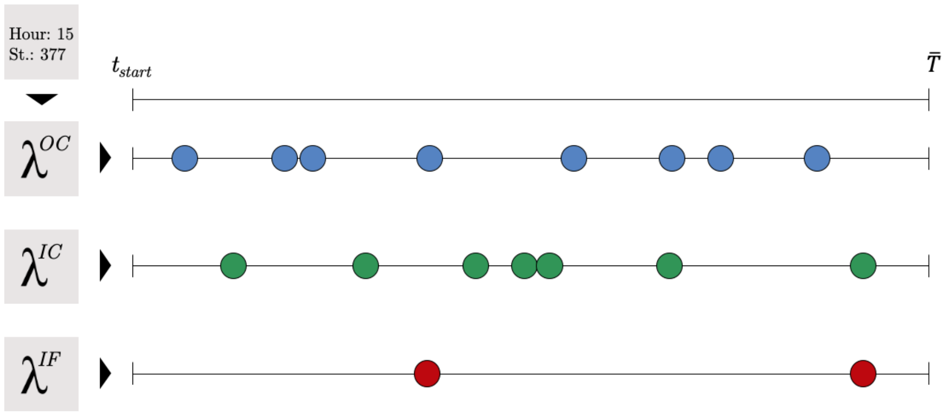


Figure 7.4: Illustration of how a subproblem scenario is generated based on three intensity rates in a Poisson arrival process.

E. Scenario probabilities

All scenarios are assumed to have equal probabilities. Thus, parameter P_s in the master problem is calculated using this simple formula:

$$P_s = \frac{1}{\text{number of scenarios}}$$

7.2 Example of a DSBRCP Test Instance and Solution

In this section, we illustrate and elaborate on a toy sized test instance and solution of the DSBRCP. The chosen instance has six stations, representing a subset of the Oslo BSS. One of the stations, station 6, is a charging station. One service vehicle operates in the system. The vehicle is located at station 2, Hausmanns gate. The initial vehicle load is 8 charged bicycles, 3 flat bicycles and 21 charged batteries. Table 7.2 shows a subset of the station input parameters, aiming at illustrating important characteristics for each station within the next time horizon $\overline{T}$ of 25 minutes. The customer arrivals listed in the table are based on UIP's estimates for the specified time of day. Their estimates are used as intensity rates in a simulated Poisson arrival process over 25 minutes. The presented numbers are calculated from the outcome of this process. Based on the data in Table 7.2 and the other parameters described in Section 7.1, the service vehicle makes a first stage decision by solving the DSBRCP.

Table 7.2: Overview of stations used in the example instance. Initial inventory is defined at 11.00 am, Ideal state is defined at 11.25 am. Customer arrivals of type OC, IC and IF correspond to outgoing charged, incoming charged and incoming flat bikes, respectively.

			Initial inventory		**Customer arrivals**			
ID	**Name**	**Cap**	**Charged**	**Flat**	**OC**	**IC**	**IF**	**Ideal**
1	Vår Frelsers gravlund sør	18	8	2	3	1	0	12
2 🚚	Hausmanns gate	21	1	2	3	0	0	10
3	Kjeld Stubs gate	12	11	1	0	5	1	7
4	Slottsparken Øst	40	28	8	5	6	1	21
5	Huitfeldts gate	18	8	0	2	4	0	9
6	Over Rådhusgata	21	11	2	4	2	0	11

Initially, the vehicle is located at station 2, Hausmanns gate. This station has one charged bicycle and two flat bicycles, as observed from the second row in Table 7.2. Within the next time horizon, three customers requesting a charged bike are expected to arrive. No incoming bikes are expected. Naturally, the station cannot meet the desired demand for charged bicycles with its current inventory. Hence, two starvation violations will occur if the service vehicle does not perform any actions to increase the charged bicycle inventory at the station. Also, note that the ideal state at the station is ten charged bicycles at the end of the time horizon, 11.25 am.

The solution of the DSBRCP for this instance is visualized in Figure 7.5. The numbers in the green, red and light blue circles above the service vehicle, represent net change in charged bicycles, net change in flat bicycles and net change in charged batteries on the vehicle, respectively. Observe from the illustration that the DSBRCP heuristic suggests to unload ten charged bikes and swap two batteries. Note that performing only battery swap actions would be sufficient to meet the expected demand within the next time horizon. However, this would lead to high expected deviation from ideal state at the time horizon. To minimize the expected deviation at the time horizon, i.e. approximate the ideal state inventory of charged bicycles, the service vehicle unloads ten charged bikes. Ultimately, the DSBRCP solution suggests to visit station 3, Kjeld Stubs gate, next. Observe from Table 7.2 that station 3 is currently full, and that the station expects arrival of six bikes in total within the next time horizon. Rebalancing decisions at the station may prevent some of the six arrivals from generating congestion violations. The other stations in Table 7.2 are not subject to similar risks of violations within the next time horizon. Hence, the choice of visiting station 3 next seems reasonable.

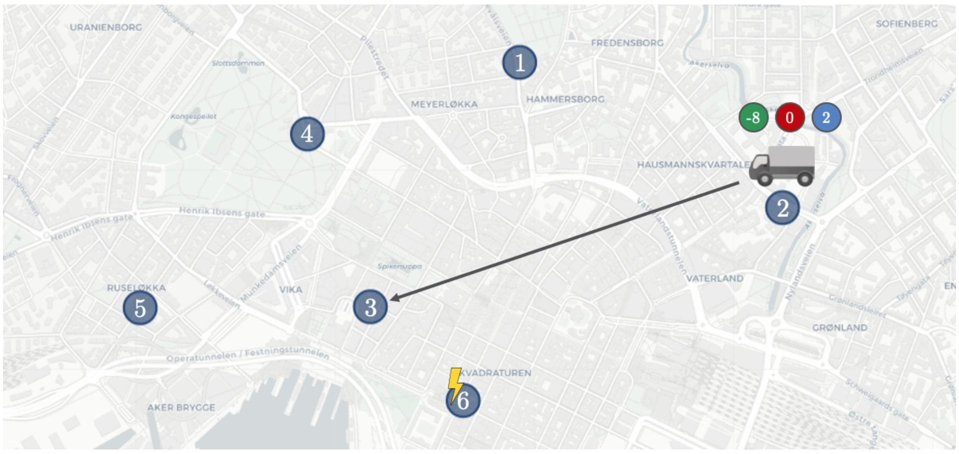


Figure 7.5: Illustration of the first stage decision made by the service vehicle based on the test instance described in Table 7.2.

The visualization in Figure 7.5 demonstrates an example of a DSBRCP solution. In a real-world environment, the DSBRCP is solved sequentially. Figure 7.6 shows how a sequence of station visits, i.e. a vehicle's travel route within for example an hour, could look like.

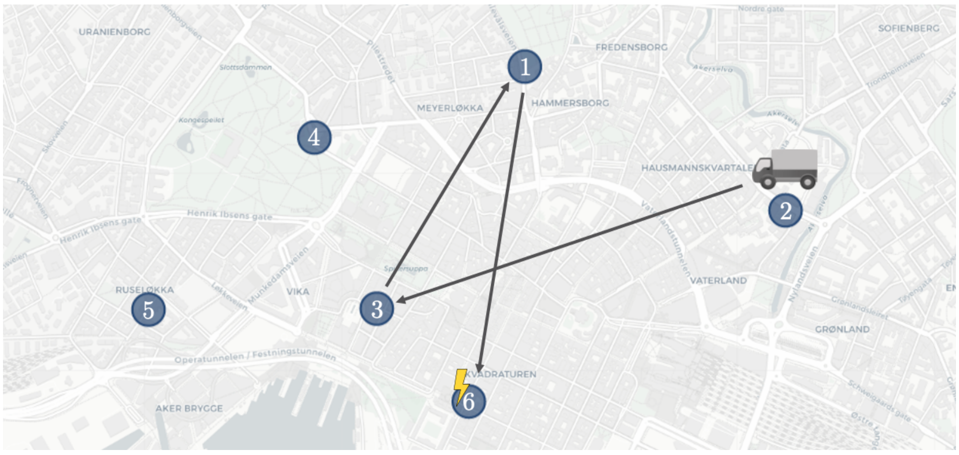


Figure 7.6: Example of a sequence of station visits performed by a service vehicle, based on the test instance in Table 7.2.

Chapter 8

Computational Study: Parameter Tuning

In this chapter, parameters in the solution method presented in Chapter 5 are tuned based on the performance of the decisions obtained from a sequence of individual DSBRCP in a simulator. First, we elaborate on the evaluation metrics used in our analyses in Section 8.1. Then, the weights in the objective function of the subproblem are set in Section 8.2. Further, the criticality score weights used in the route generation are analyzed in Section 8.3. Section 8.4 investigates the effect of increasing problem size on the solution time. This is followed by a presentation of stability tests on the first stage solutions in Section 8.5. Lastly, Section 8.6 summarizes the parameters to be used in further analyses.

8.1 Evaluation Metrics

In this section, we discuss how the simulation results are evaluated. First, we present the metric *prevented violations*, then how different system configurations can be compared through a statistical T-test.

8.1.1 Number of Prevented Violations

We recall from Section 5.2 that a violation represents an unsatisfied request incurred by a customer arrival. The overall objective of the BSS-operations is to minimize such occurrences, referred to as the *total violations*. When evaluating different system configurations, the simulator is run from 7 am to 11 pm, representing a day of operations in the BSS in Oslo. Every run simulates a possible sequence of customer arrivals. To increase accuracy and reduce variance of the evaluation, each configuration is tested on ten different days. The average total violations is calculated using Equation (8.1), where $\mathcal{C}$ is the set of generated days. This value can be further broken down into average number of starvation violations and average number of congestion violations.

$$\text{Average number of total violations} = \frac{\sum_{c \in \mathcal{C}} \text{Total violations in } c}{|\mathcal{C}|} \tag{8.1}$$

On the same day, the average total violations is recorded both with vehicles operating, and with no vehicles in operation, denoted *natural violations*. The evaluation metric used is the difference between these two, the average number of *prevented violations*.

8.1.2 Statistical T-test

A statistical T-test allows us to compare average values of two data sets and determine if they came from the same population. Mathematically, the t-test takes a sample from each set and establishes the problem statement by assuming a null hypothesis that the two means are equal Malcata (2020). In our case, the null hypothesis states that two configurations perform equally well, while the alternative hypothesis states that one configuration outperforms the other.

In our analyses, we want to make a pairwise comparison between two configurations. This is done by calculating the mean difference in violations $\overline{z}$ over multiple scenarios before estimating the probability p that this difference is equal to zero. In other words, estimating the likelihood that the two configurations perform equally well. This is done with Equation (8.2), where s_D is the standard deviation of the differences and n is the size of the population. The denominator represents the *standard error* of the population.

$$p = P\left(|Z| > \frac{\overline{z} - 0}{s_D/\sqrt{n}}\right) \tag{8.2}$$

Ordinarily, if $p < 0.05$ then the mean is unlikely to be zero whereas $p > 0.05$ provides no such evidence (Teetor, 2011). Based on this, the null hypothesis is accepted or rejected accordingly. Table 8.1 shows an example where the performance of configuration X is compared to configuration Y. We want to test whether there are reasons to believe that one of the configurations performs better than the other. The values of $\overline{z}$ and $s_D/\sqrt{n}$ are inserted into Equation (8.2). The received value is used to look up in the standard distribution table, giving $p = 0.006$. This result implies a 0.6% chance of the two configurations performing equally well. Thus, we reject the null hypothesis and conclude that Y performs better than X.

Table 8.1: T-test calculation example comparing configuration X and Y on 5 different scenarios

	Total violations		
Scenario s	X_s	Y_s	$Z_s = X_s - Y_s$
1	20	16	4
2	22	18	4
3	19	14	5
4	18	12	6
5	23	17	6
		$\overline{z}$	5
		$s_D/\sqrt{n}$	0.398

8.2 Weights in the Subproblem Objective

In the subproblem, there are five weights emphasizing different aspects in the objective function: W^V, W^D, W^R, W^N, and W^L. W^V denote the weight on violations, and the weight on deviation is represented by W^D. Further, W^R is the weight of reward obtained by delivering flat bikes to a charging station. W^N weight the effect of actions performed at the origin station, and W^L weight the effect of charging and rebalancing decisions performed at later station visits. We require that these two weights sum to one. This requirement does also apply to the total of the weights W^V, W^R, and W^D.

Allowing values between zero and one with a step size of 0.1, there are 726 possible distinct weight sets consisting of the five above presented weights. After conducting a preliminary analysis, a subset with the most promising weight values were further explored to choose the best weight configuration. Table 8.2 shows an overview of the values tested for each weight parameter, resulting in 108 possible combinations.

Table 8.2: Weight values used to generate weight sets for testing. For each component, the grey-colored weight values are tested, while the weights on white background are not.

Notation	**Associated component**	**Tested values**										
W^V	Prevented violations	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
W^D	Prevented deviations	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
W^R	Reward	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
W^N	First-stage decisions	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
W^L	Second-stage decisions	0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1

The weight sets were then evaluated using simulation to determine the best weight set configuration to use in further analyses. The average number of prevented violations obtained in the system was recorded over ten different days per weight set configuration. Table 8.3 presents the average, minimum and maximum by values of of W^N and W^L. The number of violations reflects the average across all possible combinations of W^V, W^D and W^R.

Table 8.3: Comparison of average number of prevented violations filtered on the value of W^N with corresponding value of W^L. The number of violations reflects the average across all possible combinations of W^V, W^D and W^R.

$|\mathcal{S}| = 200$, $|\mathcal{V}| = 1$, $|\mathcal{C}| = 10$, $|\mathcal{F}| = 10$, $B = 5$

W^N	W^L	**Average prevented no. of violations**	**Max**	**Min**
0.5	0.5	526	1 163	43
0.6	0.4	645	1 465	102
0.7	0.3	770	1 791	124
0.8	0.2	816	1 337	266
0.9	0.1	800	1 553	259
1	0	601	1 141	119

We observe that choosing $W^N = 0.8$ and $W^L = 0.2$ produce the best performing model with the highest number of average prevented number of violations. With these two weights fixed we moved on to tune the last three weights in the subproblem objective function. The average number of prevented violations with varying values is shown in Table 8.4. Note that the value of W^R is given by $1 - W^D - W^V$.
Based on the values seen in this table, we conclude that it is reasonable to set $W^V = 0.6$, $W^D = 0.3$, and $W^R = 0.1$. However, we also note that the variance in the table is low, implying low sensitivity of altering these weights within the investigated interval.

8.3 Weights in the Criticality Score for Route Generation

The criticality score for route generation is detailed in 5.5.2. The score has four components with an associated weight parameter. To decide the weight parameter for each component, we use the same procedure as when tuning the subproblem weights in Section 8.2. The system parameters are also set equally as in the previous section. The best performing weight combination, evaluated across ten days of simulation with five operating service vehicles, is displayed in Table 8.5. The four weights shown was tested with values in the interval between zero and one with a step size of 0.1. Their values was also required

Table 8.4: Comparison of average number of prevented violations with different weight values

$|\mathcal{S}| = 200$, $|\mathcal{V}| = 1$, $|\mathcal{C}| = 10$, $|\mathcal{F}| = 10$, $B = 5$
$W^N = 0.8$, $W^L = 0.2$

	W^D			
W^V	0.1	0.2	**0.3**	0.4
0.5	847	719	773	804
0.6	846	785	871	799
0.7	802	791	839	
0.8	743	845		
0.9	821			

to sum to one, resulting in a total of 286 tested weight combinations. The displayed result achieved the highest average number of prevented violations.

Table 8.5: Final configuration of weights in the criticality score for route generation

Notation	**Associated component**	**Value**
W^T	Time to violation	0.5
W^S	Driving time	0.2
W^D	Deviation at $\overline{T}$	0.1
W^N	Net demand	0.2

When evaluating the weight sets, one vehicle was utilized with a branching factor of one, such that only one route was generated when solving the problem. In theory, it could be possible to create optimal routes solely based on the criticality score. Thus, we set the weights to the values that generally built the best routes for a vehicle in the system.

Value of applying criticality score and filtering

To assess the value of applying criticality scores and filtering, the average prevented number of violations over ten days was recorded both applying and not applying criticality score. When not applying criticality score, the seven closest stations in terms of driving time were added to the route when branching, similar to the policy used in the alternative greedy strategy. Table 8.6 shows the results.

Table 8.6: Average prevented number of violations over 10 days applying and not applying criticality score and filtering

	Average prevented no. of violations	
	Criticality score and filtering	
Day	TRUE	FALSE
1	4 964	4 777
2	5 038	4 565
3	4 437	4 469
4	4 698	4 401
5	4 516	4 765
6	4 684	4 727
7	4 767	4 340
8	4 760	4 960
9	4 442	4 553
10	4 803	4 414
Average	**4 711**	**4 597**

A conducted T-test using Excel indicates a 22% chance of better performance with criticality score and filtering applied. As this is higher than the defined threshold of 5%, we cannot conclude that applying criticality score and filtering generally performs better, i. e. the value of applying criticality scores and filtering is low. This must however be seen in light of the high system demand, which makes more stations in need of s service vehicle visit. When applying the chosen branching factor not using criticality score, it is thus likely that one of the selected stations added during route generation also obtains good results, especially as driving time also affects the criticality score.

8.4 Size of Solvable Instances

Recall that the DSBRCP is solved every time a service vehicle arrives at a new station. For effective utilization of operational resources, rapid decision support is required. Hence, we define *reasonable time* to solve the DSBRCP as less than 60 seconds, the estimated average time it takes for a service vehicle to park.

To get an initial idea of the limit in size of solvable instances, the model was run with three varying subproblem parameters; number of stations, number of vehicles and number of customer arrival scenarios. Further, the computational time sensitivity for each of the parameters were examined. These analyses are helpful to explore the applicability of the model to real-world operations, by addressing the maximum BSS size to which the model can assist. Each configuration was run ten times. The average results for each configuration is presented in Table 8.7. The grey colored cells mark configurations where the average computational time was higher than the defined reasonable time. The branching factor was fixed to five for the first branching step, and three for the second branching step. The time horizon used as planning period in the subproblems is discussed by Fosen and Haldorsen (2019) and set to 25 minutes. In their work, this value proved to demonstrate the best balance between having a long planning period and maintaining a high marginal activity level.

From Table 8.7, we observe a clear trend of increased computational times by increased number of vehicles, holding all other parameters constant. The same trend is seen for the number of scenarios. Notably, the table also indicate low impact on the computational time when increasing the number of stations. This can be explained by the fact that the full set of stations in the BSS is only considered in the master problem of the heuristic, which is only solved once. In some cases, the computational time even decreases slightly when increasing the number of stations due to natural variance in the initialization. Further, the table shows that most grey cells, i.e. configurations with a higher average computational time than the reasonable time, occur when the number of scenarios and vehicles are close to the maximum tested value.

From the findings elaborated above, we construct a hypothesis that the computational time of the heuristic is heavily dependent on the number of created columns, i.e. the number of subproblems that is solved in the heuristic. Recall from Section 5.6 that a subproblem is solved for a specific vehicle, scenario, route, and pattern. Counting the total number of subproblems solved by the heuristic does however impose a challenge, as the cardinality of $\mathcal{P}_v$, the number of patterns created for service vehicle v, may vary depending on the vehicle and start station inventories. Additionally, the cardinality of $\mathcal{R}_v$, the number of routes created for service vehicle v, depends on the number of stations that can be added to a route before reaching a total driving time larger than the planning horizon of 25 minutes, i. e. the number of branching steps.

An upper bound for the number of subproblems solved by the heuristic may however be estimated by Equation (8.3). $\mid \mathcal{V} \mid$ and $\mid \mathcal{F} \mid$ are simply the number of vehicles and the number of subproblem scenarios that the heuristic is solved for. The constant B_1 represents the branching factor at the first branching point, while B_2 denotes the branching factor at the second branching point. Later branching steps apply a branching factor equal to one. Further, P is the number of extreme patterns in a route. Recall from Subsection 5.5.4 that the upper bound for this value when solving the DSBRCP is 45 patterns. When testing the

Table 8.7: Overview of average DSBRCP computational time in seconds for varying number of stations, number of vehicles, and number of subproblem scenarios. Combinations with average computational time higher than the defined reasonable time are marked in grey. The branching factor was fixed to B_1 = 7, B_2 = 3 and $B_n = 1$ for all $n \geq 3$.

		Vehicles, $\|\mathcal{V}\|$				
Scenarios, $\|\mathcal{F}\|$	**Stations, $\|\mathcal{S}\|$**	1	2	3	4	5
1	20	0.9	1.1	2.0	2.2	2.5
	50	0.8	1.1	2.0	2.6	2.7
	100	0.5	1.0	2.0	2.2	2.6
	150	0.4	0.7	1.8	2.1	2.5
	200	0.4	0.7	1.8	2.3	2.5
10	20	8.2	10.7	19.3	21.9	24.1
	50	9.4	11.5	21.7	24.4	28.1
	100	4.2	8.7	17.9	20.8	25.7
	150	5.0	6.3	17.7	21.2	23.6
	200	3.3	6.0	16.5	19.1	23.4
20	20	16.4	21.3	38.1	43.4	48.4
	50	20.6	23.9	42.9	48.5	55.7
	100	7.6	15.6	36.2	42.0	50.2
	150	11.0	15.4	34.5	42.0	48.3
	200	7.5	11.4	31.4	37.5	45.8
30	20	24.8	32.9	57.5	66.1	73.7
	50	32.2	37.6	63.7	74.6	83.0
	100	11.6	21.7	54.0	65.6	77.0
	150	16.9	23.8	51.8	64.3	72.6
	200	10.4	17.8	48.1	57.5	70.2

solution time, we made sure that all the instances reached this number of patterns through a careful choice of initial values. The presented values in Table 8.7 thus represent an upper bound for the computational time of the problem. Note that the number of stations in the instance, $\mid \mathcal{S} \mid$ is not included in the equation.

$$\mid \mathcal{V} \mid \times \mid \mathcal{F} \mid \times B_1 \times B_2 \times P \tag{8.3}$$

Equation (8.3) is used to estimate the number of subproblems solved by the heuristic for all instance configurations listed in 8.7. Figure 8.1 shows a plot of the computational time of each configuration by the estimated number of subproblems. From the plot, we observe a clear trend in increased computational time with higher estimated number of subproblems. Further, there seems to be a greater variance among the computational times for higher estimated numbers of subproblems. This may be due to differences in how the computational time of the master problem is affected by changes in $\mid \mathcal{V} \mid$ and $\mid \mathcal{F} \mid$.

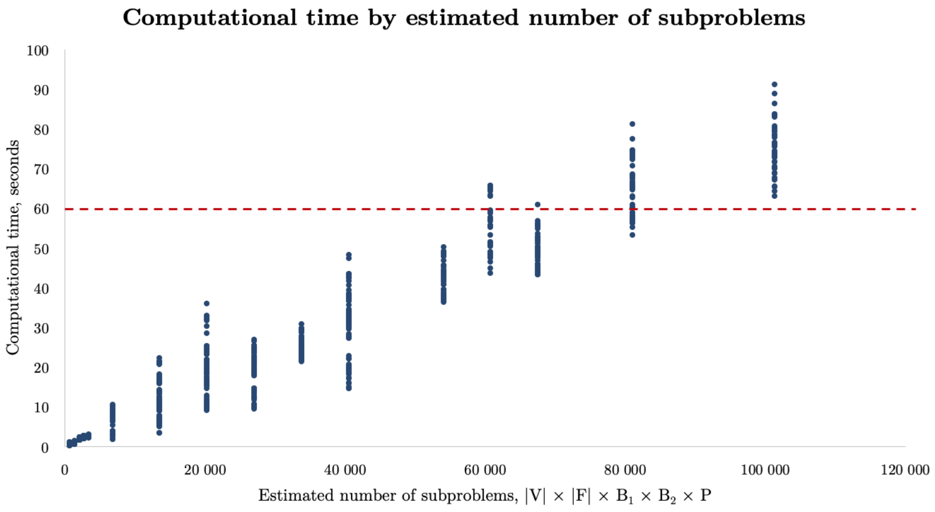


Figure 8.1: Computational time by estimated number of subproblems solved for the test instances presented in Table 8.7. The red line indicates the defined reasonable time.

Regarding computational time, it is important to note that the subproblems are independent of each other. Consequently, computations of the subproblems may be parallellized. Parallellizing of the subproblems can improve the computational time of the heuristic significantly, without compromising the solution quality. Even though some of the computational times in Table 8.7 exceed the defined reasonable time, we may therefore conclude that the DSBRCP heuristic has potential to assist in even larger sized BSS environments in terms of computational time than tested.

8.5 Stability of First Stage Decision

The results in Section 8.4 show that the computational time of the heuristic is drastically affected by the estimated number of subproblems it solves. This fact creates an incentive to examine how the solution changes when varying the parameters that are included in Equation (8.3). This section presents stability analyses of the first stage decision with respect to variance in the number of subproblem scenarios, $\mid \mathcal{F} \mid$ and branching factor, B.

8.5.1 Stability Across Number of Subproblem Scenarios, $\mid \mathcal{F} \mid$

As stated in the discussion of stochastic two-stage formulations in Section 5.3.1, the main benefit of a two-stage formulation is to find first stage decisions that value flexibility. In other words, we want to find smart first stage decisions that perform well, regardless of the outcome of the stochastic information. The flexibility lies in how the second stage decisions can adapt to the stochastic information that is revealed at a later stage. A hypothesis that further support stochastic formulations is that deterministic models are unable to discover the same first stage decisions.

To measure the value of a stochastic model as compared to a deterministic formulation, we performed a stability test of first stage decisions when varying the number of scenarios used in the subproblem. For each tested number of scenarios, we analyzed how the first stage decisions corresponded to a first stage decision from the heuristic using the maximum tested value of scenarios. All other parameters were kept constant. To obtain first stage decisions from different system states, all heuristic decisions during a full day BSS simulation were recorded. This amounted to 73 decision points. For each decision point, the heuristic solution was found with 1, 10, 20, 30, 40 and 50 subproblem scenarios. The results are presented in Table 8.8. Each row presents the average similarity in first stage solutions when compared to the solutions of the heuristic with 50 subproblem scenarios. Note that an underlying assumption for this analysis is that using 50 subproblem scenarios will give the most flexible, and hence better, first stage decision. This fact is in general supported by two-stage stochastic programming theory, but might not be true for our particular data set.

In the analysis, we distinguish between similarity in the next station decision and the quantity decision, which includes rebalancing and battery swap decisions. All quantity decisions must be equal to achieve quantity similarity. If a first stage decision obtains both next station similarity and quantity similarity to the decision using 50 subproblems, it is defined as full similarity, presented in the rightmost column of Table 8.8.

Contrary to our initial hypothesis that flexible solutions will only be found using many sub-

Table 8.8: Average first stage similarity compared to decisions made based on 50 subproblem scenarios.

$|\mathcal{S}| = 200$, $|\mathcal{V}| = 1$, $|\mathcal{F}| = x$, $B = 5$

$W^N = 0.8$, $W^L = 0.2$, $W^V = 0.6$, $W^D = 0.3$, $W^R = 0.1$

Scenarios	Station similarity	Quantity simliarity	Full similarity
1	47 %	75 %	37 %
10	62 %	73 %	48 %
20	52 %	69 %	38 %
30	56 %	70 %	43 %
40	62 %	73 %	45 %

problem scenarios, Table 8.8 shows that inclusion of many subproblem scenarios seems to have little effect. More specifically, the inclusion of more subproblems do not lead to first stage decisions that are more similar to the first stage decisions performed with 50 subproblem scenarios. Also, we observe that full similarity applies to less than 50% of the decision points for all presented number of scenarios. This fact makes it difficult to conclude that an increased number of subproblem scenarios impact the solution quality. However, we see that a configuration with only one subproblem scenario obtains the lowest score of full similarity among the tested configurations. This might indicate that it is favorable to use ten subproblem scenarios, even though the effect of using more than this appears to be small. Note that similarity is not necessarily a measure of quality. The solution qualities with varying number of subproblem scenarios are examined later in Section 9.1.

The remarkable low differences in similarity with varying number of scenarios, could possibly be due to a general low variance in the chosen solution given the distributions that the subproblem scenarios are generated from, i. e. the predicted customer arrivals from UIP. For instance, the outgoing demand at a station could be so high that a service vehicle chooses to maximize the number of unloaded charged bikes almost regardless of the outcome of the sequence of customer arrivals. However, this was not further explored as we assume the real-world data from UIP to be applicable. As the results indicate that the quality of the heuristic will not be drastically affected by the number of scenarios used in the subproblem, we fix the number of scenarios to ten. With reference to Equation (8.3), this makes room to explore wider ranges in the number of vehicles or number of routes generated per vehicle by the branching algorithm.

8.5.2 Stability Across Branching Factor, B

The branching constant B indicates how many new routes are created in each branching step of the algorithm. Each path from the root node to a leaf node in the branching tree represents a distinct route. Thus, a higher B generates more routes for each service vehicle, implying more subproblems to solve and consequently more columns in the master problem. A branching factor equal to the number of stations in the system explores all possible routes, guaranteeing that the heuristic chooses the optimal next station. However, we recall from Section 8.4 that a large number of routes increases the computational time drastically. This makes it necessary to reduce the search space. When combined with criticality scores, the branching factor is intended to shrink the search space until the point where high quality routes are always included in the solution sets. Hence, it is of interest to evaluate the first stage decisions for different values of B. An important assumption for the analysis is that the first stage decision made with the highest test value of B, indicate the best choice of the next station to visit at each decision point.

Since only the first added station to a route under construction is used to give operational guidance, we have investigated the use of a *dynamic branching factor*. With a dynamic branching factor, B is gradually decreased for every branching step, i.e. every time a route extension is conducted. The effect of this is increased exploration of the station space early in the routes than later on. For a first stage decision analysis, it is appropriate to focus on testing different values of B at the root node, denoted B_1. Thus, we fix B = 3 at the second branching step, and B = 1 at later branching steps.

Table 8.9: First stage similarity compared to decisions taken with branching factor 9 at the first branching step

$|\mathcal{S}| = 200$, $|\mathcal{V}| = 1$, $|\mathcal{F}| = 10$, $B = x$

$W^N = 0.8$, $W^L = 0.2$, $W^V = 0.6$, $W^D = 0.3$, $W^R = 0.1$

B_1	**Station similarity**	**Quantity simliarity**	**Full similarity**
1	7 %	9 %	7 %
3	39 %	92 %	39 %
5	76 %	99 %	76 %
7	92 %	99 %	92 %

A similar analysis as the previous presented scenario stability analysis is conducted for varying values of B_1. The results are shown in Table 8.9. At each decision point during a full day of simulation, the first stage solution was found with B_1 = 1, 3, 5, 7 and 9 at the root node. As 9 was the maximum tested value, the first stage decisions made with root node branching factor 9 acts as reference of the ideal choice. All other parameters were kept constant, with values indicated in the top bar of Table 8.9. Each row presents the average similarity of first stage decisions made with branching factor B_1 in the root node as compared to using the heuristic with branching factor 9 in the root node.

We notice from the second column of Table 8.9 that the station similarity significantly increases with a larger branching factor. This implies that higher values of B_1 increase the chance of identifying the preferable next station for service vehicles. From this, we can also conclude that the criticality score function is not optimal. A perfect criticality score would give 100% next station similarity for all values of B_1, as the aim of the criticality score is to always choose the most advantageous next station, i.e. to generate the optimal route first. It is also interesting to observe that the quantity similarity is close to 100% for all values of B_1 except $B_1 = 1$, which means that the branching factor has less impact on the quantity decisions. Considering this and the discussed budget on the number of subproblems solved by the heuristic, a reasonable choice is to fix B_1 to 7 in further analyses.

8.6 Final Configuration of Parameters

The final configuration of parameters used in the simulator is shown in Table 8.10. These values are used in further analyses in Chapter 9, unless otherwise specified. We refer to this configuration as the *primary test configuration*.

Table 8.10: Primary test configuration of weights in the criticality score for route generation.

Parameter	Symbol	Value
Weights in subproblem objective		
Current station operations	W^N	0.8
Later station operations	W^L	0.2
Violation	W^V	0.6
Deviation	W^D	0.3
Reward	W^R	0.1
Weights in criticality score		
Time to violation	W^T	0.5
Driving time	W^S	0.2
Deviation at $\overline{T}$	W^D	0.1
Net demand	W^N	0.2
System parameters		
Number of stations	$\lvert\mathcal{S}\rvert$	200
Initial branching factor	B_1	7
Subproblem customer arrival scenarios	$\lvert\mathcal{F}\rvert$	10
Simulation start time	t^{start}	7 am
Simulation end time	t^{end}	11 pm
Number of vehicles	$\lvert\mathcal{V}\rvert$	5

Chapter 9

Heuristic Performance and Managerial Insights

In this chapter, the simulation framework presented in Chapter 6 is used to analyze the quality and value of the service vehicle decisions constructed from the heuristic solutions. Unless otherwise specified, the analyses are based on an average of 10 days of simulation using the primary test configuration specified at the end of the previous chapter in Section 8.6. In Section 9.1, we compare the heuristic strategy with alternative strategies to address the DSBRCP. Further, managerial insights are presented in Section 9.2. The analyses use the same evaluation metrics as presented in Section 8.1.

9.1 Performance Comparison with Alternative Strategies

In this section we look at the performance of the DSBRCP heuristic. First, we compare it the performance of alternative strategies. Then, we see how well it performs when using a different number of subproblem scenarios and when employed in a system with lower demand.

9.1.1 UIP-inspired Strategy and the DSBRCP Heuristic Strategy

Rebalancing and charging of a BSS is important to meet customer demand. However, service vehicle operations are costly as it requires operators and service vehicles. Thus, it is important to utilize these resources in the best possible way. To examine the performance of the two-stage stochastic programming column generation heuristic presented in this thesis, we have conducted simulations using three alternative service vehicle strategies at the same day of simulation. A *day* is defined as a specified sequence of requested customer trips within the time interval 7 am to 11 pm generated based on the customer arrival intensity rate parameters. The three alternative strategies are listed below.

1. *No operations*. No service vehicles are in operation. The system evolves naturally through customer trips between charging stations and non-charging stations.
2. *UIP-inspired strategy*. Service vehicles aim to maximize the activity level of battery swaps and bicycle rebalancing. The station with the highest criticality score after filtering is chosen as the next station. This strategy is elaborated in Appendix B.
3. *DSBRCP heuristic*. Service vehicles aim to make efficient long-term decisions that maximizes the customer utility in the BSS. This strategy is elaborated in Chapter 5. The primary test configuration parameters from Section 8.6 are used unless otherwise is specified.

Table 9.1 shows a comparison of the number of violations occurring in a simulation from 7 am to 11 pm with five vehicles on ten different days. The ten days have an average of 33 961 requested trips. We observe from the table that the heuristic achieves the lowest average of total violations, with an average total improvement relative to a system with no operations of 23.5%, whereas the UIP-inspired strategy achieves an average total improvement of 15.5%. It is noteworthy that the number of violations with no operations is about 62% of the total requests, revealing that the simulated BSS is unable to accommodate most of the customer demands. Even though the heuristic strategy greatly improves this metric, the system still struggles to maintain high customer satisfaction. This is knowledge already recognized by UIP.

Table 9.1: Comparison of results from simulation with different service vehicle strategies

Primary test configuration, **Average requested trips** = 33 961

Strategy	**Average total starvations**	**Average total congestions**	**Average total violations**	**Average total improvement**
No operations	16 425	4 637	21 053	-
UIP inspired	14 212	3 577	17 789	15.5 %
Heuristic	13 502	2 641	16 143	23.3 %

The second and third columns of Table 9.1 display the average number of violations of each type. Remark that the number of starvations generally is significantly higher than the number of congestions. In continuation of the previous discovery that most customer requests are not met, these numbers show that starvations are the main source of violations. In other words, many customers desire to start a trip from stations where there are no bikes with sufficient charge available.

It is also of interest to examine the daily variance in performance across strategies. Each day is based on different sequences of trips leading to variance in the number of occurred violations. Hence, the decisions made using the same strategy vary between days. To highlight the performance differences, the metric used in further presentation is the number of *prevented* violations. Following from this definition, the strategy of no operation always achieves zero prevented violations, and is therefore omitted from the analyses. Figure 9.1 illustrates the number of prevented violations by day. Observe that the DSBRCP heuristic performs better than the UIP-inspired strategy on all days, but the difference in prevented violations varies. Performing a T-test on the two strategies reveals that there is a $\sim 0\%$ chance that the greedy strategy generally performs better than the heuristic. We can thus conclude that the heuristic is generally better than the UIP-inspired strategy. The outcome of the T-test is also reflected in the average value of prevented violations for each strategy. The heuristic prevents 4,910 violations on average, while the UIP-inspired strategy prevents 3,264 violations on average. This implies that by adhering to the DSBRCP heuristic strategy, 50% more violations are on average prevented in a day. In conclusion, we confirm that the proposed heuristic performs well in the described simulation environment.

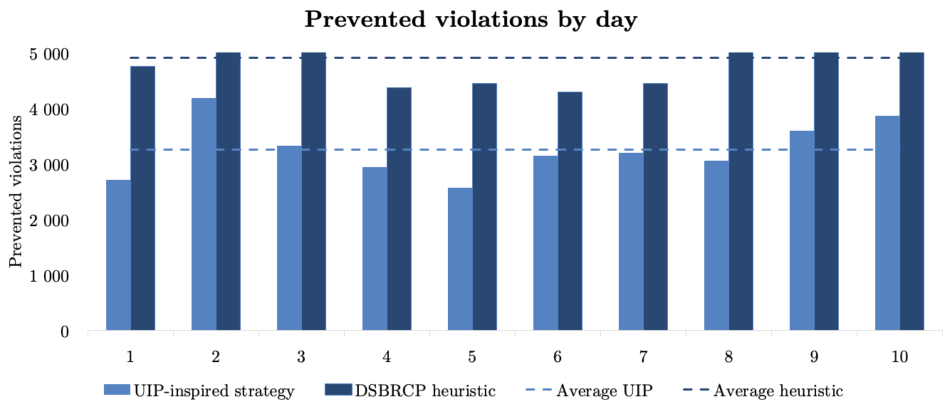


Figure 9.1: Prevented violations by the UIP-inspired strategy and the DSBRCP heuristic strategies per day

9.1.2 Number of Subproblem Scenarios in the DSBRCP Heuristic

Further, we find it valuable to examine the robustness of the primary test configuration defined in 8.6. Recall from Section 8.5 that the number of subproblem scenarios, $| \mathcal{F} |$, and the branching factor at the root node B_1, were determined by stability analyses. A key assumption made in these analyses, was that higher values of the tested parameters would retrieve better solutions. With this assumption, the determination of the number of subproblem scenarios proved to be challenging, as many of the tested parameters yielded similar results. To confirm or disprove whether it was reasonable to choose ten subproblem scenarios in the primary test configuration, a comparison over ten days of simulations was conducted with two separate heuristic configurations; one using one subproblem scenario and one using ten subproblem scenarios. Note that a choice of more than ten subproblem cannot be combined with the other chosen primary configuration parameters while still maintaining a computational time below the defined reasonable time.

A comparison of the number of prevented violations at ten separate days for the primary test configuration and a configuration using one subproblem scenario is shown in Figure 9.2. The figure reveals small differences between the number of prevented violations for the two configurations. This confirms the hypothesis that a stochastic formulation of the subproblem seems to have little effect with the current input data. Nonetheless, using ten subproblem scenarios produce the best results on average, preventing an additional 5.4% violations compared to using only one scenario. However, a conducted T-test between the configurations showed a 7.7% probability that using one subproblem scenario outperforms the primary test configuration. Hence, we cannot conclude that it is better to use ten subproblem scenarios as opposed to only one in general.

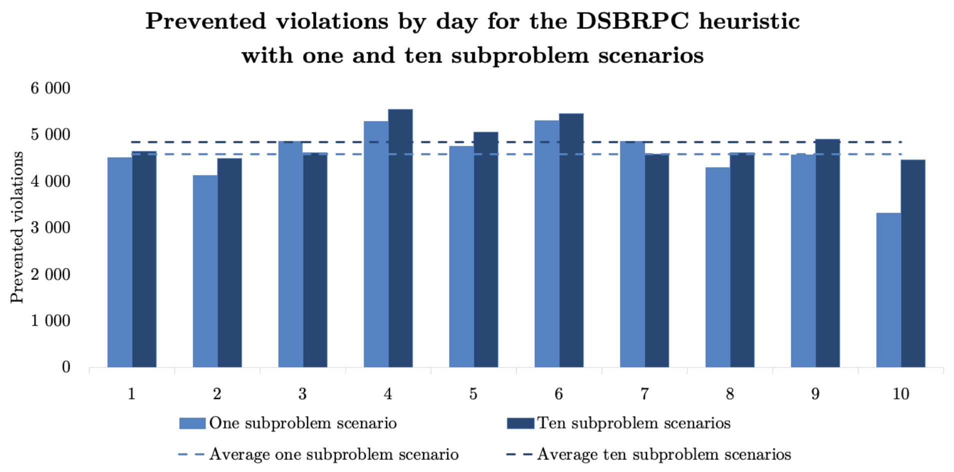


Figure 9.2: Daily prevented violations for the DSBRCP heuristic with one and ten subproblem scenarios

9.1.3 Reduced Numbers of Customer Arrivals

The predicted number of customer arrivals estimated by UIP reveals that the system is under high pressure and unable to satisfy most customers. Due to this, it is of interest to also evaluate how the heuristic performs with a lower number of customer arrivals. In order to reduce demand in the system, UIP's original predictions of customer arrivals were multiplied by a factor α in the interval from 0.3 to 1. For instance, an α-value of 0.4 corresponds to 40% of UIP's original predictions of customer arrivals. Similarly, calculation of customer arrivals with $\alpha = 1$ is equal to using UIP's original predictions.

First, we conducted a preliminary analysis to evaluate the natural violations in a system with lower demand. The results are shown in Table 9.2. The rightmost column displays the success rate of requested trips, i.e. the percentage of total requests to use a charged bike that did *not* result in a starvation. Surprisingly, the system achieves request success rates below 70% for all tested values of α, implying that less than seven out of ten desired initiations of trips are met. This even applies when the customer request level is as low as 30% of the original predictions. There are two possible explanations to this: either the station capacities or number of charged bikes are insufficient, or the system is not balanced well initially to accommodate to the requests.

Table 9.2: Outgoing charged customer arrival data for different values of α. All numbers are average numbers based on ten days of simulation

α	**Total requests**	**Natural starvations**	**Initiated trips**	**% request success**
0.3	10 179	3 644	6 535	64 %
0.4	13 557	5 218	8 339	62 %
0.5	17 036	7 011	10 025	59 %
0.6	20 297	8 783	11 514	57 %
0.7	23 733	10 480	13 253	56 %
0.8	27 016	12 395	14 621	54 %
0.9	30 488	14 411	16 077	53 %
1.0	33 961	16 425	17 536	52 %

Further, the effect of employing a fleet of five service vehicles is examined for $\alpha = 0.3$ and $\alpha = 1$. The effect is measured by the percentage of natural violations that are prevented through service vehicle operations. Figure 9.3 shows that the service vehicles operations prevent a larger fraction of the natural violations when the demand is lower. This result is expected as the absolute value of the number of violations is smaller. However, Figure 9.3 also reveals two interesting tendencies. First, it shows that the percentage of prevented natural violations is nearly the same in the morning hours regardless of the level of customer demand. This fact supports high importance of a well-balanced BSS at the beginning of the day. Second, the continued high rate of prevented natural violations for the line representing α = 0.3 from 12 am onward is notable. In a strained BSS, the importance of what station to visit next is reduced as there are many stations that would benefit from a service vehicle visit. The importance of this increases drastically in a system with less customer demand, as there are larger variations in the benefit of visiting different stations. Following from this, Figure 9.3 proves an important property of the heuristic, namely that the proposed heuristic also pertain high quality operational decisions in a system with lower customer demands.

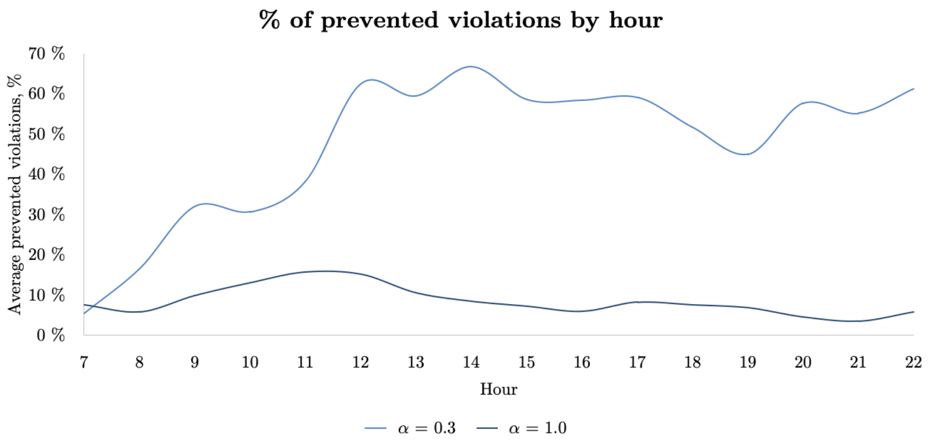


Figure 9.3: Percentage of prevented violations by hour for α = 0.3 and α = 1.0

9.2 Managerial Insights

In this section we conduct analyses that provide managerial insights. First, we explore the value of service vehicles and employing a heterogeneous fleet. Further, we look at the value of geofencing. Lastly, the effect of charging stations is evaluated.

9.2.1 The Value of Service Vehicles

Section 9.1 showed that the proposed DSBRCP heuristic works well as an operational aid to make clever rebalancing and charging decisions in a BSS with five vehicles. Naturally, the number of prevented violations will increase as a function of employed service vehicles. However, both high investment costs and high operational costs are important factors to consider when deciding on the number of service vehicles in a BSS. Thus, it is of interest to evaluate the effect of increasing the number of service vehicles on the number of prevented violations.

Size of the service vehicle fleet

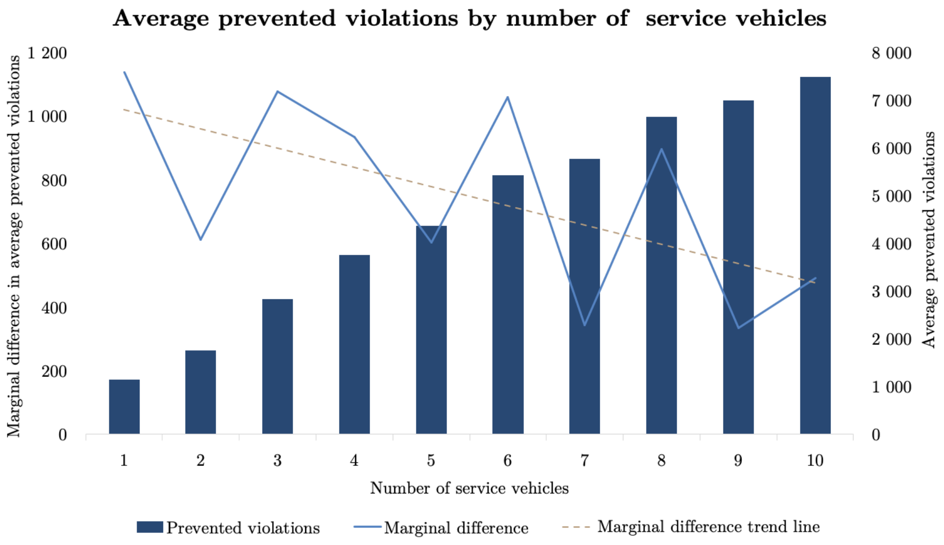


Figure 9.4: Prevented violations by number of vehicles, and the marginal difference of a vehicle

The bars in Figure 9.4 show the average number of prevented violations per day by the number of operative service vehicles. The height of the bars correspond to the values on the secondary axis. It is not surprising that the number of prevented violations increases as the number of employed service vehicles increases. The absolute value of the increase is however not constant, implying that the value of an additional vehicle varies depending on the existing fleet employed. The latter is captured by the marginal differences in prevented violations when adding a vehicle, represented by the plotted light blue colored line linked to values on the primary axis. Note that the marginal effect of an added vehicle varies, but displays a decreasing trend indicated by the dashed line. An additional service vehicle investment or employee should be evaluated in light of this marginal effect.

Varying number of employed service vehicles throughout the day

Figure 9.5 shows the average hourly natural violations occurring for the primary test configuration. The diagram reveals two violation peaks during a day; early morning and late afternoon. These time intervals correspond well with the typical rush hours in Oslo when people commute to work. These observed spikes in the number of hourly violations provide reason to investigate the effect of having a varying number of employed service vehicles at different hours during a day.

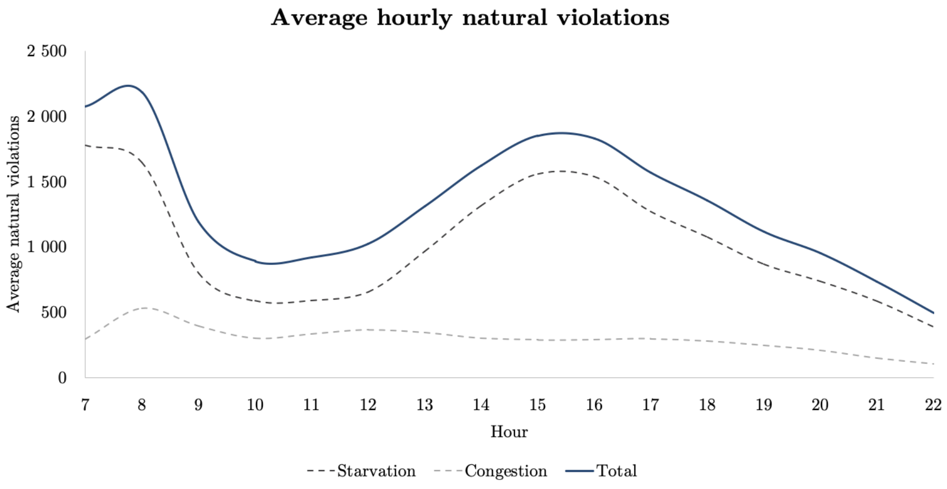


Figure 9.5: Average hourly naturally occurring violations with no service vehicles in operation

As a first step towards discovering the value of varying the employed number of service vehicles during the day, we analyze the average number of prevented violations per hour of a day using different fleet sizes. The results are plotted in Figure 9.6. An important insight from Figure 9.6 is how the vertical distance between the lines fluctuates throughout the day. These can be used to categorize time intervals with similar characteristics. Recall that the vertical distance between the line representing x vehicles and the line representing $x + 1$ vehicles corresponds to the marginal effect of adding an extra vehicle to the already employed x vehicles. We recognize that the plotted lines have little vertical spread during the first hours of operation. At 10 am, the vertical spread starts to increase, before the maximum vertical distance between the top and bottom line is obtained between 1 pm and 2 pm. At 4 pm, the average marginal effect of an employed vehicle is again reduced. These tendencies remain relatively constant for the rest of the day, before we see another decrease during the two last hours of simulation.

The above findings are used to divide the day into time intervals with similar characteristics, indicated by vertical, dashed lines in Figure 9.6. For each interval, the marginal effect of additional service vehicles were analyzed for a varying fleet size of three to seven. The results are displayed in Figure 9.7. The morning interval from 7 am to 9 am is characterized by a high number of natural violations. Recall that the initial state when entering the morning rush hours highly impacts the number of occurring violations. When starting in a poorly rebalanced initial state, a larger service vehicle fleet might only be capable of preventing a few additional violations. Combined with the violations stemming from the spike in demand, an assumed imbalanced initial state can explain the displayed low marginal effects of additional vehicles in the morning interval.

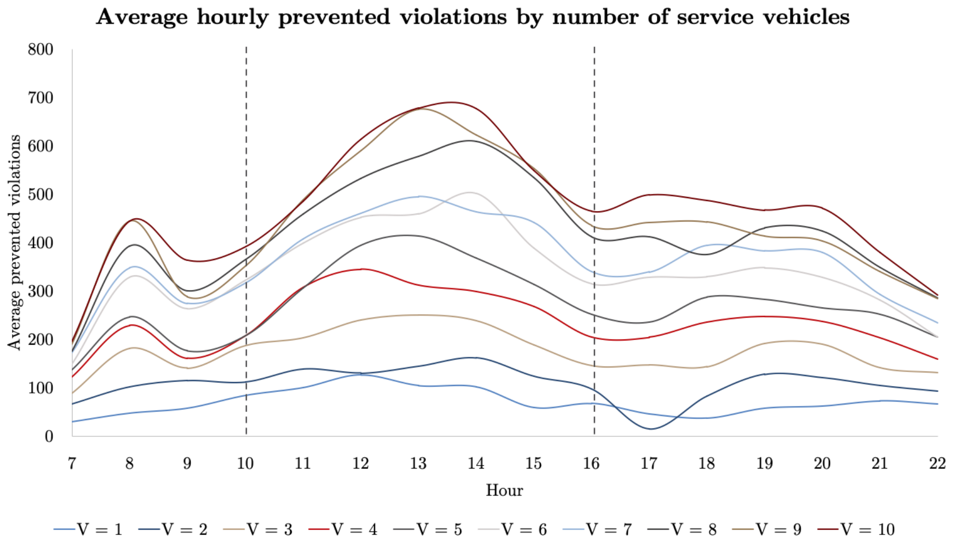


Figure 9.6: Average number of prevented violations with varying fleet size per hour in a day

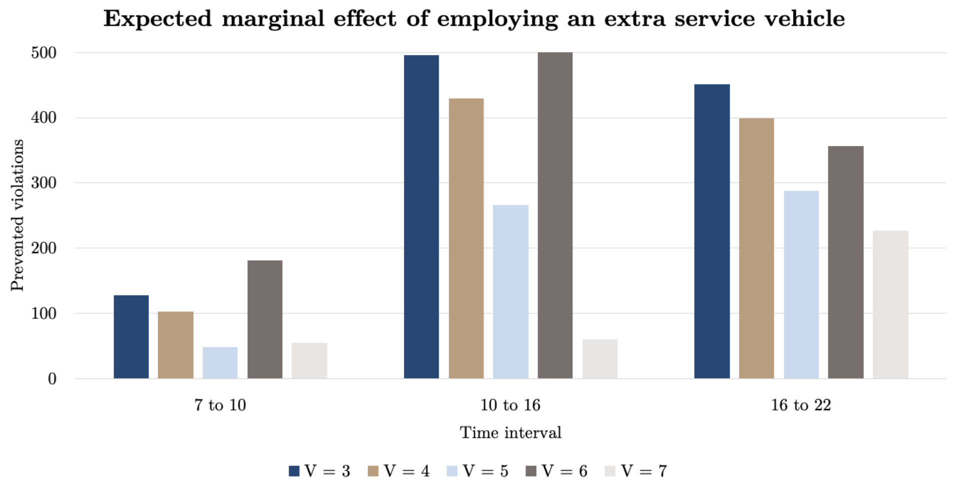


Figure 9.7: Expected marginal effect of employing additional vehicles at different time intervals during a day

The interval in the middle of the day contains the peak of prevented violations for all fleet sizes, and poses the highest marginal effects of additional service vehicles. This provides reason to employ a larger fleet in this interval. Lastly, the evening interval has a stable number of average prevented violations per hour until 8 pm, where the number decreases for all fleet sizes. The value of an extra service vehicle is also generally high in the evening interval. In conclusion, the results show that prioritizing service vehicle efforts in the middle of the day and early evening produce the highest marginal improvement in prevented violations. However, it is important to note that initial state of the system greatly affects these numbers. Efficient use of service vehicles in one interval may hence reduce the need of service vehicles in subsequent intervals. For instance, the marginal effect of a service vehicle might actually prove higher during the night when preparing for the morning rush.

Heterogeneous fleet

The service vehicles employed in previous analyses are dual purpose service vehicles, which can transport both bikes and batteries. These are simply referred to as *service vehicles*, and have a capacity of 20 bikes and 40 batteries. In a BSS, battery swaps and bike rebalancing may however be performed as separate tasks by distinct vehicle types. The following analysis evaluates the effect of separating rebalancing and battery swap activities between two different types of vehicles. We split the inventory capacity of the presented service vehicle between two new vehicle types: a *rebalancing vehicle* with capacity for 20 bikes, and a *battery swap vehicle* with capacity for 40 batteries. This split creates increased flexibility as battery swaps and rebalancing do not have to occur at the same stations at the same time. Another potential advantage is lower investment costs related to a battery swap vehicle. However, if choosing to operate more vehicles in total, other operating costs, like wage expenses, will increase. Table 9.3 displays the average prevented violations when employing combinations of up to five battery swap vehicles and five rebalancing vehicles.

We observe in the grey cells that only employing battery vehicles does not necessarily improve the total number of prevented violations. This is due to an increased number of congestions. With only battery swap vehicles in operation, more trips are initiated as starvations are prevented, but many of these lead to congestion violations at the end station unless accompanied by rebalancing efforts or increased parking capacity at stations.

The table diagonal, marked in blue, is comparable with the results from using one to five dual purpose service vehicles displayed in Table 9.4, as the total inventory capacity is the same. The small differences in performance indicate that the value of splitting up the activity of the service vehicles seems to be low. However, an important note is that the same criticality score policy and driving times are used to generate routes for all vehicle types. In addition, none of the vehicles are allowed to visit the same station simultaneously, which creates a small bias in favor of the service vehicle.

Table 9.3: Average number of prevented violations by number of employed rebalance vehicles and battery vehicles

Reb.	**Battery swap vehicles**					
vehicles	0	1	2	3	4	5
0	-	103	120	-111	36	-37
1	913	863	1033	766	939	938
2	1680	1730	1916	1974	1914	1998
3	2208	2403	2657	2769	2832	2708
4	2781	3169	3617	3612	3763	3542
5	3620	3707	4180	4481	4314	4510

Table 9.4: Average number of prevented violations by number of dual purpose service vehicles

Service vehicles	**Prevented violations**
0	-
1	1135
2	1744
3	2818
4	3749
5	4351

9.2.2 The Value of Geofencing

Introducing geofencing can be seen as increasing the parking capacity of stations and is a measure to prevent congestions in a BSS. Table 9.5 shows the average number of congestions in a day with increasing station capacity. A station capacity factor of 3.00, which makes the total system capacity three times larger than the initial capacity, eliminates congestions from occurring with service vehicles in operation.

Table 9.5: Average number of observed congestions when multiplying the initial station capacity with an increasing factor

Station capacity factor	**Congestions with no service vehicles**	**Congestions with service vehicles**
1.00	4688	2800
1.25	1920	908
1.50	984	306
1.75	607	101
2.00	422	33
2.25	329	8
2.50	263	6
2.75	213	1
3.00	174	0

Even though increasing station capacities improves the problem of congestion, it may concurrently increase the number of starvations occurring in the system, illustrated in Figure 9.8. The figure shows in increasing number of starvations when increasing the station capacity multiplier. This is due to greater imbalance in the system, where more bikes end up in the same areas. Consequently, more starvations will occur on stations in other areas which receive a lower supply of bikes. Therefore, the total number of violations must be taken into account when altering station capacities without also changing other system characteristics like the number of bikes or level of service vehicle activity.

Figure 9.8: Average number of starvations and congestions by station capacity multiplier

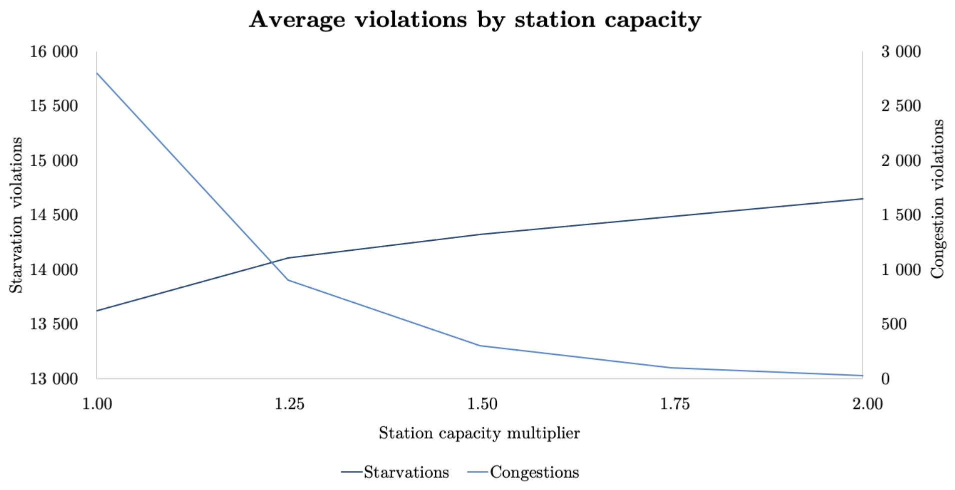


Figure 9.9 shows the average improvement in total violations when increasing the station capacities. The trend visible in this graph suggests that increasing the station capacities with a factor of more than 1.5, i. e. by more than 50%, does not further improve the total violations in the system.

9.2.3 The Value of Charging Stations

In the analysed BSS, 20 out of 200, or 10% of the stations, are charging stations. Previous analyses have shown that the predicted customer arrivals by UIP cause approximately 16,000 natural starvations per day in a system with no service vehicle operations, corresponding to around 50% of unmet outgoing charged customer demand. Starvations can either occur at empty stations, or at stations with only flat bikes. The latter case is denoted *flat station starvation*. As opposed to starvations at empty stations, flat station starvations may be prevented solely by offering charging at the station. Charging can either be performed by battery swaps or through a charging station. Charging stations incur high

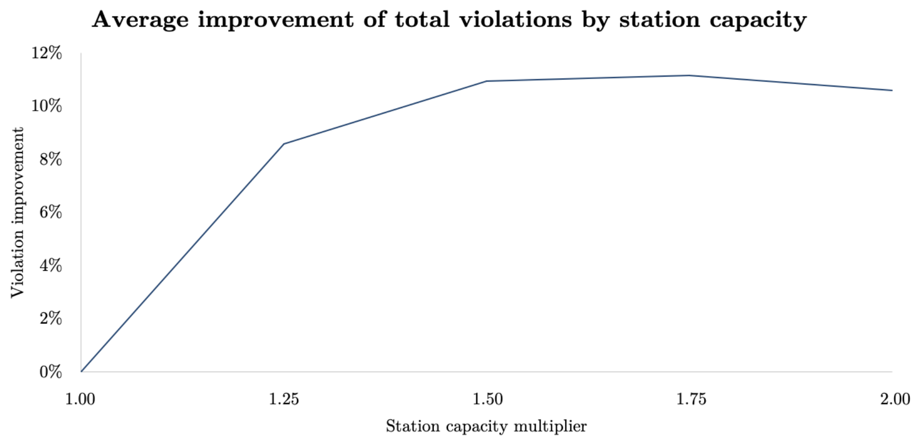


Figure 9.9: Graph illustrating the average improvement in total violations when increasing the station capacities

investment costs, and it is hence of interest to explore the value of introducing these. The following analysis presents the effect of introducing charging stations on the number of natural starvations in the system, disregarding rebalancing efforts in the BSS.

Charging stations can influence the number of natural flat station starvations occurring in a day. Assuming no charging delay, no flat station starvations will occur in a system with 100% charging stations. As a result, the absolute difference between the natural starvations observed in a system with 100% charging stations, and the natural starvations observed in a system with a lower level of charging stations, corresponds to the number of flat station starvations. Figure 9.10 shows the development in natural starvations for every ten percent increase in number of charging stations. The numbers represent an average across ten days of simulation.

We observe from Figure 9.10 that approximately 2,400 of the natural starvations in a system can be categorized as flat station starvations. This corresponds to ∼15% of the natural starvations occurring in the system. The rest of the starvations are due to empty stations, and cannot be prevented by charging in isolation.

An important note to the analysis is that changes in the number of congestions are not addressed. Through closer inspection of the data, we observe an increase in the number of natural congestions with added number of charging stations. The same tendency was discovered in the previous analysis performed on heterogeneous fleets, where addition of more battery vehicles, unless accompanied by increased rebalancing efforts, proved to have negative marginal effect. This further establishes a general discovery made in this chapter; the high level of inter-dependencies of system parameters.

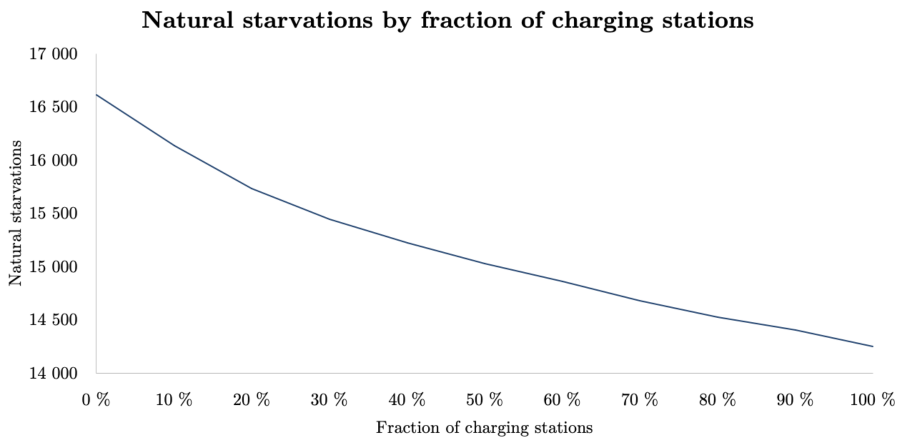


Figure 9.10: Average of natural starvations occurring in a BSS with increasing fraction of charging stations

Chapter 10

Concluding Remarks

In this thesis, the Dynamic Stochastic Bike Rebalancing and Charging Problem (DSBRCP) encountered in electric bike sharing systems (BSS) is solved with a two-stage stochastic programming column generation heuristic. The solution method combines exact optimization and approximating heuristics. The main objective of solving the problem is to provide a decision making policy for efficient rebalancing of bikes and charging of batteries. This policy aims at maximizing customer satisfaction by preventing occurrences of unsatisfied customer requests.

The DSBRCP heuristic is composed of three main steps; (i) initializing routes and patterns, (ii) solving subproblems, and (iii) solving a master problem. The first step is implemented heuristically, while the second and third step find optimal solutions. In the first step, a set of routes and pattern combinations are generated using a branching algorithm. For each of the initialized route and pattern combinations, a score for the combination is found by solving a subproblem in step two. Together, the first and second step construct a column that is passed to the master problem. The master problem combines these columns to find the optimal first stage decisions for each vehicle. The DSBRCP heuristic solves the problem for a time horizon of 25 minutes. In order to limit shortsightedness, predictions of an ideal target state are considered in the solutions.

The BSS in Oslo, operated by Urban Infrastructure Partners (UIP), is used as a sample case and source of data in this thesis. Based on this data and industry knowledge provided by UIP, a simulation framework mirroring a real-world electric BSS has been designed to analyze the performance of our solution and derive valuable managerial insights. It was also used to optimize parameter configurations and assess solution time. Requiring a reasonable solution time of less than 60 seconds, we are able to achieve high quality solutions when employing up to eight service vehicles in a real-sized BSS, e.g. the BSS in Oslo with about 200 stations.

To examine the performance of the DSBRCP heuristic, we conducted simulations with five service vehicles guided by the heuristic and an alternative strategy. The alternative strategy was constructed based on the characteristics of UIP's current rebalancing efforts. In a pairwise comparison, the number of violations occurring using the same sequence of generated trips were recorded and evaluated for each strategy. The results show that the DSBRCP heuristic are able to prevent 50% more violations during a day than the UIP-inspired strategy. The improvement corresponds to a 23.3% reduction of violations occurring per day when compared to a system with no service vehicle operations. This number has potential to be greatly improved if employed in a less strained system, or with better predictions for customer arrivals and ideal states.

Based on data from simulations, useful managerial insights have been identified. We have seen that the marginal effect of increasing the number of service vehicles in operation generally decreases, and is higher during hours with lower number of customer requests. During the busy time periods such as the morning rush hours, more trips are ongoing, making rebalancing efforts difficult. Thus, entering the busy time period in an ideal state is more effective. Further, we have identified a benefit of increasing all station capacities by 50% through geofencing. However, a larger increase in station capacities uncoupled by other system alterations, proves counterproductive. This is because higher capacity at stations cause increased aggregation of bikes, which leads to more starvations in other parts of the system. Through simulations using a heterogeneous service vehicle fleet, we revealed a larger utility of rebalancing efforts than battery swaps and additional charging stations when evaluated in isolation. However, these results could change with longer simulation periods or with different demand levels.

In conclusion, this thesis presents applications of optimization techniques in order to improve customer satisfaction in electric BSSs. The approach combining exact solution methods with heuristic approximations in a stochastic formulation has proven effective to solve the DSBRCP. Current literature focuses primarily on the relocation problem, omitting the problem of securing a charged fleet. Thus, our contribution is to solve the extended rebalancing problem that also incorporates charging decisions by means of battery swaps and charging stations. The work may also serve as a basis for future research to further optimize electric BSSs.

Chapter 11

Future Research

This chapter highlights further research opportunities based on the findings presented in this thesis. The opportunities can be split into two categories: (i) suggested areas of development for the presented heuristic, and (ii) research on the input data that influences the heuristic solutions.

A possible area of improvement for the DSBRCP heuristic is incorporation of a more sophisticated policy for pattern generation. The presented heuristic uses pattern generation, representing the rebalancing and battery swap decisions, to limit the solution space and hence the required solution time of the master problem. The current patterns are generated based on assumptions about the maximum activity that it may be beneficial to complete at a station when the service vehicle arrives. It could be of interest to include more combinations of fractional patterns or develop patterns that allow for more activity than assumed reasonable. This could be done by developing a pricing problem.

One of the reasons why it is difficult to make a good rebalancing policy, is limitations in predicting the customer requests in the system. The requests are affected by numerous factors that are prone to sudden variations, which make them difficult to predict. Typically, the forecasts are solely based on historical data, but sometimes they also consider external factors like the weather or information about surrounding events. Yet, the stochastic aspects of the demand often lead to gaps between the forecasts and the realizations. Further research on demand predictions in an attempt to close this gap could lead to better estimates of ideal states and consequently increased customer satisfaction when solving the DSBRCP. Furthermore, the approach used to predict demand patterns in a traditional BSS is not necessarily transferable to a setting with electric bikes.

When optimizing customer satisfaction as presented in this thesis, the overall objective is to maximize the number of prevented violations constrained by the resources available. In this regard, it would be interesting to explore how the natural flow of bicycles may influence the number of violations. Bikes placed at certain stations might prove to generate a longer sequence of trips before ending up at a station without outgoing customer demand. This insight could be used to make smarter rebalancing decisions by rewarding unloading of charged bikes at advantageous stations. This insight could also be combined with user incentivization, where users are rewarded for ending the trips at the same stations.

Bibliography

Baltic Sea Region Electric. E-bike sharing scheme in Gdansk Metro Region. 2018. `https://www.bsr-electric.eu/news/https-bsr-electric-eu-news` (Accessed: 2019-08-12).

M. Barth and M. Todd. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7:237–259, 1999.

R. Bellman. A Markovian Decision Process. *Indiana University Mathematics Journal*, 6: 15, 1957.

T. Bieliński and A. Ważna. Hybridizing bike-sharing systems: The way to improve mobility in smart cities. *Transport Economics and Logistics*, 79, 2019.

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

P. Ceres. How Jump designed a global electric bike, 2018. `https://www.wired.com/story/how-jump-designed-a-global-electric-bike/` (Accessed: 2019-08-12).

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

F. Chen, M. C. Klos, K. Turon, and G. Sierpinski. Fifth generation of bike-sharing systems - Examples of Poland and China. *Scientific Jourunal of Silesian University of Technology*, 99:5–13, 2018.

Z. Chen, Y. Hu, J. Li, and X. Wu. Optimal deployment of electric bicycle sharing stations: Model formulation and solution technique. *Networks and Spatial Economics*, 2019.

C. Cherry, S. Worley, and D. Jordan. Electric bike sharing–system requirements and operational concepts. 2010. University of Tennessee, Knoxville, TN (United States). Department of Civil and Environmental Engineering., Technical report.

K. Cripps. Bike share boom: 7 cities doing it right. *CNN Travel*, 2013. `https://edition.cnn.com/travel/article/bike-share-boom-global-report/index.html` (Accessed: 2019-09-16).

P. DeMaio. The bike-sharing phenomenon - The history of bike-sharing. *Carbusters Magazine, November*, 2008.

L. C. Drazek. Intensity estimation for Poisson processes. *The University of Leeds, School of Mathematics*, 2013.

H. M. Espegren and J. Kristianslund. Modeling the Static Bicycle Repositioning Problem. Master's thesis. *Norwegian University of Science and Technology*, 2015.

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

S. Fosen and O. Haldorsen. A battery swap model for electrical bike sharing systems. Project thesis. *Norwegian University of Science and Technology*, 2019.

C. Fricker and N. Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, 3:1–31, 2014.

R. Gallager. Discrete stochastic processes. *Journal of the Operational Research Society*, 48, 1997.

P. C. Guedes and D. Borenstein. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. *Computers Industrial Engineering*, 90:361–370, 2015.

K. Hagen and M. Gleditsch. A dynamic rebalancing model for bike sharing systems. Project thesis. *Norwegian University of Science and Technology*, 2017.

Z. Han, Y. Yang, Y. Jiang, and W. Liu. SDVRP-based reposition routing in bike-sharing system. In *Algorithms and Architectures for Parallel Processing*, pages 596–610. Springer International Publishing, 2018.

C. Hughes. Bike share: The dawn of the smartbike (and the death of dock-blocking). 2017. `https://medium.com/social-bicycles/bikeshare-the-dawn-of-the-smartbike-and-the-death-of-dock-blocking-9f52bb642ae` (Accessed: 2019-08-12).

H. Jiang and H. Jamba. To solve China's bike-sharing woes, Hangzhou and Shanghai turn to bluetooth and geofencing. *TheCityFix*, 2019. `https://thecityfix.com/blog/solve-chinas-bike-sharing-woes-hangzhou-shanghai-turn-bluetooth-geofencing-hui-jiang-harshita-jamba/` (Accessed: 2019-08-12).

Jump. Jump: About us, 2019. `https://www.jump.com/us/en/about/` (Accessed: 2019-08-12).

A. Kadri, I. Karim, B. Hu, and G. R. Raidl. A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems. *Computers Industrial Engineering*, 95:41–52, 2016.

B. Legros. Dynamic repositioning strategy in a bike-sharing system how to prioritize and how to rebalance a bike station. *European Journal of Operational Research*, 2019.

J. Liu, L. Sun, W. Chen, and H. Xiong. Rebalancing bike sharing systems: A multi-source data smart optimization. pages 1005–1014, 08 2016.

F. Malcata. *Statistical Hypothesis Testing*, pages 919–926. 06 2020.

M. R. K. Mes and A. P. Rivera. *Markov Decision Processes in Practice*, pages 63–101. Springer International Publishing, 2017.

Mevo. Mevo has been launched, 2019. `https://rowermevo.pl/en/english-mevo-ruszylo/` (Accessed: 2019-12-08).

P. Midgley. Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas. *United Nations, Department of Economic and Social Affairs*, 8:1–12, 2011.

M. Modijevsky. One of EU's largest electrical bike-sharing systems launched in Gdansk - Gdynia - Sopot metropolitan area. *Eltis - The Urban Mobility Observatory*, 2019. `https://www.eltis.org/discover/news/one-eus-largest-electric-bike-sharing-systems-launched-gdansk-gdynia-sopot` (Accessed: 2019-09-16).

N. T. Moungla, L. Létocart, and A. Nagih. Solutions diversification in a column generation algorithm. *Algorithmic Operations Research*, 5(2):86–95, 2010.

B. Neumann Saavedra, T. G. Crainic, B. Gendron, D. Mattfeld, and M. Römer. Service network design of bike sharing systems with resource constraints. pages 352–366, 2016.

M. Nourinejad and M. Roorda. A dynamic carsharing decision support system. *Transportation Research Part C: Logistics and Transportation Review*, 4:36–50, 2014.

Oslo Bysykkel. How it works, 2020a. `https://oslobysykkel.no/en/how-it-works` (Accessed: 2020-05-25).

Oslo Bysykkel. About us, 2020b. `https://oslobysykkel.no/en/about` (Accessed: 2020-05-25).

J. Oyola and H. Arntzen. The Stochastic Vehicle Routing Problem, a literature review, part i: Models. *EURO Journal on Transportation and Logistics*, 7:193–221, 2016.

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

J. Pfrommer, J. Warrington, G. Schildbach, and M. Morari. Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15:1567–1578, 04 2013.

M. Rainer-Harbach, P. Papazek, B. Hu, and G. Raidl. Balancing bicycle sharing systems: A variable neighborhood search approach. pages 121–132, 2013.

M. Rao. Measuring urban traffic congestion – A review. *International Journal for Traffic and Transport Engineering*, 2:286–305, 2012.

J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257, 2013.

S. Shaheen, S. Guzman, and H. Zhang. Bikesharing in europe, the americas, and asia: Past, present, and future. *Institute of Transportation Studies, UC Davis, Institute of Transportation Studies, Working Paper Series*, 2143, 2010.

J. Shuguang, C. R. Cherry, L. D. Han, and D. A. Jordan. Electrical bike sharing: Simulation of user demand and system availability. *Journal of Cleaner Production*, 85: 250–257, 2014.

P. Teetor. *R Cookbook*. O'Reilly Media, Inc., 1st edition, 2011.

M. Toll. Uber's new Jump electric bicycles just fixed everything wrong with shared e-bikes. 2018.

P. Toth and D. Vigo, editors. *Vehicle Routing: Problems, Methods, and Applications, Second Edition*. Number 18 in MOS-SIAM Series on Optimization. SIAM, 2014.

P. Venkateshan and K. Mathur. An efficient column-generation-based algorithm for solving a pickup-and-delivery problem. 38(12):1647–1655, 2011.

P. Vogel. *Service Network Design of Bike Sharing Systems : Analysis and Optimization*. 2016. doi: 10.1007/978-3-319-27735-6.

P. Vogel, J. Ehmke, and D. Mattfeld. Cost-efficient allocation of bikes to stations in bike sharing systems. pages 498–512, 2017.

X. Wang, G. Lindsey, J. E. Schoner, and A. Harrison. Modeling bike share station activity: Effects of nearby businesses and jobs on trips to and from stations. *Journal of Urban Planning and Development*, 142, 2015.

J. Warrington and D. Ruchti. Two-stage stochastic approximation for dynamic rebalancing of shared mobility systems. *Transportation Research Part C: Emerging Technologies*, 104:10–34, 2019.

L. Zhang. Sustainable bike-sharing systems: Characteristics and commonalities across cases in urban China. *Journal of Cleaner Production*, pages 124–133, 2015.

Appendix

Appendix A

Mathematical Model

This chapter describes a mathematical model of the Markov Decision Process (MDP) subproblem, discussed in Section 5.1. As opposed to the stochastic heuristic algorithm presented in the thesis, the mathematical model is deterministically formulated. A similar, yet simplified, version of the model is studied in Fosen and Haldorsen (2019), where analyses show that the formulation suffers from long computational times. This chapter starts with a presentation of the assumptions made in Section A.1, before describing the notation used in Section A.2. Section A.3 lists the constraints that the MDP subproblem is subject to. Note that this model needs to be linearized before implementation.

A.1 Assumptions

The time horizon for the problem is short. The customer movement rates are therefore assumed to follow a constant rate within each time horizon, reflecting the average number of bicycles that depart and arrive. The arrival rate of bicycles is split into bicycles arriving with sufficient charge for a new trip, *charged bicycles*, and bicycles that require charging or battery swaps before being available for use again, *flat bicycles*. Both these rates and the demand rate are assumed known and based on historical numbers. The rate of departure from a station equals the demand rate while there are enough available charged bicycles. Otherwise, if the station does not have any charged bicycles available, the departure rate is set to zero. Violations are defined similarly as in Chapter 5, and we separate between starvations and congestions. A starvation occurs when there is demand at a station with no charged bicycles available. Congestion occurs when a station is full and the net rate of incoming bicycles is positive. Once a station is full, we assume that there are no changes in station inventory before the station is visited. Each occurrence of starvation and congestion is penalized equally in the model.

Flat bicycles will not be available for use, and the battery level of a bicycle will not change during a time horizon unless the battery is swapped. Charging stations are assumed to not need battery swaps and it should not be necessary to unload charged bicycles. The effect of charging is not seen before the next time horizon. The rate of incoming flat bicycles is assumed to be a constant fraction of the rate of incoming charged bicycles.

A service vehicle can visit multiple stations during a given time horizon. To reduce the dimensionality of the problem, a station can only be visited once in each time horizon. Each vehicle can also visit the depot once, which restores the number battery inventory of a vehicle to the vehicle's capacity. The depot is hence the only station that can be visited by multiple vehicles within the same time horizon. We assume the service time per battery swap, bicycle handling and parking time at the station and driving time between each station to be constant. We also assume a battery swap to result in a fully charged electrical bicycle. The discharged batteries need to be recharged at the depot before put in use again.

A.2 Notation

Sets

$\mathcal{S}^E$	Set of electric charging stations
$\mathcal{S}^N$	Set of non-charging stations
$\mathcal{S}$	$\mathcal{S}^E \cup \mathcal{S}^N$, the set of stations where one can deliver or pick up bicycles
$\mathcal{S}^+$	Set of stations in $\mathcal{S}$, including the depot and the artificial end station
$\mathcal{V}$	Set of service vehicles

Indices

i, j	Station $i, j \in \mathcal{S}^+$
v	Service vehicle $v \in \mathcal{V}$
$o(v)$	Starting station for service vehicle v
d	Artificial destination
b	Depot station

Parameters

$\overline{T}$	Length of the time horizon
Q_v^B	Storage capacity of batteries at service vehicle v
Q_v^C	Storage capacity of bicycles at service vehicle v
Q_i^S	Docking capacity at station i
L_v^{BV}	Initial load of batteries at the beginning of the time horizon for vehicle v
L_v^{CV}	Initial load of charged bicycles at the beginning of the time horizon for vehicle v
L_v^{FV}	Initial load of flat bicycles at the beginning of the time horizon for vehicle v
L_i^{CS}	Initial load of charged bikes at station i
L_i^{FS}	Initial load of flat bikes at station i
O_i	Ideal state at station i at time $\overline{T}$
T_{ij}^D	Driving time between station i and j
$T_v^{o(v)}$	Driving time for service vehicle v to the starting station $o(v)$
T^P	Service vehicle parking time at station
T^B	Unit handling time at station used for performing a battery swap
T^C	Unit handling time at station used for performing a bicycle pick-up or delivery
I_i^{OCR}	Estimated rate of demand for charged bikes at station i
I_i^{ICR}	Estimated rate of incoming charged bikes to station i within $\overline{T}$
I_i^{OCR}	Estimated rate of incoming flat bikes to station i within $\overline{T}$
W^V	Weight of violations in the objective function
W^D	Weight of deviations in the objective function
W^R	Weight of reward in the objective function

Variables

x_{ijv}	1 if service vehicle v drives directly from station i to j

t_i	Time when station i is visited
t_v^B	Time when the depot is visited by vehicle v
q_{iv}^B	Number of battery swaps performed a station i by service vehicle v
q_{iv}^{CCU}	Number of charged bicycles unloaded from service vehicle v at station i
q_{iv}^{FCU}	Number of flat bicycles unloaded from service vehicle v at station i
q_{iv}^{CCL}	Number of charged bicycles loaded to service vehicle v at station i
q_{iv}^{FCL}	Number of flat bicycles loaded to service vehicle v at station i
l_i^{CS}	Number of bikes with sufficient battery at station i when service starts
l_i^{FS}	Number of flat bikes at station i when service starts
l_{iv}^{BV}	Number of batteries in vehicle v at the beginning of a visit at station i
l_{iv}^{CV}	Number of charged bicycles in vehicle v at the beginning of a visit at station i
l_{iv}^{FV}	Number of flat bicycles in vehicle v at the beginning of a visit at station i
s_i^C	Number of charged bikes at station i at time $\overline{T}$
s_i^F	Number of flat bikes at station i at time $\overline{T}$
v_i^S	Number of starvations at station i upon its visit
v_i^C	Number of congestion at station i upon its visit
d_i	Deviation at station i at $\overline{T}$

A.3 Constraints

This section presents the constraints in the mathematical model by groups. First, the constraints to generate feasible routes are described. These constraints are divided into subgroups for readability purposes. Further, we present the objective function consisting of three parts with its associated constraints, ending in a complete expression for the objective function. The notation described in Section A.2 is used with additional variables introduced when needed.

A.3.1 Ensure Feasible Routes

Routing constraints

Constraints (A.1) force all vehicles to start the route, and (A.2) force the routes to end in d.

Flow balance is ensured by (A.3), which implies that the routes are continuous between the start and end stations. Each station can only be visited once during the time horizon, with exception of the depot and end destinations which can be visited once by each vehicle. This is handled in constraints (A.4) and (A.5), respectively. Note that symmetry breaking constraints are intentionally not included in the model, as we force each vehicle to start at unique stations, and all stations only can be visited once within the time horizon.

$$\sum_{j \in \mathcal{S}^+} x_{o(v)jv} = 1 \quad v \in \mathcal{V} \tag{A.1}$$

$$\sum_{i \in \mathcal{S}^+} x_{idv} = 1 \quad v \in \mathcal{V} \tag{A.2}$$

$$\sum_{i \in \mathcal{S}^+} x_{ijv} - \sum_{i \in \mathcal{S}^+} x_{jiv} = 0 \quad v \in \mathcal{V}, j \in \mathcal{S}^+ \setminus \{o(v), d\} \tag{A.3}$$

$$\sum_{i \in \mathcal{S}^+} \sum_{v \in \mathcal{V}} x_{ijv} \leq 1 \quad j \in \mathcal{S} \tag{A.4}$$

$$\sum_{i \in \mathcal{S}} x_{ibv} \leq 1 \quad v \in \mathcal{V} \tag{A.5}$$

Time Constraints

The problem is solved over a fixed time horizon $\overline{T}$. To limit shortsightedness, each vehicle is allowed to initiate one visit that is scheduled to take place after $\overline{T}$, as illustrated in Figure A.2. This reduces the idle time that may occur at the end of a time horizon shown in Figure A.1.

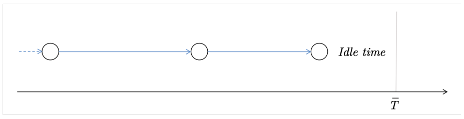


Figure A.1: Idle time occurs at the end of the time horizon if no station visits are allowed after the time horizon

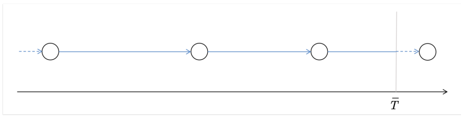


Figure A.2: One station visit allowed after the time horizon, preventing idle time for the vehicles

As discussed in the model assumptions, each station can be visited only once, while the depot can be visited once by each vehicle. The depot has a unique time of visit for each vehicle, while the other stations only have one time of visit. Due to this fact, we have defined the term *depot related station*. A depot related station is a station that is visited right before or right after the depot, as illustrated in Figure A.3. Further, the time constraints are divided into two groups: Time constraints for depot related stations, and time constraints for other stations.

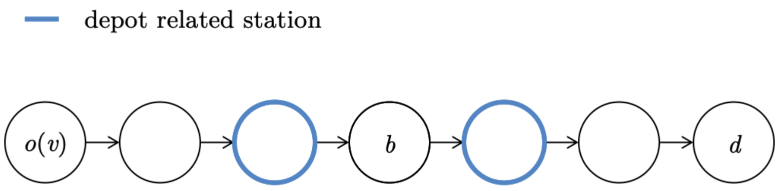


Figure A.3: Illustration of a vehicle's route, with depot related stations highlighted. The depot is labeled with a b

For the stations *not* related to depot, the arrival time of a station visit i should be before the arrival time on a later visit j if the service vehicle drives directly from i to j, enforced by constraints (A.6). These constraints also include the parking, battery swap, bicycle handling and driving time. Constraints (A.7) ensure that the service vehicles do not arrive at the first station before the time it takes to drive there. The time of arrival should be within the time horizon for all visits except for the last, enforced by Constraints (A.8). Constraints (A.9) make the time of a visit zero if a station is not visited.

$$\Bigg(t_i + T^P + T^B \sum_{v\in\mathcal{V}} q_{iv}^B + T^C \sum_{v\in\mathcal{V}} \left(q_{iv}^{CCL} + q_{iv}^{FCL} + q_{iv}^{CCU} + q_{iv}^{FCU}\right) + T_{ij}^D - t_j\Bigg) \sum_{v\in\mathcal{V}} x_{ijv} \leq 0 \quad i \in \mathcal{S}, j \in \mathcal{S}^+ \setminus \{b\} \tag{A.6}$$

$$t_{o(v)} \geq T_v^{o(v)} \quad v \in \mathcal{V} \mid o(v) \neq b \tag{A.7}$$

$$\left(1 - \sum_{v\in\mathcal{V}} x_{idv}\right) t_i \leq \overline{T} \quad i \in \mathcal{S} \tag{A.8}$$

$$t_i \left(1 - \sum_{j\in\mathcal{S}^+} \sum_{v\in\mathcal{V}} x_{ijv}\right) \leq 0 \quad i \in \mathcal{S} \tag{A.9}$$

For the depot related stations, constraints (A.10) set the arrival time at the depot for a specific vehicle. Note that instead of using separate variables, t_v^B, for setting the arrival times at the depot, we could have created a copy of the depot node for each service vehicle. Constraints (A.11) ensure that the arrival time at the next station after the depot, is after the time of visit to the depot. Further, Constraints (A.12) ensure that the arrival at the depot is later than the driving time from start, if the depot is a vehicle's start station. Lastly, the visit time at the depot for a vehicle is set to zero if the vehicle does not visit the depot. This is enforced in Constraints (A.13).

$$\begin{split}(t_i + T^P + T^B q^B_{iv} + T^C(q^{CCL}_{iv} + q^{FCL}_{iv} \\ + q^{CCU}_{iv} + q^{FCU}_{iv}) + T^D_{ib} - t^B_v)x_{ibv} \leq 0 \quad i \in \mathcal{S}, v \in \mathcal{V}\end{split} \tag{A.10}$$

$$\left(t^B_v + T^P + T^D_{bj} - t_j\right) x_{bjv} \leq 0 \quad j \in \mathcal{S}^+ \setminus \{b\}, v \in \mathcal{V} \tag{A.11}$$

$$t^B_v \geq T^{o(v)}_v \quad v \in \mathcal{V} \mid o(v) = b \tag{A.12}$$

$$t^B_v \left(1 - \sum_{j \in \mathcal{S}^+} x_{bjv}\right) \leq 0 \quad v \in \mathcal{V} \tag{A.13}$$

Vehicle loading constraints

Constraints (A.14) restrict that the service vehicle cannot swap more batteries than the amount of available flat bikes after the bicycle rebalancing decisions are performed at a station. Constraints (A.15) ensure that the sum of bikes that are unloaded from a station does not exceed the number of available bicycle slots at the vehicle. Constraints (A.16) set the available battery load of the vehicles on the first station visit to the initial battery inventory of the vehicles. Constraints (A.17) and (A.18) work similarly for charged and flat bikes, respectively. The battery inventory balance for a vehicle after a depot visit is handled in Constraints (A.19) and for all other stations in Constraints (A.20). The vehicle inventory balance for bicycles are handled in Constraints (A.21) and (A.22). Lastly, Constraints (A.23) and (A.24) ensure that there are not performed any bicycle handling in the depot.

$$q^B_{iv} \leq l^{BV}_{iv} \quad i \in \mathcal{S}^N, v \in \mathcal{V} \tag{A.14}$$

$$q^{CCL}_{iv} + q^{FCL}_{iv} \leq Q^C_v - l^{CV}_{iv} - l^{FV}_{iv} + q^{CCU}_{iv} + q^{FCU}_{iv} \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.15}$$

$$l^{BV}_{o(v)v} = L^{BV}_v \quad v \in V \tag{A.16}$$

$$l^{CV}_{o(v)v} = L^{CV}_v \quad v \in V \tag{A.17}$$

$$l^{FV}_{o(v)v} = L^{FV}_{v} \quad v \in V \tag{A.18}$$

$$x_{bjv}(l^{BV}_{jv} - Q^{B}_{v}) = 0 \quad j \in \mathcal{S}^{+}, v \in \mathcal{V} \tag{A.19}$$

$$x_{ijv}(l^{BV}_{jv} - l^{BV}_{iv} + q^{B}_{iv}) = 0 \quad i \in \mathcal{S}, j \in \mathcal{S}^{+}, v \in \mathcal{V} \tag{A.20}$$

$$x_{ijv}(l^{CV}_{jv} - l^{CV}_{iv} + q^{CCU}_{iv} - q^{CCL}_{iv}) = 0 \quad i \in \mathcal{S}^{+} \setminus \{d\}, j \in \mathcal{S}^{+}, v \in \mathcal{V} \tag{A.21}$$

$$x_{ijv}(l^{FV}_{jv} - l^{FV}_{iv} + q^{FCU}_{iv} - q^{FCL}_{iv}) = 0 \quad i \in \mathcal{S}^{+} \setminus \{d\}, j \in \mathcal{S}^{+}, v \in \mathcal{V} \tag{A.22}$$

$$q^{CCL}_{bv} - q^{CCU}_{bv} = 0 \quad v \in \mathcal{V} \tag{A.23}$$

$$q^{FCL}_{bv} - q^{FCU}_{bv} = 0 \quad v \in \mathcal{V} \tag{A.24}$$

Station loading constraints
Constraints (A.25) handle the flat bike inventory balance per station for a visit. Constraints (A.26) handle the same for battery bikes. It should also not be possible to swap more batteries than there are flat bikes at a station after rebalancing actions, handled in Constraints (A.27). Constraints (A.28) and (A.29) ensure that the number of bicycles unloaded from a station does not exceed the station inventory for flat and charged bikes, respectively. The total number of loaded bikes to the station is restricted by the number of available slots, secured in (A.30). Constraints (A.31) and (A.32) ensure that all bicycle handling variables are set to zero if the station is not visited by the vehicle. Lastly, Constraints (A.33) work similarly for the number of battery swaps.

In our model, q^{FCU}_{iv} is forced to 0 for all stations in $\mathcal{S}^{N}$, implying that one can not deliver flat bikes to non-charging stations. Further, q^{FCL}_{iv} and q^{B}_{iv} is set to 0 for all charging stations, implying that one can not remove flat bikes nor swap batteries at charging stations. These assumptions are used to prevent unpractical solutions to the model.

$$l^{FS}_{i} = L^{FS}_{i} + I^{IFR}_{i} t_i \quad i \in \mathcal{S} \tag{A.25}$$

$$l^{CS}_{i} = L^{CS}_{i} + (I^{ICR}_{i} - I^{OCR}_{i}) t_i + v^{S}_{i} - v^{C}_{i} \quad i \in \mathcal{S} \tag{A.26}$$

$$\sum_{v \in \mathcal{V}} q_{iv}^{B} \leq l_i^{FS} + \sum_{v \in \mathcal{V}} q_{iv}^{FCL} \quad i \in \mathcal{S}^N \tag{A.27}$$

$$\sum_{v \in \mathcal{V}} q_{iv}^{FCL} \leq l_i^{FS} \quad i \in \mathcal{S}^N \tag{A.28}$$

$$\sum_{v \in \mathcal{V}} q_{iv}^{CCL} \leq l_i^{CS} \quad i \in \mathcal{S} \tag{A.29}$$

$$\sum_{v \in \mathcal{V}} \left(q_{iv}^{FCU} + q_{iv}^{CCU} - q_{iv}^{FCL} - q_{iv}^{CCL}\right) \leq Q_i^S - l_i^{CS} - l_i^{FS} \quad i \in \mathcal{S} \tag{A.30}$$

$$q_{iv}^{FCU} + q_{iv}^{CCU} - Q_v^C \sum_{j \in \mathcal{S}^+} x_{ijv} \leq 0 \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.31}$$

$$q_{iv}^{FCL} + q_{iv}^{CCL} - Q_v^C \sum_{j \in \mathcal{S}^+} x_{ijv} \leq 0 \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.32}$$

$$q_{iv}^{B} - Q_v^B \sum_{j \in \mathcal{S}^+} x_{ijv} \leq 0 \quad i \in \mathcal{S}^N, v \in \mathcal{V} \tag{A.33}$$

Non-negativity, integer, and binary constraints
In the model, only necessary variables are created. This is ensured by making the relevant sets as tight as possible. Constraints (A.34) make the decisions of which station to visit next in the route binary. Arches from $i = b$ to $j = d$ are not instantiated, which make it impossible for the depot to be the last station visit. The number of batteries to change at each station visit and the quantity of bicycles to load or unload should be integer, given by constraints (A.35) to (A.39) . All variables should be non-negative, ensured in constraints (A.40) to (A.51).

$$x_{ijv} \in \{0,1\} \quad i \in \mathcal{S}^+ \setminus \{d\}, j \in \mathcal{S}^+ \setminus \{o(v) \forall v \in \mathcal{V}\} v \in \mathcal{V} \tag{A.34}$$

$$q_{iv}^{B} \geq 0, integer \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.35}$$

$$q_{iv}^{CCU} \geq 0, integer \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.36}$$

$$q_{iv}^{FCU} \geq 0, integer \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.37}$$

$$q_{iv}^{CCL} \geq 0, integer \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.38}$$

$$q_{iv}^{FCL} \geq 0, integer \quad i \in \mathcal{S}, v \in \mathcal{V} \tag{A.39}$$

$$t_i \geq 0 \quad i \in \mathcal{S}^+ \tag{A.40}$$

$$t_v^B \geq 0 \quad v \in \mathcal{V} \tag{A.41}$$

$$l_i^{CS} \geq 0 \quad i \in \mathcal{S} \tag{A.42}$$

$$l_i^{CS} \geq 0 \quad i \in \mathcal{S} \tag{A.43}$$

$$l_{iv}^{BV} \geq 0 \quad i \in \mathcal{S}^+, v \in \mathcal{V} \tag{A.44}$$

$$l_{iv}^{CV} \geq 0 \quad i \in \mathcal{S}^+, v \in \mathcal{V} \tag{A.45}$$

$$l_{iv}^{FV} \geq 0 \quad i \in \mathcal{S}^+, v \in \mathcal{V} \tag{A.46}$$

$$s_i^C \geq 0 \quad i \in \mathcal{S} \tag{A.47}$$

$$s_i^F \geq 0 \quad i \in \mathcal{S} \tag{A.48}$$

$$v_i^S \geq 0 \quad i \in \mathcal{S} \tag{A.49}$$

$$v_i^C \geq 0 \quad i \in \mathcal{S} \tag{A.50}$$

$$d_i \geq 0 \quad i \in \mathcal{S} \tag{A.51}$$

A.3.2 Objective Function

The objective function has three subobjectives: 1) Minimize total violations, 2) Minimize total deviation, and 3) Maximize reward from initiating trips that exceed the time horizon. The latter subobjective is done to reduce idle time at the end of the time horizon.

We use the weighted sum method to treat the multi-objective optimization problem (MOP) as a single-objective optimization problem (SOP). The weights W^V, W^D and W^R are weights for the total violations, deviations and reward, respectively. The objective function is written as follows:

$$\min W^V \cdot [\text{Sum of violations}] + W^D \cdot [\text{Total deviation}] - W^R \cdot [\text{Reward}]$$

Subobjective 1: Violations

The objective of the mathematical model includes minimization of the total violations. These violations are counted for each station from the beginning of the time horizon until the end of the time horizon, $\overline{T}$. This captures the violations across all stations regardless of whether the station was visited and the time of the visit.

The violations occurring before a station visit at station i is captured by v_i^S and v_i^C for starvations and congestions, respectively. Recall that a station's visit may happen within the time horizon, after the time horizon or not at all. This gives three possible situations for each station:

- **Situation 1:** A station does not get any visits, illustrated in Figure A.4
- **Situation 2:** A station is visited within the time horizon, illustrated in Figure A.5
- **Situation 3:** A station is visited after the time horizon, illustrated in Figure A.6

To accurately capture the violations in these three situations, additional variables are needed. First, we define four new violation variables:

$v_i^{S,f}$	Starvation from visit i to $\overline{T}$ if the visit is before $\overline{T}$, or total starvation if i is not visited
$v_i^{C,f}$	Congestion from visit i to $\overline{T}$ if the visit is before $\overline{T}$, or total congestion if i is not visited
$v_i^{S,F}$	Starvation from $\overline{T}$ to the visit if i is visited after $\overline{T}$
$v_i^{C,F}$	Congestion from $\overline{T}$ to the visit if i is visited after $\overline{T}$

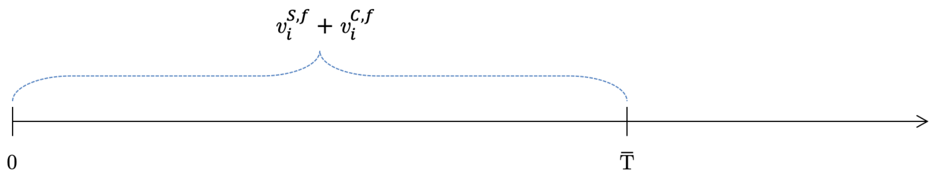


Figure A.4: Situation 1: Station i is not visited

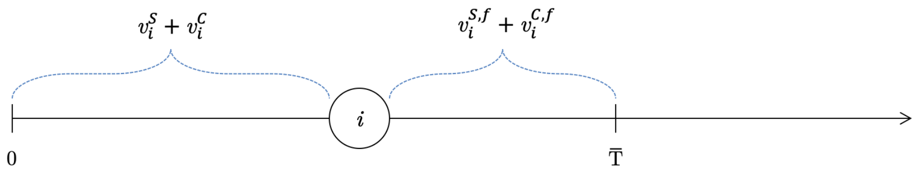


Figure A.5: Situation 2: Station i is visited within the time horizon

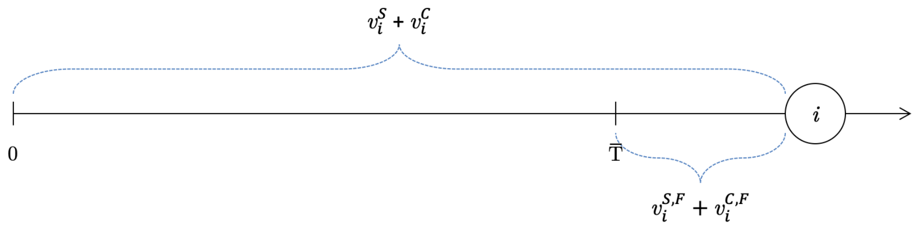


Figure A.6: Situation 3: Station i is visited after the time horizon

The total violation is then calculated as follows:

$$\sum_{i\in\mathcal{S}}\left(v_i^S + v_i^C\right) + \sum_{i\in\mathcal{S}}\left(v_i^{S,f} + v_i^{C,f}\right) - \sum_{i\in\mathcal{S}}\left(v_i^{S,F} + v_i^{C,F}\right)$$

In the rest of this section, the constraints that capture these violation variables are introduced. To distinguish between the different situations, we introduce two binary variables: These variables are determined by Constraints (A.52) - (A.55).

$$t_i\left(1-\delta_i\right) \le \overline{T} \quad i \in \mathcal{S} \tag{A.52}$$

$$t_i \ge \overline{T}\delta_i \quad i \in \mathcal{S} \tag{A.53}$$

$$\delta_i \le \sum_{v\in\mathcal{V}} x_{idv} \quad i \in \mathcal{S} \tag{A.54}$$

δ_i 1 if station i is visited after the time horizon $\overline{T}$, 0 otherwise

γ_i 1 if station i is visited, 0 otherwise

$$\gamma_i = \sum_{j \in \mathcal{S}^+} \sum_{v \in \mathcal{V}} x_{ijv} \quad i \in \mathcal{S} \tag{A.55}$$

Further, Constraints (A.56) - (A.59) ensure non-negativity, and Constraints (A.60), and (A.61) are binary constraints for the newly introduced variables.

$$v_i^{S,f} \geq 0 \quad i \in \mathcal{S} \tag{A.56}$$

$$v_i^{C,f} \geq 0 \quad i \in \mathcal{S} \tag{A.57}$$

$$v_i^{S,F} \geq 0 \quad i \in \mathcal{S} \tag{A.58}$$

$$v_i^{C,F} \geq 0 \quad i \in \mathcal{S} \tag{A.59}$$

$$\delta_i \in \{0,1\} \quad i \in \mathcal{S} \tag{A.60}$$

$$\gamma_i \in \{0,1\} \quad i \in \mathcal{S} \tag{A.61}$$

Situation 1: A station is not visited within the time horizon. The total violation is the sum of $v_i^{S,f}$ and $v_i^{C,f}$. The station load at $\overline{T}$ is s_i^C charged bikes and s_i^F flat bikes. Constraints (A.62) and (A.63) capture the violations for stations that are not visited within the time horizon for charged and flat bikes, respectively.

$$\left(-s_i^C + L_i^{CS} + (I_i^{ICR} - I_i^{OCR})\overline{T} + v_i^{S,f} - v_i^{C,f}\right)(1 - \gamma_i) = 0 \quad i \in \mathcal{S} \tag{A.62}$$

$$\left(-s_i^F + L_i^{FS} + I_i^{IFR}\overline{T}\right)(1 - \gamma_i) = 0 \quad i \in \mathcal{S} \tag{A.63}$$

Situation 2: A station is visited within the time horizon. The total violation is the sum of v_i^S, v_i^C, $v_i^{S,f}$ amd $v_i^{C,f}$. Constraints (A.64) and (A.65) capture the violations from the visit until the end of the time horizon for charged and flat bikes, respectively. Note that these constraints do only apply when $\gamma_i = 1$ and $\delta_i = 0$, i.e. the station is visited *before* the time horizon.

$$(-s_i^C + l_i^{CS} + \sum_{v \in \mathcal{V}} \left(q_{iv}^B - q_{iv}^{CCL} + q_{iv}^{CCU}\right) + (I_i^{ICR} - I_i^{OCR})(\overline{T} - t_i) + v_i^{S,f} - v_i^{C,f})(\gamma_i - \delta_i) = 0 \quad i \in \mathcal{S} \tag{A.64}$$

$$\left(-s_i^F + l_i^{FS} - \sum_{v \in \mathcal{V}} \left(q_{iv}^B + q_{iv}^{FCL} - q_{iv}^{FCU}\right) + I_i^{IFR}(\overline{T} - t_i)\right)(\gamma_i - \delta_i) = 0 \quad i \in \mathcal{S} \tag{A.65}$$

Situation 3: A station is visited after the time horizon. Constraints (A.66) and (A.67) capture the violations from the time horizon $\overline{T}$ until the visit. These constraints are only realized when a station visit occurs after the time horizon, as they are multiplied by δ_i.

$$\left(-l_i^{CS} + s_i^C + (I_i^{ICR} - I_i^{OCR})(t_i - \overline{T}) + v_i^{S,F}\right)\delta_i = 0 \quad i \in \mathcal{S} \tag{A.66}$$

$$\left(-l_i^{FS} + s_i^F + I_i^{IFR}(t_i - \overline{T})\right)\delta_i = 0 \quad i \in \mathcal{S} \tag{A.67}$$

The total starvations at a station in situation 3, is different from 0 only if the station is actually flat or empty upon visit, i.e. it does not have any battery bikes in its inventory. Constraints (A.68) ensure this. Similarly, the total congestions is different from 0 only if the station is full upon visit, handled in Constraints (A.69).

$$l_i^{CS}\left(v_i^S - v_i^{S,F}\right) = 0 \quad i \in \mathcal{S} \tag{A.68}$$

$$\left(Q_i^S - l_i^{CS} - l_i^{FS}\right)\left(v_i^C - v_i^{C,F}\right) = 0 \quad i \in \mathcal{S} \tag{A.69}$$

The violations captured in $v_i^{S,F}$ and $v_i^{C,F}$ will be subtracted from the total violations in the objective function. Hence, the model will aim to maximize these variables. Following from this maximization, we need constraints that ensure that the violation $v_i^{S,F}$ and $v_i^{C,F}$ can be greater than zero only if the station visit occurs after $\overline{T}$. This is done by Constraints (A.70) and (A.71).

$$v_i^{S,F}(1 - \delta_i) = 0 \quad i \in \mathcal{S} \tag{A.70}$$

$$v_i^{C,F}(1 - \delta_i) = 0 \quad i \in \mathcal{S} \tag{A.71}$$

Lastly, we need to specify the relationship between the violations after the time horizon and the violations before the station visit occurs. These constraints must be included to ensure that the violations after the time horizon does not take a higher value than the actual violation, ensured in Constraints (A.72) and (A.73).

$$v_i^{S,F}\delta_i \leq v_i^S \quad i \in \mathcal{S} \tag{A.72}$$

$$v_i^{C,F}\delta_i \leq v_i^C \quad i \in \mathcal{S} \tag{A.73}$$

Subobjective 2: Deviations

The objective function also attempts to minimize the total deviation of the system at the end of each time horizon. The deviation is defined as the absolute value between the ideal state, O_i, and current station inventory of charged bikes, s_i^C, both measured at $\overline{T}$. The absolute value of the deviation is captured by the combination of Constraints (A.74) and (A.75).

$$d_i \geq O_i - s_i^C \quad i \in \mathcal{S} \tag{A.74}$$

$$d_i \geq s_i^C - O_i \quad i \in \mathcal{S} \tag{A.75}$$

The total deviation is

$$\sum_{i \in \mathcal{S}} d_i$$

Subobjective 3: Reward

The effect of visiting a station, in terms of reduction in future violations, is not visible until *after* the station is visited. Similarly, deviation is measured at the time horizon, and will not be affected by later visits. Hence, a vehicle is not incentivized to initiate a station visit after the time horizon if the objective function only considers minimization of violations and deviations within the time horizon. This creates a need to introduce reward for station visits after the time horizon, in order to reduce the idle time of a vehicle at the end of the time horizon. Note that the model restricts the number of station visits after the time horizon to maximum one per vehicle.

As we do not impose a time constraint on this last visit, a challenge arises if the desired station to visit after $\overline{T}$ is far away. This is a challenge as it may detain the service vehicle for most of the next time horizon. Consequently, we impose a reward for starting a station visit that takes place after $\overline{T}$, and penalize the driving time to that particular station in the objective function.

We also want to reward moving flat bikes to charging stations. This is both to reduce the need for battery swaps in the system and to favor flat bikes over charged bikes when unloading bikes at charging stations. In the objective function we reward the number of bikes unloaded to the charging stations with a weight W^F.

There are a number of ways to introduce the reward aspect in the objective functions. In this report, we have chosen to give reward based on the number of batteries the vehicles plan to swap at the station they choose to visit. This reward is denoted r_i^B and is weighted by W^B. In addition, a penalty t_v^F, equal to the driving time to the chosen station, is subtracted from the reward function with weight W^T. The complete reward function thus becomes:

$$W^B \sum_{i \in \mathcal{S}} r_i^B - W^T \sum_{v \in \mathcal{V}} t_v^F + W^F \sum_{i \in S^E} q_{iv}^{FCU}$$

Note that this implementation of the reward aspect does not reward depot visits. This is because we only solve the subproblem of the MDP, where the battery inventory at a vehicle at $\overline{T}$ is not examined. The effect of depot rewards should be investigated in future research.

Further, we introduce parameters, variables and constraints to calculate the correct total reward in the subproblem:

Parameters

W^B	Weight for battery swap reward
W^T	Weight for penalty on driving time
W^F	Weight for moving flat bikes to charging stations

Variables

r_i^B	Reward for driving to station i after the time horizon
t_v^F	Driving time to the last planned station visit at $\overline{T}$ for vehicle v

Constraints

Constraints (A.76) secure that the reward r_i^B at station i never takes a higher value than the number of charged bicycles made available at the station i. Furthermore, Constraints

(A.77) ensure that no reward is given for visits that is not a vehicle's last station visit. Finally, the values of t_v^F are captured in Constraints (A.78).

$$r_i^B \leq \sum_{v \in \mathcal{V}} \left(q_{iv}^B - q_{iv}^{CCL} + q_{iv}^{CCU} \right) + Q_i^S \left(1 - \delta_i \right) \quad i \in \mathcal{S} \tag{A.76}$$

$$r_i^B \leq \delta_i Q_i^S \quad i \in \mathcal{S} \tag{A.77}$$

$$t_v^F \geq \sum_{i \in \mathcal{S}} x_{idv} t_i - \overline{T} \quad v \in \mathcal{V} \tag{A.78}$$

Lastly, non-negativity, integer and binary constraints for the variables are presented in Constraints (A.79) to (A.80).

$$r_i^B \geq 0 \quad i \in \mathcal{S} \tag{A.79}$$

$$t_v^F \geq 0 \quad v \in \mathcal{V} \tag{A.80}$$

Final expression for the objective function

The objective function is now formulated in Equation (A.81) with the expressions presented in this chapter inserted.

$$\begin{aligned} \min W^V & \left[\sum_{i \in \mathcal{S}} \left(v_i^S + v_i^C \right) + \sum_{i \in \mathcal{S}} \left(v_i^{S,f} + v_i^{C,f} \right) - \sum_{i \in \mathcal{S}} \left(v_i^{S,F} + v_i^{C,F} \right) \right] \\ & + W^D \left[\sum_{i \in \mathcal{S}} d_i \right] - W^R \left[W^B \sum_{i \in \mathcal{S}} r_i^B - W^T \sum_{v \in \mathcal{V}} t_v^F + W^F \sum_{i \in S^E} q_{iv}^{FCU} \right] \end{aligned} \tag{A.81}$$

Appendix B

UIP-Inspired Alternative Service Vehicle Strategy

To estimate how well the model heuristic performs, the simulator is run with an alternative service vehicle strategy to provide data for comparison. This is done through different handling of vehicle arrivals occurring under the same day of simulation, i. e. replacing the DSBRCP box in Figure 6.1. Algorithm 3 is used to make the decisions a service vehicle faces when arriving at a station. The decisions are inspired by how UIP currently operates. Typically, a service worker tries to maximize the level of assumed beneficial operational activity when visiting a station. The assumption of what is considered beneficial is based on experience and know-how, but we model this by attempting to reach the ideal state. We also assume that a flat battery should always be swapped at a non-charging station if possible, and that no batteries should be swapped at charging stations.

When deciding what station to visit next in a route, the service worker examine a live overview of the system displayed on an iPad. Based on this information, experience, and current vehicle inventory, he decides on the next station. We assume that our presented criticality score is a reasonable approximation of how a service worker would value station visits, which can be seen as our best effort to decide the same. In addition, each vehicle in operation receive a dedicated driving zone with a subset of the stations in the system to evaluate to avoid simultaneous visits at the same station. These subsets contain an equal number of stations and are disjoint with exception of the depot, which is included in all subsets. Following from this, the service vehicles using this strategy make independent decisions that do not consider the operations of other vehicles in the system. Note that when visiting the depot, the next station to visit is the only decision made.

Algorithm 3: UIP-inspired policy for determining decision variables

```
Data: $t^{current}$:=current time, $T^H$:=unit handling time, $T^P$:=parking time,
      $T^D$:=next station driving time, current station and vehicle inventory levels
Result: $q^B$:=battery swaps, $q^{CCU}$:=charged bike unload, $q^{CCL}$:=charged bike
        load, $q^{FCU}$:=flat bike unload, $q^{FCL}$:=flat bike load, $j$:= next station,
        $t$:=next station arrival time
1  $q^B$ = min(vehicle batteries, station flat bikes);
2  if current station has more charged bikes than ideal state then
3      $q^{CCL}$ = max(0, min(charged bikes at station, available vehicle bike capacity,
         charged bike difference to ideal state)) ;
4      $q^{CCU}$ = 0 ;
5  end
6  else
7      $q^{CCL}$ = 0 ;
8      $q^{CCU}$ = max(0, min(charged bikes at vehicle, available parking at station,
         charged bike difference to ideal state));
9  end
10 if current station is charging station then
11     $q^{FCU}$ = min(flat bikes on vehicle, available parking at station);
12     $q^{FCL}$ = 0 ;
13 end
14 else
15     $q^{FCU}$ = 0 ;
16     $q^{FCL}$ = min(available vehicle bike capacity, station flat bikes);
17 end
18 $j$ = station in dedicated zone with highest receiving criticality score ;
19 $t = t^{current} + (q^B + q^{CCU} + q^{CCL} + q^{FCU} + q^{FCL}) \times T^H + T^D + T^P$
```