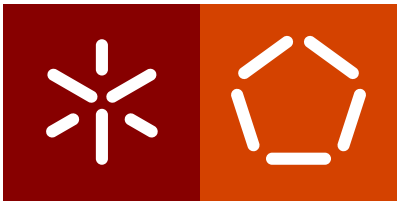
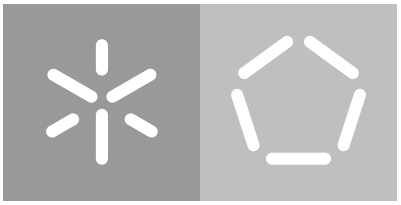


University of Minho
School of Engineering
Department of Information Systems

Guilherme Jesus Sousa Fernandes

A Decision Support System for the Management of Smart Mobility Services

January 2020



University of Minho
School of Engineering
Department of Information Systems

Guilherme Jesus Sousa Fernandes

A Decision Support System for the Management of Smart Mobility Services

Master Dissertation
Master Degree in Engineering and Management of Information Systems

Supervisors
Professor Dr. Paulo Cortez
Dr. Nuno Oliveira

January 2020

ACKNOWLEDGEMENTS

This dissertation represents the culmination of 5 years of learning and continuous growth, widely reflected in this work. For this to take place, I always had the support of several people to whom I address my special gratitude:

To Professor Dr. Paulo Cortez and Dr. Nuno Oliveira, for the generosity and trust when they guided me throughout this journey, and, above all, for the time spent in their orientation, which, despite the full schedule, was reflected in enriching moments and great learning.

To my mother and brother, for the unconditional love and affection at all times. Thank you for your knowledge, for your confidence in me at the most difficult times, for your support in the hours of happiness and sadness that the writing of this thesis brought.

To Bernardo, Felipe, João and Tiago, my closest university friends, for their friendship, companionship and help. Thank you for making this journey easier and for your help at all times, for the pleasure of sharing with you the joys and sorrows. To all my friends and colleagues who made this journey memorable: thank you for all the unforgettable moments.

Thanks to my friends, from times when youth and innocence ruled life, Alexandre, Miguel, and Rui. Thank you for your patience and understanding in my absence. Thank you for your kindness, trust, and courage that you gave me to pursue and strive for my goals.

To Daniel, my longtime friend who closely followed this journey, always with a friendly word to give me. For all the fellowship, friendship and patience, thank you.

To my co-worker Pedro for the tips, advice and willingness to help.

To Afonsina, for the second family that always accompanied me along this journey. Thank you for the companionship, friendship and alliance that always brightened my days and that will surely accompany me for the rest of my life.

Finally, I would like to share the merits of this achievement with you, Joana. Because you have always walked by my side, been my safety net and best friend; half is yours and half is mine... half is you and half is me. You are the right probability in this Russian roulette life. As James LaBrie said in that song: 'Never in my dreams could I deserve to ever see a vision quite like her; then unexpectedly I'm taken by surprise; an angel just appeared before my eyes'. You really are my Arkenstone.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

RESUMO

Nos dias que correm, a mobilidade assume especial importância no quotidiano das áreas metropolitanas em crescimento no país. . Com o notório crescimento das cidades, torna-se necessária e urgente uma transformação dos costumes e formas de mobilidade dentro das áreas urbanas, alterando as realidades aparentes que hoje conhecemos. Inseridos numa sociedade cada vez mais consciencializada e alerta para as questões ambientais, é essencial transportar esta mentalidade renovada para a resolução das problemáticas citadinas. Assim, o conceito de "Cidade Verde" levanta uma série de questões que exigem uma resposta eficaz para o bem-estar dos seus habitantes.

Por entre as várias soluções apresentadas para estas patologias, uma das mais promissoras é, sem dúvida, o sistema de mobilidade partilhada. Pela sua dimensão, é pertinente expor o caso prático da cidade de Barcelona, em Espanha, explorando o seu sistema de partilha de scooters, um meio que adquire especial importância como meio de transporte urbano. Como qualquer sistema em constante aprimoramento, procura-se uma solução para a problemática da variação de procura, que apresenta oscilações constantes, tanto a nível temporal como geográfico, resultando na falta de veículos em algumas áreas e excesso noutras. Assim sendo, o rebalanceamento do sistema torna-se crucial para uma possível maximização na utilização de veículos, satisfazendo a procura e potenciando um aumento da sua utilização.

No correr desta dissertação, foram estudados e utilizados vários métodos de otimização moderna (metaheurísticas) para a procura de soluções (sub)ótimas para o(s) percurso(s) a percorrer pelo(s) veículo(s) que executam a redistribuição das scooter/bicicletas pelas diversas áreas abrangidas pelo sistema de partilha. Deste modo, foi desenvolvido um sistema de apoio à decisão para satisfazer estas necessidades, garantindo ao utilizador toda a informação relevante para um trabalho mais eficiente e preciso.

Palavras-Chave: Sistemas de Mobilidade Partilhada, Problema de Rebalancemanto de Bike Sharing, Sistemas de Apoio à Decisão, Otimização Moderna, Mobilidade como um Serviço.

ABSTRACT

Nowadays, mobility is especially important in the daily life of the country growing metropolitan areas. With the increasing influx of people and development of these large cities, the reality of mobility that we know becomes increasingly unsustainable. Along with mobility, the environmental concerns are one of the main topics of discussion worldwide and the population is starting to act and change the way they live to find a more "green" and sustainable way of doing it.

Several proposals have been put forward, trying to mitigate this issue and, one of the most promising is, undoubtedly, shared mobility systems. In this case study will be addressed the Barcelona scooter sharing system, characterized by its great size and importance as a mean of urban transport. One of the problems presented by these sharing services is that demand varies widely, both temporal and geographical. Thus, there are several cases where there is a lack of vehicles in some areas and an excess in others. The rebalancing of the system is crucial to maximize vehicle utilization and meet customer demand.

In this thesis, several modern optimization methods (metaheuristics) were used to search for (sub)optimal solutions for the redistribution route(s). A decision support system was developed to meet this end, giving the end user relevant information for a more efficient and precise work.

Keywords: Modern Optimization, Metaheuristics, Machine Learning, Neuroevolution, Online Learning, Mobile Advertising, Performance-based Advertising

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ACO Ant Colony Optimization
ANN Artificial Neural Networks

B&B Branch-and-bound
B&C Branch-and-cut
BRP Bike Sharing Rebalancing Problem
BSS Bike Sharing System(s)

DBMS Database Managements System
DSS Decision Support Systems

EA Evolutionary Algorithms

FPT Full-Port Time

GA Genetic Algorithms
GUI Graphical User Interface

HC Hill Climbing

IP Integer Programming

LP Linear Programming

MaaS Mobility-as-a-Service
MBMS Model Base Managements System
MILP Mixed Integer Linear Programming
MO Modern Optimization

NP Nondeterministic Polynomial-time
NR Number of Relocations

OTS Optimization-trend-simulation

PBS Population-Based Search
PSO Particle Swarm Optimization

SA Simulated Annealing
SM Smart Mobility

TS Tabu Search
TSP Traveling Salesman Problem

VRP Vehicle Routing Problem
VSS Vehicle Sharing Systems

ZVT Zero-Vehicle Time

1

INTRODUCTION

1.1 PROBLEM DESCRIPTION AND PROPOSED SOLUTION

This project was born due to a concern of CEiiA, a Centre of Engineering and Product Development that designs, develops and operates innovative products in the mobility industry. CEiiA is fully aware of the exponential growth of urban areas and their difficulties to maintain a fluid traffic jam and offer a smart and sustainable mobility paradigm to its dwellers. Along with mobility, the environmental concerns are one of the main topics of discussion worldwide and the population is starting to act and change the way they live to find a more "green" and sustainable way of doing it. Consequently, CEiiA saw an opportunity to make a difference and offered sharing services to the citizens of these overflowing urban areas.

These services offer a mobility solution whereby vehicles (cars, bicycles, scooters, etc.) located at different stations across the urban areas, are available for shared use. These systems contribute to a more sustainable mobility in these areas, decreasing traffic and pollution caused by individual transportation. In these systems (sharing services), the demand is really dynamic and changes quickly, both at temporal and geographic level. There are many scenarios where there is a lack of vehicles in certain areas and an excess in others.

A few challenges arise when setting up these systems to operate efficiently. Consequently, the decision makers need some sort of aid to give strength to their assumptions or give them some information that they were not taking into consideration. More than that, they need to optimize the way the system is currently working. With this in mind, this project comes to light, where the goal is to implement a decision support system to assist the management of smart mobility services.

"If we have some concern about 'sustainability' we need to anticipate what effects our use might have on future generations - and we have some clear indicators that there's a problem."

— (Allwood and Cullen, 2012)

1.2 EXPECTED RESULTS

The main goal of this project is to develop a decision support system that optimizes the relocation operations in CEiiA's vehicle sharing systems. Through modern optimization techniques, the operational repositioning of the vehicles will be optimized.

The system should:

- deliver the route(s) that the relocation vehicle(s) should take to rebalance the system;
- detail the outcome (distance and time) to the user; and
- adapt to new information (constraints and assets).

1.3 BIBLIOGRAPHIC RESEARCH STRATEGY

Every scientific development starts with a proper scientific research, in which the developer aims to learn about fundamental concepts of the research theme and discover its state of the art. Therefore, the researcher needs to follow a search methodology to find the documentation to support the outcome.

To that end, the investigation process initiated with the investigation of the most important concepts and related keywords and then went to more specific and practical case studies. A few search platforms were defined as essential to this step: *Google Scholar*, *Web of Science*, *ScienceDirect* and *ResearchGate*.

Aside from the platforms, the keywords choice to unveil the most relevant documents was also truly important. "Vehicle Routing Problem (VRP)" was the first description of the project that was found. Since that early definition, others started to appear, as "Vehicle Sharing Systems (VSS)" or "Bike Sharing Rebalancing Problem (BRP)". All of these are important and slightly different concepts. The foundation concepts of this project were also key-concepts to be defined and learned: "Decision Support Systems (DSS)", "Smart Mobility (SM)", "Mobility-as-a-Service (MaaS)" and "Modern Optimization (MO)".

As expected, several irrelevant articles and books resulted from the search queries. Thus, a filtering criterion was adopted, as shown in Figure 1, to select the documents to build the groundwork for this project. All selected works were manually inspected, in terms of their titles and abstracts.

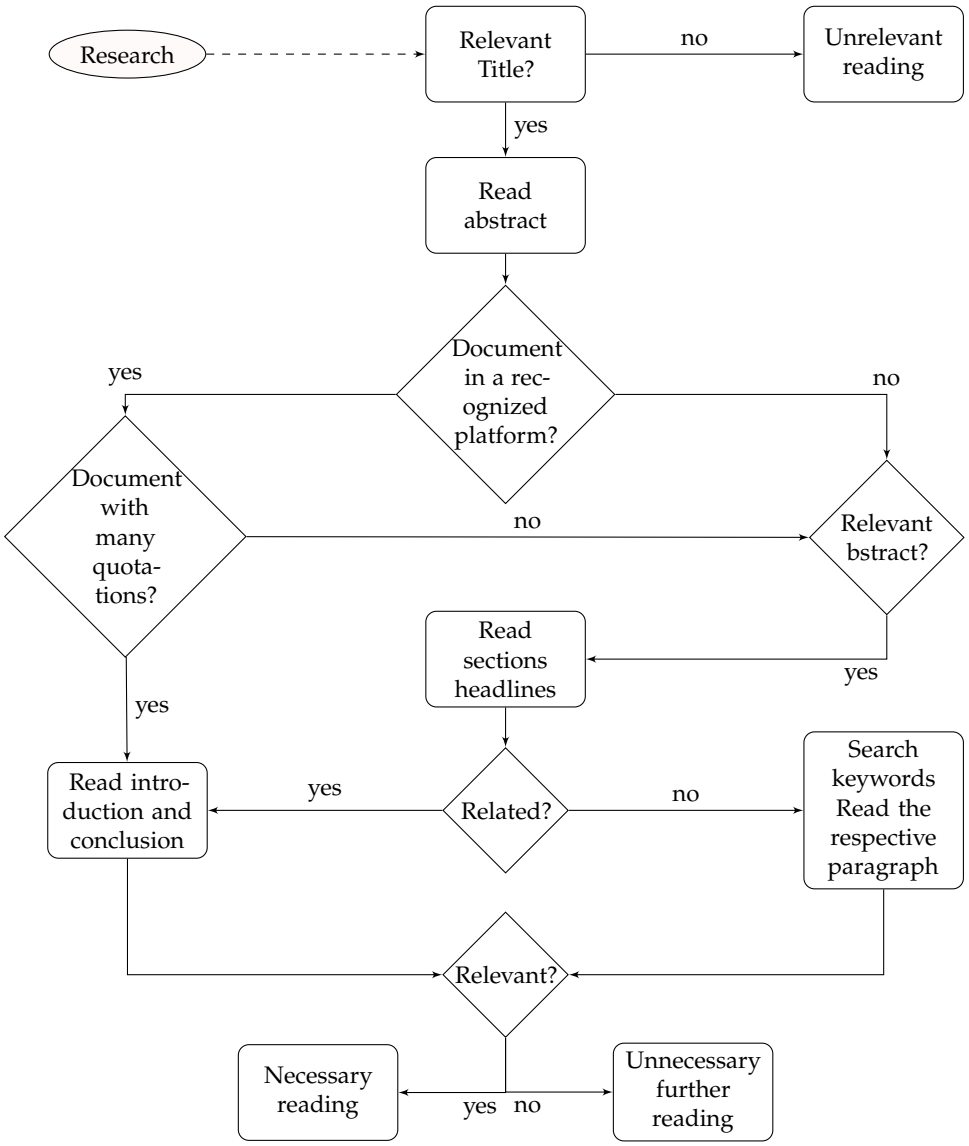


Figure 1: Literature selection process

1.4 DOCUMENT OUTLINE

This document is organized in six chapters which are structured as follows: Chapter 1 is the introduction where the problem is described and a solution is proposed; expected results are outlined and the bibliographic research strategy is depicted. The literature review is presented in Chapter 2. It has six subchapters: Decision Support Systems; Smart Mobility; Optimization Approaches; Decision Support Systems for Dynamic Vehicle Relocation; Optimization of Dynamic Vehicle Relocation. Decision Support Systems subchapter approaches the basic concepts of DSS; Smart Mobility covers the concepts of Smart Mobility, Mobility-as-a-Service, Operational Repositioning of Vehicles and the well-known Bike Sharing Rebalancing Problem. Optimization Approaches presents the main differences between the operational research and modern optimization methodologies, detailing some of the most popular algorithms used in the area. Decision Support Systems for Dynamic Vehicle Relocation document some of the state of the art DSS developments in vehicle routing problems, with particular attention to vehicle relocation cases. Optimization of Dynamic Vehicle Relocation subchapter states some of the most common algorithms in the area.

Then, Chapter 3 formalizes the problem and specifies the environment that underpins the system development. In Chapter 4 the implementation construction and the flow of the system are detailed, presenting a step by step design. The execution of the algorithms and their behavior in the search for the sub(optimal) solution is also specified. Chapter 5 demonstrates and discusses the experimental results and Chapter 6 summarizes the system conception with closing statements and concludes with recommendations of potential future work.

2

LITERATURE REVIEW

This chapter covers the theoretical background associated with this thesis: "A Decision Support System for the Management of Smart Mobility Services". The first section (Section 2.1) provides an overview of what a Decision Support System is, and which benefits business managers can expect with his adoption. The next section (Section 2.2)– covers the rising interest of Mobility-as-a-Service in urban areas. Then, Section 2.3 describes the most technical aspects of the development of the project itself, presenting an introduction of modern optimization concepts and some of the algorithms used in operational research ands metaheuristics areas, within this project context. Sections 2.4 and 2.5 we address some real cases examples, proving a practical perception of success cases to be considered when designing the development phase.

2.1 DECISION SUPPORT SYSTEMS

This section introduces the concept of Decision Support Systems (DSS), presenting definitions and contextualization about the project. It was strongly influenced by the book *Decision Support and Business Intelligence Systems, 9th edition* (Turban et al., 2010), a reference book in the area. We note that the term DSS is used in this work to refer to both its plural and singular forms.

2.1.1 *Concept*

Decision Support Systems (DSS) does not have a universally accepted definition among the experts of the area for it is a content-free expression[1]. However, we are going to see some of the early definitions and relevant concepts that characterize it.

In the early '70s, Scott Mortin defined DSS as an "interactive computer-based systems which help decision makers utilize data and models to solve unstructured problems". A few years later Keen and Scott-Morton (1978) came with another classic DSS definition that is still employed nowadays: "Decision support systems couple the intellectual resources

1 Has different meaning to different people.

of individuals with the capabilities of the computer to improve the quality of decisions. It is a computer-based support system for management decision makers who deal with semistructured problems."

By these definitions, we can perceive a DSS as a computer-based system intended to support managers in a decision situation. It should be seen as a way to extend decision makers capabilities and never as a machine with the intent to replace people's judgment.

2.1.2 *DSS as an Umbrella Term*

As previously said, DSS definitions are open to several interpretations. DSS can also be used as an umbrella term. By this characterization, DSS encompasses separate support systems within an organization, like marketing, finance, accounting, production, and many other systems.

2.1.3 *DSS as a Specific Application*

Although DSS is usually applied to refer the umbrella term, some use it to refer to the application itself. Most of the times, DSS are developed to evaluate opportunities or and support the solution of a specific problem or set of them.

Usually, DSS have their own databases to support the storage of the data; provide a straightforward user interface; support all decision-making phases and can be used by a single user on a personal computer (PC) or by many on a Web-based interface.

2.1.4 *The Architecture of DSS*

Typically deployed online, DSS have three main components: data, models and a user interface. There is still a fourth, which is an optional one - Knowledge. The individuality of the support provided, and overall capabilities of the system are defined by the way these components are assembled (typically via internet). This suggests that a DSS application can incorporate a data management subsystem, a model management subsystem, a user interface subsystem, and a knowledge-based management subsystem.

Data Management Subsystem All the relevant data for the DSS to operate is stored in a database and managed by Database Managements System (DBMS) software.

Model Management Subsystem

This subsystem is a software package which contains statistical, financial and other quantitative modelsthat provide analytical capabilities and appropriate software management. This software is generally called a Model Base Managements System (MBMS)

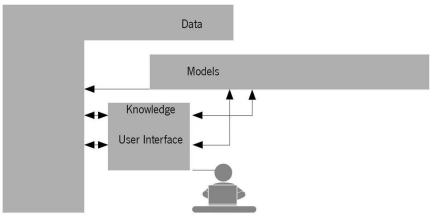


Figure 2: High-level architecture of a DSS, adapted from (Turban et al., 2010)

User Interface Subsystem

The final user is also considered a part of the system. He communicates with the DSS through the user interface subsystem. Most of Decision Support Systems provide a Graphical User Interface (GUI) over a *Web Browser*.

Knowledge-Based Management Subsystem

A DSS must include the three components described until now - DBMS, MBMS, and GUI. The *knowledge-based management subsystem* is nonmandatory, but it can grant intelligence and other benefits to the system. This subsystem can act as an independent component of the system or support the other three. This knowledge is usually granted by *Web Servers*.

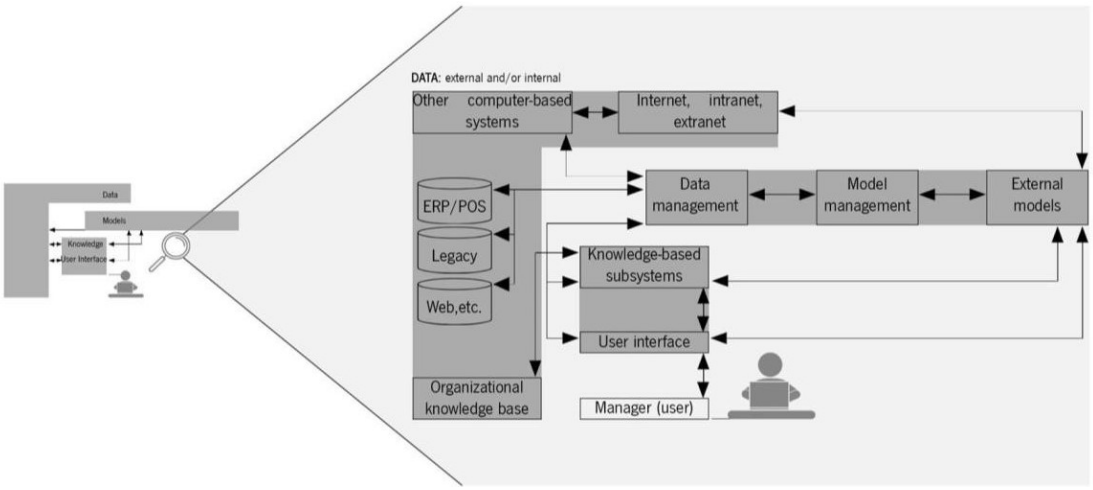


Figure 3: Schematic view of a DSS, adapted from (Turban et al., 2010)

2.1.5 *Decision Support System Classification*

The expected outcome of this project is a DSS that, with the proper input, should give the best answer to the final user. Therefore, the emphasis of this system will be optimization. This purpose fits in the *Model-Driven* DSS classification. The focus of such systems lays on optimizing one or more objectives, like increase profit, reduce costs, etc.

2.1.6 *The Decision Support System User*

This project intends to improve Decision Support Systems of CEiiA's sharing services by proposing better routes for their maintenance operations. The main objective is to provide DSS users with real-time information that permits more efficient maintenance operations; not only for the rebalancing of the system but also for the preservation of the vehicles functionality (e.g., battery change).

2.2 SMART MOBILITY

Smart Mobility is a new concept that came in the last few years with the growth of *internet of things* applications and other emerging technologies. Marked by the promise of environmental change and new mobility solutions, this key component to urban transportations change aims to reduce car use and related problems, like traffic congestion, crashes, poor air quality and more (Docherty, 2018).

Private vehicle ownership is still the solution adopted by the great majority of the population to address their everyday mobility purposes. A massive adoption of the Mobility-as-a-Service (MaaS) paradigm requires a mindset shift from vehicle ownership to vehicle 'usership' (Wockatz and Schartau, 2015). *Smart Mobility* acceptance and implementation can change the way we live in society, avoiding much of the waste, pollution and environmental degradation (Docherty, 2018).

Nowadays, we can already see the numbers and the change associated with this shift. In 2014, car sharing systems had almost 5 million members and 92000 vehicles worldwide (Vine et al., 2014). The International Transport Forum (ITF-CPB, 2017) went even further when predicted that an integrated system of on-demand *taxis* and *taxi buses* in a rail system in Lisbon, Portugal could achieve a 44% reduction in peak vehicle kilometres and 53% reduction in CO_2 emissions in the city, and release fully 95% of parking spaces for other public uses.

Hietanen (2014) sets out his view of future mobility as seeing "the whole transport sector as a co-operative, interconnected ecosystem, providing services reflecting the needs of customers". The boundaries between different transport modes are blurred or disappeared completely. The ecosystem consists of transport infrastructure, transportation services, transport information and payment services.

2.2.1 *Mobility as a Service*

Mobility-as-a-Service, or MaaS, is a recent concept in the *Smart Mobility* arena (Pangbourne et al., 2018) that presents itself as being a new business model for delivering sustainable transport services (Signorile et al., 2018). As all "as-a-service" instances, Mobility-as-a-Service assures a personalized use of a bundle of transportation means such as taxis, buses, cars or bikes for a certain amount of time. Mobility-as-a-Service (MaaS) focuses on addressing the needs of the everyday person, providing a set of trip options with different modes, time and cost (Mulley, 2017).

According to Bude (2016), to work properly, MaaS requires:

- widespread penetration of smartphones on $3G/4G/5G$ networks;
- secure, dynamic, up-to-date information on travel options, schedules, and updates;
- high levels of connectivity; and
- cashless payment systems.

To enable these conditions, different actors need to cooperate (Goodall et al., 2017):

- mobility management players;
- telephone companies;
- payment processors;
- public and private transportation providers; and
- local authorities with responsibility for transportation and city planning.

Mulley (2017) emphasizes that the conditions to make this change already exist. He says that the millennials generation is the most predisposed segment of the population to accept this new paradigm of mobility, due to their level of acceptance of technology, great flexibility and adaptation capabilities.

2.2.2 *Vehicle Sharing Systems*

Vehicle Sharing Systems are an application of Mobility-as-a-Service, where vehicles such as bicycles, low emission cars, and electric vehicles (Nair and Miller-Hooks, 2011), are strategically located in stations across the urban transportation network and represent an emerging transportation scheme.

For example, users of car sharing systems can rely on a service that provides the comfort and convenience of private automobiles without having to support their buying cost, insurance and maintenance requirements, fuel expenses or parking headaches (Mitchell, 2008). Generally, there are three types of car sharing systems implementations (Ait-Ouahmed et al., 2017):

Stations-based car sharing There are two types of stations-based systems:

1. *Round trip car sharing* where users must bring the cars back to the departure station;
2. *One-way car sharing* where users can park the car at a chosen station, among all the available.

Free-float car sharing This service operates without stations. Users can pick up and return a vehicle anywhere within a defined area.

In light of the previous example, bike sharing systems offer a one-way approach where public bicycles, located at different stations across a specific region, are available for shared use (Dell'Amico et al., 2014). Customers can pickup a vehicle at any location and not necessarily return it to the departure one.

This project is based on a one-way sharing system. It offers an enormous advantage for the customers due to the commodity to the user, once he can return the bike to a area/station close to his destination. This approach tends to lead to an unbalanced system, where some stations are fully occupied or congested (more than required) and others are empty or with fewer vehicles than required. Hence, the need for a balanced system urges and a redistribution of vehicles among stations is required.

2.2.3 *Operational Repositioning of Vehicles*

The design and management of vehicle sharing systems is a tough and meticulous exercise that raise several optimization problems and design decisions such as the location, number and capacity of the stations; vehicles fleet size; and others.

The criteria of success of these systems is the guarantee of vehicles availability, so every decision should be made to ensure enough coverage (i.e., satisfy user demand with appropriate quality of service) while minimizing investments and maintenance costs (Gavalas et al., 2016). These decisions will not be covered in this project, being the responsibility of CEiiA's management board. Then again, the strategy adopted to verify the demand needed for each day and in which hour must be discussed in this report.

2.2.3.1 *Predicted Demand*

Until now we have learned that, due to spatio-temporal variation of requests, vehicle relocation is important to enhance the performance of the system. Having to adapt to user demand dynamics, vehicle relocation activities are typically needed several times on the course of a day. Thus, relocation decisions are bound, not only to spatial constraints but also temporal (Gavalas et al., 2016).

Various authors and researchers have different perspectives on how to address these problems. For example, Froehlicj et al. (2008) predicts the user demand for available bicycles in a certain station at a certain time. O'Mahony (2015) predicts the service level requirements for rush hours through the analysis of historical data. For this practical case, an approach was chosen and implemented: *Data Mining* approach.

A *Data Mining* approach (Vogel et al., 2011), through data mining methods and geographical information technology, grant insight into the complex vehicle (car, bicycle) activity patterns at stations. A proper design and management of the system based on historical data and demand prediction should enhance relocation strategies.

2.2.4 *Traveling Salesman Problem*

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization (Cortez, 2014) and can be stated as follows: "A salesman is required to visit each of the n given cities once and only once, starting from any city and returning to the original place of departure. What route, or tour, should he choose in order to minimize the total distance traveled?" (Lin, 1965). The goal is to find the cheapest way of visiting all cities of a given map (Cortez, 2014), with the cost being *distance*, *time* or *expense* (Lin, 1965).

2.2.5 *Bike Sharing Rebalancing Problem*

A system like vehicle sharing has costs that can fluctuate intensely, depending on a group of variables, like the population density of the area, the fleet size or the system itself. The repositioning task is usually assigned to capacitated vehicles based on a central depot that transfer vehicles from congested stations to ones with fewer vehicles than required, rebalancing the system. The literature refers to the resulting optimization of which route the vehicles should take, to perform the redistribution at minimum cost, as Bike Sharing Rebalancing Problem (BRP) (Dell'Amico et al., 2014). There are two ways to address the Bike Sharing Rebalancing Problem - a *Static* and a *Dynamic one*.

2.2.5.1 *Static Approach*

Also known as *Single-Periodic Rebalancing*, this version only allows the repositioning of the vehicles when users cannot act on them during the process (Chemla et al., 2013) – during the night in most cases. The level of occupation at the stations is measured once and then the redistribution is planned (Dell'Amico et al., 2014). Static approaches, however, may not be enough to avoid network failures during the day (Chiariotti et al., 2018) .

2.2.5.2 *Dynamic Approach*

Also known as *Multi-Periodic Rebalancing*, is the opposite of the static approach. In other words, the real-time usage of the system is considered, and the redistribution plan can be updated as soon as the information required to make decisions is revealed (Dell'Amico et al., 2014). This suggests that dynamic rebalancing aims at redistributing bikes throughout the day according to the current network state (Chiariotti et al., 2018) .

2.3 OPTIMIZATION APPROACHES

The concept of optimization is a principle underlying the analysis of complex decision problems (Luenberger and Ye, 2008). When addressing a complex decision problem, we focus on an objective. That same objective can be maximized or minimized (depending on the problem) and have several constraints that can hinder the search of the solution.

Decision problems can have truly different complexities. In this project, we are going to address a decision problem that exhibits a major complexity level. The *Static* BRP is NP-Hard (Schuijbroek et al., 2017). The *Dynamic* variant has a higher level of complexity so is, at least, NP-Hard (Ghosh et al., 2017).

NP-HARD

Nondeterministic Polynomial-time (NP) (Moura, 2006) is a complexity class used to classify decision problems. NP is the collection of decision problems for which the problem instances are verifiable in polynomial time[2] (Kleinberg and Tardos, 2006). There have been numerous efforts to find polynomial time algorithms for problems in NP.

NP-hard problems are the ones which are, at least, as hard as the hardest problems in NP. In real practical applications, potentially suboptimal solutions might be determined in polynomial time (instead of looking for optimal ones, consuming computational resources). Some algorithms are following presented.

2.3.1 *Operational Research Methods*

As the name suggests, operational research[3] involves 'research on operations' and is applied to problems related to operations within an organization. It has been practiced in several areas such as transportation, manufacturing, public services[4].

Usually, operational research tries to find the best (optimal) solution to the problem, identifying the best course of action. To do this, it uses systematic solution procedures, famously known as algorithms. To clarify, we are going to see some of those algorithms, with special attention to those used in previous work, related to vehicle routing and rebalancing problems. The objective is not to give you a full understanding of each algorithm but to describe in global terms how operational research tends to approach real-world problems.

2 *polynomial time* is an efficiency measure of an algorithm. It refers to how quickly the number of operations needed by an algorithm grows.
3 Also known as Operations Research.
4 This section was strongly influenced by Hillier and Liberman (2014).

2.3.1.1 *Linear Programming*

Before we proceed to the algorithms, there is a concept that we need to clarify - Linear Programming (LP). LP uses a mathematical model to represent the problem of interest. *Linear* indicate that all the mathematical functions in the model must be linear functions[5]. *Programming* is a synonym for planning. Therefore, LP suggests the planning of activities to reach an optimal result, among all feasible alternatives.

2.3.1.2 *Integer Programming*

Integer Programming (IP) designates a *Linear Problem* that requires the variables to be integers; it is usually necessary to assign people, machines, and vehicles to activities in integer quantities. If only some of the variables are expected to have integer values, the model is referred to as Mixed Integer Linear Programming (MILP)[6].

2.3.1.3 *Branch-and-bound*

In a pure IP problem, only a few solutions can be listed as feasible. The most popular way to find optimal solution while working with IP algorithms is the Branch-and-bound (B&B) technique (to identify feasible integer solutions). Considering the original problem to be too heavy to be solved in a reasonable amount of time, the same is divided into smaller 'subproblems' until this assumption is no longer applicable.

2.3.1.4 *Branch-and-cut*

The very first algorithms developed for IP were based on cutting planes but, in practice, the outcome was not the expected. Over time, the combination of cutting planes and Branch-and-bound[7] algorithms proved to be a powerful algorithm approach to solve large-scale IP problems. And so, the Branch-and-cut (B&C) algorithm was born.

2.3.1.5 *Nonlinear Programming*

The assumption that the real-world problems are all linear does not hold. Many economists say that in economic planning problems, linearity is an exception. In the last decade, the modeling of nonlinear problems grew substantially. So, nonlinear programming uses a mathematical model to represent the problem of interest and not all the mathematical functions in the model are linear.

5 A function whose graph is a straight line - a polynomial function of degree zero or one (Gel'fand, 1989).
6 When all variables are expected to have integer values, the model is referred to as *pure integer programming*.
7 B&C is a B&B technique, where search space is reduced by adding new constraints (cuts) - (Vaira, 2014).

2.3.2 *Modern Optimization*

The discipline of Operational Research produced many techniques to address multiple real-world problems. These optimization techniques are often referred to as 'classic' techniques (Michalewicz, 2007). On the other hand, Modern Optimization[8] (Michalewicz et al., 2006) methods are appropriate to a wide range of distinct problems, with no need for extensive domain knowledge, or problems for which no specific optimization algorithm has been developed, in contrast with these classical techniques (Cortez, 2014). Another differentiating aspect is that Modern Optimization does not assure that the best possible solution (optimal) is always found but, instead, with reasonable use of computational resources, a high-quality solution (sub-optimal) is determined.

Once the objective of the optimization[9] is clear and defined, only two details need to be specified: The **representation of the solution** and the **evaluation function**. The representation of the solution is a predominant specification. It will determine the search space and its scope and, if we do not select the correct domain, in the beginning, we might end up digging holes in a territory with no gold. The next thing to do is design an evaluation function that allows the algorithms to evaluate how good a solution is and compare the quality of different ones (e.g., Formalization 1).

A general constrained optimization problem can be formally stated as follows (Rao, 2009):

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f(X) \\ \text{subject to} \quad & g_j(X) \leq 0, \quad j = 1, \ldots, m \\ & h_j(X) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{1}$$

where X is an n-dimensional vector called the design vector, $f(X)$ is the objective function and $g(X)$ and $h(X)$ are constraints. The components of the X vector are the design or decision variables x_i, $i = 1, 2, \ldots, n$. Finally, the set of all possible solutions define the search space.

After clarifying the core aspects of modern optimization, their most popular algorithms (related to this project) are next detailed. The blueprint of the algorithms was based on Cortez (2014) and Blum and Roli (2003).

8 Also known as "modern heuristics" (Michalewicz and Fogel, 2004) and "metaheuristics" (Luke, 2013)
9 Minimization/maximization expression, regarding the domain constraints.

2.3.2.1 *Local Search*

Also known as *single-state* search, local search optimization algorithms concentrate in a local neighborhood of a given initial solution, generating new solutions from current ones. All local optimization techniques use iterative improvement (Michalewicz, 2007).

Hill Climbing

Hill Climbing (HC) is a local optimization technique that hikes up and down a hill until a local optimum is obtained. It starts by considering a single current solution in the search space and comparing it to a new one from its neighborhood. If that solution has a better fit to the problem than the current one, this one starts to be considered the best solution and the former is forgotten. Otherwise, nothing happens and another neighbor is selected, repeating the procedure. The procedure ends when no further improvements are feasible, or the designated time runs out (Michalewicz, 2007).

Algorithm 1 Hill Climbing

Input: s (initial solution), f (evaluation function), ... (other parameters)
  while termination conditions not met **do** ▷ E.g. maximum number of iterations
    $s' \leftarrow change(s, ...)$ ▷ Pick a solution from s neighborhood
    $s \leftarrow best(s, s', f)$ ▷ Select best solution for next iteration
  end while
Output: s

Simulated Annealing

In the '80s, a variation of the Hill Climbing was proposed: Simulated Annealing (SA). Inspired by the annealing phenomenon of metallurgy, SA considers a control parameter T (commonly referred to as temperature) to decide whether to accept an inferior solution or not[10]. The parameter T starts the execution with high values (randomizing the search) and then, gradually decreases. Towards the end, the T value is quite low, making the algorithm behave like Hill Climbing.

Algorithm 2 Simulated Annealing

Input: s (initial solution), T (initial temperature), f (evaluation function), ... (other parameters)
  while termination conditions not met **do**
    for i iterations **do** ▷ Iterate i times for each T
      $s' \leftarrow change(s, ...)$ ▷ Pick a solution from s neighborhood
      if $s' = best(s, s', f)$ **then**
        $s \leftarrow s'$ ▷ Select best solution for next iteration
      else
        $s \leftarrow accept(T, s', s)$ ▷ accept worse solution
      end if
    end for
    $T \leftarrow cooldown(\mathrm{T})$ ▷ Cooling schedule
  end while
Output: s

10 The higher the temperature (T value), the higher the probability to accept an inadequate solution is.

Tabu search

Tabu Search (TS) is a variation of Hill Climbing (Cortez, 2014). Its purpose is simple - prevent an iteration of the search that had already been performed (during a specified amount of iterations). The way that this happens is also simple. The algorithm includes a 'memory' (tabu list of length L) that forces the search to move to unexplored search space, preventing it from being stuck in a local optimum. The memory stores the most recent solutions and does not let the algorithm 'visit' them in the next iterations, avoiding unnecessary searches. After the number of iterations surpasses the length L of the memory, the positions turn, once again, available.

Algorithm 3 Tabu Search

Input: s (initial solution), f (evaluation function), ... (other parameters)
  $list \leftarrow \{\}$ ▷ Tabu list
  while termination conditions not met **do**
    $s' \leftarrow change(s, list, ...)$ ▷ Pick a solution from s neighborhood, not in $list$
    if $s' = best(s, s', f)$ **then**
      $update(list, s)$ ▷ Update $list$
      $s \leftarrow s'$ ▷ Select best solution for next iteration
    else
      $update(list, s')$ ▷ Update $list$
    end if
  end while
Output: s

2.3.2.2 *Population-Based Search*

Another interesting class of search methods is the Population-Based Search (PBS). Instead of using a single search point, PBS uses a set of feasible solutions. Because of this, these methods tend to avoid being stuck at local minimums by easily locating worthy regions of the search space (Michalewicz and Fogel, 2004).The usage of more computational power can be justified by the diversity of solutions that can be found, using these methods. Most population-based methods steal concepts from biology (Luke, 2013).

Genetic and Evolutionary Algorithms

Based on evolution theory, Genetic Algorithms (GA)[11] follow the principle in which only the fittest entities survive. More recently, the term Evolutionary Algorithms (EA) was adopted to address genetic algorithm variants, which include real value representations and flexible genetic operators (Michalewicz, 1996). In a simple way, following the biological terminology associated (Luke, 2013), the behavior of the algorithms goes something like this: It starts with an initial generation of candidate solutions and then generates new ones (*breeding*) through *genetic operators* like *crossover* and *mutation*. *Crossover* generates children through a combination of two or more parent solutions, while *mutation* performs a small change to an individual (Cortez, 2014).

Algorithm 4 Evolutionary Algorithms

Input: f (evaluation function), ... (other parameters)
  $P \leftarrow initialize(\ldots)$ ▷ Random (or not) initial population
  while termination conditions not met **do**
    $eval(P)$ ▷ Evaluate P individuals
    $p \leftarrow select\_parents(P)$ ▷ Select a set of parents from P
    $p' \leftarrow crossover(p)$ ▷ Create offspring
    $p'' \leftarrow mutation(p')$ ▷ Apply mutations to some children
    $eval(p'')$ ▷ Evaluate new individuals
    $P \leftarrow select(p \cup p'')$ ▷ Set next population
  end while
Output: P

11 Proposed John Holland at the University of Michigan in the 1970s (Luke, 2013).

Swarm Intelligence

With a family of algorithms like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), *Swarm Intelligence* is inspired by the swarm behavior manifested by several animals, like ants, bees, and others (Cortez, 2014). The essence of Swarm Intelligence lays on the self-organized behavior that the agents' population presents (Michalewicz, 2007).

particle swarm optimization is a stochastic[12] optimization technique that is essentially a form of guided mutation (Luke, 2013). Unlike other population-based methods, PSO does not focus on expanding the population but in its preservation. When changes in the search space are detected, the population members are tweaked. In other words, candidate solutions are mutated towards the best solution (the guided mutation moves the particles in the search space).

Algorithm 5 Particle Swarm Optimization

Input: f (evaluation function), ... (other parameters)
  $Population \leftarrow initialize\_population()$ ▷ Initialize population
  $Best \leftarrow best(P, f)$ ▷ Best particle
  while termination conditions not met **do**
    for $p \in Population$ **do** ▷ Iterate over all particles
      $s \leftarrow move(s, ...)$ ▷ Update particle's position
      $s \leftarrow adjust(s, ...)$ ▷ adjust s position (if outside bounds)
      if $f(s) < f(p)$ **then**
        $p \leftarrow s$ ▷ Update previous best
      end if
      if $f(s) < f(B)$ **then**
        $p \leftarrow s$ ▷ Update best value
      end if
    end for
  end while
Output: B

12 Something that was randomly determined.

ANT COLONY is an algorithm inspired by real colonies of ants, which pour pheromone on the ground to influence the behavior of other ants; the greater the amount of pheromone on a particular path, higher the probability that other ants will select it. In these systems, the pheromone levels guide the creation of new solutions (Michalewicz, 2007).

Algorithm 6 Ant Colony Optimization

Input: f (evaluation function)
  initialize_pheromone() ▷ Initialize pheromone levels
  while termination conditions not met **do**
    for $ant \in Colony$ **do** ▷ Iterate over all ants
      $s \leftarrow constructSolution()$ ▷ Construct solution
      $f(s)$ ▷ Evaluate s
    end for
    update_pheromone() ▷ Update pheromone levels
  end while
Output: s

2.4 DECISION SUPPORT SYSTEMS FOR DYNAMIC VEHICLE RELOCATION

As we saw, these sharing services need some sort of system that offer some aid to the management when it comes to decision making. To this day, different Decision Support Systems were developed to help these managers, allowing them to have a clearer view of the day to day scenarios and provide information about what repercussions each decision should lead. Some real-world examples are presented in the next section.

Kek et al. (2009) present a three-phase optimization-trend-simulation (OTS) decision support system to help car sharing operators determine a set of near-optimal manpower and operating parameters for the vehicle relocation problem. This DSS was tested on real-world operational data from a car sharing company in Singapore. The results are fascinating. They suggest that the manpower recommended by the system lead to a cut in staff cost of 50%, a reduction in zero-vehicle-time ranging between 4.6% and 13.0% and a contraction in the number of relocations between 37.1% and 41.1%. The structure of this approach is presented in Figure 4 .

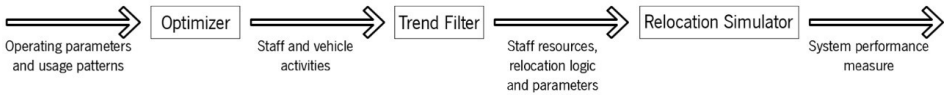


Figure 4: Structure of the three-phase OTS DSS, adapted from (Kek et al., 2009)

The first phase of the OTS is an Optimizer. It receives, as input from the car sharing system, characteristics such as the number of parking stalls, staff costs, vehicle relocation costs; then proceeds to determine the resource allocation that minimizes the cost, displaying the optimized staff needs and activities, relocations and the number of vehicles at the stations at each time.

Trend Filter, the second phase, receives the output from the Optimizer and 'filters' them through heuristics. The expected output is a set of recommended operating parameters. The third and last phase of OTS evaluates the effectiveness of the recommended parameters using three performance indicators - ZVT (Zero-Vehicle Time), FPT (Full-Port Time) and NR (Number of Relocations).

We note that in this work, when ZVT occurs at a station, the station has no available vehicles to satisfy the demand. When FPT occurs, the station has no empty parking stalls and users want to return the vehicles to that station (not being able to do so). A greater value of NR means a higher cost of vehicle relocation operations and, both ZVT and FPT reduce the attractiveness of the carsharing system to users. Consequently, for an optimal performance of the system, the values for ZVT, FPT and NR should all be minimized (close to zero). The implementation of OTS presented interesting results. In Figure 5 we can see that all the performance indicators are either maintained or reduced. The most relevant value is the NR. NR levels dropped to an exciting 41.1%.

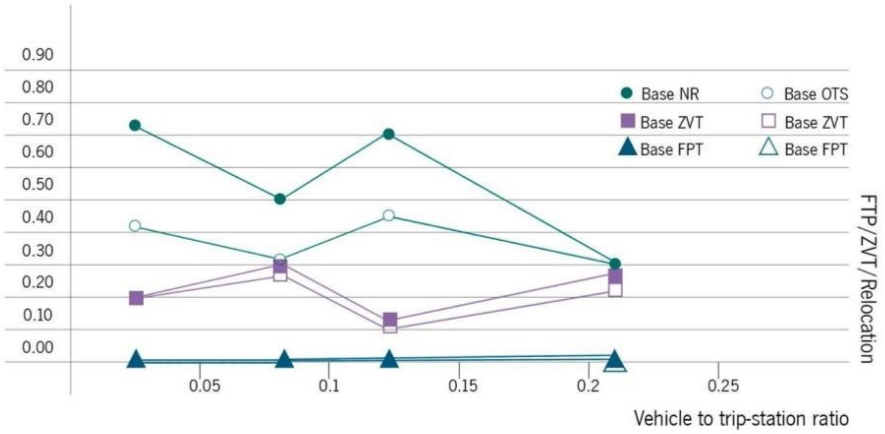


Figure 5: Comparison of OTS against base model, adapted from (Kek et al., 2009)

From the values shown in Table 1, we can see the performance improvement when compared to the base model.

Table 1: Percentage improvement in permonce indicator of OTS, adapted from (Kek et al., 2009)

Vehicle to Trip-Station Ratio	Improvements		
	ZVT	FPT	NR
0.03	4.6	0.0	41.1
0.07	6.0	8.1	38.0
0.12	11.2	0.0	37.1
0.21	13.0	71.3	0.0

In another example, Caggiani and Ottomanelli (2012) developed a flexible *fuzzy* DSS (based on a Fuzzy Inference System[13]) for the vehicles relocation in a bike sharing system. The goal was to determine the optimal repositioning flows, time intervals and distribution patterns that would minimize the redistribution costs and assure user satisfaction.

The system starts to predict future demand among stations, basing the forecast demand method on Artificial Neural Networks (ANN) and Fuzzy Logic[14]. The Neural Net translates the spatio-temporal variation of requests, giving the user important feedback about the vehicle usage for that day or time period. Being based on fuzzy logic suggests that the result can be better than a traditional one, especially when dealing with uncertain, imprecise or ambiguous environment.

13 A fuzzy inference system is a system that uses fuzzy set theory to map inputs to outputs.
14 Fuzzy logic is an approach based on 'degrees of truth' rather than the usual 'true or false' logic on which the modern computer is based.

When the demand is known, the system needs to be rebalanced. The relocation paths are calculated by solving the well-known TSP, formulated as *fixed-point Non-Linear Integer Optimization*. The solution algorithm is based on a Branch-and-bound algorithm. The proposed DSS leads to a diminution of lost users, increasing their probability of finding available bikes or free slots. The operation of the system was only simulated and not deployed, but easily could be extended to wider and real sized systems.

Many were the initiatives that emerged in the last few years, regarding vehicle routing problems. These examples were detailed because of the similarities to this project, but we are going to see many other important developments in this specific area. Not all of them are about Decision Support Systems per se, but the fundamentals of redistribution systems are covered.

In the late '90s, Dror et al. (1998) proposed that a fleet of electric vehicles relocation should be made by a fleet of capacitated tow trucks. Two decades after, we are still making progress in this area, that is more actual than never. A year later, Barth and Todd (1999) simulated a rebalancing system and showed that the vehicle relocation is minimized when 18-24 vehicles are available for every 100 users.

A few years later, Wang et al. (2010) presented a forecast-based relocation model that showed an improvement in the overall efficiency of the system. Correia and Antunes (2012) formulated a Mixed Integer Linear Programming (MILP) strategy to optimize the depot location in a one-way carsharing system and Boyacı et al. (2015) proposed a multi-objective MILP to develop one with an electric fleet.

The benefit of sharing systems in coexistence with other networks such (as public transportation) was studied by Chow and Sayarshad (2014) and, in the same year, Nourinejad and Roorda (2014), by taking the user needs and individuality into account (departure, arrival and request time), defined a better model that can be used, not just as a DSS for optimal relocation operations, but also as a strategic decision-making aiding tool to find an optimal fleet size.

In round-trip systems, the requirements are slightly different. The fact that the users have to return the vehicles to the departure location forces the system to have a higher fleet size (Nourinejad and Roorda, 2015).

When the system has uncertain requirements, some adjustments need to be made. Studies were made and articles were published to address these particular cases. Fan et al. (2008) and Nair and Miller-Hooks (2011) developed different models to address these issues. The first had the objective of maximizing revenues and minimizing relocation costs; the second wanted to minimize the cost of vehicle relocation.

When it comes to bike sharing systems, some things are not the same. The literature covers that as well. Some examples will be following presented.

Froehlicj et al. (2008), using data from a Bike Sharing System(s) in Barcelona (Spain), tested a few predictive techniques to forecast the demand. The results present an astonishing average error of 8% over all days. Another experiment, led by Borgnat et al. (2011), through data from BSS on Lyon (France), predicted hourly rental of vehicles.

Many different approaches were used in the last few years to predict user behavior and habits towards demand. Hemdersom and Fishman (2013) predicted departures and arrivals from customers on time window of 1 hour and the expectation of stations occupancy. The model was tested on the BSS from Chicago (EUA).

Borgnat et al. (2011) and Caggiani and Ottomanelli (2013) went the extra mile and included weather and holidays in their studies. With that detail, they were able to improve their predictions accuracy in more than 50%.

Predictions of users demand only give companies the knowledge to design better systems. The real value emerges when inventory and routing problems start to take part in the equation. Some authors described their work with these aspects. Caggiani and Ottomanelli (2013) proposed a DSS that, based on the user demand prediction, minimized the repositioning costs. Schuijbroek et al. (2013) also address both problems, combining them into a single framework. To summarize, through historical data, estimate optimal inventory levels, which serve as a base to the routing problem, minimizing the distance to travel. The model performance was tested on systems from Boston and Washington (USA). A solution was identified within one or two minutes (better than other often-used formulations after two hours).

Regue and Recker (2014) proposed to find the optimal inventory levels at each station and the optimal routes for the repositioning vehicles to redistribute bikes through a BSS. They combined different datasets to foresee future station inventory levels. Based on the results, a stochastic linear integer problem was resolved to determine the estimated number of bikes required at each station.

As we can see, this area has experienced a boom in the last few years derived to the necessities of today's society.

2.5 Optimization of Dynamic Vehicle Relocation

Now that we covered the fundamentals of some algorithms, we can have a better understanding of why they are being used in this area. We are going to see some work from multiple authors to have a clearer perception of the state of the art. We are only going to address routing optimization and leave out other system design initiatives, like the determination of size and location of stations. To see some of the related work to that end consult authors/articles like Lin and Yang (2011).

Brinkmann et al. (2016) presented a multi-periodic inventory routing problem, evaluated by a cost-benefit routing algorithm. Two search algorithms were implemented in their work: Hill Climbing to choose the best solution (until a local optimum is found) and Simulated Annealing, for further exploitation of the search space.

From the system of Reggio Emilia (Italy) was collected the data that served for testing of the Dell'Amico et al. (2014) research. They addressed the Bike Sharing Rebalancing Problem as a four Mixed Integer Linear Programming formulation, including an exponential number of constraints, and Branch-and-cut algorithms. Caggiani and Ottomanelli (2012) formulated the problem as a fixed-point Optimization, basing the solution also in a Branch-and-bound procedure. Chemla et al. (2013) also proposes a Branch-and-cut algorithm for the problem: to obtain the lower bound. Through Tabu Search, they obtained the upper bound of the optimal solution.

Raviv et al. (2013) presented two MILP formulations to a static rebalancing problem in which they minimized traveling costs and user dissatisfaction by assuring a proper inventory level in each station. Each one describes a distinct version of the problem. Another MILP formulation was developed by Ghosh et al. (2017). Their ultimate objective was to maximize profit for a bike sharing system, considering a trade-off between demand (maximize) and expenses (minimize). On the other hand, Contardo et al. (2012) offer a formulation for the dynamic rebalancing problem. The purpose of the optimization was the maximization of the service demand.

For the dynamic rebalancing of a bike sharing system, Chiariotti et al. (2018) came up with a greedy rebalancing path, implementing a nearest-neighbor heuristic, with the goal of improving the availability of the service.

In the vehicle routing Marinakis et al. (2010) proposed a hybrid algorithm based on Particle Swarm Optimization. Gong et al. (2012) also worked with Particle Swarm Optimization to address the Vehicle Routing Problem (VRP) with time windows. Three years later, Marinakis et al. (2013) solved a variant of the problem with the same approach, the Vehicle Routing Problem with stochastic demands. Bianchi et al. (2006), in the same problem, explored the hybridization of metaheuristic by means of two objective functions and Mak and Guo (2004) explored Genetic Algorithms, taking into account soft time windows. In the

same problem, but with a distinct strategy, Vaira (2014) proposed a genetic algorithm based on a random insertion heuristics. They also used Branch-and-bound (B&B). Lei et al. (2012) went the extra mile and addressed the Vehicle Routing Problem with Stochastic Demands and Split Deliveries, using an adaptive large neighborhood search heuristic.

Rizzoli et al. (2007) used an Ant Colony Optimization metaheuristic and applied their research to real-world variants of the vehicle routing problem. A VRP with time windows for an important supermarket in Switzerland; VRP with pickup and delivery for a leading distribution company in Italy; a time-dependent VRP for load distribution in Padua, Italy (trip depends on the time of day); on-line VRP in Lugano, Switzerland (customers' orders arrived during the delivery process). Table 2 summarizes the most important related literature.

Table 2: Summary of related literature

Study	Static (S)/ Dynamic (D)	Modern Optimization (M)/ Operational Research (O)	Bike Sharing	Car Sharing
(Caggiani and Ottomanelli, 2012)	D	O	✓	×
(Dell'Amico et al., 2014)	D	O	✓	×
(Raviv et al., 2013)	S	O	✓	×
(Ghosh et al., 2017)	S	O	✓	×
(Chemla et al., 2013)	S	O	✓	×
(Brinkmann et al., 2016)	D	M	✓	×
(Bianchi et al., 2006)	S	M	✓	✓
(Gong et al., 2012)	S	M	✓	✓
(Marinakis et al., 2010)	S	M	✓	✓
(Rizzoli et al., 2007)	D	M	✓	✓
(Mak and Guo, 2004)	D	M	✓	✓
(Fan et al., 2008)	D	O	×	✓
(Kek et al., 2009)	D	O	×	✓
(Correia and Antunes, 2012)	D	O	×	✓
(Boyacı et al., 2015)	D	O	×	✓

3

PROBLEM FORMALIZATION

3.1 PROBLEM DESCRIPTION

CEiiA is a Portuguese company that has projects all over the world. The output of this thesis is expected to be implemented in the Decision Support Systems of several services worldwide, implemented by CEiiA.

In conversations with Nuno Oliveira, Data Scientist at CEiiA, it was decided that the testing environment should be the Barcelona scooter sharing system. This system already has a significant lifetime, an obvious need for relocation (several times during the day) and trustworthy predictive models. Despite being a Bike Sharing Rebalancing Problem[15], ideally, the system should be applied to any service without adaptations (the specifications of each sharing service would be characterized in the input data for the system and not in the system itself). It would have as input the relocation needs, the capacity of the transport vehicle(s) and eventually the time limit to do the relocation and would return the route(s) of the relocations and associated time and cost.

Scooter sharing services are associated with the BRP. As they are the best-known reference in this variety of studies, throughout the development documentation we will refer to scooters as bikes.

3.1.1 *Descriptive Example*

In the initial phase, the problem will be simplified. Over time, incrementally, we will add complexity to it, making the system more versatile and powerful. As already mentioned, the system to be developed aims to solve the needs of a dynamic rebalancing system. The following example (selected from the data) demonstrates the system requirements: to balance the service for a certain hour, while minimizing the distance traveled by the vehicles. The system, querying the data to a certain hour or a day, will discover the relocation requirements for specific areas of the sharing service.

15 The literature refers to the resulting optimization of which route the vehicles should take, to perform the redistribution at minimum cost, as BRP (Dell'Amico et al., 2014).

Table 3 displays the relocation needs for 2017-11-26 12:00:00.

Table 3: Relocation needs

Hour	*Area*	*Relocation*
2017-11-26 12:00:00	177	10
2017-11-26 12:00:00	107	8
2017-11-26 12:00:00	192	8
2017-11-26 12:00:00	9	6
2017-11-26 12:00:00	108	6
2017-11-26 12:00:00	162	6
2017-11-26 12:00:00	176	6
2017-11-26 12:00:00	84	-30
2017-11-26 12:00:00	57	-15
2017-11-26 12:00:00	148	-5

Figure 6 depicts the selected areas in Barcelona.This system intends to identify the shortest route that facilitates the delivery of 10, 8, 8,6, 6, 6, 6 vehicles respectively to areas 177, 107, 192, 9, 108, 162 and 176 by removing 30, 15 and 5 vehicles from areas 84, 57 and 148, departing from area 84 and returning to it in the end.

Figure 6: Relocation instance depicted

3.1.2 *Problem Formulation and Definitions*

In this section we define the problem under study and the notations used throughout the report. As the areas under study are defined by a geographical point (*latitude, longitude*), we will refer to the centroid of each area as *station*. This will facilitate the comprehension of the problem formulation. Therefore, the problem is given by a set of n stations $N = \{1, 2, 3, \ldots, n\} | \{84\}$ and the depot $O = \{84\}$. Each station i has a request qi , which can be either positive ($qi > 0$ - delivery node) or negative ($qi < 0$ - pickup node). Every pair of locations (i, j) , where $i, j \in N$ and $i \neq j$, is associated with a distance traveled dij that, according to the system data, is symmetrical ($dij = dji$). We can formulate a complete directed graph $G = (V, A)$, where $V = \{1, 2, \ldots, n\}$ (N stations and O depot) and $A = \{(i, j) : i, j \in V, i \neq j\}$, with a traveling distance dij associated with each arc $(i, j) \in A$.

A fleet of m identical vehicles of capacity Q is available at the depot to transport the bikes. The bikes removed from pickup nodes can either go to a delivery node or back to the depot (depending on the demand) and bikes supplied to delivery nodes can either come from the depot or from pickup nodes. Depending on the capacity of the vehicles and demand (or business regulations), the balance of the system can be restored by one vehicle (Figure 7) or more (Figure 8).

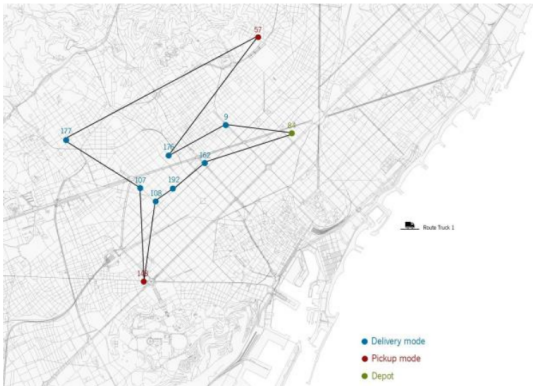


Figure 7: One route rebalancing

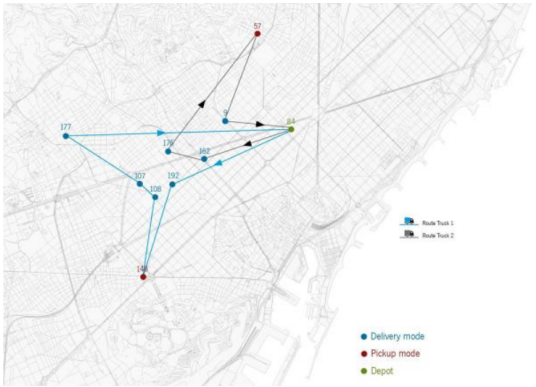


Figure 8: Two routes rebalancing

Table 4: Notation summary

N	Station	$n+1$	Number of Stations (depot inclusive)
O	Depot	m	Number of vehicles
V	Set of vertices	Q	Vehicle capacity
A	Set of arcs	qi	Demand at vertex i
n	Number of stations	dij	Distance of the arc (i, j)
$i^{+/-}$	Delivery/pickup Station	l_i	truck load leaving station i

The BRP determines where to send m vehicles through the graph, with the purpose of minimizing the total distance traveled. There are many constraints that challenge the system and require special attention from the developers, like:

- each vehicle operates a course that starts and finishes at the depot;
- each vehicle leaves the depot vacant or with some initial load;
- a station can be visited multiple times until the request is fulfilled;
- a station can serve as temporary storage to improve the route; and
- others.

These are some of the most adopted or discussed constraints in the field. These business decisions shape the way that the system operates. There are other constraints that restrict the feasibility of the solution:

- time window for rebalancing operations;
- size of the fleet; and
- others.

Figure 9 presents a feasible solution for the example above (with two trucks, each one can carry 15 vehicles). It should be noted that station 9 and 107 were visited two times until the request was satisfied. The constraint that allows multiple visits to a station is not always followed (e.g., Dell'Amico et al. (2014)), but in some cases, can make a difference. This is just an example that helps us to understand the complexity of a large sharing system and its rebalancing requirements.

The constraints that we just saw are soft constraints (related to user-goals). Michalewicz (2007) recognizes hard and soft constraints as being the two main types of constraints in modern optimization problems. Hard constraints cannot be infringed and are due to factors such as laws or physical restrictions (Cortez, 2014). Such constraints must be addressed to guarantee real value to the system:

- vehicles capacity shall not be exceeded;
- speed limits shall not be broken;
- regulations restricting truck drivers' working and driving hours must be attended; and
- others.

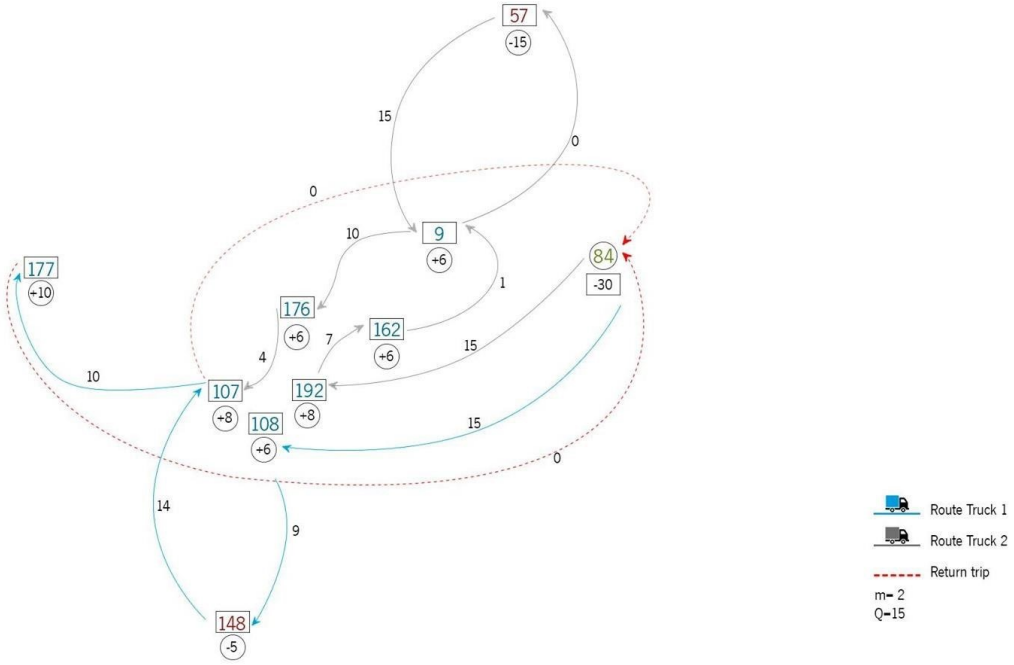


Figure 9: Example of a feasible folution

4

DEVELOPMENT OF THE OPTIMIZATION SYSTEM

4.1 FORMULATION

This project attempts to address the lack of real, dynamic and adaptable solutions to solve this problem; adaptable to the size of the available truck fleet, their features, environment characteristics, and business rules. The system should be easily adjusted to the different types of vehicles to be transported and not attached to a specific type. Despite several studies and papers on this subject, there is no holistic approach that contemplates the end-to-end process, developed and planned for a mild and effective implementation. The fact that the need for such systems is more popular than ever raises the need for a solution that meets the constraints of most BSS businesses in major cities. There is an obvious urgency for such solutions to cease to be the focus of academic studies alone and focus on developing cohesive and robust solutions.

The project will compare the performance of algorithms that follow different approaches and converge to a solution that can address the day to day business requirements. This is a a problem that can either be solved by a single trip, carried by a single truck, or by t trips, performed by m vehicles. The final solution will be represented by a matrix in which each row describes a trip. The solution should look like this:

$$\mathbf{sol} = \begin{pmatrix} l_{11} & l_{12} & \dots & l_{1n} \\ l_{21} & l_{22} & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{t1} & l_{t2} & \dots & l_{tn} \end{pmatrix}$$

Where t represents each trip and n the station visited, l is the load that the truck has when leaving each station n. To explain the development and associated logic of the system, we are going to follow the already listed example (see Chapter 3.1). The system can be represented by the flow represented in Figure 10. Each step of the flow is explained in the following sections.

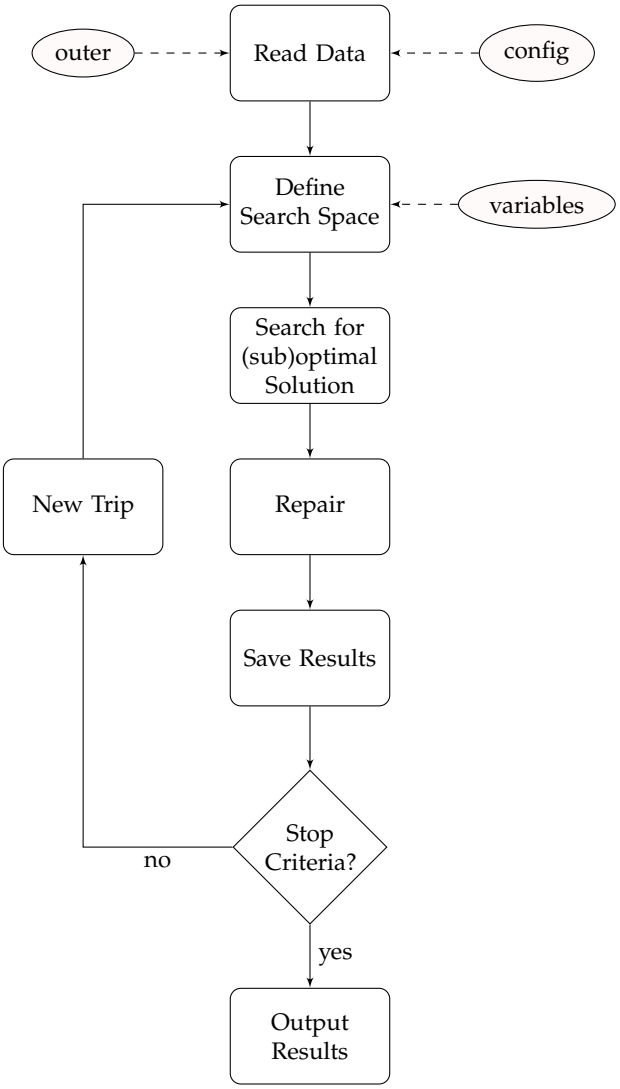


Figure 10: Optimization system flow

4.2 READ DATA

In general terms, the first phase of the system is the building of its infrastructure and the loading of the specifications in which the rebalacing optimization is going to be built upon. Global variables like stations demand, distance between them, available fleet and truck atributtes are loaded. The description of the variables are presented in Tables 5 and 6.

Table 5: Data provided

Name	Extension	Purpose	Description
relocations	.csv	Necessities to be met	A dataset with 3 columns and 8267 rows, each one of them expressing the necessities associated with a particular station, at a particular time. Example: Table 7
distances	.csv	Distances between stations	A dataset with 264 columns and 264 rows, each one of them representing the distance (meters) between areas. Example: Table 8
areas	.csv	lat e lon of each station	A dataset with 3 columns and 264 rows, each one representing the latitude and longitude of each area's centroid. Example: Table 27

Table 6: User defined data

Name	Extension	Purpose	Description
fleet	.csv	Truck Fleet	Dataset with 3 columns and x rows, representing a fleet of x trucks. Examaple: Table 9
config	.yml	Configuration Parameters	YAML file with configuration variables that are defined by the user and are volatile over time. See Figure 11

Table 7: Relocations example

Hour	*Area*	Relocation
2017-11-26 12:00:00	177	10
2017-11-26 12:00:00	107	8
2017-11-26 12:00:00	192	8
2017-11-26 12:00:00	9	6
2017-11-26 12:00:00	108	6
2017-11-26 12:00:00	162	6
2017-11-26 12:00:00	176	6
2017-11-26 12:00:00	84	-30
2017-11-26 12:00:00	57	-15
2017-11-26 12:00:00	148	-5

Table 8: Distances example

	X_1	X_2	X_3	X_4
Y_1	0	332.8	881.3	730.6
Y_2	332.8	0	776.1	398.3
Y_3	881.3	776.1	0	851.8
Y_4	730.6	398.3	851.8	0

Table 9: Fleet example

Plate	*Capacity*	*available*
$11-XX-22$	15	*TRUE*
$22-YY-33$	15	*TRUE*
$33-ZZ-44$	10	*FALSE*

```yaml
---   config.yml
journey:
    limit: 60 #time, in minutes, to rebalance the system
    velocity: 750 #m/min - 45km/h average speed during a trip
    depot: 84 #depot

operations:
    bycicles:
        pickup: 1 #average time spent on picking up a bycicle
        deliver: 1 #average time spent on delivering a bycicle

     scooters:
        pickup: 1 #average time spent on picking up a scooter
        deliver: 1 #average time spent on delivering a scooter
---
```

Figure 11: YAML configuration file

To relate to the example with the formulation of the problem present in Section 3.1, Figure 12 tends to clarify the reader about the construction of the graph, corresponding to each problem instance (fictitious values).

Hour	*Area*	*Relocation*
2017-11-26 12:00:00	177	10
2017-11-26 12:00:00	107	8
2017-11-26 12:00:00	192	8
2017-11-26 12:00:00	9	6

di, j	*Distance*	di, j	*Distance*
$d_{177,107}$	2 988	$d_{107,192}$	7 990
$d_{177,192}$	680	$d_{107,9}$	9 219
$d_{177,9}$	10 000	$d_{192,9}$	5 500

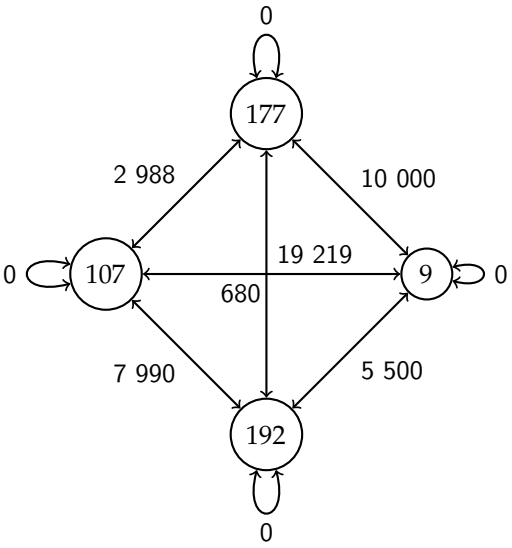


Figure 12: Example of a complete directed graph G=(V,A) corresponding to a fictitious instance of the problem

Some adaptations were implemented to facilitate the algorithm to always consider feasible solutions. For example, as depicted in Table 7, the stations do not follow an incremental approach (Figure 13), making it arduous to impress a search logic to them. The problem was adressed by maping each station to an identifier (Figure 14), folowing an incremental approach, allowing the algorithm(s) to easilly evaluate the solution and making the boundaries much more ristrict (see Section 4.3.2).

In the execution of the optimization module, the algorithm(s) is going to run based on a data frame with the auto increment identifier of each station, the actual need to be addressed and the need after the journey returned by the search (see Table 14).

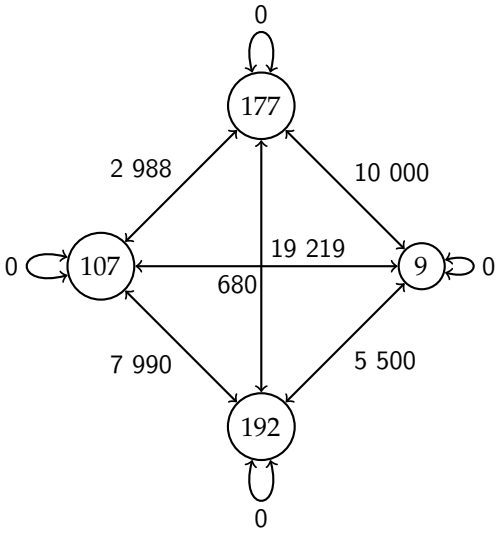


Figure 13: Complete directed graph G = (V, A) with real data

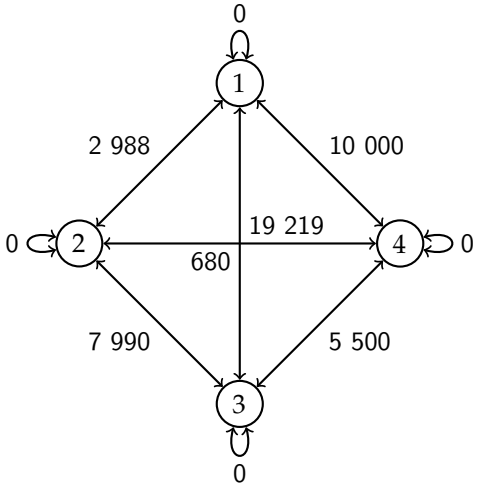


Figure 14: Complete Directed Graph G = (V, A) to feed the system

4.3 DEFINE SEARCH SPACE AND SOLUTION REPRESENTATION

With all global variables defined, all the requirements to represent the solution and the search space are satisfied. Both the search space and the representation of the solution are dynamically derived by the specifications of each instance of the problem. This use case has three distinct redistribution hours each day. Each one of them differs in complexity and demand, requiring different constraints that require consideration. The need to perform each task with high adaptability is crucial and, on account of all the constraints, the solution can be highly variant. It should be noted that the Bike Sharing Problem is a non trivial variant of of the well known Traveling Sales Problem, thus it does not assume just the distance between stations. In effect, the best solution is not the shortest path between stations, but the shortest path that allows a fleet of trucks to fully restore the service level of each station. With this in mind the solution should not only represent the stations, but also the vehicles delivered/picked on each one of them.

The most direct way of representing this task is to consider, for each station, the pair $station \rightarrow quantity$, in which the station would be its identifier ($\in \mathbb{N}^+$) and the quantity to be delivered ($\in \mathbb{N}_0^+$) or picked ($\in \mathbb{N}_0^-$). This approach would have a straightforward implementation but the search space associated with it could be smaller. Consider that the truck drives to station number 3. This station has a demand for 8 vehicles. So the solution should be $3 \rightarrow$ -8. So, considering that the truck can carry 15 vehicles simultaneously ($Q = 15$), the algorithm should consider from $3 \rightarrow$ -15, to $3 \rightarrow 15$. This is not an unacceptable search space but it could be more advantageous. Instead of considering the pair $station \rightarrow quantity$, we are going to consider the pair $station \rightarrow load$, in which the station would be its identifier ($\in \mathbb{N}^+$) and load, the truckload leaving the station ($\in \mathbb{N}_0^+$). As the vehicles that can be picked or delivered are constrained by the truck capacity Q, the search space can be optimized, only considering, in this example, $3 \rightarrow 0$, to $3 \rightarrow 15$, allowing the algorithm, with the same computational power, to explore regions that can be more meaningful. It should be noted that many Modern Optimization methods are implemented by assuming solutions with a numeric vector. Thus, we assume this representation of solutions, putting each trip in a loop, that will only stop when all service level requirements are quenched or the time window constraint is corrupted. The solution representation is depicted below (Figure 15):

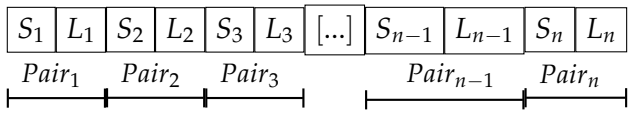


Figure 15: Theoretical solution sepresentation

where each pair corresponds to the *station identifier* and the *truck load* when leaving it.

As the solution is driven by constraints, in order to deliver a valid and meaningful solution, its construction should reflect them. As one of the constraints is that each truck operates a course that starts and finishes at the depot, S_1 should always be considered as the depot and should not be part of the solution representation on the optimization tasks, being the first pair a false one, making the first value of the solution the truck load leaving the depot.

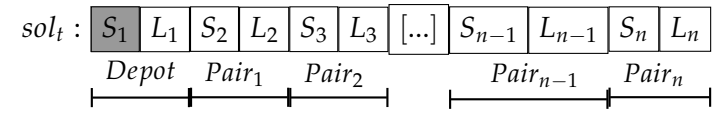
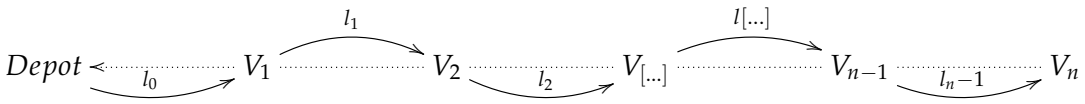


Figure 16: Practical solution representation

To better understand what the solution represents, the following flow tries to facilitate its interpretation.



4.3.1 *Dimension*

This is a problem that can either be solved by a single trip, carried by a single truck, or by t trips, performed by m vehicles. We have to consider this factor when we think about the dimension of the solution. To make sure that the system can find a solution for every instance of the problem, its representation and search space should consider that one trip can restore the equilibrium of the system. Consequently, the dimension should be considered as being V number of vertices, multiplied by two (truckload for each one of them), less one (excluding *depot*).

$$D = (V \times 2) - 1, V \in \mathbb{N}^{+} \backslash \{1\} \tag{2}$$

By the example given in the previous chapter (Section 3.1), we have 10 areas to visit, including depot. So, $10 \times 2 - 1 = 19$.

4.3.2 *Bounds*

The search space is not only composed by the dimension, but also by the bounds of the solution. These intend to control the dispersion of the generated solutions, maintaining the coherence of the solutions generated with the case study expectations.

So, as seen in Figure 16, odd numbers represent the load of the truck when leaving each station, and the even ones, the station identifier. So, following the logic of the auto-increment identifier, for each station and capacity Q, the bounds could be granted by:

$$lower = \begin{cases} 0, & \text{if } i \text{ is odd} \\ \min(N), & \text{if } i \text{ is even} \end{cases} \qquad upper = \begin{cases} Q, & \text{if } i \text{ is odd} \\ \max(N), & \text{if } i \text{ is even} \end{cases}$$

i, representing the index of the solution vector — i, representing the index of the solution vector

Consider the example given in the previous chapter (Section 3.1):

$lower$:

0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

$upper$:

15	10	15	10	15	10	15	10	15	10	15	10	15	10	15	10	15	10	15

Attending to the problem constraints, we ought to consider that the truck must return vacant to the depot. To do so, the last station visited should have a positive demand (*delivery node*). To meet this business logic in mind, we have to add conditions to our bounds.

$$lower = \begin{cases} 0, & \text{if } i \text{ is odd} \\ \min(N), & \text{if } i \text{ is even, i}<\{n\times 2\} \\ \min(N^+), & \text{if } i \text{ is even, i}\geq \{n\times 2\} \end{cases} \qquad upper = \begin{cases} Q, & \text{if } i \text{ is odd, i}<\{n\times 2\} \\ \max(N), & \text{if } i \text{ is even, i}<\{n\times 2\} \\ 0, & \text{if } i \text{ is odd, i}\geq \{n\times 2\} \\ \max(N^+), & \text{if } i \text{ is even, i}\geq \{n\times 2\} \end{cases}$$

i, representing the index of the solution vector — i, representing the index of the solution vector

which translates into:

$lower$:

0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

$upper$:

15	10	15	10	15	10	15	10	15	10	15	10	15	10	15	10	15	7	0

4.4 SEARCH FOR (SUB)OPTIMAL SOLUTION

Modern Optimization, also known as metaheuristics, are exceptional when it comes to finding a 'good enough' solution when there is no simple way to find the absolute best. Two different metaheuristic approaches were considered when modeling this problem; local and population-based search. In the first one, Hill Climbing and Simulated Annealing were implemented, and in the second, Evolutionary Algorithms were considered. The two approaches will be compared throughout this section, giving a thorough explanation of each step of the development, opposing one approach to another and comparing the results at the end.

4.4.1 *Objective Goal Formulation*

The evaluation function, also known as fitness, translates the project desired goal; the minimization of the distance traveled in a redistribution instance.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & d(x) \\ \text{where} \quad & d(x) = \sum_{\substack{i=1 \\ i \neq n}}^{n} d(i, i+1) \quad i, n \in \mathbb{N}^{+} \end{aligned} \tag{3}$$

But, if we only consider the distance, this project would not care if the business goals were met or not, making this project just the study and development of a solution to the Traveling Salesman Problem. The goal of the solution is the minimization of the total distance traveled, that can meet the service level requirements of all stations. Since considering the solutions under the single objective of the distance traveled would lead to unacceptable outcomes, this factor should be incorporated in the evaluation formula.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & d(x) + \alpha r(x) \\ \text{where} \quad & d(x) = \sum_{\substack{i=1 \\ i \neq n}}^{n} d(i, i+1) \quad i, n \in \mathbb{N}^{+} \\ & r(x) = \sum_{i=1}^{n} r(|i|) \qquad i, n \in \mathbb{N}^{+} \end{aligned} \tag{4}$$

In some cases, solutions can be improved by incorporating some domain knowledge. As this is an incremental approach, the redistribution can be performed by t trips. To align with business goals and to enhance the algorithm performance, each trip should also maximize the demand. For instance, supplied stations should no longer be considered in

the next trips, resulting in a logical distance/time advantage and increasing the probability of finding better solutions.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & d(x) + \alpha r(x) - \beta f(x) \\ \text{where} \quad & d(x) = \sum_{\substack{i=1 \\ i \neq n}}^{n} d(i, i+1) \qquad i, n \in \mathbb{N}^{+} \\ & r(x) = \sum_{i=1}^{n} r(|i|) \qquad i, n \in \mathbb{N}^{+} \\ & f(x) = \sum_{i=1}^{n} [r(|i|) = 0] \quad i, n \in \mathbb{N}^{+} \end{aligned} \tag{5}$$

So, the ultimate goal is to have a solution that minimizes the distance required to redistribute as many vehicles as possible, trying to reduce the number of stations left to visit for the next trip.

As for α and β values, these were set by executing some experiments, as discussed in Section 5.

4.4.2 *Initial Solution and Population Definition*

4.4.2.1 *Initial Solution*

As seen in Section 2.3.2, one of the characteristics that differentiate local and population based approaches is the initialization of the search environment. Local search only considers one initial solution and generates new ones from the old ones. The most common approach is to set the initial solution randomly (Cortez, 2014), but better results can come when applying domain knowledge to it, putting the initial solution on a more desirable local neighborhood, increasing the probabilities of discovering a more suitable one.

Although, too much domain knowledge can harm the search. Generating a solution that can be one (sub)optimal can cause the algorithm to get stuck at a local optimum, hardly granting the chance to explore other interesting areas; especially in single initial solution approaches. In this case, some ground rules were determined to try to add some domain knowledge to it, but not too much.

So, the initial solution could be seen as depicted in Figure 17, representing a trip that could rebalance the system. Algorithm 7 exhibits the principle behind the construction of the single initial solution.

Algorithm 7 Initial solution with domain knowledge

Input: f (fleet), d (demand matrix),$depot$ (depot index), ... (other parameters)
  $Q \leftarrow max\$capacity(f)$ ▷ Capacity of 'bigger' truck
  $n\_stations \leftarrow nrows(d) - 1$ ▷ Number of stations to visits
  $pickup\_nodes \leftarrow neg(d)$ ▷ Stations with negative demand
  $delivery\_nodes \leftarrow pos(d)$ ▷ Stations with positive demand
  $s \leftarrow []$ ▷ Empty Array
  $i \leftarrow 1$ ▷ Number of iterations
  while $i \leq n\_stations$ **do** ▷ $n\_stations$ is the termination criteria
    if $i = 1$ **then**
      $s \leftarrow max(Q, d[depot])$ ▷ Load leaving depot (max possible)
      if $Q/2 \geq max(Q, d[depot])$ **then** ▷ If truck is more than half full, go deliver. Go pick when not
        $s \leftarrow get(delivery\_nodes)$ ▷ Go to delivery node
      else
        $s \leftarrow get(pickup\_nodes)$ ▷ Go to pickup node
      end if
    end if
    if $i = n\_stations$ **then** ▷ Last station selection
      $s \leftarrow get(delivery\_nodes)$ ▷ Must be a delivery node
      $s \leftarrow 0$ ▷ Must return empty to depot
    else
      $s \leftarrow get(d)$ ▷ Go to random station
      $s \leftarrow sample(0 : Q)$ ▷ Leave that station with random load (not exceeding truck capacity)
    end if
    $i \leftarrow i + 1$
  end while
Output: s

Algorithm 7 provides a brief demonstration of the domain knowledge printed in the initial solution creation. The focus is, as usual, to impress human vision on the system's operation. As we mentioned how dangerous it is to create a local optimum as an initial solution, the truckload is randomly generated in the process. Since we want the first station to be visited after the depot not to be completely random, the load for the depot is first generated. Considering this load, if it fills half the capacity of the truck, the station to visit will be a delivery node. Otherwise, it is a sign that the truck still has a lot of cargo space, so it should visit a pickup node. Given the constraints of the business, as already mentioned, the truck returns without a single vehicle to the depot. For this, the last station to visit must be a delivery node. These are the basic principles of creating the initial solution.

Example

According to the example in Section 3.1.1, reported in Table 7, Figure 17 represents the instance of the problem, arranged by logical sense.

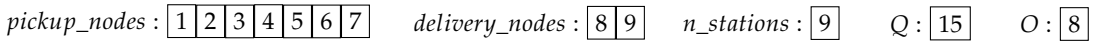


Figure 17: System variables from the Section 3.1.1 example

Following Algorithm 7, and using the information from Figure 17, we can derive an initial solution to feed the local search algorithm(s).

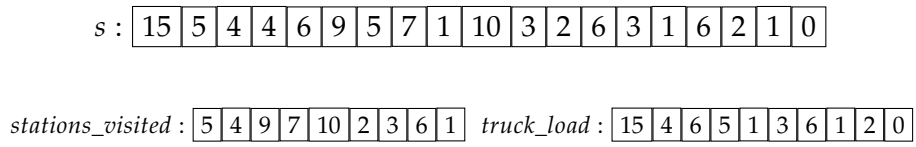


Figure 18: Initial solution example

We can translate the vector s to a practical representation, which can give us a clear understanding of what it should express.

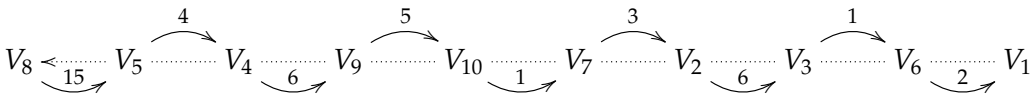


Figure 19: Practical representation of an initial solution

Leaving Depot, V_8, with 15 bicycles, the truck goes to the station labeled 5, V_5, leaving loaded with 4 bicycles, dropping 11 in V_5 $(15 - 11)$, and so on. The vector should be read this way.

4.4.2.2 *Initial Population*

In population-based algorithms, like evolutionary algorithms, instead of considering a single initial solution, it considers a set of them, commonly known as population. Common population size values are $N_p \in \{20; 50; 100; 200; 500; 1000; 2000\}$ (Cortez, 2014). It gives us more liberty to use domain knowledge on the initial population, being able to explore more distinct regions of the search space. Taking advantage of this fact, we developed a method of creating non random solutions to incorporate the population. We started with the stream of thought previously considered, but then we were left with the sensation that the versatility of the region considered should be bigger, so we only considered the previous method to generate 1/5 of the population. The possibility of generating a set of local optima was considered and explored to determine the rest. Consider Figure 22 as an example.

Figure 20: Solution with no repeated stations

Figure 21: Solution with repeated stations

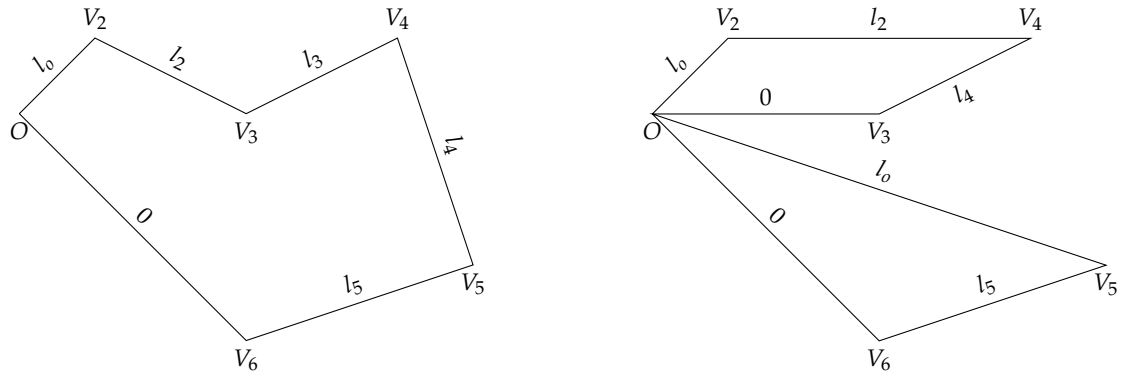


Figure 22: Example of two possible individuals of the population

Figure 22 depicts two example trips. Figure 20 can have the solution with only one trip but, when that it is not the case, it delivers no advantage because it has no focus on emptying a set of stations to not have to include them in eventual next journeys, as Figure 21 does. The principles to each one are based on the Algorithm 7 but, for one, the load is not random, forcing the truck to deliver/pick the maximum possible, for each station. For another, this principle is followed but another one is added: the $get()$ method allows repeated stations to be incorporated in the solution, allowing the truck to visit the same station multiple times throughout the journey. In Figure 21 example we can perceive that this variant can be an advantage in the pursuit of better solutions, being a differentiation to another approaches known in the field. Notice the examples shown on the next page.

Example

Following the example already described in Section 4.4.2.1 we illustrate the two new solution generation techniques. Recognize the variables:

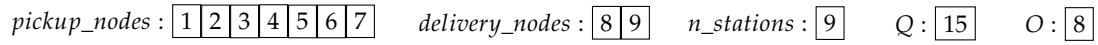


Figure 23: System variables from the Section 3.1.1 example

The first technique illustrated earlier and detailed in Algorithm 7, creates a solution similar to:

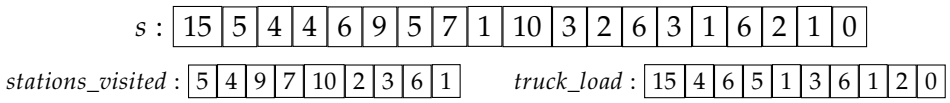


Figure 24: Output example of the first solution generation technique

where each truck load (except depot) is random. This means that some work is necessary to give real value to the output. Table 10 represents the impact that the solution would have on the system service level requirements. Details on Table 10 calculations are given in Section 4.8.

Table 10: System new state

Station	Demand	Need After
1	10	8
2	8	11
3	8	3
4	6	8
5	6	-5
6	6	7
7	6	2
8	-30	-15
9	-15	-16
10	-5	-3

The different colors observed express the different types of impact that the redistribution had on the status of the system. The red color represents that vehicles were delivered where they should have been picked and vice versa. Green means this did not happen, i.e., vehicles were loaded/unloaded where it was supposed to. Yellow means that vehicles were loaded/unloaded where it was supposed to but in excess, i.e., it is still necessary to return there to restore the service level.

Example (continuation)

As we have seen, a considerable effort is still required to make the solution acceptable in terms of the load. In the second technique, this obstacle is overcome. Considering the same stations as the previous example we can notice the upgrade that this technique can be to the optimization task.

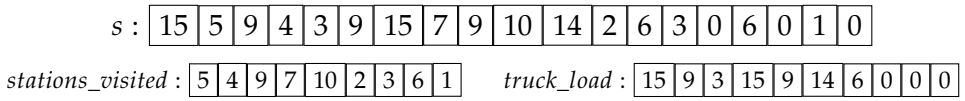


Figure 25: Output example of the second solution generation technique

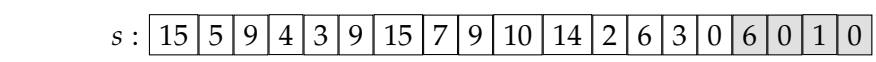
Table 11 represents the impact that the solution would have on the system service level requirements. The color blue is shown when station demand is completely fulfilled, no longer being necessary to include them in the next solutions. When the color of the table cell is not changed it means that there was no change in demand for the station.

Table 11: System new state

Station	Demand	Need After
1	10	10
2	8	0
3	8	2
4	6	0
5	6	0
6	6	6
7	6	0
8	-30	-15
9	-15	-3
10	-5	0

As we can see, with a change in the loads, we were able to find an initial solution that can be a local optimum, fully satisfying the needs of 5 stations, leaving two of them close to the ideal status.

Despite having visited all stations, it was not possible to update the demand of the last ones. Due to this technique not being able to repeat stations in the solution creation, this situation may happen. To try to avoid these obstacles the third technique was developed.



Example (continuation)

This technique allows the return to stations already visited on the same trip. This addresses a purpose that can be crucial to making the most of the available time. So we can look at the third technique as an improved version of the second, giving the solution generation more flexibility.

With the repetition of stations, we could exchange one that could not be updated with any that could. For example, just by switching from 6 to 9, and recalculating the loads, we could build an even more relevant initial solution. Table 12 represents the impact that this small change would have on the system service level requirements.

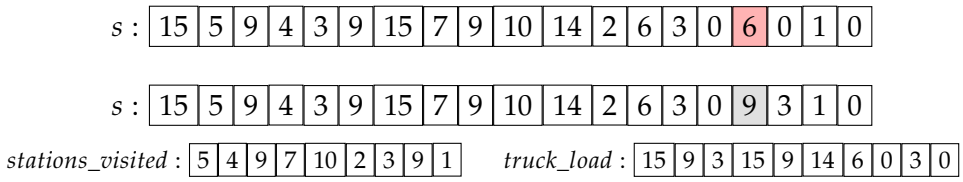


Figure 26: Output example of the third solution generation technique

Table 12: System new state

Station	Demand	Need After
1	10	7
2	8	0
3	8	2
4	6	0
5	6	0
6	6	6
7	6	0
8	-30	-15
9	-15	0
10	-5	0

With this minor change, it was possible to prepare an initial solution that satisfies the service requirements from 60% of the stations. These techniques are undoubtedly an asset to the final solution. Practical results of the implementation of these techniques detailed in Section 5.1.2

4.4.3 *Change and Breeding (Genetic Operators)*

To consider solutions in the neighborhood of the one(s) in memory, some adjustments must be made to the current ones, in order to discover and exploit some more interesting areas of the search space. These adjustments differ for each instance, some being more challenging than others. The system logic and the solution representation will dictate these adjustments.

The problem that we have in hands is a '*hybrid integer*'. *Integer* because all elements are treated as integers; *Hybrid* because of its nature, composed by elements that hold a different meaning, being addressed differently. On one hand, with the stations we are in the presence of a combinatorial problem and, on the other hand, with the load of the trucks, we have a (discrete) numerical problem. Thus, we have to be very careful when addressing this problem, applying different techniques in the same change.

4.4.3.1 *Change*

When working with a single solution, the adjustments made in the solution to exploit its neighborhood are denominated 'change'. In this hybrid problem we, first need to divide the solution in stations visited and truck load, as in Figure 18.

As we can split this problem in two, it makes sense to have at least three change options:

- only stations are modified;
- only the truck load is changed; and
- both stations and truck load are modified.

Combinatorial Problem

In this problem, the objective is to find combination of stations that optimizes the score of the evaluation function (see Section 4.4.1). Since the main goal is the visit all stations by the shortest path, this problem can be solved by considering the permutation of the stations $\{1, 2, 3, ..., n\}$.

A permutation describes an arrangement or ordering of items. There are $n!$ permutations of n items. This grows so exponentially that the amount of time to generate all permutations would be gigantic, for $n > 12$, since 12! = 479, 001, 600 (Skiena, 2010). With (meta)heuristics we can set a maximum number of iterations to assure that the algorithm does not take too long to generate a satisfying solution. The vector that stores the sequence of stations to be visited can be reordered, through the permutation of the elements. This is a typical combinatorial problem.

From (Cortez, 2014) we can see examples of mutation operators for the Traveling Salesman Problem. They were considered and implemented but were forced to satisfy the problem constraints. These examples are depicted in Figure 27: exchange, insertion, and displacement. The first operator swaps two randomly selected cities, the following inserts a city into an arbitrary position and the third inserts a random subtour into another position.

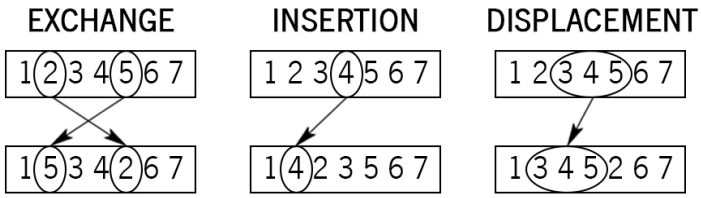


Figure 27: Example of three mutation operators, adapted from (Cortez, 2014)

However, this is not a pure Traveling Salesman Problem. For instance, the repetition of a station may improve the quality of the solution. This was previously discussed in Section 4.4.2.2, but here we examine how it works. Instead of just permutate the stations' vector, the *injection* operator stores in memory the stations to visit; selects a random element and, instead of swapping it with another from the same vector, it removes it from the solution and adds another element from the vector in memory.

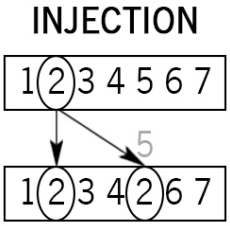


Figure 28: Injection operator

There is also a key factor in this operator. If the user decides so (through defining a Boolean variable as $TRUE$), it can allow the operator to accept one station to appear followed by the same one, shortening the trip - through a logic repair, Algorithm 8.

Using some domain knowledge with some trial and error we can also associate with each type of change, a probability of occurring. The probability of *exchange* and *insertion* operators has an associated probability of 0.3 each, while *displacement* has a probability of 0.2. We just want a trip to be introduced from afar; therefore, the probability associated with the *injection* operator is 0.1. Sometimes we also want no changes in the stations, but only in the associated load, thus leaving 0.1 probability to no modifications.

Examples

Starting from the solution represented in Figure 29, the operators will be exemplified.

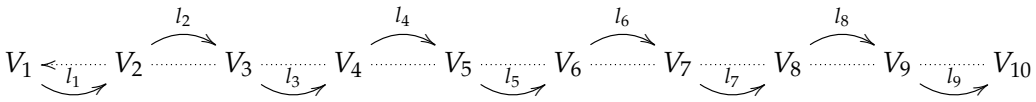


Figure 29: Practical representation of a solution

Exchange

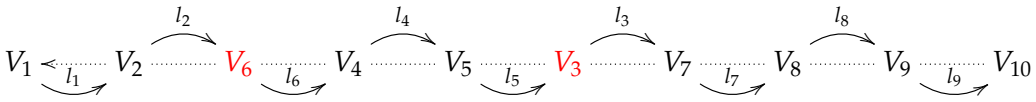


Figure 30: Practical representation of *exchange operator*

Insertion

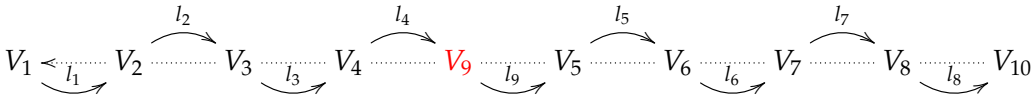


Figure 31: Practical representation of *insertion operator*

Displacement

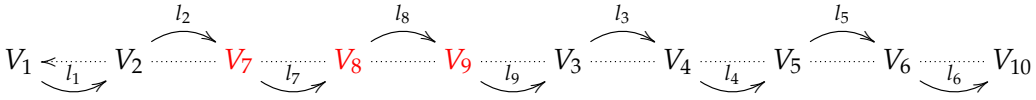


Figure 32: Practical representation of *displacement operator*

Injection

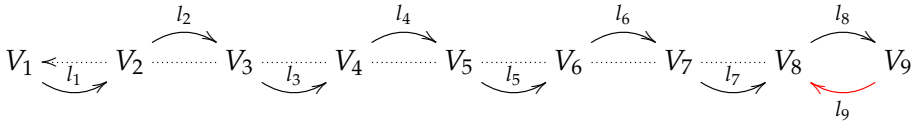


Figure 33: Practical representation of *injection operator*

Numerical Problem

The Bike Sharing Rebalancing Problem is different from the Traveling Salesman Problem because it has the mission of rebalancing a sharing system. This increases the problem difficulty but has a tremendous potential to be adapted to a mundane problem, being able to make life easier for many. To work with the loads, we needed an operator that would allow us to make small changes in values, trying to come up with new ones that would allow us to get an interesting trade-off for future solutions. Thus originated the differential operator, which chooses v values from $[1, n], v \in \mathbb{N}^+$, and adds x from $[-1, 1], x \in \mathbb{N}^+$. That is, choose a set of indexes from the vector and add or remove the values 0 or 1. Examples of this operator are shown in Figures 34 and 35.

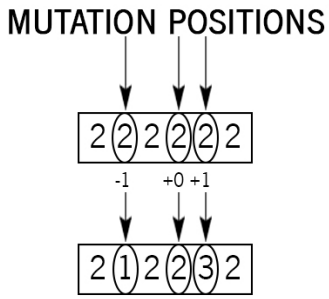


Figure 34: Differential operator

Example

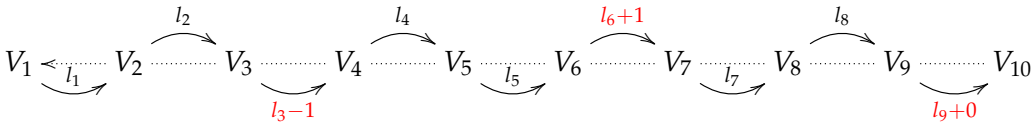


Figure 35: Practical representation of *differential operator*

4.4.3.2 *Breeding (Genetic Operators)*

In the presence of a population, rather than a single solution, the possibilities of generating new solutions increase, being able, in addition to changing the solution individually, to cross with others, taking advantage of their diversity to explore regions of the search space that otherwise it would not be possible. So, in addition to the mutation operators we just discussed, we added a crossover technique.

The crossover operator is a genetic operator that combines two solutions (parents) to produce a new one (children). The ideia behind crossover is that the children might be better than both parents, if he takes the best characteristics from each parent.

One of the most basic operators to understand and implement has been adopted in the system development. Despite its simplicity, it fits properly in this particular problem. The operator is the single point (or one point) crossover, where both parents are split at a randomly determined crossover point. Then, a new child is created by appending the first part of the first parent with the the second part of the second (Altenberg, 1995). Figure 36 depicts the single point crossover process and Karlin and Liberman (1978) describes it by:

$$R(r) = \begin{cases} 1/(L-1) & \text{if} \quad \sum_{i=1}^{L-1} |r_{i+1} - r_i| = 1, \\ 0, & \text{otherwise} \end{cases}$$

Both crossover and mutation operators occur during the evolution, according to an a priori defined probability. The probability applied to the crossover is 0.4, and the probability of a mutation is 0.6. Each mutation operator has its own associated probability (see Section 4.4.3.1).

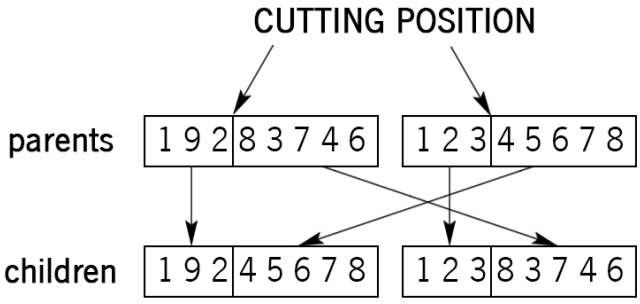


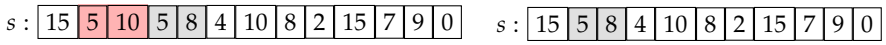
Figure 36: Single-point crossover

4.5 REPAIR

The goal of this section is not to give an exact blueprint of the way that the functions perform, but to elucidate the reader about the inherent logic.

For each optimized trip that the algorithm returns, there is a repair at the station level, ensuring that it does not make unnecessary trips. The algorithm can visit a station and do nothing (does not leave or lift vehicles). When this occurs, the repair procedure removes such unnecessary trip, therefore optimizing the travel time to the maximum.

Attached to this, as the same station can appear more than once in the solution, there is the possibility that they will appear one after the other, meaning that a trip will link to the same geographical point. When this happens, the procedure detects the occurrence, eliminating the redundancy. With the logic of the solution representation, dictating the obliviation of the first station and load.



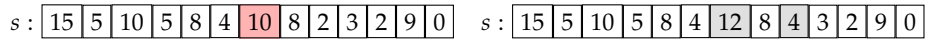
Algorithm 8 demonstrates in detail the execution of the logical repair of the solution.

Algorithm 8 Solution repair method

Input: s (solution)
- $truck\_load \leftarrow odd(s)$ ▷ Truck load throughout the journey
- $stations\_visited \leftarrow even(s)$ ▷ Stations visited throughout the journey
- $waste \leftarrow []$ ▷ Empty Array to store useless trips
- **for** $load$ **in** $truck\_load$ **do** ▷ Go through the array
  - **if** $load = next\_load$ **then** ▷ See if loads are equal
    - $waste \leftarrow append(load\_position)$ ▷ Save position in the solution array
    - $waste \leftarrow append(station\_position)$ ▷ Save also the station position
  - **end if**
- **end for**
- $s \leftarrow remove(s, waste)$ ▷ Remove useless trips from the solution
- $waste \leftarrow []$ ▷ Empty Array to store repeated stations
- **for** $station$ **in** $stations\_visited$ **do** ▷ Go through the array
  - **if** $station = next\_station$ **then** ▷ See if stations are equal
    - $waste \leftarrow append(station\_position)$ ▷ Save position in the solution array
    - $waste \leftarrow append(load\_position)$ ▷ Save also the load position
  - **end if**
- **end for**
- $s \leftarrow remove(s, waste)$ ▷ Remove useless trips from the solution

Output: s

After this, there is also a repair at the vehicle load level. The algorithm determines that a truck, on a journey, can only carry a certain number of vehicles, but in some situations, it may carry more than indicated. This adjustment deals with those situations, making the system more reliable.



Algorithm 9 demonstrates in detail the execution of the logical repair of the solution.

Algorithm 9 Load repair

Input: s (solution), t (truck), l (service levels)
  $truck\_capacity \leftarrow capacity(t)$ ▷ Truck load throughout the journey
  $situation \leftarrow demand(l, s)$ ▷ Number of vehicles left to redistribute
  $imp \leftarrow improvement(t, l, s)$ ▷ Load improvement for each station until Q
  $indexes \leftarrow (imp, l, s)$ ▷ Indexes that can be improved

  for i in $indexes$ **do** ▷ Go through the array
    $s \leftarrow reform(indexes, imp, s)$ ▷ Change indexes load
  end for
Output: s

More than logical corrections to the solution, the implemented repairs attempt to address the weaknesses that the randomness of metaheuristics bring, enriching the system with domain knowledge. While Algorithm 8 only compares the stations and loads with those immediately followed in the solution, the Algorithm 9 tries to figure out where to add/remove load throughout the solution, i.e., without fiddling with the sequence of stations that the search output delivered, tries to find a possible solution that the system may not have been able to acknowledge.

4.6 SAVE RESULTS

As we saw in Section 4.1, the final solution should deliver the final journey collection distributed by the fleet available at the time of redistribution. Its structure is theoretically represented by:

$$\mathbf{sol} = \begin{pmatrix} l_{11} & l_{12} & \dots & l_{1n} \\ l_{21} & l_{22} & \dots & l_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{t1} & l_{t2} & \dots & l_{tn} \end{pmatrix}$$

Where, t represents each trip, n the station visited and l the load that the truck has when leaving each station n. For system understanding, ease of implementation and readability, the results are stored as a matrix, composed by the vectors of the solutions generated in the loop.

Table 13: System output with final journeys

t	s_0	l_{s_0}	s_1	l_{s_1}	s_2	l_{s_2}	s_3	l_{s_3}	$\cdots$	s_{n-1}	$l_{s_{n-1}}$	s_n	l_{s_n}
1	8	15	1	5	10	10	5	8	$\cdots$	3	2	6	0
2	8	12	5	2	3	0	10	5	$\cdots$	6	0	$\times$	$\times$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t	8	0	7	4	10	0	$\times$	$\times$	$\cdots$	$\times$	$\times$	$\times$	$\times$

Where s_0 represents the depot indicator, l_{s_0} the truck load leaving the base, and so on, representing, in order, the stations to visit and the load, leaving each one of them.

4.7 STOP CRITERIA

Generally speaking, the objective is to route a fleet of vehicles to minimize the deviation of service levels. Business decisions aim to ensure that, within limited time, the possible redistributions are made. The minimization of distances and redistribution time increases operational efficiency. For instance, it permits to cut operating costs (e.g., less time spent by staff, less fuel, less maintenance vehicles) and to increase the probability of additional revenues because relocated vehicles are available sooner. Thus, the solution must satisfy the defined requirements as efficiently as possible. Therefore, the optimization system has two principles as stopping criteria: satisfaction of the service level requirements and/or compliance with the service time window constraint.

When the first event is triggered, the service will be possible within the time stipulated by the business rules. When the second event is triggered, that will not be the case. In this situation another approach is taken by the system to maximize the remaining time: to narrow down the search size to try to get another possible trip in the remaining time. This means that the system will try to choose a subset of stations to attempt to decrease the deviation of service levels as much as possible.

Algorithm 10 Stop criteria

Input: s (solution), w (time window), t (truck), l (service levels), o (operational times), r (results)
  $time \leftarrow remaining(t, w)$ ▷ Time left to break the time constraint
  $situation \leftarrow demand(l, s)$ ▷ Number of vehicles left to redistribute
  $duration \leftarrow duration(s, o)$ ▷ Total Redistribution Duration
  if $situation = 0$ **then** ▷ Check if service levels are met
    $save\_results(s, r)$ ▷ Save result
    $end\_search()$ ▷ Cease search task
  else
    if $duration \geq w$ **then** ▷ Check trip duration
      $new\_trip()$ ▷ New search with less stations to visit
    else
      $save\_results(s, r)$ ▷ Save result
    end if
  end if
Output: r

Thus, we realize that the time criteria does not force a strict stop event, but rather a change of approach to the search, trying to make the most of the system structure and behavior, to avoid waste (see Section 4.8).

The calculation of deviation of service levels is nothing less than the sum of the absolute values of the matrix column that stores the demand, iteratively, after each solution. If this sum is 0, then the need has been suppressed by the system final solution (see Table 16).

The time estimation is performed for each vehicle by the sum of the distance traveled on the journey, divided by the average truck speed along the journey, then adding the vehicles loading and unloading time (loading and unloading times set in config.yml).

An explanation of the time evaluation is presented in Section 5.3.

4.8 NEW TRIP

As long as the service level requirements are not met, or the user-defined time boundary is not surpassed (see Section 4.7), the search is not over and a new trip ought to be generated. The system has to dynamically recalculate the size of the search space and prepare the environment for a new search.

The system maintains in memory the redistribution requirements. As solutions are generated and secured, those requirements are renewed. The update logic is presented next. Table 14 represents the initial state of the system and Figure 18 is a possible initial solution.

Table 14: System initial state

Station	Demand	Need After
1	10	10
2	8	8
3	8	8
4	6	6
5	6	6
6	6	6
7	6	6
8	-30	-30
9	-15	-15
10	-5	-5

Considering a solution like the one depicted in Figure 37, the system determines the remaining necessities (represented in Table 15) and which stations are in a position to be suppressed (represented in Table 16). With the information updated, the system can continue the search (defined in Section 4.3).

s :

8	15	1	5	10	10	5	8	2	2	7	0	9	15	4	9	3	2	6	0

Figure 37: Possible solution

Table 15: System after solution

Station	Demand	Need After
1	10	0 [10+(5-15)]
2	8	2 [8+(2-8)]
3	8	1 [8+(2-9)]
4	6	0 [6+(9-15)]
5	6	4 [6+(8-10)]
6	6	4 [6+(0-2)]
7	6	4 [6+(0-2)]
8	-30	-15 [-15+(15-0)]
9	-15	0 [-15+(15-0)]
10	-5	0 [-5+(10-5)]

Table 16: System new state

Station	Demand	Need After
1	0	0
2	2	2
3	1	1
4	0	0
5	4	4
6	4	4
7	4	4
8	-15	-15
9	0	0
10	0	0

4.9 OUTPUT RESULTS

In the end, the system should return, not only the trips (t) that must be performed by the truck drivers but also complementary information, to clarify the final user of the quality of the solution. As for the representation of the stations to be visited, they will be delivered to the user in their true identity, not as the system represents in the search phase. Moreover, it will be transmitted information regarding the individual distance for each route, the total and individual duration, how many trips each truck makes, as well as the truck (v) license plate. This information is crucial to the redistribution operator.

Apart from this, relevant information will also be presented for the business itself, such as which vehicles are to be redistributed and how many stations are to be visited at the end of each route (and at the end of redistribution, if applicable). Table 17 shows the final results presented to the user.

Table 17: System deliverable

v	t	s_0	l_{s_0}	s_1	l_{s_1}	s_2	l_{s_2}	s_3	l_{s_3}	$\cdots$	s_{n-1}	$l_{s_{n-1}}$	s_n	l_{s_n}
1	1	8	15	1	5	10	10	5	8	$\cdots$	3	2	6	0
2	2	8	12	5	2	3	0	10	5	$\cdots$	6	0	$\times$	$\times$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2	t	8	0	7	4	10	0	$\times$	$\times$	$\cdots$	$\times$	$\times$	$\times$	$\times$

$\mathbf{c}_j$	*Plate*	*Length*	*Duration*	*Trip*	*Missed*	*Stations*
$\mathbf{c}_1$	*XYZ*	10033.3	13.37	1	20	8
$\mathbf{c}_2$	*ZYX*	8093.8	10.72	1	12	5
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{c}_k$	*ZYX*	2100.2	2.80	*nm*	0	0
Total	–	L	D	–	M	S

5

EXPERIMENTAL RESULTS AND DISCUSSION

To try to understand if the approaches proposed in Chapter 4 make sense and are useful for the business, some experiments are reviewed and presented, opposing approaches and results.

This chapter is divided into two sections.The first one describes the advantages of using some domain knowledge in the generation of solutions, both in local and population-based optimization. Different weights were also tested for the evaluation function. In the second one, two analysis scenarios are compared by contrasting the results of different proposed algorithms.

5.1 SOLUTION GENERATION APPROACHES AND WEIGHT EXPERIMENTATION

At this stage we will not restrict the process by a time limit, allowing the optimization task to execute until the service levels are assured. The operational times will also not be accounted for, as these will only make sense for the business and not for this study. Therefore, for each solution it is considered the total duration of the redistribution, the distance covered in meters, and the execution time of the optimization task. The tests will run 10 times for each algorithm, and an average result will be displayed.

5.1.1 *Weight Experimentation*

As we saw in Section 4.4.1, the evaluation function is given by

$$\underset{X}{\text{minimize}} \quad d(x) + \alpha r(x) - \beta f(x) \tag{6}$$

where α and β values are defined weights, establishing a Weighted-Formula Approach for a Multi-Objective Optimization (Cortez, 2014). First, we will consider the weights of the evaluation function. After testing several scenarios based on intuition, three promising weight scenarios were selected and executed for comparison purposes. Their values are represented in Table 18 and search results are described in Table 19.

Table 18: Weights for each Scenario

values	α	β
Scenario I	1100	700
Scenario II	1500	100
Scenario III	1500	1000

Table 19: Results of the weighted evaluation function experimentation (best values in bold)

Scenario I	Length	Duration	Execution
Hill Climbing	21750.08	16.81	17.74
Simulated Annealing	21670.51	16.87	**15.34**
Evolutionary Algorithms	22617.71	18.40	16.21
Scenario II			
Hill Climbing	21712.66	16.87	17.89
Simulated Annealing	**21521.51**	16.53	18.28
Evolutionary Algorithms	21605.63	**16.50**	16.42
Scenario III			
Hill Climbing	21460.20	16.83	16.85
Simulated Annealing	21783.91	17.52	14.68
Evolutionary Algorithms	22043.42	17.63	15.99

As we can see from Table 19 values, *scenario II* presented the best results, both in terms of distance traveled, as well as in travel time, distributed by the two distribution vehicles (Q = 15). Thus, the values to consider for the evaluation function weights are 1500 for α and 100 for β. Figures 38, 39 and 40 represent graphically the values from Table 19.

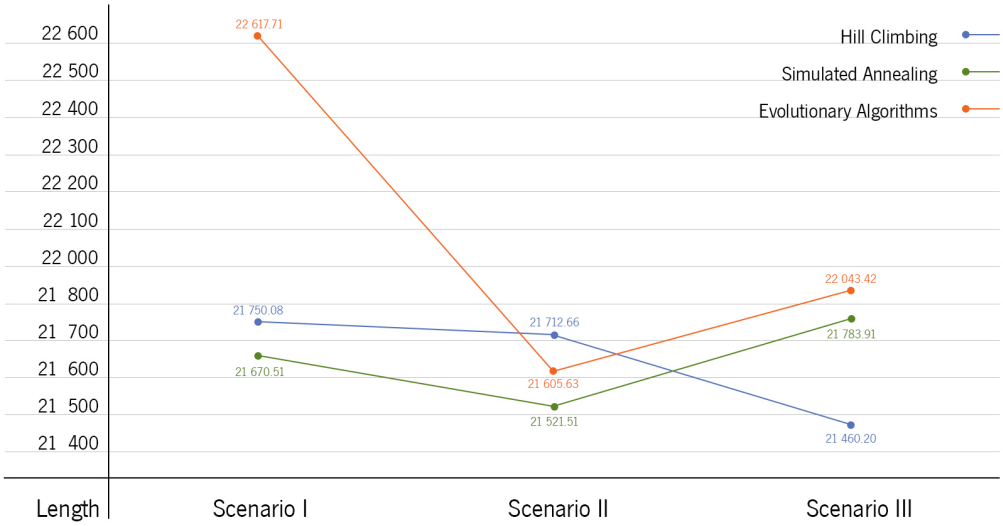


Figure 38: Length for different weight scenarios

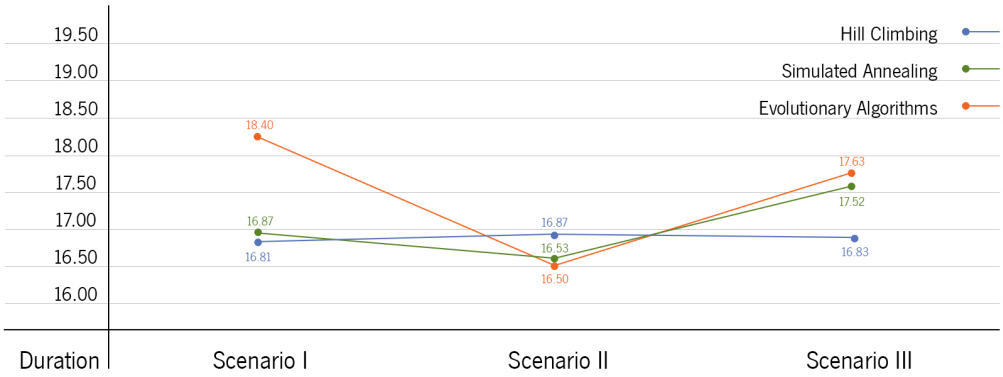


Figure 39: Duration for different weight scenarios

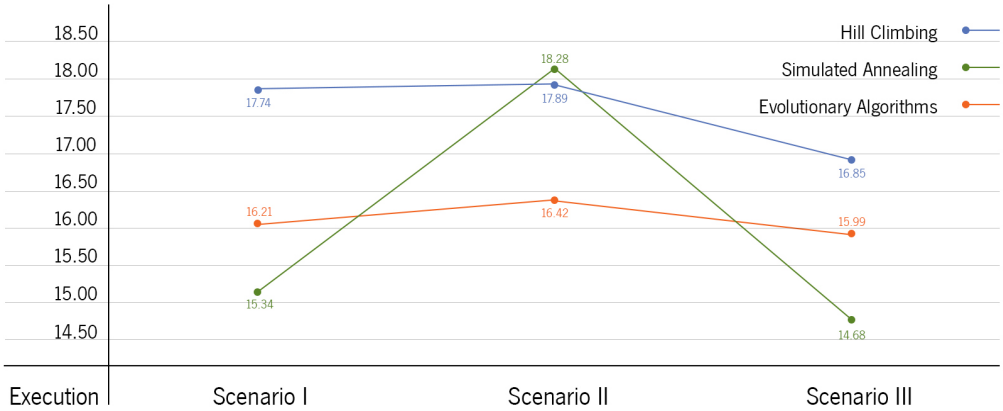


Figure 40: Execution time for different weight scenarios

5.1.2 *Initial Solution/Population Experimentation*

Local Based Approach Experimentation

Two algorithms were implemented from those mentioned and explained in Section 2.3.2: Simulated Annealing and Hill Climbing. With the same principles but with different behavior, we are going to examine which one best fits the issue at hand. First, we try to understand the impact that the custom generation of initial solutions have in the search for solutions. These results are compared with the previously obtained results, described in the Table 19. These values were obtained following the methodology for custom solution generation promoted by this research work (see Section 4.4.2.1).

When the initial solution generation is random, the search begins at a random starting point within the search space. That position can be good or bad. By not using domain knowledge, we are using a completely random initial search position. However, the search task could benefit from the utilization of a much more attractive initial point by assuming domain knowledge.

Table 20: Optimization results for the single-state algorithms using random and customized generation of initial solutions (best values in bold)

Random Generation	Length	Duration	Execution
Hill Climbing	29373,54	21.34	25.56
Simulated Annealing	28914.40	20.45	24.21
Custom Generation			
Hill Climbing	21712.66	16.87	**17.89**
Simulated Annealing	**21521.51**	**16.53**	18.28

Table 20 confirms the gain of using the methodology presented in this document, being advantageous in all the assessment metrics considered: the length (*m*), the duration (*min*) of the journey, and the execution time of the search task (*sec*).

Population Based Approach Experimentation

Above we look at the benefits of generating a single initial solution, but when the search algorithm considers an initial population, does the benefit remain? As described in Section 2.3.2.2, population-based algorithms, such as the Evolutionary Algorithms implemented in this paper, consider several solutions as a starting point.

The importance of setting the initial population in meaningful places of the search space can become even more fundamental than in single-state approaches. The utilization of various initial points in the search space allows to explore diverse interesting and relevant areas of the problem and not to restrict to a specific area. Hence, as we also saw in Section 4.4.2.2, the strategy adopted for early population generation is to consider multiple domain knowledge applications to explore most of the search space, in order to find a solution flexible enough to adapt to different instances of the problem.

Table 21: Optimization results for the population-based algorithms using random and customized generation of the initial population (best values in bold)

Random Generation	Length	Duration	Execution
Evolutionary Algorithms	28644.75	22.463	24.15
Custom Generation			
Evolutionary Algorithms	**21605.63**	**16.49**	**16.43**

Table 21 presents the results obtained when considering the strategy adopted in this paper (see Section 4.4.2.2) and also results not using domain knowledge, confirming that for this problem, the randomness in solution generation is outweighed by the benefits of biased solutions.

These results confirm that the customized method for generating initial solutions results in enhanced optimizations for both single-state and population-based methods.

5.2 SYSTEM EXPERIMENTATION

This section presents the experiments executed for different scenarios to verify the versatility and flexibility of the system to meet the many challenges of the real sharing service.

A major business concern is the conformity with a time window. Therefore, system rules had to be considered to make the most of the available time because time may not be enough to reestablish service levels. Thus, the system follows two principles:

- the initial load for each truck must be packed when the rebalancing begins and
- for each truck's last trip, the travel time back to the depot is not accounted for.

In this study only two scenarios will be presented here for analysis purposes. For reasons of compliance and understanding, the instance that underpins the scenarios is the one previously described in Section 3.1.1. *Scenario I* is the same as the one used by the author for the solution proposed in Figure 9, where two trucks with capacity for 15 vehicles are considered ($Q = 15$). For *scenario II*, only one truck is considered, also with capacity for 15 vehicles ($Q = 15$).

The results of the experiment are represented in Table 22, with each value being the result of the average from 10 observations. Each scenario environment can be revisited in *A*.

The time considered for redistribution, agreed with Nuno Oliveira, was 1 *hour* (60 *min*), with the operational loading/unloading time of 1 *min* for each vehicle.

Table 22: Optimization results for system trial (best values (for each scenario) in bold)

Scenario I	Length	Duration	Missed	Stations	Execution
Hill Climbing	**21737.86**	58.14	0	0	**31.51**
Simulated Annealing	22389.75	**57.27**	0	0	51.63
Evolutionary Algorithms	22033.04	58.75	0	0	49.46
Scenario II					
Hill Climbing	**11373.31**	58.42	22.1	5.9	**36.09**
Simulated Annealing	11987.03	**57.19**	20.2	5.8	59.56
Evolutionary Algorithms	12085.54	58.57	**20**	**5.4**	36.91

Table 22 shows that the utilization of only one truck to redistribute, conditions the main business goal of meeting the service level requirements. In general, the search algorithm is able to find a solution that enables effective system rebalancing. However, the organizational infrastructure has to be aligned with the requirements of the problem.

When the service level requirements have been met, the metrics considered and analyzed to choose the best solution and/or algorithm are the distance traveled and the duration of the redistribution, taking into account the operational time. According to Nuno Oliveira, the execution of the redistribution in the shortest possible time and distance is a big competitive advantage because it reduces costs of relocation (e.g., fuel, staff, maintenance vehicles) and monetize resources (e.g., vehicles are available sooner). Based on the same logic, we can understand that when it is not possible to meet the requirements, the most important metrics should be the vehicles that have not been redistributed and the stations that have remained in need.

In *scenario I*, although the Hill Climbing algorithm has the best result in terms of distance traveled during redistribution, Simulated Annealing has a notable advantage in averaging redistribution duration. All of them satisfy the requirements, but 1 minute can be the difference between losing a customer or not.

In *scenario II*, the results tend to give an advantage to Evolutionary Algorithms. This scenario is undoubtedly more demanding in terms of deliverables and is perhaps a more realistic scenario in terms that the conditions for full redistribution cannot always be met. Although it takes longer, and travel further, the solutions proposed by the Evolutionary Algorithms can, on average, satisfy more stations than other algorithms.

We realize that to always ensure that the best possible solution is found, we must guarantee that multiple searches are generated, various algorithms are tested and that the viability of the final output is considered. The user should be able to decide which one is the best according to their needs. The display of diverse metrics is valuable to support the user in their decision.

5.3 DISCUSSION

5.3.1 *Author Defined Solution*

To further analyze the system, we will try to contrast the system solutions with the solution defined by the author of this thesis, represented in Figure 9. It is important to realize that the handmade solution took a long time, demonstrating the importance of this type of system. A more complete assessment of the utility of this system can be done by comparing with the author's solution. Table 23 demonstrates the solution of Figure 9 in the system deliverable scheme.

Table 23: Author defined solution

v	t	l_{s_0}	s_1	l_{s_1}	s_2	l_{s_2}	s_3	l_{s_3}	s_4	l_{s_4}	s_5	l_{s_5}	s_6	l_{s_6}	s_7	l_{s_7}
1 ←	1	15	3	7	6	1	4	0	9	15	4	10	7	4	2	0
2 ←	2	15	5	9	10	14	2	10	1	0	×	×	×	×	×	×

$\mathbf{c}_j$	*Plate*	*Length*	*Duration*	*Trip*	*Missed*	*Stations*
$\mathbf{c}_1$	*XYZ*	12856.56	58.81	1	0	0
$\mathbf{c}_k$	*ZYX*	9375.44	33.36	1	0	0
Total	–	22232	58.81	–	0	0

The first trip has considerable size and its duration is very close to the $60min$ limit. We are going to consider it to understand how the *duration* metric is evaluated. The second trip takes less time because it is shorter, with fewer vehicles to redistribute. This one will not be analyzed in detail.

The first trip (t_1) is represented by:

$$s : \boxed{15}\boxed{3}\boxed{7}\boxed{6}\boxed{1}\boxed{4}\boxed{0}\boxed{9}\boxed{15}\boxed{4}\boxed{10}\boxed{7}\boxed{4}\boxed{2}\boxed{0}$$

$$stations\_visited : \boxed{3}\boxed{6}\boxed{4}\boxed{9}\boxed{4}\boxed{7}\boxed{2} \qquad truck\_load : \boxed{15}\boxed{7}\boxed{1}\boxed{0}\boxed{15}\boxed{10}\boxed{4}\boxed{0}$$

Figure 41 shows the calculation of the operational time for the first trip (t_1), considering $1min$ for vehicle loading/unloading. As explained earlier, we consider that the first loading of each truck do not count as operational time.

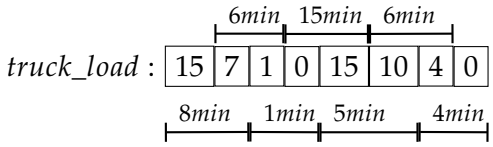


Figure 41: Operational time from t_1

Thus, the operational time is given by the sum of the difference between n and $n+1$ values. Therefore:

$$operational\_time = (15-7) + (7-1) + (1-0) + |(0-15)| + (15-10) + (10-4) + (4-0)$$

$$operational\_time = (8) + (6) + (1) + (15) + (5) + (6) + (4)$$

$$operational\_time = 45$$

As we can see from the operational time required for only one route, the travel time must be reduced to enable a redistribution that respects the time window stipulated by the user. Considering the time window of 60 minutes, there are only 15 minutes left to spend driving between stations.

Considering an average speed of $45km/h$ ($750m/min$) along the entire route, we can translate a maximum distance that the trip must meet.

$$max\_distance = remain\_time \times average\_speed$$

$$max\_distance = 15 \times 750$$

$$max\_distance = 11250$$

Looking at the Table 23 we notice that, even though it is close, the distance traveled during redistribution (12856.56 m) exceeds 11250 of the remaining time. The elapsed travel time is then:

$$travel\_time = \frac{trip\_length}{average\_speed}$$

$$travel\_time = \frac{12856.56}{750}$$

$$travel\_time = 17.14$$

Knowing that the duration will exceed the expected, the total time ($duration$) is given by:

$$total\_time = operational\_time + travel\_time$$

$$total\_time = 45 \times 17.14$$

$$total\_time = 62.5$$

As already mentioned, the return trip from the last journey of each truck does not count to the redistribution time (a measure adopted to optimize the algorithm performance and increase the chances of finding better solutions). So, the distance between the last station and the depot does not add to the journey duration, only for the distance traveled. Therefore, we have to exclude the distance between 2 (last station) and 8 (identifier of the depot).

$$return\_trip = \frac{trip\_length(ind, depot)}{average\_speed}$$

$$duration = total\_time - return\_trip$$

$$duration = 62.5 - \frac{2770.373}{750}$$

$$duration = 62.5 - 3.69$$

$$duratioin = 58.81$$

With the thoroughness of the problem, it becomes a very difficult task for a human to perform quickly and effectively. All possible combinations make each instance a complex challenge.

5.3.2 *System Solution*

The processing speed of a computer is always higher than human work particularly for small, specific and repetitive tasks. Therefore, a system that performs these sorts of tasks is always a valuable asset for the organization and the general welfare.

Table 24 presents a solution generated by Evolutionary Algorithms, where the deliverable provides a feasible and better solution than the presented by the author of the thesis, shown in Table 23. This solution visits one less station on the first trip, managing to make a trade-off with the second truck, improving the overall response time.

Table 24: System solution

v	t	l_{s_0}	s_1	l_{s_1}	s_2	l_{s_2}	s_3	l_{s_3}	s_4	l_{s_4}	s_5	l_{s_5}	s_6	l_{s_6}
1 ←	1	15	6	9	1	0	9	15	4	9	1	8	3	0
2 ←	2	15	7	9	2	1	10	6	5	0	×	×	×	×

$\mathbf{c}_j$	*Plate*	*Length*	*Duration*	*Trip*	*Missed*	*Stations*
$\mathbf{c}_1$	*XYZ*	11074.32	56.16	1	0	0
$\mathbf{c}_k$	*ZYX*	9507.44	34.54	1	0	0
Total	–	20581.76	56.16	–	0	0

In less than a minute, the metaheuristics were able to test thousands of different combinations, saving time, stress and resources to the organization.

Figures 42 and 43 represent the routes that redistribute the service. Comparing the images, we can see that the system output (Figure 42) trips are 'cleaner' and more direct than those of the author's solution (Figure 43). This again demonstrates the significance of implementing such solutions, confirming their value to organizations.

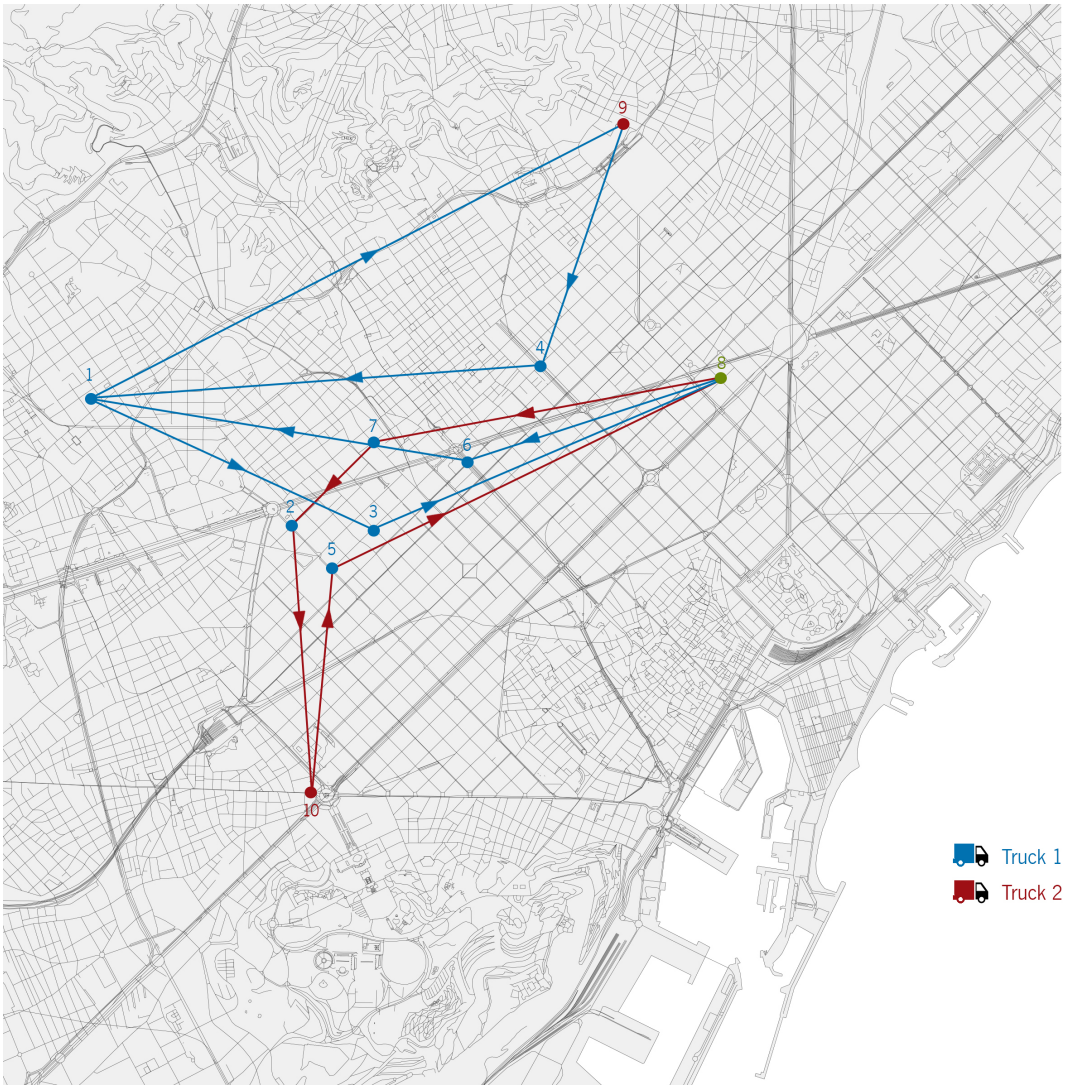


Figure 42: System solution representation on Barcelona map

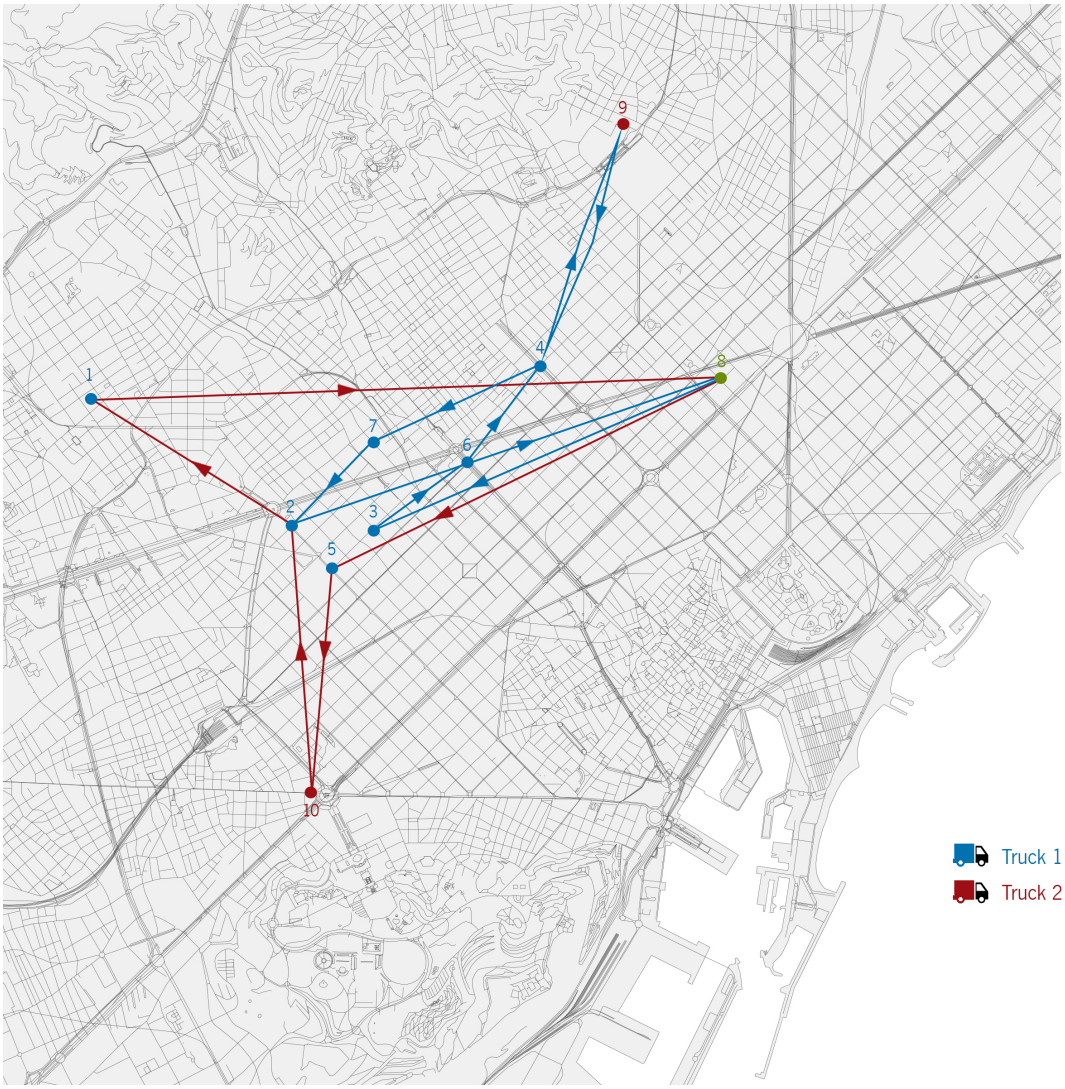


Figure 43: Author's solution representation on Barcelona map

6

CONCLUSIONS AND FUTURE WORK

This chapter presents the final aspects of this dissertation and briefly describes the work performed, the main conclusions some limitations that arose during its development and aspects to be addressed in the future.

6.1 SYNOPSYS

In this project, the goal was to test several modern optimization methods (metaheuristics) in the search for (sub)optimal solutions for redistribution route(s) of a sharing service. A decision support system was developed to meet this end.

The thesis introduced first the conceptual structure of what is a decision support system, defining it in the light of the most renowned authors in the area, converging to the project expectations and what type of decision support system will derive from the project development. Next, the Smart Mobility thematic is presented, referring to the motivations behind this project and framing it in the current mobility arena. More than that, it goes into detail about what sharing services are and their importance, not only in terms of mobility in urban areas but also about their environmental relevance and tendency to be spread.

To link the concepts, a holistic review is made, trying to elucidate the reader about basic concepts of optimization algorithms (operational research and metaheuristics). Then, a state of the art is presented, where relevant related studies are scrutinized.

After reviewing the concepts and the state of the art, the development of the system is detailed, trying to make as clear as possible the thinking behind all the development and implementation. The structure of the implementation and the flow of the system is detailed, presenting, step by step, the structure of the system, with practical examples supporting the theory, graphs and matrices that theoretically represent the solution, to the final deliverable, specifying the algorithms and their behavior in the search for the sub(optimal) solution.

Two different metaheuristic approaches were considered when modeling this problem; local and population-based search. In the first one, Hill Climbing and Simulated Annealing were implemented, and in the second, Evolutionary Algorithms were considered. These

two approaches have completely different behavior. Local based only consider one solution at a time and upgrade it until they find the best solution, considering a finite set of attempts. Population-based consider a set of solutions and initials and make adjustments to chosen solutions of the population, and cross-referencing each other. These are two different concepts that allow an interesting analysis.

Finally, we have the section with experimental results and their discussion, where several scenarios are proposed and analyzed, so that best practices can converge to a final interpretation, therefore testing the system, opposing the system output with the solution designed by the system developer.

The results were very satisfactory considering the tailor-made procedures for creating, altering and repairing solutions. All of these combinations offer the system an integrated and realistic program to design and evaluate solutions. With all the structure assembled, all algorithms can generate satisfactory results, with no significant averaging differences between optimization approaches. Even so, it is proposed that all algorithms are considered for each instance, leaving the final choice for the system user.

6.2 DISCUSSION

In a booming market, where many organizations try to achieve leadership and stand out from the competition, having a competitive advantage is not always easy. Owning something unique is difficult and innovating in such a saturated market is even more difficult. The investment in research is therefore crucial. In the case of this project, where the sharing service itself is based on a purely flexible structure, innovation and research play a critical role in the survival of the business.

A system like vehicle sharing has costs that can fluctuate intensely, depending on a group of variables, like the population density of the area, the fleet size or the system itself. These sharing services need some sort of system that offer some aid to the management when it comes to decision making, allowing them to have a clearer view of the day to day scenarios.

The execution of the redistribution in the shortest possible time and distance is a big competitive advantage because it reduces costs of relocation (e.g., fuel, staff, maintenance vehicles) and monetize resources (e.g., vehicles are available sooner). Therefore, a robust, cohesive and adaptable decision support system was designed to support a sharing system business, regardless of its features. With minimal code adaptation required, it is possible to migrate the system to integrate existing decision support platforms.

Studying the developments of the field of study it was possible to observe the notorious lack of robust, dynamic and scalable solutions, most of them being only the target of theoretical study. The development of this type of system/application is nonexistent in the market, which gives even more relevance to this project. Even so, most studies never

dynamically considered the system as a whole. The final solution was always based on assumptions like all trucks have the same capacity, or that each truck has to visit all stations on each trip. The greatest value of the proposed system is that the infrastructure is adapted to accept solutions (routes) that can visit the same station multiple times on the same trip and that it is not required to visit all, being able to make interesting trade-offs between trucks and/or between trips. These are two differentiating features that give this system unique performance, resulting in very interesting outputs.

The experimental results showed that the system output can be more relevant than a handmade solution. In less than a minute, the system can present the user a viable, all-in-one solution, featuring multiple evaluation metrics for complete transparency.

As for the performance of search algorithms (metaheuristic), none stood out holistically, i.e., each stood out for some specific metric or scenario. Hill Climbing algorithm has the best result in terms of distance traveled during redistribution, Simulated Annealing has a notable advantage in averaging redistribution duration and, although it takes longer, and travel further, the solutions proposed by the Evolutionary Algorithms can, on average, satisfy more stations than other algorithms (when the time limit for redistribution does not allow the fulfillment of all service level requirements).

Even so, the developed system is not perfect. This project considered a weighted-formula approach. This is not the most advantageous approach when working with multi-objective optimization problems. This approach was considered due to its simplicity of implementation and flexibility in handling the output. It was the most straightforward strategy for incremental integration, as defined early in the project. Because of that, interesting and valuable trade-offs could have been lost.

It is noteworthy the importance that the open-source tool R had during this project. The implementation flexibility it provided ensured the success of the project.

6.3 FUTURE WORK

Despite the system developed in this thesis being a step forward in the development of decision support applications for vehicle sharing services, there is still a very long road to walk. This project is academic but has the ambition to overcome this barrier and be applied in a real environment of sharing services. As it is, it can be deployed, requiring maintenance and support, at least for data refreshment. Nevertheless, some improvements can be made by adapting the system to more complex business rules, trying to take advantage of demand behavior, always trying to lose as few customers as possible.

Listed below are some improvements which might enhance the performance and usability of the system:

- integrate the system with the Decision Support Systems of diverse services worldwide, implemented by CEiiA;
- add the weight of *station utilization rate* to the evaluation function for interesting trade-offs and more realistic and business-friendly deliverable;
- integrate the calculation of the service level requirements in the system, creating a system that is able to feed itself, automating its execution; and
- build a Graphical User Interface for the system to work independently.

BIBLIOGRAPHY

Ait-Ouahmed, A., Josselin, D., and Zhou, F. Relocation optimization of electric cars in one-way car-sharing systems: modeling, exact solving and heuristics algorithms. *International Journal of Geographical Information Science*, 32(2):367–398, September 2017. doi: 10.1080/13658816.2017.1372762. URL https://doi.org/10.1080/13658816.2017.1372762.

Allwood, J. M. and Cullen, J. M. *Sustainable Materials - With Both Eyes Open (Without the Hot Air)*. UIT Cambridge Ltd., 2012. ISBN 190686005X. URL https://www.amazon.com/Sustainable-Materials-Both-Eyes-Without/dp/190686005X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=190686005X.

Altenberg, L. The schema theorem and price's theorem. In *Foundations of Genetic Algorithms*, pages 23–49. Elsevier, 1995. doi: 10.1016/b978-1-55860-356-1.50006-6. URL https://doi.org/10.1016/b978-1-55860-356-1.50006-6.

Barth, M. and Todd, M. Simulation model performance analysis of a multiple station shared vehicle system. *Transportation Research Part C: Emerging Technologies*, 7(4):237–259, August 1999. doi: 10.1016/s0968-090x(99)00021-2. URL https://doi.org/10.1016/s0968-090x(99)00021-2.

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., and Schiavinotto, T. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*, 5(1):91–110, December 2006. doi: 10.1007/s10852-005-9033-y. URL https://doi.org/10.1007/s10852-005-9033-y.

Blum, C. and Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

Borgnat, P., ABRY, P., FLANDRIN, P., ROBARDET, C., ROUQUIER, J.-B., and FLEURY, E. SHARED BICYCLES IN a CITY: A SIGNAL PROCESSING AND DATA ANALYSIS PERSPECTIVE. *Advances in Complex Systems*, 14(03):415–438, June 2011. doi: 10.1142/s0219525911002950. URL https://doi.org/10.1142/s0219525911002950.

Boyacı, B., Zografos, K. G., and Geroliminis, N. An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational*

Research, 240(3):718–733, February 2015. doi: 10.1016/j.ejor.2014.07.020. URL https://doi.org/10.1016/j.ejor.2014.07.020.

Brinkmann, J., Ulmer, M. W., and Mattfeld, D. C. Inventory routing for bike sharing systems. *Transportation Research Procedia*, 19:316–327, 2016. doi: 10.1016/j.trpro.2016.12.091. URL https://doi.org/10.1016/j.trpro.2016.12.091.

Bude, p. Global smart infrastructure—smart city transformation 2016. 2016.

Caggiani, L. and Ottomanelli, M. A modular soft computing based method for vehicles repositioning in bike-sharing systems. *Procedia - Social and Behavioral Sciences*, 54:675–684, October 2012. doi: 10.1016/j.sbspro.2012.09.785. URL https://doi.org/10.1016/j.sbspro.2012.09.785.

Caggiani, L. and Ottomanelli, M. A dynamic simulation based model for optimal fleet repositioning in bike-sharing systems. *Procedia - Social and Behavioral Sciences*, 87:203–210, October 2013. doi: 10.1016/j.sbspro.2013.10.604. URL https://doi.org/10.1016/j.sbspro.2013.10.604.

Chemla, D., Meunier, F., and Calvo, R. W. Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization*, 10(2):120–146, May 2013. doi: 10.1016/j.disopt.2012.11.005. URL https://doi.org/10.1016/j.disopt.2012.11.005.

Chiariotti, F., Pielli, C., Zanella, A., and Zorzi, M. A dynamic approach to rebalancing bike-sharing systems. *Sensors*, 18(2):512, February 2018. doi: 10.3390/s18020512. URL https://doi.org/10.3390/s18020512.

Chow, J. Y. and Sayarshad, H. R. Symbiotic network design strategies in the presence of coexisting transportation networks. *Transportation Research Part B: Methodological*, 62:13–34, April 2014. doi: 10.1016/j.trb.2014.01.008. URL https://doi.org/10.1016/j.trb.2014.01.008.

Contardo, C., Morency, C., Rousseau, L., and Centre interuniversitaire de recherche sur les réseaux d'entreprise, l. l. e. l. t. *Balancing a Dynamic Public Bike-sharing System*. CIRRELT (Collection). CIRRELT, 2012. URL https://books.google.pt/books?id=QEZBDQEACAAJ.

Correia, G. and Antunes, A. P. Optimization approach to depot location and trip selection in one-way carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):233–247, January 2012. doi: 10.1016/j.tre.2011.06.003. URL https://doi.org/10.1016/j.tre.2011.06.003.

Cortez, P. *Modern Optimization with R*. Springer International Publishing, 2014. doi: 10.1007/978-3-319-08263-9. URL https://doi.org/10.1007/978-3-319-08263-9.

Dell'Amico, M., Hadjicostantinou, E., Iori, M., and Novellani, S. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega*, 45:7–19, June 2014. doi: 10.1016/j.omega.2013.12.001. URL https://doi.org/10.1016/j.omega.2013.12.001.

Docherty, I. New governance challenges in the era of 'smart' mobility. In *Governance of the Smart Mobility Transition*, pages 19–32. Emerald Publishing Limited, March 2018. doi: 10.1108/978-1-78754-317-120181002. URL https://doi.org/10.1108/978-1-78754-317-120181002.

Dror, M., Fortin, D., and Roucairol, C. Redistribution of Self-service Electric Cars: A Case of Pickup and Delivery. Research Report RR-3543, INRIA, 1998. URL https://hal.inria.fr/inria-00073142. Projet PRAXITELE.

Fan, W. D., Machemehl, R. B., and Lownes, N. E. Carsharing. *Transportation Research Record: Journal of the Transportation Research Board*, 2063(1):97–104, January 2008. doi: 10.3141/2063-12. URL https://doi.org/10.3141/2063-12.

Froehlicj, J., Neumann, J., and Oliver, N. Sensing and pedicting the pulse of the city through shared bicycling. 2008.

Gavalas, D., Konstantopoulos, C., and Pantziou, G. *Design and management of vehicle-sharing systems: A survey of algorithmic approaches*, pages 261–289. 12 2016. ISBN 9780128034545. doi: 10.1016/B978-0-12-803454-5.00013-4.

Gel'fand, I. M. *Lectures on Linear Algebra (Dover Books on Mathematics)*. Dover Publications, 1989. ISBN 0486660826. URL https://www.amazon.com/Lectures-Linear-Algebra-Dover-Mathematics/dp/0486660826?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0486660826.

Ghosh, S., Varakantham, P., Adulyasak, Y., and Jaillet, P. Dynamic repositioning to reduce lost demand in bike sharing systems. *Journal of Artificial Intelligence Research*, 58:387–430, February 2017. doi: 10.1613/jair.5308. URL https://doi.org/10.1613/jair.5308.

Gong, Y.-J., Zhang, J., Liu, O., Huang, R.-Z., Chung, H. S.-H., and Shi, Y.-H. Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(2):254–267, March 2012. doi: 10.1109/tsmcc.2011.2148712. URL https://doi.org/10.1109/tsmcc.2011.2148712.

Goodall, W., Tiffany Dovey, F., Bornstein, J., and Bonthron, B. The rise of mobility as a service. 2017.

Hemdersom, J. and Fishman, A. Divvy: Helping chicago's new bike share find its balance. *Data Science for Social Good*, 2013.

Hietanen, S. Mobility as a service - the new transport model? 2014.

Hillier, F. S. and Liberman, G. J. *Introduction to Operations Research*. McGraw-Hill, 2014. ISBN 0073523453. URL https://www.amazon.com/Introduction-Operations-Research-Frederick-Hillier/dp/0073523453?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0073523453.

ITF-CPB. Transition to shared mobility: How large cities can deliver inclusive transport services. page 54, May 2017.

Karlin, S. and Liberman, U. Classifications and comparisons of multilocus recombination distributions. *Proceedings of the National Academy of Sciences of the United States of America*, 75(12):6332–6336, 1978. ISSN 00278424. URL http://www.jstor.org/stable/68958.

Keen, P. G. W. and Scott-Morton, M. S. *Decision Support Systems: An Organizational Perspective (Addison-Wesley series on decision support)*. Addison-Wesley, 1978. ISBN 0201036673. URL https://www.amazon.com/Decision-Support-Systems-Organizational-Addison-Wesley/dp/0201036673?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0201036673.

Kek, A. G., Cheu, R. L., Meng, Q., and Fung, C. H. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):149–158, January 2009. doi: 10.1016/j.tre.2008.02.008. URL https://doi.org/10.1016/j.tre.2008.02.008.

Kleinberg, J. and Tardos, E. *Algorithm Design*. Pearson, 2006. ISBN 9780321295354. URL https://www.amazon.com/Algorithm-Design-Jon-Kleinberg/dp/0321295358?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321295358.

Lei, H., Laporte, G., and Guo, B. The vehicle routing problem with stochastic demands and split deliveries. *INFOR: Information Systems and Operational Research*, 50(2):59–71, May 2012. doi: 10.3138/infor.50.2.059. URL https://doi.org/10.3138/infor.50.2.059.

Lin, J.-R. and Yang, T.-H. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):284–294, March 2011. doi: 10.1016/j.tre.2010.09.004. URL https://doi.org/10.1016/j.tre.2010.09.004.

Lin, S. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, Dec 1965. doi: 10.1002/j.1538-7305.1965.tb04146.x.

Luenberger, D. G. and Ye, Y. *Linear and Nonlinear Programming*. Springer US, 2008. doi: 10.1007/978-0-387-74503-9. URL https://doi.org/10.1007/978-0-387-74503-9.

Luke, S. *Essentials of Metaheuristics (Second Edition)*. lulu.com, 2013. ISBN 1300549629. URL https://www.amazon.com/Essentials-Metaheuristics-Second-Sean-Luke/dp/1300549629?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1300549629.

Mak, K. and Guo, Z. A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows. In *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium, 2004.* IEEE, 2004. doi: 10.1109/sieds.2004.239880. URL https://doi.org/10.1109/sieds.2004.239880.

Marinakis, Y., Marinaki, M., and Dounias, G. A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 23(4):463–472, June 2010. doi: 10.1016/j.engappai.2010.02.002. URL https://doi.org/10.1016/j.engappai.2010.02.002.

Marinakis, Y., Iordanidou, G.-R., and Marinaki, M. Particle swarm optimization for the vehicle routing problem with stochastic demands. *Applied Soft Computing*, 13(4):1693–1704, April 2013. doi: 10.1016/j.asoc.2013.01.007. URL https://doi.org/10.1016/j.asoc.2013.01.007.

Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996. ISBN 3540606769. URL https://www.amazon.com/Genetic-Algorithms-Structures-Evolution-Programs/dp/3540606769?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540606769.

Michalewicz, Z. *Adaptive Business Intelligence*. Springer, 2007. ISBN 3540329285. URL https://www.amazon.com/Adaptive-Business-Intelligence-Zbigniew-Michalewicz/dp/3540329285?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540329285.

Michalewicz, Z. and Fogel, D. B. *How to Solve It: Modern Heuristics*. Springer, 2004. ISBN 3540660615. URL https://www.amazon.com/How-Solve-Heuristics-Zbigniew-Michalewicz/dp/3540660615?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540660615.

Michalewicz, Z., Schmidt, M., Michalewicz, M., and Chiriac, C. *Adaptive Business Intelligence*. Springer, 2006. ISBN 3540329285. URL https://www.amazon.com/Adaptive-Business-Intelligence-Zbigniew-Michalewicz/dp/3540329285?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540329285.

Mitchell, W. Mobility on demand. 2008.

Moura, L. Introduction to the theory of np-completeness. page 66, 2006.

Mulley, C. Mobility as a services (MaaS) – does it have critical mass? *Transport Reviews*, 37 (3):247–251, March 2017. doi: 10.1080/01441647.2017.1280932. URL https://doi.org/10.1080/01441647.2017.1280932.

Nair, R. and Miller-Hooks, E. Fleet management for vehicle sharing operations. *Transportation Science*, 45(4):524–540, November 2011. doi: 10.1287/trsc.1100.0347. URL https://doi.org/10.1287/trsc.1100.0347.

Nourinejad, M. and Roorda, M. Carsharing operations policies: a comparison between one-way and two-way systems. *Transportation*, 42, 05 2015. doi: 10.1007/s11116-015-9604-3.

Nourinejad, M. and Roorda, M. J. A dynamic carsharing decision support system. *Transportation Research Part E: Logistics and Transportation Review*, 66:36–50, June 2014. doi: 10.1016/j.tre.2014.03.003. URL https://doi.org/10.1016/j.tre.2014.03.003.

O'Mahony, D. B., Eoin an Shmoys. Data analysis and optimization for (citi)bike sharing. 2015.

Pangbourne, K., Stead, D., Mladenović, M., and Milakis, D. The case of mobility as a service: A critical reflection on challenges for urban transport and mobility governance. In *Governance of the Smart Mobility Transition*, pages 33–48. Emerald Publishing Limited, March 2018. doi: 10.1108/978-1-78754-317-120181003. URL https://doi.org/10.1108/978-1-78754-317-120181003.

Rao, S. S. *Engineering optimization: theory and practice*. John Wiley & Sons, 2009.

Raviv, T., Tzur, M., and Forma, I. A. Static repositioning in a bike-sharing system: models and solution approaches. *EURO Journal on Transportation and Logistics*, 2(3):187–229, January 2013. doi: 10.1007/s13676-012-0017-6. URL https://doi.org/10.1007/s13676-012-0017-6.

Regue, R. and Recker, W. Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:192–209, December 2014. doi: 10.1016/j.tre.2014.10.005. URL https://doi.org/10.1016/j.tre.2014.10.005.

Rizzoli, A. E., Montemanni, R., Lucibello, E., and Gambardella, L. M. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151, September 2007. doi: 10.1007/s11721-007-0005-x. URL https://doi.org/10.1007/s11721-007-0005-x.

Schuijbroek, J., Hampshire, R., and van Hoeve, W.-J. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992–1004, March 2013. doi: 10.1016/j.ejor.2016.08.029. URL https://doi.org/10.1016/j.ejor.2016.08.029.

Schuijbroek, J., Hampshire, R., and van Hoeve, W.-J. Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research*, 257(3):992–1004, March 2017. doi: 10.1016/j.ejor.2016.08.029. URL https://doi.org/10.1016/j.ejor.2016.08.029.

Signorile, P., Larosa, V., and Spiru, A. Mobility as a service: a new model for sustainable mobility in tourism. *Worldwide Hospitality and Tourism Themes*, 10(2):185–200, April 2018. doi: 10.1108/whatt-12-2017-0083. URL https://doi.org/10.1108/whatt-12-2017-0083.

Skiena, S. S. S. *The Algorithm Design Manual*. Springer, 2010. ISBN 1849967202. URL https://www.amazon.com/Algorithm-Design-Manual-Steven-Skiena/dp/1849967202?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1849967202.

Turban, E., Sharda, R. E., and Delen, D. *Decision Support and Business Intelligence Systems (9th Edition)*. Prentice Hall, 2010. ISBN 013610729X. URL https://www.amazon.com/Decision-Support-Business-Intelligence-Systems/dp/013610729X?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=013610729X.

Vaira, G. Genetic algorithm for vehicle routing problem. page 158, 2014.

Vine, S. L., Lee-Gosselin, M., Sivakumar, A., and Polak, J. A new approach to predict the market and impacts of round-trip and point-to-point carsharing systems: Case study of london. *Transportation Research Part D: Transport and Environment*, 32:218–229, October 2014. doi: 10.1016/j.trd.2014.07.005. URL https://doi.org/10.1016/j.trd.2014.07.005.

Vogel, P., Greiser, T., and Mattfeld, D. C. Understanding bike-sharing systems using data mining: Exploring activity patterns. *Procedia - Social and Behavioral Sciences*, 20:514–523, 2011. doi: 10.1016/j.sbspro.2011.08.058. URL https://doi.org/10.1016/j.sbspro.2011.08.058.

Wang, H., Cheu, R., and Lee, D.-H. Dynamic relocating vehicle resources using a microscopic traffic simulation model for carsharing services. In *2010 Third International Joint Conference on Computational Science and Optimization*. IEEE, May 2010. doi: 10.1109/cso.2010.98. URL https://doi.org/10.1109/cso.2010.98.

Wockatz, P. and Schartau, P. Im traveller needs and uk capability study: supporting the realisation of intelligent mobility in the uk. In *IM traveller needs and UK capability study: supporting the realisation of intelligent mobility in the UK*, pages 1–35. October 2015. doi: 10.1108/978-1-78754-317-120181002.

A

SYSTEM ENVIRONMENT BY SCENARIO

This appendix serves as a reminder, if necessary, of the environment behind the scenarios for system experimentation (Section 5.2).

A.1 SCENÁRIO I

A.1.1 *Evaluation Formula*

$$
\begin{aligned}
\underset{X}{\text{minimize}} \quad & d(x) + \alpha r(x) - \beta f(x) \\
\text{where} \quad & \alpha = 1500 \;\&\; \beta = 100 \\
& d(x) = \sum_{\substack{i=1 \\ i \neq n}}^{n} d(i, i+1) \qquad i, n \in \mathbb{N}^{+} \\
& r(x) = \sum_{i=1}^{n} r(|i|) \qquad i, n \in \mathbb{N}^{+} \\
& f(x) = \sum_{i=1}^{n} [r(|i|) = 0] \qquad i, n \in \mathbb{N}^{+}
\end{aligned}
\tag{7}
$$

A.1.2 *Fleet*

Table 25: Fleet (available and not available)

Plate	*Capacity*	*available*
XYZ	15	*TRUE*
ZYX	15	*TRUE*
YXZ	10	*FALSE*

A.1.3 *Service Level Requirements*

Table 26: Service level requirements for 2017-11-26 12:00:00

Hour	*Area*	*Relocation*	*ID*
2017-11-26 12:00:00	177	10	1
2017-11-26 12:00:00	107	8	2
2017-11-26 12:00:00	192	8	3
2017-11-26 12:00:00	9	6	4
2017-11-26 12:00:00	108	6	5
2017-11-26 12:00:00	162	6	6
2017-11-26 12:00:00	176	6	7
2017-11-26 12:00:00	84	-30	8
2017-11-26 12:00:00	57	-15	9
2017-11-26 12:00:00	148	-5	10

A.1.4 *Geospatial Coordinates*

Table 27: Stations(areas) locations

ID	*Area*	*Latitude*	*Longitude*
1	177	41.39446	2.155730
2	107	41.39134	2.146347
3	192	41.39084	2.153818
4	9	41.40213	2.165954
5	108	41.38893	2.149614
6	162	41.39574	2.160796
7	176	41.39682	2.152529
8	84	41.40098	2.180908
9	57	41.41730	2.172617
10	148	41.37510	2.147933

A.1.5 *Distances*

Table 28: Distance between areas/stations

di,j	*Distance*	di,j	*Distance*	di,j	*Distance*	di,j	*Distance*
$d_{177,192}$	431.7352	$d_{107,162}$	1303.422	$d_{192,148}$	1817.079	$d_{108,148}$	1542.504
$d_{177,9}$	1207.32	$d_{107,176}$	798.5859	$d_{9,108}$	2004.482	$d_{162,176}$	701.7969
$d_{177,108}$	799.2059	$d_{107,84}$	3082.193	$d_{9,162}$	830.7663	$d_{162,84}$	1779.739
$d_{177,162}$	446.9816	$d_{107,57}$	3625.005	$d_{9,176}$	1268.168	$d_{162,57}$	2591.11
$d_{177,176}$	375.0533	$d_{107,148}$	1809.014	$d_{9,84}$	1257.084	$d_{162,148}$	2532.482
$d_{177,84}$	2226.706	$d_{192,9}$	1612.832	$d_{9,57}$	1774.846	$d_{7,84}$	1858.355
$d_{177,57}$	2904.002	$d_{192,108}$	411.0985	$d_{9,148}$	3359.673	$d_{7,57}$	1439.637
$d_{177,148}$	2246.887	$d_{192,162}$	797.4612	$d_{108,162}$	1202.867	$d_{7,148}$	3801.518
$d_{107,192}$	627.2566	$d_{192,176}$	672.356	$d_{108,176}$	909.9641	$d_{8,57}$	1605.997
$d_{107,9}$	2030.982	$d_{192,84}$	2529.699	$d_{108,84}$	2939.563	$d_{8,148}$	3525.419
$d_{107,108}$	382.7962	$d_{192,57}$	3332.652	$d_{108,57}$	3692.28	$d_{9,148}$	3359.673

A.1.6 *Configurations*

```yaml
---   config.yml
journey:
    limit: 60 #time, in minutes, to rebalance the system
    velocity: 750 #m/min - 45km/h average speed during a trip
    depot: 84 #depot

operations:
     scooters:
        pickup: 1 #average time spent on picking up a scooter
        deliver: 1 #average time spent on delivering a scooter
---
```

Figure 44: YAML configuration file

A.2 SCENÁRIO II

The only difference between *scenario I* and *scenario II* is the availability of the rucks. Table 29 shows that only one truck is available.

A.2.1 *Fleet*

Table 29: Fleet (available and not available)

Plate	*Capacity*	*available*
XYZ	15	*TRUE*
ZYX	15	*FALSE*
YXZ	10	*FALSE*