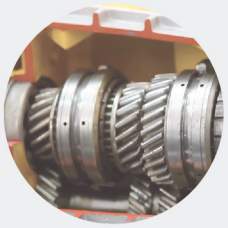
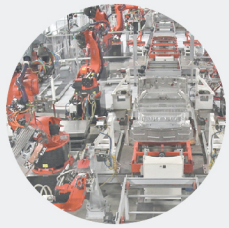
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL



Districting in Last-Mile Delivery Routing: Route creation using Shortest Hamiltonian Path-based algorithms

JOSÉ MIGUEL RIBEIRO E SILVA
julho de 2024

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Districting in Last-Mile Delivery Routing: Route creation using Shortest Hamiltonian Path-based algorithms

Instituto Superior de Engenharia do Porto
Master in Industrial Engineering and Management

José Miguel Ribeiro e Silva
ID number 1180689

Advisor
Prof. António Galrão Ramos

Academic Year 2023/2024

Thesis defended on 19 July 2024
in front of a Board of Examiners composed by:

Prof. Alzira Mota (chairman)

Prof. Elsa Silva

Prof. António Ramos

Districting in Last-Mile Delivery Routing: Route creation using Shortest Hamiltonian Path-based algorithms

Instituto Superior de Engenharia do Porto

This thesis has been typeset by LaTeX and the isepthesis class.

Author's email: 1180689@isep.ipp.pt

Acknowledgments

This dissertation was a significant challenge and represents a long journey that would not have been possible without the help of many contributors.

First, I would like to thank my advisor, António Ramos, for all the patience and, above all, for all the guidance.

I would also like to thank INESC-TEC for the great opportunity to develop this work with a research grant. I was received with open arms, and all the help I received was undoubtedly essential to this work's conclusion.

To my family - my mom, my dad, and my sisters Inês and Clara, I would not be here without you, and I will never be able to thank you enough. I am sorry for the endless rants, for asking your opinions on things that you don't understand, and for taking up space at the dinner table for a year with my computer screens.

To my girlfriend, Carolina, I thank you for always being there for me no matter what, for listening to what is troubling me, even for the 100th time, for always calming me down, and above all, for making me happy.

To all my friends, thank you for easing my mind, making me laugh, and making me love this short life we are living.

Abstract

Districting can reduce the complexities of delivery problems by segmenting its dimensions while facilitating drivers' familiarity with their work areas, fostering personal connections with customers, and enhancing satisfaction.

This dissertation presents and evaluates multiple heuristics for route creation intending to identify the most efficient method for intra- and inter-district routing. Three types of distances, two TSP-to-SHPP converters, and three types of inter-district connectors were tested, resulting in a total of 18 distinct variants from the combination of these components.

Out of 18 tested variants, the best-performing developed approach used a Lin-Khernigan-based heuristic, later converting it to a Shortest Hamiltonian Path in each district, creating inter-district connections to a hypothetical medoid in the next district to visit and utilizing asymmetric road distances. The obtained results were satisfactory, and the best components for route creation were identified, allowing for the continuation of the analysis of the model's adaptability to day-to-day implementation in a further project.

The models were developed and tested using real-world data from a parcel delivery company operating in the Porto metropolitan area of Portugal.

Resumo

O processo de districting pode reduzir a complexidade de problems de parcel delivery, particularmente problemas de Last-Mile Routing, ao segmentar suas dimensões, facilitando a familiaridade dos motoristas com suas áreas de trabalho, fomentando conexões pessoais com os clientes e aumentando a satisfação.

Esta dissertação apresenta e avalia múltiplas heurísticas de criação de rotas com o objetivo de identificar o método mais eficiente para roteamento intra e inter-district. Foram testados três tipos de distâncias, dois conversores TSP-para-SHPP e três tipos de conectores inter-districts, tendo a combinação desses componentes resultado num total de 18 variantes distintas.

Das 18 variantes testadas, a abordagem desenvolvida com melhor desempenho utilizou uma heuristica Lin-Kernighan, posteriormente convertida num Shortest Hamiltonian Path em cada distrito, criando conexões inter-district com um medoide hipotético no próximo district a visitar e utilizando distâncias rodoviárias assimétricas. Os resultados obtidos foram satisfatórios, e os melhores componentes para a criação de rotas foram identificados, permitindo a continuidade da análise da adaptabilidade do modelo à implementação diária num projeto futuro.

Os modelos foram desenvolvidos e testados usando dados práticos relativos às rotas def uma empresa de entrega de encomendas operando na área metropolitana do Porto, em Portugal.

Contents

List of Abbreviations

ALNS	Adaptive Large Neighborhood Search
ANN	Artificial Neural Network
CARP	Capacitated Arc Routing Problem
CluVRP	Clustered Vehicle Routing Problem
CTSP	Clustered Travelling Salesman Problem
CVRP	Capacitated Vehicle Routing Problem
EAX-GA	Edge Assembly Crossover Genetic Algorithm
GA	Genetic Algorythm
HRP	High-Level Routing Problem
ILS	Iterated Local Search
KNIES	Kohonen Network Incorporating Explicit Statistics
LK	Lin-Kernighan
LKH	Lin-Kernighan Heuristic
LMD	Last-Mile Delivery
LRP	Low-Level Routing Problem
MDSDRP	Multi-objective Dynamic Stochastic Districting and Routing Problem
MILP	Mixed Integer Linear Programming
MOEA	Multi-Objective Evolutionary Algorithm
MTSP	Multisalesman Problem

OSM	OpenStreetMap
POPMUSIC	Partial OPtimization Metaheuristic Under Special Intensification Conditions
PSO	Particle Swarm Optimization
RPP	Rural Postman Problem
RTR	Record-To-Record
s–t TSP	Path Travelling Salesman Problem
SHP	Shortest Hamiltonian Path
SHPP	Shortest Hamiltonian Path Problem
SoftCluVRP	Soft Clustered Vehicle Routing Problem
SOM	Self Organizing Maps
TSP	Travelling Salesman Problem
UHGS	Unified Hybrid Genetic Search
VND	Variable Neighbourhood Descent
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
VSR-LKH	Variable Strategy Reinforced Lin-Kernighan Heuristic

List of Tables

List of Figures

Chapter 1

Introduction

This chapter outlines the research problem that forms the central focus of this dissertation, offering context and discussing its significance. It sets forth the research question, delineates the project's objectives, and presents the chosen methodological approaches. Lastly, the chapter provides a detailed overview of the dissertation's structure.

1.1 Background and relevance

In an increasingly consumerist society, customers around the globe, either individuals or businesses, are increasingly expecting faster, more customized, and more efficient product deliveries.

Ensuring customer satisfaction is of utmost importance for any business, which is the reason for the growing emphasis on supply chain management (Agus, 2013). In the supply chain management problem, one of the most challenging components is Last-Mile Delivery (LMD). According to Boysen et al. (2021), the relevance of the LMD is caused by challenges like the increasing volume of demand, sustainability focus, cost reduction, time pressure, and an aging workforce.

However, a supposed optimal route is not in practice the optimal route. Many factors such as traffic, parking, traffic lights, or shortcuts are overlooked and ignored. Therefore, more and more emphasis is placed on drivers' experience-based knowledge as a method for improvement in LMD. For drivers to acquire this experience-based knowledge, consistency is important, i.e., it is advantageous for a company to allocate a driver to a specific area. Vidal et al. (2020) argue that the quality of a route is increased if it has temporal consistency, person-oriented consistency, regional consistency, and delivery consistency. Consequently, more and more approaches are based on the creation of groups of clients, named districts, or clusters.

In certain routes where the number of stops to perform is considerable, it would

be advantageous for the driver if a good performing route were presented identifying the order of customers to visit, especially when the driver is not familiar with a certain area. Moreover, for less experienced drivers, the proposal of optimal routes may lead to the creation of more efficient driving habits and patterns. As such districting (or clustering) routing remains of great relevance, and therefore, research in this area could be extremely relevant and lead to improvements in the field of LMD.

As such, this work focuses on identifying, implementing, and analyzing the best overall performing methods for route creation using districting approaches. In this task, computational costs are very relevant, since in real-life operations the time available for generating routing plans is very short.

1.2 Research proposal and objectives

With the problem presented in the previous subtopic, it is understood that the following work is developed to answer the following research question: "In a districting approach to Last-Mile route creation, what is the best strategy for determining the sequence of stops that each vehicle will have to perform to serve all the clients?"

To answer this question, the following general objective was established: "Develop and validate routing algorithms that define the sequence of stops that each vehicle will have to perform in each district, to serve all the clients with the best possible performance."

To achieve the general objective of this research, the following objectives were formulated:

- Pre-processing the data received from the districting phase;
- Identify routing creation components;
- Develop routing algorithms using the identified components;
- Critical analysis of the results and the performance of the components;

1.3 Methodological options

Regarding the methodology, this research adopts a quantitative approach. The primary objectives are to identify, analyze, and understand the best components for route creation using concrete data comparisons, such as computational times and total traveled distances.

The research method will be a case study, as the data used throughout the work refers to real data obtained from the daily operations of a parcel delivery company operating in the Porto Metropolitan Area.

1.4 Structure of the Paper

The structure of this dissertation will follow the following structure. Chapter 1, Introduction, identifies the research problem's significance, defines the research question and objectives, outlines the methodology, and previews the dissertation's structure. In Chapter 2, Literature Review, the state-of-the-art in the relevant project areas is examined. Chapter 3, Dataset description, introduces the dataset used in this study, as well as the prior works that influence the current work. In Chapter 4, the raw data undergoes preparation for analysis, including attribute adaptation, handling missing values, stop aggregation, outlier management, and final data tuning. Chapter 5 describes the developed route creation components. The multiple intra, inter, or general components are detailed. In chapter 6 the results obtained from the implemented methodologies are analyzed and interpreted, and an effort to identify the best-performing components, as well as the best-performing variant. Lastly, Chapter 7 draws relevant conclusions to the dissertation.

Chapter 2

Literature Review

Throughout this chapter, a comprehensive literature review of the relevant elements for the development of this project is provided. Section 2.1 focuses on the Vehicle Routing Problem (VRP), by defining it and presenting its main formulations. Section 2.2 delves into one of the many approaches to simplify and, consequently, attempt to solve this problem, the two-phase solution for VRP: Clustering or Districting, and Routing, further analyzed in subsections 2.2.1, 2.2.2, and 2.2.3, respectively. Sections 2.3 and 2.4 define two very important concepts for the current project, the Travelling Salesman Problem (TSP) and the Shortest Hamiltonian Path Problem (SHPP), respectively, explaining the concepts, revealing their importance in VRPs and identifying their many variations. Section 2.5 delves into the state-of-the-art approaches for solving routing problems, focusing on many approaches that authors used to attempt to solve problems: subsection 2.5.1 focuses on approaches to solve the Travelling Salesman Problem, subsection 2.5.2 focuses on solutions to create Shortest Hamiltonian Paths in the context of graph theory, section 2.5.3 explores the many approaches that use clustering as a way to solve Vehicle Routing Problems, section 2.5.4 centers on the Districting approach as a method for solving Vehicle Routing Problem. Finally, 2.6 provides some concluding remarks on the methods described throughout the chapter.

2.1 The Vehicle Routing Problem

The Vehicle Routing Problem, commonly known as VRP, is one of the most famous and studied combinatorial optimization problems worldwide. This is a generic name assigned to problems involving the routing of vehicles to meet the demand of multiple geographically dispersed customers. "In broad terms, it deals with the optimal assignment of a set of transportation orders to a fleet of vehicles, and the sequencing of stops for each vehicle." (Hartl et al., 2006).

This problem has been studied for almost 70 years. Dantzig and Ramser (1959) were the first to mention this problem in 1959. By studying the optimal routes of a fleet of gasoline delivery trucks between a large number of gas stations and a given supplier terminal, they developed the first algorithmic approach that allowed the user to achieve a near-optimal solution. This algorithm was later improved by Clarke and Wright (1964).

The VRP is based on the objective of identifying the best vehicle distribution and routing so that, starting from a common initial depot, the demand of each customer is served to minimize a certain routing cost. This cost may be any given parameter like the number of vehicles, euclidean distance, road distance, monetary cost, and many others. It can also be a combination of multiple parameters (Christofides, 1976).

The VRP and its variants are considered NP-hard problems, which means that finding an optimal solution is either impossible or very demanding in terms of computational times, so exact methods to solve these types of problems are not commonly used in practical, real-world situations. Most approaches are based on heuristics or meta-heuristics (Brajevic, 2011).

According to Munari et al. (2017) two formulations are the basis for almost every other VRP-related problem: the Capacitated Vehicle Routing Problem (CVRP), consisting of a variation of the VRP where vehicles contain limited capacity, and the Vehicle Routing Problem with Time Windows (VRPTW), consisting of a variation of the VRP where to clients are associated certain time windows, i.e. certain clients must be visited between a certain time window.

Considering a set of customers ($C = \{1, ..., n\}$), there is a positive demand associated with each customer $i \in C$. To fulfill this demand, a set of K vehicles of Q capacity have to carry out a route starting from a certain depot, visiting a set of customers, and finally returning to the given depot. The locations of the depot and customers are known and each customer must be visited once. The problem may be represented using a graph $G(N, \varepsilon)$. N corresponds to the set of nodes $n \in N$ associated to the customers C and $N = C \cup \{0, n + 1\}$. c_{ij} represents the cost of crossing each arc (i, j). The demand in each node is represented by q_i, where $i \in C$ and it is always positive except in the depot, in which its value is 0 (Nanda Kumar and Panneerselvam, 2012).

2.2 Two-phase solution for VRP: Clustering or Districting and Routing

The VRP is, and for many years has been, a rather challenging problem. Thus, many authors defend that to simplify it, it should be divided into different components.

Bowerman et al. (1994) mention 5 different solution techniques to solve the VRP problem: The cluster-first/route-second, consisting of assigning demand nodes into clusters or districts that then are routed; The route-first/cluster second, which consists in establishing a large tour of all the demanded nodes, and later broken to satisfy capacity constraints of the vehicles; The savings/insertion that consists in iterative comparisons between current best-performing solutions and alternative solutions to progressively identify the largest savings until a feasible configuration is found; the improvement/exchange that consists in progressively improving an initial route through the exchange of arcs; mathematical programming, that is based in optimally solving a relaxed formulation of the VRP. **This dissertation will mainly focus on the cluster-first[1]/route-second approach.**

2.2.1 Clustering

According to Sisodia et al. (2012), clustering is the task where a certain set of data is grouped according to several categories. The grouping of this data must be based on the principle of maximizing intra-class similarity while minimizing inter-class similarity. The authors define clustering algorithms into 3 categories: partitioning, hierarchical, and density-based:

Partitioning algorithms attempt to decompose N objects into k clusters. These algorithms are usually relatively simple and scalable but also have a very high sensitivity to the initiation phase. A few examples of famous partitioning algorithms are the K-means and K-medoids.

Hierarchical algorithms are based on division, where the whole set of nodes is progressively divided into progressively smaller clusters, or agglomeration where singular customers are progressively grouped into bigger and bigger clusters. These algorithms are too expensive for high-dimensional problems.

Density-based clustering algorithms are based on grouping the objects based on the local density and not based on proximity between objects. These algorithms are resistant to noise and outliers but are also unsuitable for high-dimensional sets of data.

[1] Clustering should be understood as the process of grouping customers, so that this component includes both clustering and districting.

Clustering is extremely helpful in many areas like marketing, economy, biology, libraries, insurance, city planning, pattern recognition, image processing, logistics, and image processing (Omran et al., 2007).

Given the problem presented in this project, the focus will be on the logistics area.

This concept was first introduced in logistics by Chisman (1975), where a new perspective regarding the TSP problem was proposed. A perspective in which it would make sense to visit certain grouped customers before advancing to another set of customers. Those sets of customers were defined as clusters, creating the basis for the Clustered Vehicle Routing Problem (CluVRP).

Clustering makes sense because of the resources and the number of customers. As resources are limited for a high number of customers, partitioning them into different subareas in such a way that they are served on specific days or by specific drivers. The clustering essentially changes the VRP problem's order of magnitude from a customer basis to a cluster basis (Expósito-Izquierdo et al., 2016).

According to Expósito-Izquierdo et al. (2016), customers should be clustered regarding 3 components: based on their distribution, as well as the characteristics of the land on which they are located; based on customers sharing certain delivery characteristics; precedence before points, because sometimes customers need to be served before others. According to Vidal et al. (2015), clusters can be imposed by the geography, and nature of the application, as well as by practitioners aiming to achieve compact and easy-to-implement routing solutions.

As stated by Hintsch and Irnich (2018), regarding the CluVRP, "(...) all customers of a cluster must be served by the same vehicle in consecutive visits. It means that if one customer of a cluster is served by a vehicle, all other customers of the same cluster are served by the same vehicle. Moreover, there is no customer from another cluster visited in between two customers of the same cluster."

This means that all the goods to be delivered to customers in the same cluster should be loaded in a single vehicle. Once a certain customer of a certain depot is visited, the vehicle should not leave the cluster, nor return to the depot unless all the other customers in that cluster have been visited. More than one cluster may be visited in one go, so, a trip can also be viewed as the cycle containing a certain sequence of clusters that should be visited (Barthélémy et al., 2010).

Although the CluVRP demands a certain vehicle not to leave the cluster before all the customers in it are served, according to Islam et al. (2021), there are 2 versions to the CluVRP: CluVRP with strong restrictions (CluVRP) and the CluVRP with soft restrictions - the Soft Clustered Vehicle Routing Problem (SoftCluVRP). In the CluVRP the customers in the same cluster must be visited without any interruptions

by the same vehicle. However, in the SoftCluVRP, customers in the same cluster are visited by the same vehicle, but the vehicle is allowed to leave and come back to the cluster multiple times. As seen in Le et al. (2022), VRP problems with time windows (VRPTW) can also be solved by resorting to clusters.

Barreto et al. (2007) mention that there are four different ways to achieve clustering: one-phase hierarchical method, two-phase hierarchical method, direct assignment (non-hierarchical), and sequential assignment (non-hierarchical).

2.2.2 Districting

Districting is the process of dividing a vast geographical region or network into smaller subareas. This is done to facilitate organization and administration (Kalcsics et al., 2005).

In literature, most of the districting approaches are of the agglomeration type. These types of problems are usually solved by allocating units to centers taking into consideration some constraints. Very often, these constraints ensure contiguity, compactness, and workload balance (Muyldermans et al., 2003).

Districting is very helpful in many diverse areas and takes into consideration a vast number of criteria. Political districting is one of the areas where the districting problem is applied. It takes into consideration demographic, geographic, and political criteria to effectively draw the boundaries for electoral votes, as done in Mehrotra et al. (1998).

Another area where the districting problem is applied is the sales and service territory design. This takes into consideration organizational criteria, geographical criteria, and activity-related criteria, among others, to subdivide all customers into smaller areas of responsibility, where the needs of these customers are more effectively assured. An example of this type of problem was tackled in Fleischmann and Paraschis (1988), where the authors proposed a manufacturing and delivery company to update their 8-year-old obsolete districts, to create a more even customer distribution and reduce workloads.

You can also use the districting problem to solve other common problems like school location districts, waste collection areas, emergency service sectors, electrical power districting, etc). The district design should try to fulfill the maximum possible number of technical, ethical, ecological, social, and other constraints (Kalcsics et al., 2005).

According to Miranda et al. (2011), districts are created to be used for longer periods, usually at least one year. This is due to the big effort that is applied in defining them, caused by all of the factors that are taken into consideration, like operational, physical, and managerial parameters.

One of the advantages of the districting process, as previously mentioned, and similar to clustering, is partitioning the area of the problem so that its dimensions are segmented, decreasing the degree of complexity of its resolution. However, the main reason to divide greater areas into districts is usually the drivers need to be acquainted with the area where they work (Haugland et al., 2007). In addition, while visiting the same area, and as a result, the same customer, a personal connection is created between the driver and the customer, increasing customer satisfaction (Bender et al., 2020).

In some cases, a district containing a set of regular customers is assigned to a driver, but other occasional customers also arise. As Lei et al. (2016) mention "Besides the regular customers, there are also some stochastic customers in each period. The stochastic customers are independent from one period to the next. Unlike what happens for regular customers, both the location and presence of stochastic customers are uncertain." In contexts like this, it is desirable to assign the usual sets of customers to the same drivers. These stable districts will achieve better results and superior service to customers (Lei et al., 2012).

2.2.3 Routing

After the creation of districts or clusters, it is necessary to create routes to visit all the customers in all the clusters or districts.

In problems where the clustering approach is utilized, authors like Izquierdo et al. (2013), Expósito-Izquierdo et al. (2016), Defryn and Sörensen (2017), Sevaux and Sörensen (2008), and others have divided this problem into two different components:

- **The High-Level Routing Problem (HRP).** This component defines the routes to serve clusters (inter-cluster). This means that it decides the order of clusters visited by each vehicle. It is a tactical problem that has the goal of identifying the routes with minimum length to visit each cluster exactly once. This component identifies the cycles that leave the depot, visit a certain number of clusters, and later return to the depot. There is a resemblance between the HRP and the Travelling Salesman Problem (TSP) if each cluster is compacted into its centroid or an arbitrary node;

- **The Low-Level Routing Problem (LRP).** After ordering the clusters, in the LRP the routes inside the clusters are created. There is one LRP associated with each cluster. In this phase, the goal is to find the Shortest Hamiltonian Path (SHP) from a certain starting point, where the vehicle enters the cluster, to a certain ending point, where the vehicle leaves the cluster, visiting all of the customers in that cluster.

Hintsch and Irnich (2018) follow a similar perspective, but the authors divide the problem into three subproblems, i.e., the assignment of clusters to routes, the routing inside each cluster, and only after the sequencing of the clusters in the routes is done.

In the districting problem, as the districts are created with a long-term perspective, the routing is done on a daily basis and is based on the districting as well as the demand in each customer and district (Expósito-Izquierdo et al., 2016). This means that on one hand districts are more robust and are not influenced by changes in operational character statistics, while on the other hand, routes are sensitive to variations and constraints like capacity, time, distance, or others. Different guidelines and perspectives should be used for each of them while attempting to keep interactions between the two levels.

2.3 The Travelling Salesman Problem

Combinatorial optimization is the area that studies and attempts to solve problems that look for the best option in a set of objects. The Travelling Salesman Problem (TSP) was proposed in Flood (1956) in a seminar talk at Princeton University and is still now one of the most famous and widely studied combinatorial optimization problems.

Given a set of cities and the distances between them, the TSP attempts to identify the shortest possible route that visits each city exactly once and returns to the starting point (Junior Mele et al., 2021). Regardless of its simplicity, even when broken down into its components, it is one of the most complex problems to solve in its area. For example, in a problem using only 10 nodes n, the number of possible routes for the Traveling Salesman Problem would be $\frac{(n-1)!}{2}$, so over 180000. If the number of nodes were 15, the number of possible routes would be over four billion (Singh et al., 2020).

The TSP can be divided into different classes based on the arrangement of the distances between the cities. In the symmetric TSP, the distance between two cities is the same in each direction. In the asymmetric TSP, paths may not exist in both directions, or the distance between two points might be different in each direction (Goyal, 2010).

There are many versions and variations of the Traveling Salesman Problem, like the Multisalesman Problem (MTSP), the Rural Postman Problem (RPP), the Clustered Travelling Salesman Problem (CTSP), and many others. A deeper enumeration and explanation of these variations may be seen in Jünger et al. (1995).

2.4 Shortest Hamiltonian Path Problem

Graph theory is the branch of mathematics that studies networks of points or nodes connected by lines or edges. These nodes and edges represent connections between entities, allowing graph theory to be an important method in multiple areas like computer science, electrical engineering, logistics, and many other scientific areas (Tamilarasi and Dhenakaran, 2018). In logistics, graph theory is very helpful because it allows the study of locations (nodes) and the connections between them (edges).

In graph theory, the term Hamiltonian refers to the fact that each node in a graph has to be visited exactly once. A Hamiltonian circuit is a spanning cycle in a graph, meaning that it cycles through every vertex exactly once, returning to the same starting point. A Hamiltonian path, like the Hamiltonian cycle, passes through all the nodes in the graph but starts and finishes in two different nodes. Every path with a Hamiltonian cycle has a Hamiltonian path, but the opposite might not be true (Rahman and Kaykobad, 2005).

The shortest Hamiltonian path problem (SHPP) attempts to find a minimum cost path between a starting node v_s and terminating node v_t visiting each node v in the graph exactly once (Kawashima and Sugai, 2014).

According to Sevaux and Sörensen (2008) there are 4 different variants of the shortest Hamiltonian path problem, depending on the structure of the graph and the demands of the customers:

- The **shortest Hamiltonian path problem** is, as mentioned before, the problem that identifies the shortest Hamiltonian path in a graph, from a certain point v_s to a terminating node v_t, but these points are not specified.
- The **shortest Hamiltonian path problem between two nodes** is identical to the one above, except for the starting and terminating nodes that are known.
- The **shortest pre-Hamiltonian path problem** is a variation of the shortest Hamiltonian path problem where instead of every node being visited exactly once, every node would be visited **at least** once.
- The **shortest pre-Hamiltonian rural path problem** is the same problem as above, but only a certain subset of nodes has to be visited.

The shortest Hamiltonian path problem is an NP-complete problem. Because of being an NP-complete problem, most of the authors focus on finding one Hamiltonian path and not the shortest Hamiltonian path, and even when the main goal is finding the shortest path, many authors only find an acceptable approximation in order to minimize computational efforts (Ding et al., 2007).

In the CluVRP, as mentioned by Izquierdo et al. (2013) and Expósito-Izquierdo et al. (2016), the Low-Level Routing Problem (LRP) can be interpreted as finding the Hamiltonian path with the minimum distance within the cluster at hand, i.e. the SHPP.

The starting and finishing nodes are usually dependent on the approach utilized to identify the routes, and the connections between clusters, among other factors. It can also favor intra-cluster Hamiltonian paths over inter-cluster paths. As mentioned in Ding et al. (2007), "The algorithms favoring the intra-cluster Hamiltonian paths have the disadvantage that the shortest Hamiltonian paths specify the end vertices for each cluster which do not necessarily lead to the best inter-cluster paths."

Authors like Alshamlan and Menai (2012) and Tamilarasi and Dhenakaran (2018) mention a *general method*, also known as brute force method, for solving the SHPP between two nodes:

1. List all possible Hamiltonian paths with each vertex being visited once;
2. Identify the ones starting in v_s and finishing in v_t;
3. Finding the costs associated with each of those routes;
4. Pick the shortest route.

This solution is considered acceptable for a small set of nodes, but for bigger graphs, it is not applicable given the $N!$ number of possible permutations.

The SHPP is closely related to the TSP, even admitting a similar dynamic programming solution (Höner zu Siederdissen et al., 2014). The Path Travelling Salesman Problem (s–t TSP), also known as Path TSP resembles the SHPP, given that the shortest path between s and t must be found.

Even though the literature reports some independent exact or heuristic algorithms, the majority of the algorithms for the Hamiltonian Path Problem utilize fast solutions to its cousin, the TSP. According to Altinel et al. (2000), to achieve an answer for finding a Hamiltonian path by means of a TSP solution, one of two changes should be applied:

1. The distance between the starting and finishing node is set to zero, in order to guarantee that this connection is verified. After the cycle is created, remove the connection between those two nodes, transforming the cycle into a path.
2. A new node is added to the set of nodes. The distances between the new node and the starting and finishing nodes are then set to zero in order to guarantee their connection. After the cycle is created, remove the node and its connections, transforming the cycle into a path.

2.5 State-of-the-art in Routing Problems

A thorough analysis was conducted on various topics related to solving various types of Routing Problems relevant to the current work, including Solving the Travelling Salesman Problem, Solving the Shortest Hamiltonian Path Problem in the context of graph theory, clustering for solving the Routing Problem, and districting for solving Routing Problems.

2.5.1 Solving the Travelling Salesman Problem

The TSP is one of the most broadly studied routing problems, meaning that many interesting solutions were developed in order to solve this routing problem. Also, as previously mentioned, it is possible to transform the TSP into an SHPP by applying a few simple transformations to the problem, making these solutions even more in line with the problem proposed in this report, as they allow the creation of routes inter and intra cluster or district. Therefore, there is interest in analyzing different approaches to solving this problem.

Methods for solving the TSP are usually divided into three phases: a starting point, a solution generation scheme, and a termination rule. If the termination rule is to only stop when the optimal solution is found, the method is considered exact (Bellmore and Nemhauser, 1968). This means that it is possible to optimally solve the Travelling Salesman Problem. Examples of these algorithms are the Brute Force Algorithm, Dynamic Programming, and Branch and Bound. (Singh et al., 2020)

The Concorde solver is a Branch-and-Bound solver-based solver and it can optimally solve instances within a reasonable computation time, given that the largest solved instance had 85900 cities. It is still currently regarded as the fastest TSP optimal solver (Lu et al., 2022).

Although these achieve an optimal solution to the TSP, there's a large time complexity associated with them, making them a less desirable approach in case faster solutions are required or larger areas are being studied. In those cases, the approximate, heuristic, and metaheuristic approaches are more suitable (Jünger et al., 1995).

According to Singh et al. (2020), some of the most well-known heuristic and metaheuristic algorithms are: Hill climbing, nearest neighbor, genetic algorithm, ant colony optimization, simulated annealing, tabu search, and multi-start local search. According to the authors, the best overall solution is the multi-start local search, providing a good solution in reasonable computational time. The best solution in their study, however, was obtained by the genetic algorithm.

In their works, Jünger et al. (1995) mention some improvement heuristics, that

can be applied to functional heuristics, in order to provide ways to leave local optima. These improvement heuristics include the two-opt exchange, three-opt exchange, and the Lin-Kernighan (LK) type exchange.

A current and very promising approach to solving the Travelling Salesman Problem includes Machine Learning techniques. These approaches were introduced to attempt to reduce the number of experts necessary to solve combinatorial optimization problems. As stated by Junior Mele et al. (2021) "The scope is to learn heuristics directly from instance definition, or in case of supervised training extracting knowledge from existing solutions."

Usually, these systems are divided into two procedures: The *Input Procedure* - where manageable information is provided to the artificial neural network; and the *Output Procedure* - where a feasible solution is created. Currently, these approaches still suffer from some disadvantages like the usual scaling problem that affects all algorithms, or the fact that these approaches currently can only deal with non-Euclidean distances, etc. A more detailed study on this type of TSP resolution technique was carried out in Junior Mele et al. (2021).

Another promising and relatively recent approach for solving the TSP is Self Organizing Maps (SOM). According to Sarikyriakidis et al. (2023), glsSOM is an Artificial Neural Network (ANN) that uses unsupervised learning to build a 2D map. The SOM approach is based on correction learning instead of error learning. The network propagates from a central node until it reaches the whole population of nodes, tracking the best positions for nodes meanwhile.

The propagation varies with the method the SOM model is based. The most common approaches are based on the hybrid algorithm, greedy algorithm, evolutionary algorithm, genetic algorithm, memetic algorithm, chaotic self-organizing map, fuzzy logic, and simulated annealing. According to the author, the best results are obtained with ORC-SOM for small-scale problems and PMSOM for large-scale problems. For a more extensive view of the topic, the work by Sarikyriakidis et al. (2023) can be analyzed.

According to Goyal (2010), out of all the approaches to the Travelling Salesman Problem, Lin-Khernigan (LK) based heuristics are generally considered to be one of the most effective methods for generating optimal or near-optimal solutions for the TSP.

According to Lu et al. (2022): "Among these iterated LK algorithms, Helsgaun's Lin-Kernighan Heuristic (LKH) is the uncontested state-of-the-art heuristic TSP solver." The latest version proposed by Helsgaun (2018) utilizes the Partial OPtimization Metaheuristic Under Special Intensification Conditions (POPMUSIC) to locally optimize sub-parts of a solution, further improving the LKH algorithm.

In their works, Zheng et al. (2023) propose a new version of the LKH, the Variable Strategy Reinforced Lin-Kernighan Heuristic (VSR-LKH), which outperforms the LKH algorithm for solving the TSP, by utilizing three reinforcement learning algorithms, the Q-learning, Sarsa, and Monte Carlo. The authors also mention another kind of state-of-the-art heuristic, the Edge Assembly Crossover Genetic Algorithm (EAX-GA), which is another excellent approach for solving the Travelling Salesman Problem.

As previously alluded, the routing problem may be defined in two different phases, the High-Level Routing Problem, and the Low-Level Routing Problem. Given that the Travelling Salesman Problem is also a routing problem, this is also possible. If the set of customers is grouped into different districts or clusters, the complexity of the TSP may be reduced, by performing the Shortest Hamiltonian Path Problem inside the clusters and solving the Travelling Salesman Problem using the clusters as if they were nodes, organizing them to reduce the traveled distance, as done in Ding et al. (2007), Lu et al. (2022) or Xu and Lan (2023). The Travelling Salesman Problem containing clusters is the Clustered Travelling Salesman Problem (CTSP).

Regarding the CTSP, Lu et al. (2022) state that it is possible to solve the CTSP as a normal TSP. To do that, one must add a cost M to all inter-cluster edges, in order to force the salesman to visit all cities inside each cluster before leaving it. The authors also answer 3 questions regarding the CTSP:

1. "How do state-of-the-art exact TSP solvers perform on clustered instances converted from the CTSP?", where the authors concluded that the Concorde solver can optimally solve all the medium and large CTSP instances;

2. "How do state-of-the-art inexact (heuristic) TSP solvers perform on clustered instances converted from the CTSP?", where the authors concluded that the heuristics LKH and Genetic Algorythm (GA) based algorithms perform very well not only in terms of solution quality but also computational efficiency because of very good scalability;

3. "Do state-of-the-art TSP solvers compete well with the best-performing methods specifically designed for the CTSP?", where the authors concluded that in general, TSP solvers are considerably better than the best CTSP designed heuristics, regarding solution quality, but also computational efficiency.

2.5.2 Solving the Shortest Hamiltonian Path Problem in the context of graph theory

In graph theory solving the Hamiltonian Path Problem is often the problem of only identifying if a Hamiltonian Path exists and not which one of the existing

Hamiltonian Paths is the shortest one. Examples of such types of problems might be found in Rahman and Kaykobad (2005). Although this type of problem is also NP-hard, there is no interest in analyzing it for the sake of this report.

In Tamilarasi and Dhenakaran (2018), the authors also propose a brute force exact method for solving the SHPP, by creating a matrix with the distance value of a graph and creating different combinations of routes that passed all nodes in a graph, storing the possible routes and their distance weight. In the end, the route with the shortest weight would be the shortest Hamiltonian path. This approach has the same associated problems as the previously mentioned alternatives.

An interesting approach worth mentioning is Alshamlan and Menai (2012) which, although proposing a very simple and brute force way to solve the SHPP, applies DNA computing to solve it. Even though this area is still growing, it is very promising, given that DNA processes can take place millions of times per second, much faster than computational approaches.

In Bouazzi et al. (2021), an amelioration of the genetic algorithm was proposed. This version denominated as the Improved Genetic Algorithm, uses Dijkstra's algorithm to find the best fitness cost, the roulette wheel selection method, and the one-point crossover as the crossover method. This approach obtained similar results in less computational time in comparison to other Genetic Algorithm approaches. Even though this article proposes a solution to the Shortest Hamiltonian Circuit, as previously discussed, it is possible to alter this solution to the Shortest Hamiltonian Path with simple alterations.

All the previous approaches provide solutions to find SHPs, not the SHP between two given nodes.

In Altinel et al. (2000), three neural solutions to the Shortest Hamiltonian Path Problem between two nodes were proposed. The first approach is a generalization of Kohonen's self-organizing map, known as GSOM_HPP, and the second and third use the Kohonen Network Incorporating Explicit Statistics (Kohonen Network Incorporating Explicit Statistics (KNIES)) that differ from SOM because of two distinctive modules in training phases, the attracting module and the dispersing module. These approaches are named KNIES_HPP and KNIES_HPP_Global.

The authors mention that: "The only difference between KNIES_HPP and KNIES_HPP_Global is that in the latter, the mean of the neurons on the band always moves towards (not onto) the global mean of all the cities, instead of moving exactly onto the local mean only of the cities represented by the neurons."

The KNIES approach is an approach that previously has been used to achieve a very accurate solution to the TSP and was extended to solve the Shortest Hamiltonian Path. It is important to mention that all distances were Euclidean and symmetrical.

These three algorithms are considered very accurate and represent one of the few approaches that do not attempt to solve the SHPP by resorting to a modified TSP approach. Although the results vary in different instances, the solution that provided the biggest number of best solutions was the Knies_HPP followed closely by the KNIES_HPP_Global which has the advantage of having a smaller computational effort associated.

2.5.3 Clustering for solving the Routing Problem

As previously mentioned, a way of simplifying the routing problem is to subdivide the customers into clusters, creating the problem known as the CluVRP.

In Izquierdo et al. (2013), for solving the CluVRP, the problem is divided into two phases. In the first phase, the cluster route is generated (HRP), and in the second phase, the customer route is created (LRP).

To solve the HRP, the clusters are compacted into a virtual center that corresponds to the *center of mass* calculated using the nodes' positions. After compacting the clusters, the Record-To-Record (RTR) proposed by Li et al. (2005), a deterministic version of the Simulated Annealing algorithm, was used to identify the number of routes needed and to generate them.

For the LRP, the SHPP was solved for each cluster. The starting point of each path was the last node of the previous cluster visited (or the depot, in the first cluster), and the finishing point was the virtual center of the next cluster to visit (or the depot, in the final cluster). Three methods were used for solving the SHPP, a Mixed Integer Linear Programming (MILP) exact formulation, the Christofides exact algorithm, and the Lin-Kernighan heuristic. The best results were obtained with the latter.

In Expósito-Izquierdo et al. (2016), the works proposed in Izquierdo et al. (2013) were improved with the addition of an intensification phase, by exchanging each pair of clusters, obtaining the local optimum, and also, the addition of a shaking phase based on the Noising Method, allowing the discovery of unexplored regions in the search space. These changes allowed the results to be better than those previously obtained in Izquierdo et al. (2013). The authors also mention that for smaller instances, exact methods can be applied, returning the optimal solutions for the LRP, and in bigger instances, the Lin-Kernighan heuristic is more suitable, to reduce computational times.

In Le et al. (2022), a clustering approach was used to solve the VRPTW. The customers were clustered by using k-means clustering algorithms and capacitated k-means clustering algorithms. After that, a branch and bound algorithm was executed to generate the solution in a practical time. This approach, although very

limited proved that the grouping of customers benefits the solution of problems with time-window restrictions.

In Sevaux and Sörensen (2008), the authors developed a Mixed-Integer Linear Programming (MILP) model, using the shortest pre-Hamiltonian path to create a new graph containing the connections between the nodes to identify the Shortest Hamiltonian Path inside a cluster. This component is then inserted in a simple metaheuristic that first identifies the possible starting and ending nodes in each cluster, then attempts iteratively recurring to a Simulated Annealing method to order the clusters, changing the starting and finishing nodes progressively and recalculating the SHP regarding these changes. This heuristic is very simplistic and manual and is not adequate for bigger sets of data.

In Defryn and Sörensen (2017), the authors propose that the solution is explored at two different levels, the cluster level and the customer level.

First, a pre-computation process is created where the distances (not necessarily symmetrical or Euclidean) between each node are quantified, as well as the Hausdorfian distances between clusters. Later, clusters are allocated to vehicles using a best-fit decreasing strategy, to introduce a feasible initial solution that considers customer demands, followed by a redistribution that aims to increase the fill rate of the vehicles. This initial solution is then improved at the cluster level with a Variable Neighborhood Search.

Subsequently, a conversion operator translates the current solution into an individual level, recurring to a heuristically calculated TSP inside the cluster that is converted into a Hamiltonian path where the next starting node is the closest one to the current position of the vehicle. After this, a second Variable Neighbourhood Search (VNS) is used at the individual customer level using the distances between the nodes. Finally, a diversification phase uses random changes in order to attempt to leave local-optima.

A similar algorithm is also proposed for the CluVRP with weak cluster constraints. In this variant, vehicles are able to leave the cluster to visit other customers before they finish visiting all the customers in a cluster. This variant differs from the latter one only in this matter, meaning that it functions in a similar way but it analyses more permutations.

In Vidal et al. (2015), by improving the state-of-the-art knowledge in Local Search and Genetic Search to solve VRP, 3 algorithms were developed: 2 based on the Iterated Local Search (ILS) and one based on the Unified Hybrid Genetic Search (UHGS). The ILS algorithms find an initial solution and then iteratively improve it by means of local search procedures, applying inter and intra-route changes, and inter and intra-cluster changes. The UHGS-based algorithm uses selection, crossover,

and education operators to iteratively improve the current best solution, attempting to achieve a better balance between intensification and diversification. The authors compared the results between the three approaches and concluded that the best approach proposed was the UHGS-based method.

In Hintsch and Irnich (2018), the authors decompose the problem into three subproblems, the assignment of clusters to routes, routing inside each cluster, and the sequencing of the clusters in the routes. To solve the second task, several Hamiltonian path problems must be solved for each cluster. A pre-computation process identifies all the Hamiltonian paths for every pair of customers, solving the SHPP as a Travelling Salesman Problem, changing the value of the edge between the entry and exit point to $-M$. Following this step, a large neighborhood search using destroy and repair operators, and for post-optimization, a Variable-Neighbourhood Descent based on the Balas-Simonetti neighborhood.

In Battarra et al. (2014), the authors solve the CluVRP using two exact approaches, a Branch and Cut approach and a Branch and Cut and Price approach. An important preprocessing approach is used, where the SHPP inside each cluster is solved, using TSP solving methods, but using the -M edge value added to the starting and ending nodes. The results obtained by the Branch and Cut and Price revealed, in a case study, to be efficient when compared to methods utilized by certain companies obtaining better results in relatively reasonable times even in large-sized instances.

In Pop et al. (2018), the CluVRP was divided into 2 phases, an upper-lever, and a lower-level subproblem. At the upper level, a genetic algorithm that used a rank-based selection, a custom version of the two-point crossover, and a probabilistic mutation operator, was used to determine the clusters to be visited in each route.

The lower-level subproblem was one of the main advantages of this approach. This component had the goal of determining the visiting order of the vertices within the cluster for each collection of global routes. To solve this, the subproblem was converted into a TSP and solved using the Concorde TSP solver. This transformation into a TSP problem was to add an artificial M cost to all the inter-cluster edges, forcing the vehicle to visit all the vertices within the cluster before leaving. The Concorde solver would then not only solve the intra-cluster routes but also the inter-cluster routes, by determining the next node (and consequently, the next cluster) to visit.

In Islam et al. (2021), a new hybrid metaheuristic was created, based on a combination of the VNS and Particle Swarm Optimization (PSO). The two approaches complement each other with the local optimal improvement capabilities of VNS and the diversification abilities of the PSO. As other authors previously proposed, a

two-phased approach is used.

Given a known number of vehicles, clusters are iteratively added regarding their positional values via the cheapest insertion method. In case the capacity is exceeded in a certain route, a tabu search method is used to insert the cluster, sometimes recurring to swap methods. After the clusters are organized, a similar process to the route generation is utilized for the generation of the paths inside each cluster. This proposal distinguishes itself from others because, after the initial route generation, a VNS is applied to the solution, with local search moves applied to multiple levels: inter-route search, intra-route search, and intra-cluster search. Finally, a PSO is applied, with clusters and customers being removed and reinserted again by probability, using the cheapest insertion method.

2.5.4 Districting for solving Routing Problems

Regarding districting as an alternative to solve Routing Problems, the literature is much less abundant, given that most authors focus on designing the districts and do not proceed to the routing phase. Examples of that are the works by Muyldermans et al. (2003), Miranda et al. (2011), Tavares-Pereira et al. (2007), Kalcsics et al. (2005), Mehrotra et al. (1998), or Fleischmann and Paraschis (1988). Although Haugland et al. (2007) focuses on routing after the districting problem, the only concern is to develop districts that allow future routes not to violate capacity constraints. No efforts in developing an algorithm to create routes after the districting process are done.

In Lei et al. (2012), the routing problem after the districting stage was not tackled, although it is worth mentioning this article because a second stage after the districting problem was devised. This stage created an expected routing cost for each district using the Beardwood-Halton-Hammersley formula.

In Prischink et al. (2016), a two-phased approach was proposed for a Security Control problem. A districting construction heuristic and an iterative destroy and recreate algorithm were proposed to solve the first phase of this approach, which consisted in assigning objects to districts so that the number of districts is minimized (in this case, a district was based on time windows and not on spacial distribution, given that the district represented the number of objects needed to be visited at a certain time window). Besides this subproblem, a routing construction heuristic was devised by means of a Variable Neighbourhood Descent (VND). In this approach, besides time-windows, priorities, visit flexibility, and duration were considered.

In Wøhlk and Laporte (2019), a very large Danish multi-period garbage collection problem was analyzed. To tackle this large problem, the authors adopted the districting approach. The following heuristical approach was utilized: district

conception and assignment to the most adequate day; district balancing in order to use the vehicles more efficiently; and route creation in each district. The creation of routes followed the FastCARP heuristic, an algorithm designed to solve large-scale Capacitated Arc Routing Problem (CARP) within a short computational time, that consists of a series of giant tour merging and splitting processes with paste, switch, and shorten procedures. Although the results created a feasible solution with better quality for the citizens, this proposal would have a cost increase in comparison to the current solution.

In Lei et al. (2016), a Multi-objective Dynamic Stochastic Districting and Routing Problem (MDSDRP) was considered. This problem consists of designing districts for delivery vehicles with the goal of minimizing their costs, where customers of the territory include regular and stochastic customers. The number of vehicles, the compactness of districts, the dissimilarity measure of districts, and the equity measure of vehicle profit were considered the optimization objectives.

To solve this problem, a two-stage process was used. The first stage consisted of the districting process and the second stage consisted of the routing process. A Multi-Objective Evolutionary Algorithm (MOEA) called PICEA-g-mr was introduced, modeled, and solved in order to create the districts. To solve the routes, as the number of edges was always under 100, the Concorde solver was utilized. Every route fulfilled every client in a single district. Symmetric Euclidean distances were utilized.

Croci (2020) proposes a structure to solve the problem of consistent routing problem by means of a two-phased process:

1. In the first phase, the author proposed the creation of districts for each driver. To achieve this objective, a multi-period balanced $p - median$ is suggested.

2. After the first component, the author proposes that taking into account the previously created districts, a Capacitated Vehicle Routing Problem (CVRP) will be solved each day. The author considered that each route could not exceed the maximum travel cost allowed per vehicle, capacity constraints, and penalized routes that travel across district boundaries. As the instances were large, the utilized method to solve this component was a large neighborhood search metaheuristic, the Adaptive Large Neighborhood Search (ALNS).

The results obtained from this approach were satisfactory. Nevertheless, it is worth mentioning that it did not consider that the customer requests are not known at the beginning of the time horizon, meaning that the districts might need to suffer slight changes with daily demand fluctuation.

Another very interesting work can be seen in Zhong et al. (2007), where the authors develop an approach that attempts to generate routes with a great degree

of compactness, efficiency, and flexibility while obeying constraints on customer demands and maximum working duration for each driver while considering a very differentiating topic: minimizing day-to-day variations in driver territories. This article introduces an approach based on driver learning curves, considering that the more familiarized a driver is with a certain area, the more effective he will be performing a route in it. By creating core areas, that consist of areas containing stops always delivered within the same district by the same driver, cells (consisting of the minimum unit of a service area whose workload is assigned to a single driver, i.e. an aggregation of customers in a certain area, to be served by a single driver) not contained in these areas are rearranged daily to the most suitable area to minimize costs and driver variations.

So, the routing component of this work consists of two phases: the first phase is responsible for aggregating core areas to the most suitable free cells, as previously mentioned, as well as assigning the drivers to visit each of these areas. The second phase consists of route development. To create these routes, an algorithm was developed based on a parallel insertion heuristic. The authors utilize the shortest connection between two areas to connect two independent districts that are to be visited by the same driver. This approach proved to build low-cost and driver-consistent routes, achieving very satisfactory results.

In Bender et al. (2020), the authors propose a two-stage districting approach that first establishes districts and later adapts them to daily demands. The authors, in contrast with the majority of the literature, considered a not previously known number of districts, a heterogeneous fleet of vehicles, a heterogeneous set of drivers, and a stochastic demand.

In the first stage, their goal was to partition the basic areas into a proper number of districts, assign a driver type and a vehicle type to each district, and create districts based on data patterns. For that, three different models where different aspects like input data detail and workload limits vary. In the second stage, the goals were to reassign basic areas according to the demands while preserving driver and vehicle assignments mentioned in the first stage. A model was created to optimize the geographical compactness as well as the workload over all districts.

To achieve these goals, in both phases, an exact mathematical model was designed. But as expected, larger sets of data proved to have insupportable computational times. So, the authors devised heuristics based on the formulations to heuristically solve these problems. The heuristic consisted in solving the integer programming models previously devised, but with heuristic constraints (for example, in the routing stage, the variable constraint was the number of basic areas that were allowed to change compared to the districting stage). The obtained results were still time-

consuming but it was revealed that a very good operational feasibility was obtainable using the proposed heuristics.

2.6 Final considerations

After completing the literature review, the groundwork is now laid out to outline and contextualize the upcoming work in greater detail. To conclude the present chapter, the approaches adopted during the development of the thesis must be stated.

The route creation process will be divided into two distinct phases: Intra and Inter-District phase. In the Intra-District phase, where the routing within each district will be performed, will follow a LKH-based heuristic, as it is one of the best-performing approaches for solving the Travelling Salesman Problem, as mentioned by Lu et al. (2022) and further detailed throughout Section 2.5. After that, the obtained Travelling Salesman Problem solution cycle will be converted to a Shortest Hamiltonian Path using the approaches suggested by Altinel et al. (2000). In the Inter-District phase, connections between the Intra-District routes will be created following approaches proposed by Zhong et al. (2007), Izquierdo et al. (2013), and Expósito-Izquierdo et al. (2016).

Chapter 3

Dataset description

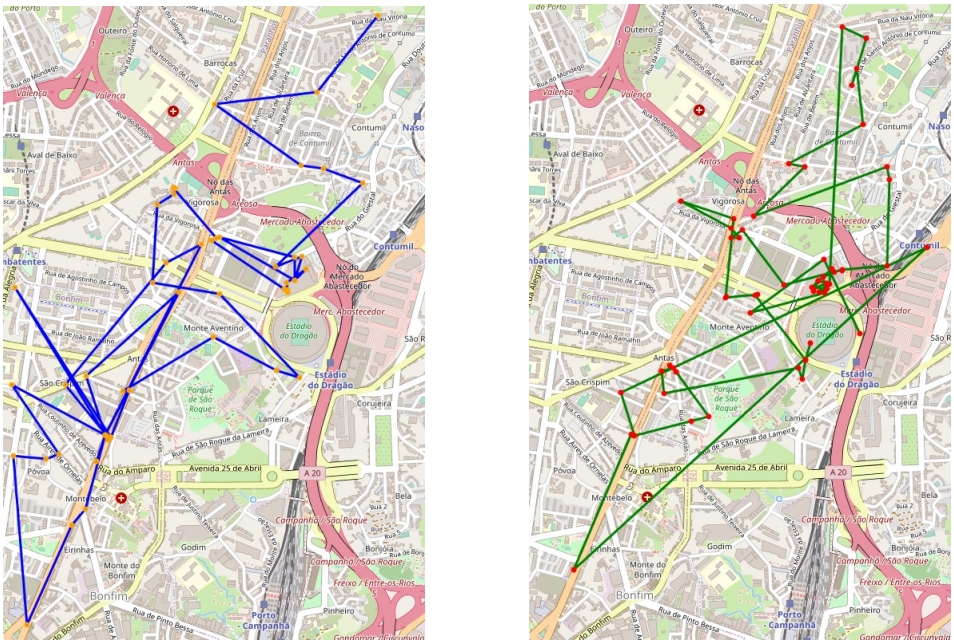
In this chapter, as the name indicates, the dataset will be described. A package delivery company provided a dataset containing real delivery information and performed route data. In Section 3.1, a comprehensive overview of the data is provided, elucidating the intricacies of the data attributes and their definitions. Section 3.2 delves into an exploration of prior works that form the foundation upon which the current project builds. This involves a thorough examination of their objectives, methodologies, and project outputs, which, in turn, serve as inputs to the current work. Section 3.3 provides a comprehensive overview and analysis of the available data.

3.1 Data source and description

A package delivery company supplied the dataset for the current project. It consists of two .xlsx files representing real data from the company's operation in the Porto district and its surroundings between 01/07/2021 to 30/09/2021 and 01/10/2021 to 31/12/2021[1]. The two files were merged, providing the project's original dataset. The dataset is presented in a tabular format, where each line represents one package, and each column contains information about that package. A package is defined as a specific parcel delivered to a certain customer, and each line refers to one single package. However, multiple packages can be delivered to the same client. This creates the definition of a stop, which consists of a group of packages delivered at a certain location to one or multiple clients, i.e. the act of actually stopping the vehicle and delivering one or multiple parcels at a given location.

Another crucial concept to grasp is the notion of a **circuit**. Each shipment is assigned a unique circuit code that signifies the type of delivery route and reflects

[1]The company does not deliver on Sundays.



(a) CIRC1 in 28/12/2021 **(b)** CIRC1 in 29/12/2021

Figure 3.1. CIRC1 Route in different dates

the geographical region where it is situated. These codes are repeated throughout different dates and in many cases occur daily. Although the concept of a circuit may seem similar to the concept of a district, addressed throughout this work, it is important to note that the two are not connected.

Additionally, one also very important concept to understand is **routes**. A route is a certain set of stops with the same circuit code made in a single day. A route can only occur once daily, but the clients, and as a consequence, the stops, vary with each day's demand. The route's order is defined by the delivery_date.

Examples of different routes can be seen in figures 3.1a and 3.1b. As seen, routes are mostly constrained to a specific area. Stops in this area are part of circuit CIRC1. Figures 3.1a and 3.1b represent the same circuit on different days, i.e. different routes. It is possible to identify similarities between both routes, considering that they share the same circuit code, are relatively in the same geographic area, and share some common stops. However, they are not identical since customers, and consequently, stops, vary from day to day.

For further information about the provided data attributes in the original provided file, see Santos (2023).

3.2 Previous works

The current work is part of a multi-phased project, divided into 3 phases.

The initial task, the districting phase, involved crafting a model that, relying on historical route execution data, partitions the geographical sphere of influence into smaller, contiguous areas known as districts. This tactical decision solidifies districts for a specified timeframe. The overarching aim is to enhance service consistency and streamline administrative processes by facilitating daily route planning.

As for the second phase, the sequencing phase, the focus shifts to devising an algorithm that employs machine learning methods. This algorithm operates with the previously defined set of districts and historical route execution datasets, ultimately yielding a sequence of districts. In essence, it determines a preference ranking for the subsequent district in the route for each district.

The third phase, which is the focus of this research, as described, involves designing delivery routes that visit all customers within each district, effectively solving the Shortest Hamiltonian Path Problem for each district. The design of these routes will allow the analysis of the best-performing used methods for route creation in districting approaches, which is the main focus of this dissertation.

Since the current work relies on previous ones, the original dataset has undergone several modifications. The data has been pre-processed, as discussed in Santos (2023), which involved removing outliers, replacing incorrect values with correct ones through checks on key data points such as customer latitudes and longitudes, delivery times among others, removing missing values, and also grouping packages into stops (grouping packages with similar locations or delivery times).

Given these modifications, the main dataset containing information about the stops now comprises data for 542,908 stops, corresponding to 139 circuits, and 8.524 distinct routes. The file containing this dataset is one of the .json components received from previous phases.

In phase 1 of the project, a hierarchical districting approach was used to create the desired districts, with 3 levels: macro, micro, and nano. A macro-district is a set of micro-districts, and, a micro-district is a set of nano-districts. Districts were developed based on maximum workload in time and demand.

The proposed solution encompassed 6 macro-districts, 49 micro-districts, and 257 nano-districts. Micro and nano-districts were delivered in .json files as polygons, i.e. the sets of coordinates defining the points connecting the districts' frontiers.

Micro and nano-district information was provided in .json files micro_polygons_after3 and polygons_nano_after_change4, respectively. The file structure follows the same pattern, given that both files are a matrix where each line represents a district and each column contains an attribute.

Attribute	Description
stop_id	Unique identifier for a stop
date	Date of delivery, in YYYYMMDD format
circuit	Circuit identification code
latitude	Latitude of the stop
longitude	Longitude of the stop
performance	Categorical variable regarding route performance varying from good, acceptable, and unacceptable
productivity	Categorical variable regarding route productivity varying from good, acceptable, and unacceptable
weight	Combined weight of all packages delivered at the stop
delivery_time	Time that the first package was delivered at the stop, in HHMMSS format
time_window	Categorical variable indicating the time constraints for deliveries on specific stops (A, D, F, M, W, Z)
no_deliveries_stop	The number of packages delivered in the stop

Table 3.1. Dataset attributes

The final relevant received component was a Python script named write_ins_solve_tsp used to solve the second component of the three-phased problem within which the current work is embedded. This component finds the order of districts to be visited by solving a TSP among all districts. However, this TSP is not based on distances but rather on probabilities of connection between districts, based on the collected driver decision data. For a more comprehensive understanding of the methods used to identify the best sequence between districts, consult Santos (2023).

3.3 Data overview

As mentioned, the new main dataset containing information about the stops now comprises data for 542.908 stops, corresponding to 139 circuits, and 8.524 distinct routes.

The new dataset's attributes and respective descriptions may be seen in table 3.1.

The new data contains no NA values in the main values such as stop_id, latitude, longitude, weight, circuit, or delivery time, as the data already underwent thorough preprocessing in previous phases.

In the 'time_windows' attribute, the categorical variables representing the delivery constraints for the stop are: A, indicating a delivery that must be completed before 10 AM; M, signifying a delivery that must be finished before 2 PM; and D, denoting a delivery that must be finalized before 6 PM. Values F, W, and Z were

Attribute	Description
level	Identifies the district level: 0 for Macro-Districts, 1 for Micro-Districts, and 2 for Nano-Districts
polygon	List of lists where each inner list represents a point in the format [latitude, longitude]. These nodes represent the district's vertices, and the connections between these nodes form the district boundaries.
center	Basic Unit (BU) representing the center of the polygon
center_up	BU that represents the center of the district where this district is inserted
code	District's unique identification code consisting of 2 letters
notOnPolygon	Connections that form the polygon but do not belong to the polygon in question (because of district boundaries overlapping)

Table 3.2. Micro-District and Nano-District Dataset attributes

deemed irrelevant and will be treated as D values, as per the company's specifications.

Beyond the data mentioned earlier, information regarding districts is also available. In the previous study, the proposed solution encompassed 6 districts, 49 micro-districts, and 257 nano-districts (Santos, 2023).

The attributes provided can be seen in table 3.2. However, see figure 3.2 for a better understanding of the districts' locations[2].

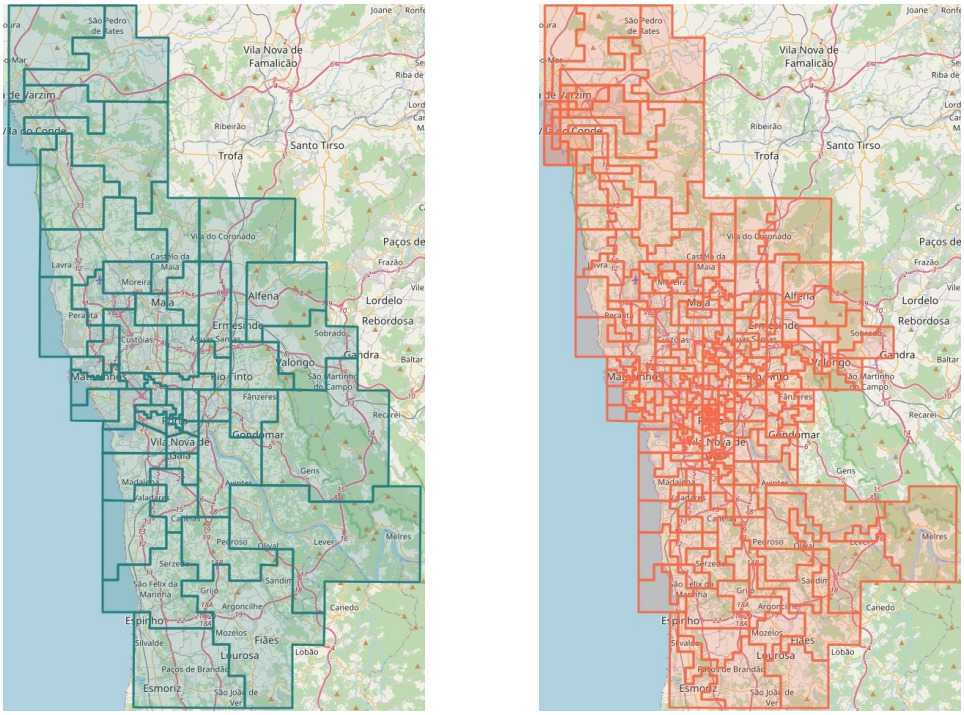
It is very important to understand what each level of district represents in the present work. In this phase, micro-districts facilitate the identification of stops for the routes in the proposed model, considering that micro-districts aim to limit the maximum number of stops a driver completes in one workday. Consequently, all stops belonging to a specific micro-district will be part of the same route.

As for nano-districts, in this work they enable the grouping of stops, practically allowing aggregation by "districts." Furthermore, the second phase of the project, as explained earlier, aims to establish the order of districts to visit. In other words, it defines the order of nano-districts to visit within each micro-district. Therefore, to develop the final route, a SHPP must be solved for each one of the nano-districts, while also following the defined order of the nano-districts to visit in a specific micro-district.

Macro-districts do not exert any influence on the ongoing phase, as the focus is solely on the importance of micro and nano-districts in this context.

Regarding the spatial distribution of the districts' stops, given the extensive service area with a varied distribution of stops, it was anticipated that a portion of these stops might extend beyond the demarcated districts. Upon analysis, it

[2]Macro-districts were not represented as they do not influence the current study



(a) Micro-Districts **(b)** Nano-Districts

Figure 3.2. Different District levels

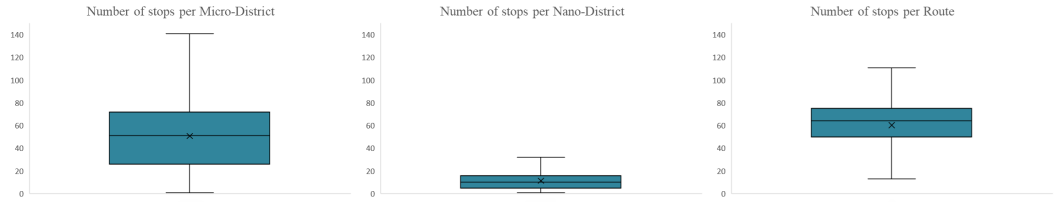


Figure 3.3. Number of Stops per Route, Micro-District, and Nano-District

was revealed that 27.144 stops reside outside the confines of any proposed district, comprising 5,00% of the overall stops. These stops are linked to 110.136 packages, contributing to 7,86% of the entire package count. Among the 8.524 routes analyzed, 814 encompass one or more stops outside the recommended districts, while 7.710 routes confine all stops within the specified districts.

In the current model, for a given day, stops contained in a circuit represent the stops a driver would have to visit. However, in the proposed model, stops contained in a micro-district correspond to the stops that a driver must perform in one day.

The number of stops can be an important factor for the route creation and should be considered. Therefore it is worth analyzing the number of stops per micro-district and nano-district that the proposed models would have if implemented, and set a comparison with the number of stops per route in the current model.

Figure 3.3 compares the number of stops per route in the current model to the number of stops per micro-district and per nano-district in the proposed model.

In the current model, on average, a route consists of 61 stops, with the minimum registered number of stops per route being 1 and the maximum being 140.

In the proposed model, on average, a micro-district, and consequently a route, would consist of 51 stops, with the minimum registered number of stops per route being 1 and the maximum being 187. As for nano-districts, on average, these would contain 12 stops, with the minimum registered number of stops per nano-district being 1 and the maximum being 77. These values would represent the lengths of the Intra-District routes created.

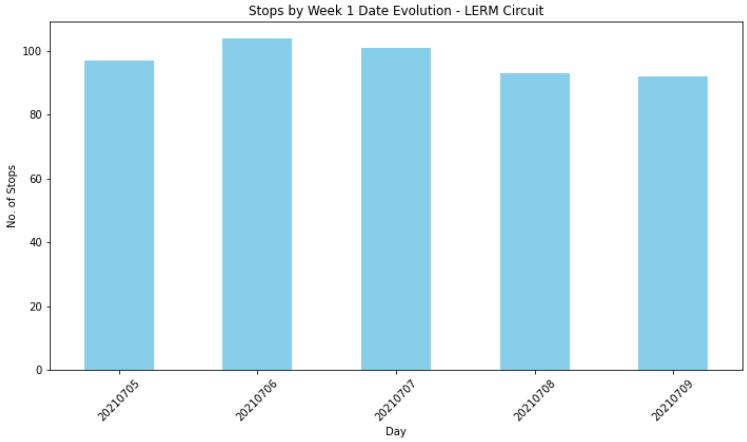
For a direct comparison, looking at routes contained in only 1 micro-district and therefore would be equal in the current and the proposed models, routes have on average 17 stops, with the maximum number of stops being 83, and the minimum 1. In these routes, nano-districts would contain on average 11 stops, with the maximum number of stops being 46, and the minimum 1.

Through the analysis of the data, it is possible to verify that, in the proposed model, the average number of stops that a driver has to visit per route decreases during a week. This means the number of drivers would have to increase in the proposed model if a 1 driver per micro-district rule is considered.

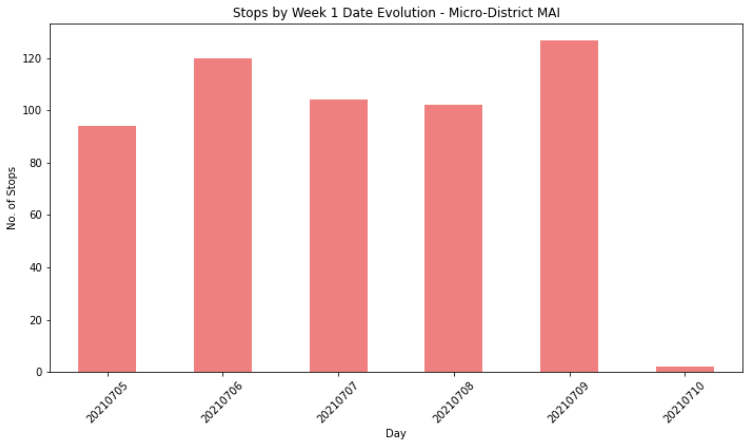
To also understand the daily behavior of demand, one should analyze the stop evolution during a week. Focusing on week 1, the circuit with the largest amount of stops would represent, at least theoretically, one of the most challenging to meet demand. For week 1, this circuit was CIRC1. In comparison, for the proposed model, the theoretical micro-district would also be the one with the highest number of stops. This micro-district corresponds, in week 1, as stated previously, to MAI.

The number of stops made in routes in the CIRC2 circuit can be seen in figures 3.4a, while the expected number of stops with the proposed district MAI may be seen in 3.4b. As seen in the graphs, the number of stops per day maintains a relatively stable number, of around 100 stops per day, in 3.4a, and of around 95 to 125 stops in 3.4b. This means that, in the proposed model, one can identify a bigger variation of stops per day. It is also clear that a substantial decrease in the number of stops happens on Saturdays (in week 1, 20210710).

The final relevant data to further analyze is the transition data. As mentioned before, a script named write_ins_solve_tsp is used to identify the order of districts to visit i.e. the order of nano districts to visit in each micro district. This script's inputs are a .json file named transition_log, which contains the connection probabilities between each district (as calculated in Santos (2023)), and also the data from the main data file. However, 2 extra attributes had to be added to each stop: the



(a) Stops by Week 1 Date Evolution - CIRC1 Circuit



(b) Stops by Week 1 Date Evolution - Micro-District MAI

Figure 3.4. Stops evolution during Week 1

Attribute	Description
micro	Name of the ordered micro-district
date	Date of the proposed order of nano-districts in YYYYM-MDD format
solution	Proposed sequence to visit the nano-districts
distance	Proposed sequence's Euclidean distance

Table 3.3. Nano-District order dataset

micro-district code, and the nano-district code to which this stop belongs. The outputs obtained from the script are a .json file named solutions_distance containing the attributes mentioned in table 3.3.

The script, and consequently, the contents of this table were slightly altered to fit this phase's requirements better. The changes consisted of altering the solution from numerical (3-2-4-1) to the corresponding district names (NAD-NAB-NAA-NAC), and also presenting only the best sequence, instead of all the sequences possible, as proposed in the original version of the script.

Chapter 4

Data adaptation

Throughout this chapter, the raw data received will be prepared and adapted to the current problem. In section 4.1, the data adaptation process is outlined, encompassing verification of essential attributes, handling potential missing values, and detailing a stop aggregation phase. Section 4.2 focuses on outlier handling. Lastly, final data tuning performed is described in section 4.3.

4.1 Overview, Missing Value Verification, and Stop Aggregation

Considering that the project in question is the third phase of a three-phase project, it is expected that the original data used has already been pre-processed, as can be verified in Santos (2023), where extensive data processing was performed. Therefore, it is expected that the data to be used in the present work will exhibit a very high degree of adequacy.

However, potential gaps must be properly identified and, if possible, corrected before proceeding with resolving the routing optimization problem.

To achieve this multiple steps were taken. The first step in the data adaptation process was to ensure that the dataset contained all the necessary information for the developed approach to function correctly. To do this, a verification of the attributes mentioned in Table 3.1 was performed, guaranteeing that the dataset presented all the expected attributes.

The second step was to check for the existence of NA values in the following fields: latitude, longitude, and delivery_time, the most essential data components. As expected, due to the preprocessing performed in the previous phases, no NA values were found in the data.

The next phase consisted of the Stop aggregation phase. A stop should be

considered a single moment where a driver leaves the vehicle to deliver one or more packages. So, one should aggregate packages to stops if in a certain route:

1. Two packages have the same latitude and longitude, as well as the same delivery time.

2. Two packages have the same latitude and longitude and are delivered in consecutive delivery times.

3. Two packages have the same delivery time while having or not having the same latitude or longitude.

In case packages were aggregated, the attributes maintained the values of the first package, except in time_windows where the string identifying the time-window for the delivery of that package would be added to a list containing all the time windows of all the aggregated packages, and no_deliveries_stop, where the number of deliveries in that particular stop is incremented in 1.

Once again, given that a similar process had already been conducted previously, no alterations were performed within the data set.

4.2 Outliers

The new proposed solution will be based on creating routes corresponding to the stops belonging to a micro-district on a given day. Therefore, not all routes will be suitable for the study in question.

As previously seen, the geographical distribution of customers is extensive, with a large number of customers whose location is out of the proposed district's borders.

An approach that might be possible would be to create a new temporary and fictitious district and to consider that all the stops without a district belong to this district. However, this approach could be challenged by the information presented in Figure 4.1. Upon closer examination, the stops located outside of any districts (represented in green) would be grouped and, consequently, visited in a sequential order. It becomes evident that entities would be grouped despite possibly being geographically distant. This would lead to an unrealistic and incongruent route with excessively long distances. Therefore, only routes that contain 100% of their stops within district borders should be considered.

As such, an algorithm that verifies if all stops on a given route belong to a micro-district and a nano-district has been developed, also taking into consideration overlapping district borders. If any stops on a route do not belong to both these levels of districts, the route is discarded.

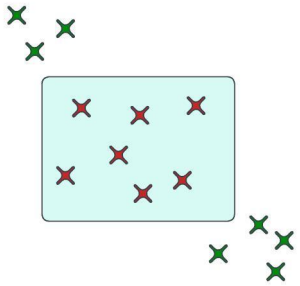


Figure 4.1. Example of a possible districting case using not processed routes

4.3 Data tuning

The last step of the preprocessing phase consisted of performing changes to adapt all the components to fit the developed approach better and allow all the elements to function more harmoniously.

Initially, an algorithm was developed to identify the micro-district and nano-district where each stop belonged. These two new attributes were added to the data file, with the names micro_district and nano_district, respectively.

Furthermore, it is important to understand that there will be two analysis phases, each requiring different dataset divisions. In both these datasets, data will not contain outliers (stops not within any district), as explained in subchapter 4.2.

Therefore, to be able to compare proposed routes with current routes directly, if one analyzes proposed routes that are 100% within only one micro-district, previous routes, and proposed routes visit the same stops, allowing a direct comparison between the proposed routes.

This would not be possible if routes that span multiple micro-districts were considered. In the new model, each micro-district would have its own route, serviced by its designated driver. Consequently, comparing the performance of the two models would not yield comparable results, and would fail to provide meaningful insights into the analyzed components.

An algorithm was developed to identify these routes, and a .json file was subsequently created for each route. In total, 179 files were created, making them the final dataset for the model comparison.

Chapter 5

Route Creation Algorithm Development

Similarly to many literature approaches, a two-step approach was followed in the current work. As such, two distinct phases are notable: an Inter-District Phase and an Intra-District Phase. In section 5.1, the components developed to tackle the Intra-District phase. Subsection 5.1.1 explains the used approach to solve a TSP in each district, and subsection 5.1.2 how to convert this TSP solution into the desired SHPP solution, with the use of converters. Subsection 5.1.3 focuses on the types of distances used to solve the problem, as well as how to obtain them. In section 5.2, the three approaches developed to solve the Inter-District Phase are detailed. Inter 1 - Centroids and Inter 1 - Medoids are both presented in 5.2.1, whereas Inter 2 - Shortest Arcs is detailed in 5.2.2. The project's main function as well as the proposed model variations are detailed in 5.3.

5.1 Intra-District Phase

In the Intra-district phase, the goal is to identify the best approach to solve the Shortest Hamiltonian Path Problem in each district. The SHPP, applied to the current problem, attempts to find a minimum-cost path visiting all the stops in each district exactly once, starting and finishing in two given stops.

In the literature, most analyzed approaches attempted to solve the SHPP tackling it as a TSP. A reason for that is the much more extensive and diverse literature regarding the TSP, along with a significantly larger number of powerful tools for solving it. Similarly, the approach adopted in this work will also attempt to utilize tools developed for solving the TSP. The results will subsequently be converted to a SHP in the most efficient manner possible.

5.1.1 LKH

According to the literature, some of the best-known heuristics currently available for solving the TSP are Lin-Khernighan-based heuristics.

Although the original version of the LKH heuristic by Helsgaun (2022) is publicly available for use, due to computational incompatibilities, an alternative approach had to be implemented. Therefore, the Elkai library was used (Dimitrovski, 2023).

This library, based on Helsgaun (2022), is proven to achieve optimal solutions in routes with up to 315 nodes. As previously seen in Chapter 3, in the whole dataset, the highest number of stops in a single nano-district is 77. As such, this heuristic is expected to find the optimal solution to each TSP problem within the nano-districts' stops, achieving similar results when compared to the original heuristic.

As inputs, this library needs one of two variants: either 2D coordinates or a Distance Matrix should be provided. If the 2D coordinates approach were used, the distances utilized by the library would necessarily be Euclidean, so the distance matrix approach was chosen, to allow more flexibility. Another reason why the matrix approach was chosen was the need to convert the TSP solution into a SHP solution.

Asymmetric or symmetric distances can be solved using this library.

5.1.2 Converters 1 and 2

Using a TSP-solving approach to solve a SHPP is possible, but demands converting a cycle into a path. As demonstrated in Chapter 2, there are two approaches to performing this conversion. Two functions performing the conversions were developed and named "**Converter 1**" and "**Converter 2**".

"**Converter 1**" uses the connection approach. By setting the distance between the first and the last stops to visit in the path to approximately zero. All other distances are also multiplied by 1000. The connection between the first and last stops is ensured, given that the TSP solver will always use the connection that minimizes the total distance. After this, the connection between the first and last stops is eliminated and the cycle becomes a path.

As of "**Converter 2**", the intermediary stop approach is used. By creating an imaginary stop X, setting the distance from this stop to the first and last stops to zero, and the distance to every other stop to an upper bound M, with far greater significance than any other distance in the route, the connection between stop X and the starting and ending stops in the route are guaranteed. After this, the stop is eliminated from the cycle, as well as the connections to it, converting the cycle into a path.

In both converters, it was crucial to ensure the cycle maintained the desired

direction (for example: depot-stops-centroid and not centroid-stops-depot), so to ensure this, only the desired direction between stops was enforced, i.e. if the desired direction was A to B, the distance from A to B would be small and the distance from B to A would be very high. This way the desired direction is guaranteed.

5.1.3 Distance Types

As the current problem tackles a routing problem, the two main types of distances utilized would be Euclidean distances or Road Distances.

Euclidean distances are the fastest approach to calculating the distance between two points and are always symmetrical. Road distances are a much more accurate distance representation between two locations for routing problems because they consider the distance a driver would have to drive from point A to B. However, this distance's calculation is significantly harder than the Euclidean distance, resulting in a much bigger associated computational time.

For the calculation of the road distances, OSMnx, a Python package that allows the user to download, model, analyze, and visualize street networks and other geospatial features from OpenStreetMap (OSM) was utilized (Boeing, 2017). NetworkX, a Python package for the creation, manipulation, and study of networks was also used (Hagberg et al., 2008).

Road distances are not symmetrical but this simplification can be done to facilitate computational complexity. So, three algorithms were developed. The first allowed the calculation of Euclidean distances between every stop, while the second and third calculated asymmetric and symmetric road distances, respectively. The three algorithms generate a $N \times N$ distance matrix.

A Porto Metropolitan Area graph was used to calculate the distances between origin and destination points. Initially, the node in the graph closest to the origin stop is identified, and this node is called the origin node. The same process is repeated with the destination point, identifying the closest node in the graph - the destination node. Finally, using the graph's edges, the distance between the origin and destination nodes is calculated. This distance is the approximation assigned to the distance between points A and B.

However, one of the problems identified with using OSMnx and NetworkX was that it was often impossible to calculate the distance between two nodes in the graph, due to the nonexistence of a connection between two given nodes. Two options were possible to solve this problem:

Option 1: Calculate the Euclidean distance between two nodes when it is impossible to calculate the road distance.

Option 2: Replace the nearest nodes with the second-closest nodes.

Option 1, although much faster, could lead to unrealistic results, as a very small Euclidean distance could connect two points, but have a very large associated road distance. For example, in cases on opposite banks of a river, the driver would have to travel to the nearest bridge.

Therefore, option 2 was adopted and the algorithm described by pseudocode 5.1 was developed.

Algorithm 5.1 Road Distance Calculation

```
graph ← Import Porto Metropolitan Area's Graph
node_list ← Identify nearest node to each stop
for node_origin in node_list do
    for node_destiny in node_list do
        if node_origin = node_destiny then
            distance ← 0
        else
            distance ← Road Distance between node_origin and node_destiny
            if not solved  then
                graph_copy ← graph
                graph_copy ← remove node_origin and dest_node from graph_copy
            end if
            while not solved do
                new_orig ← New nearest node to origin stop with graph_copy
                new_dest ← New nearest node to destiny stop with graph_copy
                distance ← Road Distance new_node_origin to new_dest_node
                solved ← True
                if not solved  then
                    graph_copy ← remove new_orig and _new_dest from graph_copy
                end if
            end while
            distancias_rodoviarias.at[node_orig, node_dest] ← distancia
        end if
    end for
end for
```

As seen, if the distance calculation fails, a copy graph is created. In this copy graph, the origin and destination nodes are deleted. The new two nodes closest to each other are then identified. This process is repeated until the distance between the origin and destination nodes can be calculated.

As seen, the previously mentioned algorithm's output would be an asymmetric matrix of distances, i.e. distance from point A to point B is not necessarily equal to the distance from point B to point A. Symmetric Distances, where the distance from A to B is the same as from B to A, are calculated similarly, where only one of the distances is calculated and its symmetric value is assumed as equal. This is done to save computational efforts, calculating only half of the distances, when compared to

the asymmetric variation, although presenting less realistic results.

5.2 Inter-District Phase

After explaining the Intra-District components and their implementation, it is important to do the same in the Inter-District components.

Inter-District components are responsible for connecting each of the Shortest Hamiltonian Paths in each nano-district considering the received nano-district visiting order identified by the sequencing phase's algorithm.

5.2.1 Inter 1

Inter 1 is the first developed approach to tackle the connection between districts. It is based on Izquierdo et al. (2013) and Expósito-Izquierdo et al. (2016).

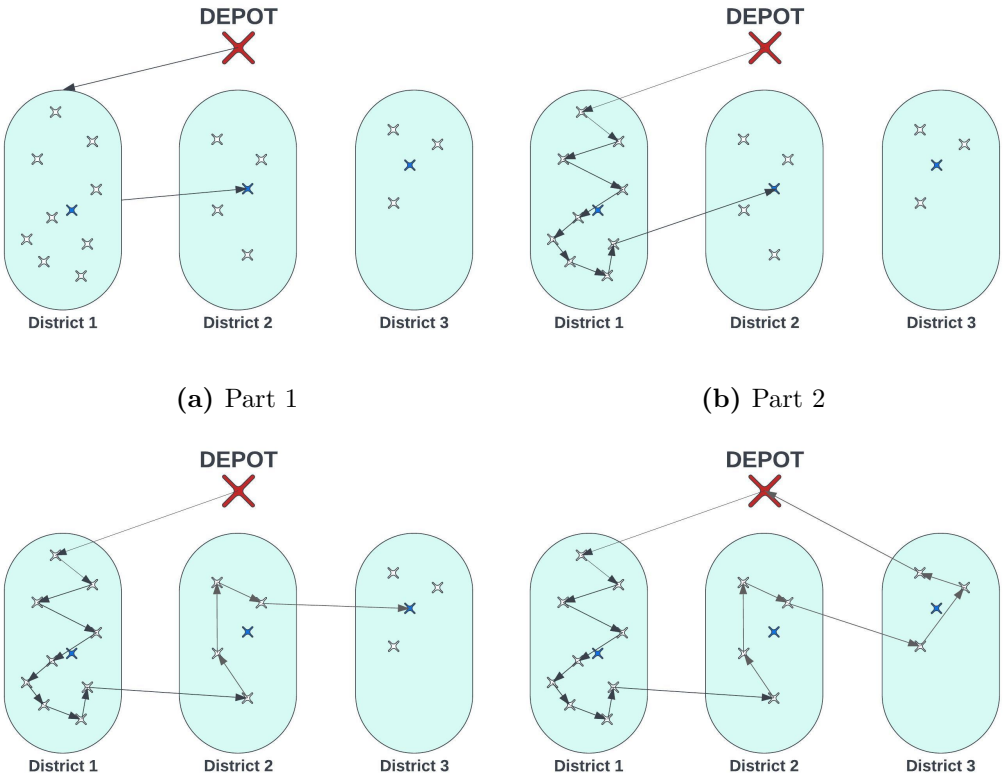
Given a certain nano-district N, contained in a sequence of nano-districts to visit, this method utilizes a temporary connection point in district $N+1$, the next district to be visited. Based on the approach, this temporary connection point can either be imaginary or real. A SHPP will be solved starting from the last stop visited in district $N-1$, passing through all the stops to visit in district N, and finishing in the imaginary point from district $N+1$. The first stop and the last stop will be different in the first and last districts respectively, as both of these will be the depot.

As seen in figure 5.1, each district is considered an independent step, starting from the last stop visited in the last district and finishing at the temporary connection point. After those connections are set, the SHPP is calculated inside the district. Lastly, the connection with the temporary connection point is eliminated. The process proceeds until all the stops in all the districts are visited. For further understanding view algorithm 5.2

The pseudocode above demonstrates the 3 stages necessary for the correct functioning of the Inter 1 approach. Stage 1 tackles the first nano-district, where the origin is the depot, the designated stops are nano-district 1's stops, and the destination is the next nano-district's temporary connection point. Stage 3 tackles the final nano-district, where the origin is the last stop in the penultimate visited nano-district, the designated stops are the final nano-district's stops, and the destination is the depot. Stage 2 tackles all the nano-districts except the first and the last, where the origin is the last stop of the previously visited nano-district, the designated stops are the stops from the nano-district in question and the destiny is the temporary connection point from the next nano-district to visit.

There are two variants of Inter 1: **Inter 1 - Centroids** and **Inter1 - Medoids**.

In **Inter 1 - Centroids**, as the name implies, the imaginary connection point



(a) Part 1

(b) Part 2

(c) Part 3

(d) Part 4

Figure 5.1. Connection between districts using the Inter 1 approach

Algorithm 5.2 Inter1

define *depot*
stops_dist ← *stops* **from district** 1
final_stop ← *centroid* **from district** 2
stops_dist ← **concatenate** *depot*, *stops_dist*, *final_stop*
route_excerpt ← TSP_LKH(*stops_dist*)
final_route ← *route_excerpt*
for each district $x \leftarrow 2$ **to** $num\_of\_districts - 1$ **do**
    remove last row from *final_route*
    initial_stop ← **last row from** *final_route*
    final_stop ← *centroid* **from district** $x + 1$
    stops_dist ← *stops* **from district** x
    stops_dist ← **concatenate** *initial_stop*, *stops_dist*, *final_stop*
    route_excerpt ← TSP_LKH(*stops_dist*)
    final_route ← **concatenate** *final_route* **and** *route_excerpt*
end for
remove last row from *final_route*
initial_stop ← **last row from** *final_route*
final_stop ← *depot*
stops_dist ← *stops* **from district** *num_of_districts*
stops_dist ← **concatenate** *initial_stop*, *stops_dist*, *final_stop*
route_excerpt ← TSP_LKH(*s_dist*)
final_route ← **concatenate** *final_route* **and** *route_excerpt*
return *final_route*

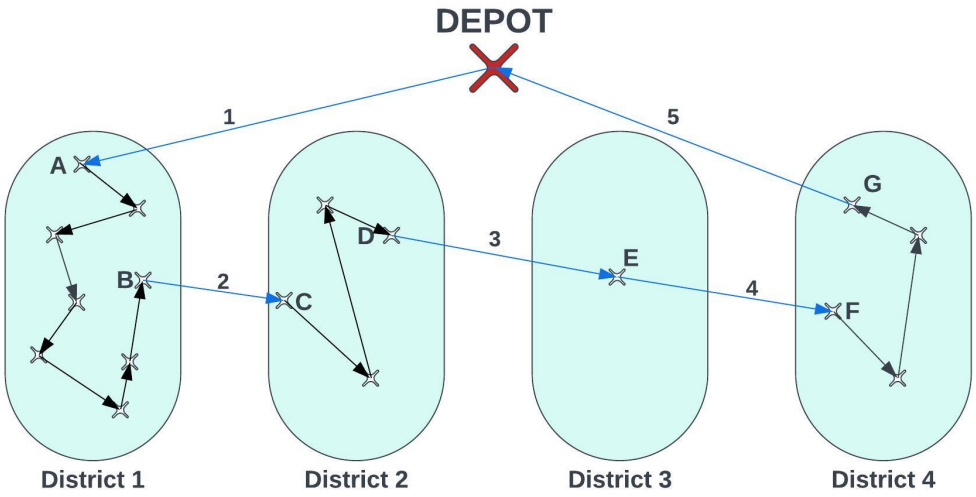


Figure 5.2. Inter 2 - Shortest Arcs

between districts N and $N+1$ is the centroid of district $N+1$. To identify the centroids in each district, an algorithm named **Calc_Centroides** was developed, calculating the means of the latitudes and longitudes of all the stops in each district. If a district has only one stop, the centroid will be coincident with the stop. Figure 5.1 illustrates the **Inter 1 - Centroids** approach.

In **Inter 1 - Medoids**, instead of using the centroids, the medoids are used. A medoid is the stop in a district that is closest to its centroid. So, to identify the medoids in each district, an algorithm named **Calc_Medoids** was developed, first calculating the centroids similarly to **Calc_Centroids**, but later identifying the closest existent stop in the district to its centroid, using Euclidean distance.

5.2.2 Inter 2 - Shortest Arcs

The last developed approach to tackle the connection between districts was named **Inter 2 - Shortest Arcs** and is based on Zhong et al. (2007) approach. This approach takes a simpler route to Inter-District connections and employs a two-phase strategy, first establishing Inter-District connections and then Intra-District connections.

In figure 5.2 it is possible to understand **Inter 2 - Shortest Arcs**'s process. The blue arrows illustrate the first step and represent the connection between the closest stops in each nano-district. In the second step, represented by the black arrows, the SHPP is solved within each nano-district, ensuring that the first and final stops in each district obey the shortest arc between nano-districts rule.

As seen, this method involves identifying the smallest connecting arcs between nano-districts and enforcing their connection before creating the SHP within the districts. In the first and last nano-districts, the depot is considered one of the nano-district's stops, while making sure that it is the first and last visited stop of the route.

Algorithm 5.3 Calculate Shortest Arcs

```
num_districts ← length of districts
shortest_arcs ← []
after ← 1
for all district and district + 1 combinations in districts do
    stops_dist1 ← stops ∈ district
    stops_dist2 ← stops ∈ district+1
    if length of stops_dist1 = 1 then
        for all stop1 in stops_dist1 do
            for all stop2 in stops_dist2 do
                distance ← Euclidean distance between stop1 and stop2
                if distance < shortest then
                    shortest ← distance
                    ind1 ← stop1
                    ind2 ← stop2
                end if
            end for
        end for
    else
        for all stop1 in stops_dist1 do
            for all stop2 in stops_dist2 do
                if indice1 ≠ after then
                    distance ← Euclidean distance between stop1 and stop2
                    if distance < shortest then
                        shortest ← distance
                        ind1 ← stop1
                        ind2 ← stop2
                    end if
                end if
            end for
        end for
    end if
    after ← ind2
    Add the connections between ind1 and ind2 to shortest_arcs.
end for
```

The approach to calculating the shortest arcs is demonstrated in algorithm 5.3, where the input to the algorithm is *districts*, a dataframe containing the information about the stops in each nano-district. The output is a dataframe containing the

shortest arcs between the nano-districts.

As demonstrated in algorithm 5.3, there is a subtle yet crucial detail that prevents the repetition of stops between different district connections, serving as a connection between two different nano-districts, as this situation would lead to a cycle rather than a path.

What must be rigorously avoided is the repetition of connections between nano-districts. This means that, while there may be multiple connections within each district, stop X should only be present in the transition between nano-district $N-1$ and nano-district N and should not reappear in the path connecting nano-district N to nano-district $N+1$. This safeguard ensures an unambiguous structure for path connections between adjacent districts. Logically, this precaution does not apply when a nano-district consists of only one stop to visit.

After identifying the shortest connections between each nano-district, it is possible to proceed with **Inter 2 - Shortest Arcs**. This algorithm is detailed in the pseudocode 5.4.

Algorithm 5.4 Inter2 - Shortest Arcs

```
function INTER2
    define depot
    final_stop ← origin_node from connection 1 of shortest_arcs
    stops_dist ← stops from district 1
    remove final_stop from stops_dist
    stops_dist ← concatenate depot, stops_dist, final_stop
    route_excerpt ← TSP_LKH(stops_dist)
    final_route ← route_excerpt
    for x range from 2 to (num_of_districts − 1) do
        initial_stop ← destiny_node from connection x of shortest_arcs
        final_stop ← origin_node from connection x + 1 of shortest_arcs
        stops_dist ← stops from district x
        remove initial_stop and final_stop from stops_dist
        stops_dist ← concatenate initial_stop, stops_dist, final_stop
        route_excerpt ← TSP_LKH(stops_dist)
        final_route ← concatenate final_route and route_excerpt
    end for
    initial_stop ← destiny_node from connection (num_of_districts − 1) of
shortest_arcs
    stops_dist ← stops from district num_of_districts
    remove initial_stop from stops_dist
    stops_dist ← concatenate initial_stop, stops_dist, depot
    route_excerpt ← TSP_LKH(stops_dist)
    final_route ← concatenate final_route and route_excerpt
    return final_route
end function
```

Inter 2 - Shortest Arcs inputs are a dataframe containing the list of stops, the dataframe created by the **Calc_Shortest_Arcs** algorithm, containing the list of shortest connections between each nano-districts, as well as the number of nano-districts. The output is, as expected, the suggested route. The process is divided into three phases:

In the first phase, the first nano-district is processed. The starting stop corresponds to the depot, the final stop is the *origin_node* of the connection between nano-district 1 and 2, and to be visited are all the stops in nano-district 1, except for the final customer, whose position is previously defined as the last one.

In the second phase, all nano-districts except the first and the last to visit are addressed. If N is the nano-district in question, its starting stop will be the *destiny_node* of the connection between nano-district $N-1$ and N, the final stop will be the *origin_node* of the connection between N and $N+1$. To be visited, are all stops except for the first and last, whose positions are previously defined.

In the third phase, the last nano-district is processed. The starting stop corresponds to the *destiny_node* of the connection between nano district $N-1$ and N, where N is the number of districts, the final stop is the depot, and to be visited are all the stops in the nano-district, except for the starting point, whose position is previously defined as the first one.

5.3 Main Algorithm

After defining the most important necessary functions for the route creation approaches, one must also look to the aggregation algorithm: **Main**. Main is responsible for structuring the previously mentioned algorithms by correctly connecting them, allowing the route generation. For a better understanding of the "main" component, check the flowchart in figure 5.3, where a summary of its operation is provided.

As seen in figure 5.3, a process called Variation X is mentioned. As depicted in Figure 5.3, Variant X is a significant component of the Main algorithm. To identify the most appropriate solution for the problem, 18 variants were created, each combining the various previously mentioned components. Table 5.1 specifies the components used in each variant.

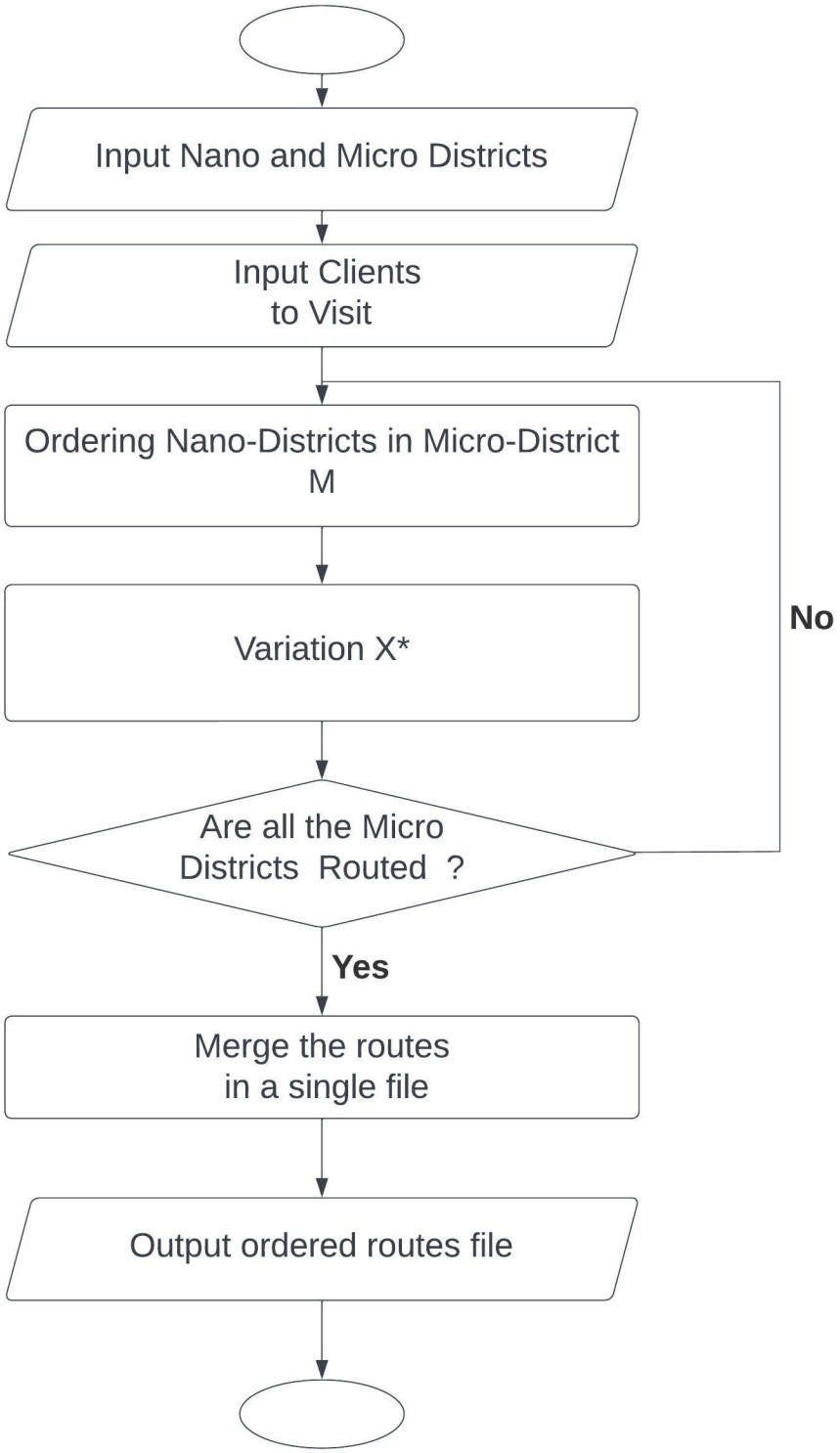


Figure 5.3. Main Function Flowchart

Variant	Inter District Connection Algorithm	Converter	Distance Type
1	Inter 1 - Centroids	1	Euclidean
2	Inter 1 - Centroids	2	Euclidean
3	Inter 1 - Medoids	1	Euclidean
4	Inter 1 - Medoids	2	Euclidean
5	Inter 2 - Shortest Arc	1	Euclidean
6	Inter 2 - Shortest Arc	2	Euclidean
7	Inter 1 - Centroids	1	Asymmetric Road
8	Inter 1 - Centroids	2	Asymmetric Road
9	Inter 1 - Medoids	1	Asymmetric Road
10	Inter 1 - Medoids	2	Asymmetric Road
11	Inter 2 - Shortest Arc	1	Asymmetric Road
12	Inter 2 - Shortest Arc	2	Asymmetric Road
13	Inter 1 - Centroids	1	Symmetric Road
14	Inter 1 - Centroids	2	Symmetric Road
15	Inter 1 - Medoids	1	Symmetric Road
16	Inter 1 - Medoids	2	Symmetric Road
17	Inter 2 - Shortest Arc	1	Symmetric Road
18	Inter 2 - Shortest Arc	2	Symmetric Road

Table 5.1. Variants

Chapter 6

Result Analysis

For the result analysis, the 18 proposed variants were tested, analyzed, and compared. In section 6.1, Variant Analysis Preparation, information about the variant analysis is provided, allowing for a clearer result analysis throughout the remainder of the chapter. After that, in section 6.2, the result analysis is performed, evaluating all the variant's components and their performance, attempting to identify the best-performing group of components, i.e. the best overall performing variant.

6.1 Variant Analysis Preparation

The variant analysis will focus on evaluating the performance of 18 variants to identify those that demonstrate superior performance, while also attempting to identify the characteristics associated with the variant's superior performance.

The provided dataset only contains information on the real routes executed by drivers, without any information regarding routes suggested by the company's model. Consequently, the results obtained may exhibit discrepancies in optimization, as the routes against which the model is being compared are influenced by day-to-day factors such as client unavailability, traffic congestion, and unforeseen road closures, which may negatively impact the obtained routes' performance. Additionally, practical route creation accounts for many other constraints, such as time windows and capacity constraints, factors that are not considered in the developed variants.

Considering this, and to provide a more realistic comparison, a Traveling Salesman Problem was solved for each route, visiting the same stops and using the same LKH algorithm employed by the proposed models, but without considering districts. Asymmetric road distances were utilized, as they are the closest to practical distances. This route development did not consider time windows, capacity constraints, and the day-to-day factors previously mentioned.

As such, the performance of these routes will be much more comparable to the

performance of the proposed routes throughout this work, unlike the original routes, which would demonstrate much worse results than any of the proposed variants. Therefore, from now on, all comparisons will be between the routes that do not use districts and the proposed variants' routes. The model that does not consider districts will, henceforth, be referred to as the No-Districts model.

Considering that the No-Districts model will present close to optimal solutions for the routes, it is expected that, due to the existence of district restrictions, all proposed variants will present similar or lower quality. However, the smaller the difference between the proposed solutions and the No-Districts model, the better the variant's quality.

For a clearer understanding when comparing variables, it is also helpful to define the concept of **homologous variants**: variants that share all components except one. For example, considering variants 1, 7, and 13: these are homologous variants, differing only in the distance used, while all other components remain the same. Variants 1 and 2, for example, are also homologous, differing only in the converter used. Analyzing homologous variants will be essential for comparing variant performance, as it isolates the influence of a specific characteristic and enables drawing meaningful conclusions about it.

The most important comparison metric will be traveled distance, considering that the Travelling Salesman Problem will be solved to minimize the total traveled distances. Nevertheless, computational times are also an important factor, and analyzing them can provide valuable insight regarding the performance of the variants.

The variant analysis will attempt to answer the following questions:

- How would the proposed variants perform in comparison to the No-Districts model?
- Are there significant differences in using Euclidean, asymmetric road, and symmetric road distances? And if yes, what are those differences?
- What is the best-performing distance type?
- What is the best TSP-to-SHPP converter?
- What is the best proposed model for Inter-District connections?
- What is the best overall performing variant?

The results presented in the present and further chapters were obtained using a computational system equipped with an Intel(R) Xeon(R) CPU X5690 running at 3,46 GHz, and 48,00 GB of RAM. The developed code was executed using Python,

Table 6.1. Variant Performance - Computational Results

Variant	Total Travelled Distance	Total Travelled Distance difference	Non-OSM Time	OSM Time	Total Comput. Time
	meters	%	seconds	seconds	seconds
No-Dist	6.875.796,97	-	71,87	69.367,643	69.439,52
1	7.145.525,52	-3,92%	31,43	1.846,42	1.877,85
2	7.151.723,92	-4,01%	31,89	1.845,71	1.877,60
3	7.144.691,34	-3,91%	31,26	1.831,08	1.862,34
4	7.151.455,12	-4,01%	31,98	1.835,53	1.867,51
5	7.141.417,04	-3,86%	32,93	1.680,75	1.713,68
6	7.147.159,83	-3,95%	33,33	1.674,67	1.708,00
7	6.985.234,66	-1,59%	29,57	37.737,44	37.767,01
8	6.985.287,05	-1,59%	30,35	37,220,60	37.250,95
9	6.972.409,07	-1,40%	29,65	37,728,64	37.758,29
10	6.972.193,68	-1,40%	30,07	37.918,76	37.948,83
11	6.983.816,37	-1,57%	31,70	35.189,87	35.221,57
12	6.983.816,37	-1,57%	32,23	35.456,57	35.488,80
13	7.125.392,81	-3,63%	30,67	19.773,50	19.804,17
14	7.125.785,43	-3,64%	31,16	19.558,28	19.589,44
15	7.115.603,96	-3,49%	29,80	19.963,72	19.993,52
16	7.121.907,98	-3,58%	29,82	19.794,58	19.824,40
17	7.127.498,89	-3,66%	31,90	17.524,70	17.556,60
18	7.128.383,37	-3,67%	31,92	17.650,79	17.682,71
Original	7.626.710,74	-	-	-	-

specifically Python version 3.11.5, and the programming and analysis tasks were conducted in the Spyder environment.

6.2 Variant Analysis

Micro-district-based routes are an important object for comparing models. As all the routes' stops are located inside one micro-district, only one driver will visit the same stops in the current, the No-District, and the proposed models, providing a direct route comparison.

Table 6.1 presents the performance of each variant in comparison to the No-District model. Total Traveled Distance refers, as the name indicates, to the variant's sum of all the traveled distances in the 179 routes, in meters. Total Traveled Distance difference refers to the percentage variation between the proposed variant and the No-District model distances, and it enables a simplified view of the observed changes. Negative numbers indicate worse results than the No-District model and vice versa.

The table also presents the computational performance of each variant in three different parameters: OSM Time refers to the computational time spent converting the stop's coordinates to OSM nodes (in all 18 variants) and calculating the distance

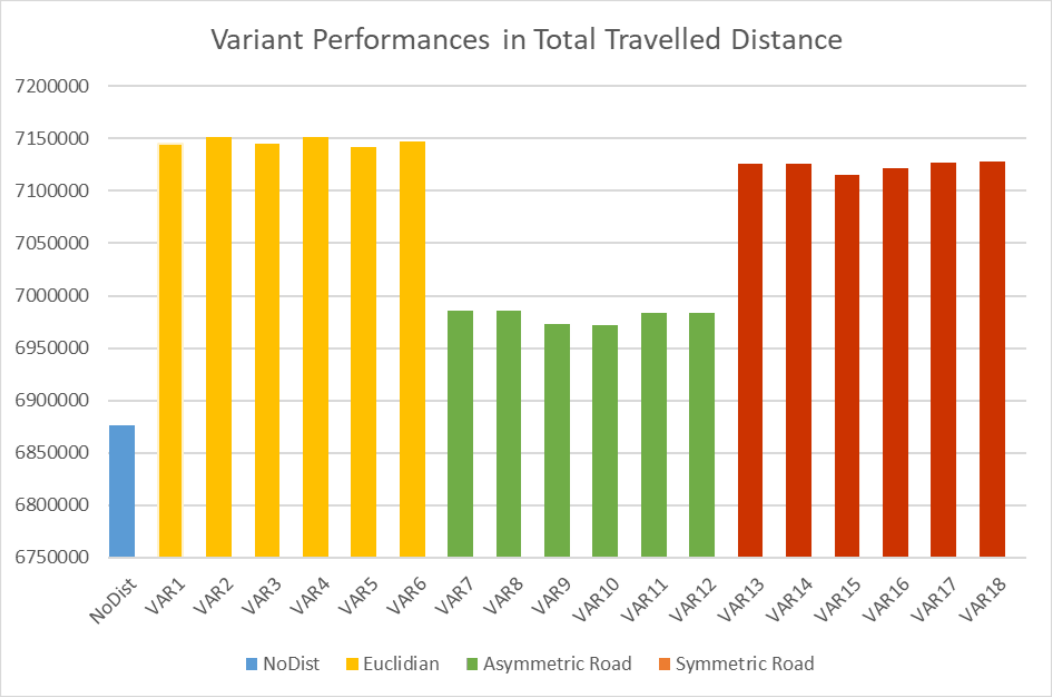


Figure 6.1. Total Travelled Distance per Variant

matrices (in all the 12 road distance variants); Non-OSM Time is the remaining computational time needed to run the non-OSM-related components; Total computational time is the sum of non-OSM time and OSM time, i.e. the total necessary time to generate the 179 routes.

Figure 6.1 demonstrates the Total Traveled Distance values in each of the 18 variants. Each variant is represented in one of three colors, referring to the distance type used in the route creation, to emphasize any performance difference between the type of distance used. The No-Districts model is also presented in the graph.

6.2.1 Result overview

For the result analysis, the first component to evaluate is: **"How would the proposed variants perform in comparison to the No-District model?"**

Column "Total Travelled Distance" in Table 6.1 demonstrates that all the proposed variants, as well as the No-Districts model, are superior in comparison to the original routes. This shows that the results are indeed influenced by day-to-day factors such as client unavailability, traffic congestion, unforeseen road closure, time windows, and capacity constraints, factors that are not considered in the developed variants but greatly influence the routes' performances.

Figure 6.1 shows that the performance of the 18 developed variants is, as expected, inferior to the No-Districts model routes. This occurs, as previously mentioned,

due to the constraints that require all stops in a given nano-district to be visited before moving on to the next nano-district. This requirement undoubtedly causes a decrease in the overall performance of the routes. A clear disparity between the results obtained with different distance types can also be observed. This shows that the distances utilized are, in fact, a defining factor in route creation.

All routes result in a total traveled distance equal to, or worse than, the distance achieved by the optimal model. In the 6 variants that use Euclidean distances, 44,7% of the routes produced inferior results, while the remaining 55,3% produced results equal to the no-district model's routes, with the greatest difference occurring in a route produced by variant 5. For the variants using asymmetric road distances, 41,9% of the routes performed worse than the No-District model, while the remaining 58,1% yielded similar results, with the greatest difference occurring in a route produced by variant 11. Finally, the variants using symmetric road distances showed similar results to those using Euclidean distances, with 44,1% of routes performing worse, and the remaining 55,9% achieving similar results.

Figures 6.2 illustrate the difference between No-District routes and routes obtained with variants developed throughout the current report. It is important to emphasize that the connections between stops are depicted as straight lines. This might lead to a misleading analysis of the graphs, as visually, routes using Euclidean distances may appear superior to the others, although practically this is not true.

The analysis of figures 6.2a, 6.2b, 6.2c, and 6.2d provides a visualization of the routes produced by variants 1, 7, and 13, as well as the No-Districts route, for the route 20211201_CIRC3, respectively. The orange polygons in the figures represent the nano-districts.

In figure 6.2d, a significant difference between the No-Districts route and the others is evident: the route does not pass through all the stops within a nano-district before visiting the next one. Instead, it follows the route that minimizes the total traveled distance.

The big disparity between figure 6.2a and figures 6.2b and 6.2c, demonstrates the impact of using Euclidean distances or road distances. Small changes are also noticeable between variants 7 and 13 in some portions of the route, although mostly, similar behavior can be noticed.

6.2.2 Distance Type differences

"Are there significant differences in routes produced using Euclidean, asymmetric road, and symmetric road distances? And if yes, what are those differences?"

In figures 6.3a and 6.3b, it is possible to see histograms that illustrate, on the

routes, the difference between the total traveled distance measured with the distance used to create the route and the total traveled distance with the closest to practical distance - the asymmetric road distances. This will allow a better understanding of the influence of the type of distance used to create the route and its actual performance. The frequency at which the difference between the two distances is 0 is presented in yellow.

Figure 6.3a refers to variant 2, which used Euclidean distances and is similar to the histograms of the remaining 5 Euclidean distances using variants. The same applies to figure 6.3b, but referring to variant 14 and, consequently, symmetric road distance using variants.

It is possible to verify that Euclidean distances cause a greater increase compared to the increase observed using symmetric road distances. The differences resemble a normal distribution, with the mode between 20% and 22%. This pattern is consistent across all five Euclidean variants, suggesting that if the use of Euclidean distances is chosen or necessary, a 20% increase should be added to the expected final route distance, to predict the real total traveled distance more accurately.

As for variant 14 and all the other symmetric road distance variants, the mode is 0, with an asymmetric distribution and a longer tail to the right of the peak. This occurs because there are considerable similarities between asymmetric and symmetric distances in most of the routes in question, but in cases where this does not hold, the distance is greater than expected. This implies that in a large number of routes, the distances will be similar. However, when this is not the case, a considerable decrease in route quality in terms of total traveled distance is to be expected.

6.2.3 Best-performing Distance Type

The next component to attempt to analyze is: **"What is the best-performing distance type?"**

Routes generated using Euclidean distances (variants 1 to 6) yield inferior results compared to the No-Districts model. The traveled distances worsen between 3,86% in variant 5 and 4,01% in variant 2, in comparison to the No-Districts model. As expected, in terms of traveled distances, these are the worst results obtained by any of the three distance types.

However, the main advantage of these six variants lies in their computational time, where 1.708,00 to 1.877,85 seconds are sufficient to solve all 179 routes, corresponding to, on average, 4,97% of the necessary time to create routes with asymmetric distances, 9,37% of the required time to create routes with symmetric distances, and markedly shorter than the No-Districts model's computational times, needing on average only 2,62 % of the required time.

In contrast, routes constructed using asymmetric road distances (variants 7 to 12) achieve the best results compared to other variants, although still worse than the No-Districts model, as evident in Table 6.1. The traveled distances are worse by 1,41%, and 1,59%, in variants 10 and 8, respectively.

Variant 10 obtained the best result out of the 18 proposed variants regarding the total distance traveled. However, variant 9 obtained very close results with a difference of just 215 meters between the two variants, corresponding to a difference of just 0,003%. Variant 10 shows equal or better results in 95%, 95%, 79%, and 79% of the routes compared to variants 7, 8, 11, and 12, respectively. Variants 9 and 10 create similar routes in 98,32% of the routes.

The major disadvantage of this type of distance is the substantial computational time required to generate the routes, due to the use of OSM. Solving 179 routes takes from 9 hours and 47 minutes (35,221.57 seconds) in variant 12 to 10 hours and 31 minutes (37,948.83 seconds) in variant 11. Although the computational times for the proposed variants are undeniably longer, they still produce results in, on average, 53.14% of the time necessary in the No-Districts model. This is one of the biggest advantages of using districting approaches for routing.

Finally, routes generated by symmetric road distances (variants 13 to 18) achieved results relatively similar to those obtained with Euclidean distances in terms of total distance values traveled. The traveled distances increase from 3,49% in variant 15 to 3,67% in variant 18. Furthermore, the associated computational time is significantly lower compared to variants 7-12 (on average 53,39% of the required time), requiring between 4 hours and 53 minutes (17.556,60 seconds) in variant 17 and 5 hours and 33 minutes (19.933,52 seconds) in variant 14. This represents a decrease compared to the No-Districts model, as the required time is only 27,41%.

The results obtained by the symmetric road distances are closer to the results obtained by Euclidean distances than to the results obtained by the Asymmetric Road distances. This means that in a case where using asymmetric road distances proves impossible due, for example, to a lack of available time to generate routes, using Euclidean distances may prove more suitable as the results are comparable (only 0,33% worse) to the ones obtained with symmetric road distances, but are obtained in significantly less time (on average, in 9,52% of the time).

6.2.4 Best TSP-to-SHPP converter

The next component to be analysed is: **"What is the best performing TSP-to-SHPP converter?"**

Regarding the TSP-to-SHPP converters, by analyzing Table 6.1, it is possible to notice that, although marginal, variants using Converter 1 obtain slightly better

Table 6.2. Variation between converter performance - Evaluation with Own-Distance

Variant	Total Travelled Distance (m)	Improvement (%)	Variation (%)
Var1	5.320.706,64	-0,38	-
Var2	5.320.739,23	-0,38	0,00
Var3	5.320.537,38	-0,37	-
Var4	5.320.569,96	-0,37	0,00
Var5	5.323.327,32	-0,43	-
Var6	5.323.327,57	-0,43	0,00
Var7	6.985.234,66	-2,06	-
Var8	6.985.287,05	-2,07	0,00
Var9	6.972.409,07	-1,82	-
Var10	6.972.193,68	-1,82	0,00
Var11	6.983.816,37	-2,04	-
Var12	6.983.816,37	-2,04	0,00
Var13	6.904.376,52	-1,31	-
Var14	6.904.359,73	-1,31	0,00
Var15	6.897.899,20	-1,19	-
Var16	6.898.699,95	-1,20	0,01
Var17	6.906.132,38	-1,34	-
Var18	6.906.132,06	-1,34	0,00

results than variants using Converter 2 in 8 out of all 9 cases, with Variant 10 being the only exception, having a 0,003% better result than Variant 9.

However, when exclusively analyzing the differences between variants 7-8, 9-10, and 11-12, it is observed that the performance differences of these homologous variants are similar, with the largest difference between the variants being only 0.003%. This could indicate that converters 1 and 2 perform identically in terms of total traveled distances.

To achieve this, the performances were evaluated in terms of total traveled distance using the same distance matrices employed in the creation of the route. The aim was to determine whether, in the 9 homologous variants, the differences between results obtained with converter 1 are equal to, or different, from converter 2. These results are shown in Table 6.2, where the difference between homologous variants with different TSP-to-SHPP converters is illustrated.

As shown in Table 6.2, all variants exhibit similar results, with no significant differences between any of them. When comparing all routes to their counterparts in the homologous variant, 99,3% of the routes have similar distances. Uncontrollable factors, such as variations caused by the OSMnx cause the differences in the remaining routes.

The differences observed between variants 1-2, 3-4, 5-6, 13-14, 15-16, and 17-18 when evaluated using the own-distance matrix versus real road distance matrix indicate that the disparities between Converter 1 and 2 are influenced by variations

between actual distances and those used in calculations. Multiple combinations of stops may achieve similar total traveled distances with the own-distance matrix but yield different values when evaluated with real road distances. This inevitably leads to variations in the proposed routes across different variants, thereby causing the observed discrepancies, which should thus be disregarded. It is concluded that Converters 1 and 2 have similar performances in terms of total traveled distances.

Nevertheless, in terms of computational times, the performances show distinct results. Considering that converters only influence the Non-OSM component of the computational times, further attention should be given to the Non-OSM Time component in Table 6.1.

When analyzing the pairs of homologous variants, it is evident that, despite a small difference, Converter 1 consistently performs better than Converter 2 in terms of computational times. This difference is due to the lower complexity associated with Converter 1, which only requires altering a value in the distance matrix, a process much simpler than Converter 2's, which requires adding a new column and row to the distance matrix, as well as solving a TSP with an additional value, making its process more complex.

Therefore, considering that Converters 1 and 2 achieve similar performances in terms of total traveled distances and that Converter 1 has superior performance in terms of computational times, Converter 1 should be considered the best TSP-to-SHPP converter.

6.2.5 Best Inter-District connector

The last of the three components to analyze are the Inter-District connections. **"What is the best-proposed model for Inter-District connections?"**

To allow a clearer understanding, table 6.3 may be observed. This table shows the same results seen in table 6.1, however, this table is grouped by homologous variants with different Inter-District connectors.

To facilitate the analysis, information regarding the variant's components is also presented in the table: Column "Dist. Type" refers to the distance type, with "EUC" referring to Euclidean distances, "ASY" referring to asymmetric road distances, and "SYM" representing symmetric road distances; Column "Converter" refers to the TSP-to-SHPP converter, with 1 and 2 representing Converters 1 and 2 respectively; Column "Inter-District Connector" refers to the used Inter-District connector, with "I1C" referring to Inter 1 - Centroids, "I1M" to Inter 1 - Medoids, and "I2" to Inter 2 - Shortest Arcs. The "Variation" column compares homologous variants, representing the difference between each variant and the one listed above it in the table.

Upon examining Table 6.3, it can be observed that each type of distance produces

Table 6.3. Inter-District Connector Performance Variation

Variant	Total Travelled Distance	Improv. (%)	Variation (%)	Dist. Type	Conv.	Inter-District Connector
NoDist	6.875.796,97	-	-	-	-	-
VAR5	7.141.417,04	-3,86%	-	EUC	1	I2
VAR3	7.144.691,34	-3,91%	0,05%	EUC	1	I1M
VAR1	7.145.525,52	-3,92%	0,01%	EUC	1	I1C
VAR6	7.147.159,83	-3,95%	-	EUC	2	I2
VAR4	7.151.455,12	-4,01%	0,06%	EUC	2	I1M
VAR2	7.151.723,92	-4,01%	0,00%	EUC	2	I1C
VAR9	6.972.409,07	-1,40%	-	ASY	1	I1M
VAR11	6.983.816,37	-1,57%	0,16%	ASY	1	I2
VAR7	6.985.234,66	-1,59%	0,02%	ASY	1	I1C
VAR10	6.972.193,68	-1,40%	-	ASY	2	I1M
VAR12	6.983.816,37	-1,57%	0,17%	ASY	2	I2
VAR8	6.985.287,05	-1,59%	0,02%	ASY	2	I1C
VAR15	7.115.603,96	-3,49%	-	SYM	1	I1M
VAR13	7.125.392,81	-3,63%	0,14%	SYM	1	I1C
VAR17	7.127.498,89	-3,66%	0,03%	SYM	1	I2
VAR16	7.121.907,98	-3,58%	-	SYM	2	I1M
VAR14	7.125.785,43	-3,64%	0,06%	SYM	2	I1C
VAR18	7.128.383,37	-3,67%	0,03%	SYM	2	I2

different results.

When using Euclidean distances, the best results are obtained using Inter 2 - Shortest Arcs, the second-best results are obtained using Inter 1 - Medoids, and the worst are obtained using Inter 1 - Centroids. Inter 2 - Shortest Arcs is the highest-performing Inter-District connector, achieving results at least 0,05% better than the other 2 options. Inter 1 - Centroids and Inter 1 - Medoids have similar performances, as 98,32% of the routes produced are equal, and total distances differ by a maximum of 0,01%.

Inter 1 - Medoids demonstrated the best performance regarding variants using asymmetric road distances. Variants 9 and 10 achieved slightly superior results by 0,16%, and 0,17%, respectively, compared to their respective homologous variants with different Inter-District connections. Inter 1 - Medoids using variants have superior performances in 11,45% more routes than variants that use Inter 1 - Centroids, and 16,20% more routes than Inter 2 - Shortest Arcs using variants. Variants using Inter 2 - Shortest Arcs achieve the second-best results, which are 0,02% better than the worst results obtained by Inter 1 - Centroids.

Finally, in symmetric road distance using variants, Inter 1 - Medoids using variants show the best results, with 3,49% in variant 15 and 3,58% in variant 16. Variants that use Inter 2 - Shortest Arc present the second-best result with a quality

decrease of only 1,57%, while the worst result among variants that use Inter 1 - Centroids show a quality decrease of 1,59%.

Regarding computational times, variants utilizing the Inter 2 - Shortest Arc method exhibit notably reduced OSM Time, consequently resulting in significantly lower computational times. This means that the calculation of the shortest arcs between districts is computationally superior to the calculation of the distances between centroids and all the nodes in each district. Inter 1 - Centroids and Inter 1 - Medoids have comparable computational times.

In summary, it is important to notice that, although some variations are observed, Inter 1 - Centroids is always inferior to Inter 1 - Medoids in all three distance types. It also is possible to conclude that there is no absolute superior Inter-District Connector; instead, each distance has a more adequate suitor: Euclidean distance using variants should use the Inter 2 - Shortest Arc connector; In both Road Distances, Inter 1 - Medoids is the most suitable option to reduce traveled distances.

Even with an inferior performance regarding traveled distances, Inter 2 - Shortest Arc is the best variant regarding computational times, with results being obtained on average between 5 to 6% faster than other variants.

6.2.6 Best overall performing variant

Taking into account all the conclusions drawn throughout the current sections, it would be expected that the best solution would be obtained with a variant that uses:

- Distance Type: Asymmetric Road Distances;
- Inter District Connector: Inter 1 - Medoids;
- TSP-to-SHPP converter: Converter 1

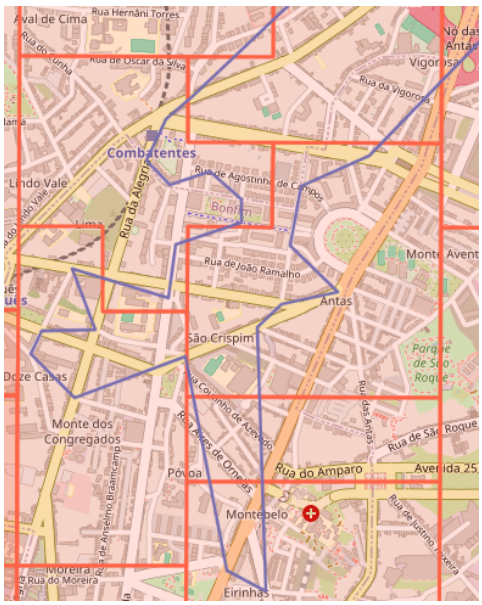
The best-performing variant in terms of total traveled distances was, as previously mentioned, variant 10, which uses Asymmetric Road Distances, and Inter 1 - Medoids and is therefore in line with the conclusions drawn throughout this chapter. However, variant 9 has similar results to variant 10, with only a 0,003% difference between the variants' traveled distances, but uses converter 1, which was considered more computationally efficient.

Finally, after all the analysis drawn throughout the current section, the best overall performing variant is Variant 9, as it contains the three overall best-performing components: Asymmetric Road Distances for route creation, the distance type that produces the best results in terms of total traveled distances; the Inter-District connector Inter 1 - Medoids, the best performing Inter-District connection approach

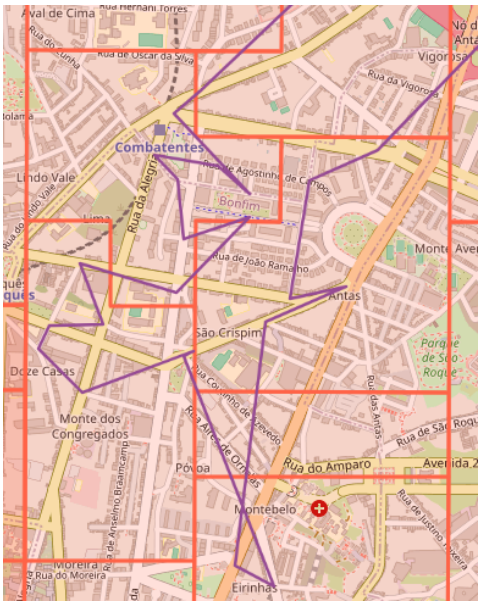
considering that the considered distance would be the asymmetric road distance; Converter 1 that although produces similar results in comparison to Converter 2, is more computationally efficient. Further analysis of the current model's adaptability to a daily model should consider this variant as the main option.

Variant 11 is also worth considering for future analysis, as it achieves results 6% faster than Variant 9, with only a 0.17% increase in total traveled distances.

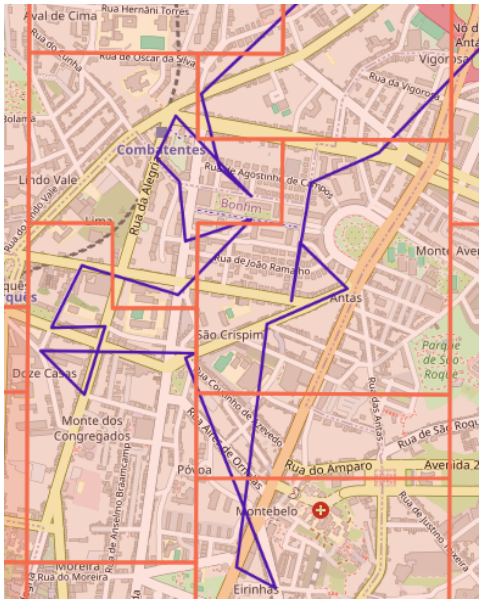
However, if the use of the OSM and the calculation of distance matrices is not possible due to, for example, lack of available time, the best-suited variant would be an Euclidean distance-using variant, because these achieve comparable results in considerably less time in comparison to symmetric road distance-using variants. In this case, Variant 5 would be the best-suiting variant because Inter 2 - Shortest Arc and Converter 1 are the better options for Euclidean distance using variants.



(a) Variant 1



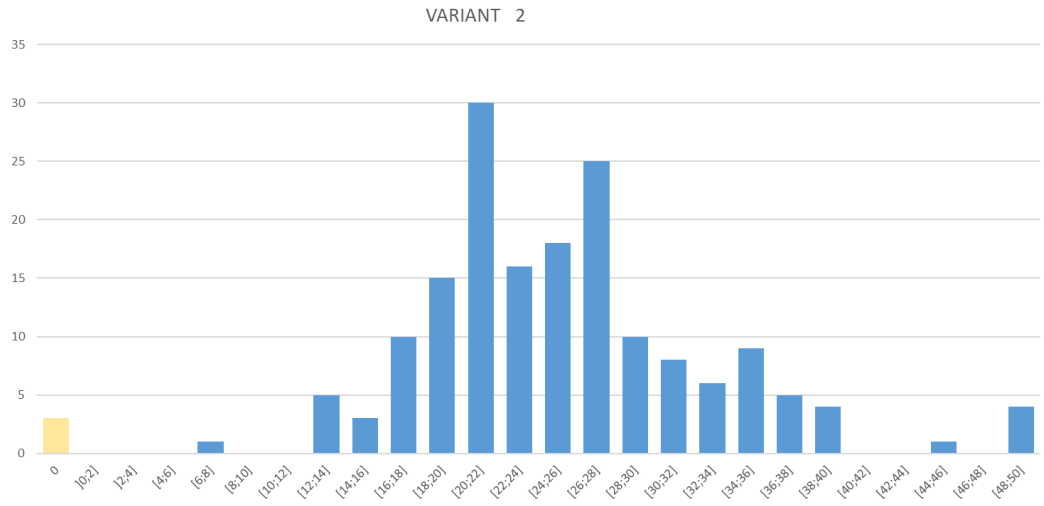
(b) Variant 7



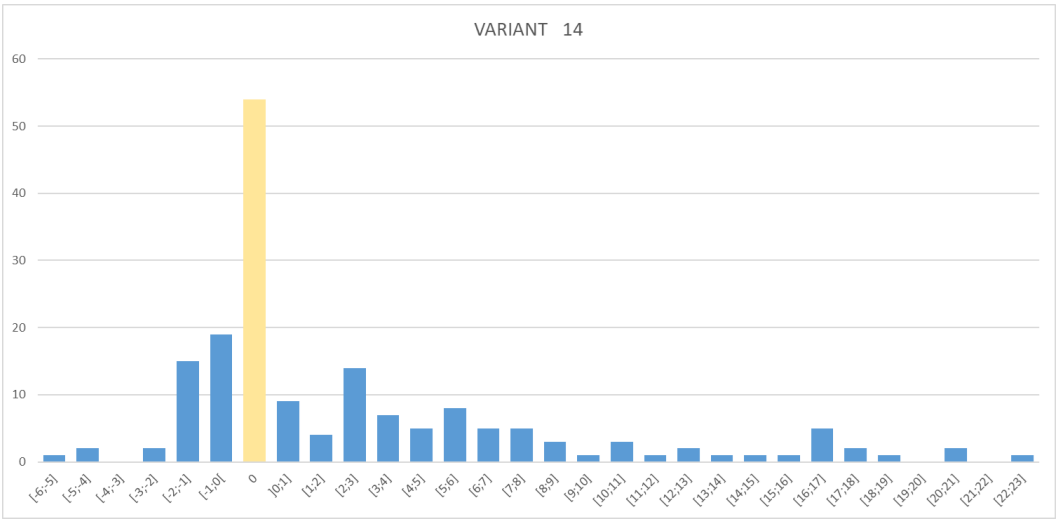
(c) Variant 13

(d) No-Districts

Figure 6.2. Route 20211201_CIRC3 obtained with different distances



(a) Histogram Variant 2



(b) Histogram Variant 14

Figure 6.3. Histograms of the Total Traveled Distances difference with Own - Distances and with Real Road Distances

Chapter 7

Conclusion

This chapter focuses on establishing the conclusions drawn from this work. In Section 7.1, conclusions are established based on the findings and analysis presented throughout the dissertation. Section 7.2 addresses limitations identified in the current work and carefully maps out several proposals for future research.

7.1 Final Conclusions

This dissertation aims to contribute to the existing vehicle routing literature, by identifying, developing, analyzing, and evaluating the best-performing Shortest Hamiltonian Path-based intra and inter-district routing approaches. The dissertation uses the case study of a parcel delivery company operating in the Porto Metropolitan Area over a period of 6 months.

The literature review highlighted that two-phased approaches are frequently used for route creation in districting approaches. In the first phase, the order of districts to be visited by each vehicle and their connection method is established (inter-district phase), and in the second phase, the route in each district is developed (Intra-District phase).

18 variants for route creation were developed within a districting strategy to improve last-mile delivery. These 18 variants differed slightly in three components identified in the literature review. Euclidean, symmetric road, and asymmetric road were the distance types used. The variables also varied concerning the type of TSP-to-SHPP converter, with two distinct options tested. Three methods were tested for inter-district connection. The Intra-District route creation algorithm was a Lin-Khernighan-based heuristic.

The 18 developed variants were subsequently tested, comparing them to optimal routes that do not utilize districts for route allocation. As anticipated, the results indicated that asymmetric road distances yielded the best outcomes. However,

using these distances required greater computational effort due to the utilization of OSMnx for calculating the required distance matrices. Results also demonstrated that although computationally superior to asymmetric road distances, symmetric road distances obtain results comparable to the Euclidean distance using variants, and in case of time restrictions impeding the use of the better-performing road distances, Euclidean distances should be used, as computational times are far superior in comparison to the other distance types.

Concerning the converters analyzed, although the performances were similar in terms of total traveled distances, it was possible to identify that Converter 1 was superior in comparison to the other option in terms of computational times.

Finally, concerning the inter-district connection approaches, it can be concluded that Inter 1 - Medoids has the best performance in terms of total traveled distances in road distance using variants, and Inter 2 - Shortest Arcs has the best performance in terms of total traveled distances in Euclidean distance using variants. Inter 2 - Shortest Arcs is also the best-performing variant in terms of computational time.

Given these conclusions, it would be expected that the best overall performance would come from a variant using Asymmetric Road Distances, Inter 1 - Medoids, and Converter 1. The variant containing all of the mentioned components is variant 9, making it this work's best overall-performing variant.

Variant 11 should also be mentioned, however, as in comparison to variant 9, it obtains results only 0,19% inferior in terms of total traveled distances, but with the computational times being reduced by 6%.

The results also demonstrated that, although the use of districts produces sub-optimal results in terms of traveled distances, computationally, due to the scalability of the data, the results obtained using districts are achieved with significantly less computational effort, which can present substantial advantages.

7.2 Limitations and Future work

The main limitation of this work relates to the use of road distances. While road distances are more accurate, they are significantly more demanding computationally. Given the existence of 18 variants, obtaining results was extremely time-consuming, causing constant delays throughout the process. Moreover, frequent code modifications required the lengthy result acquisition process to be repeated multiple times. Additionally, the limited number of routes contained in only 1 Micro-District (179 across the entire dataset) makes the results less reliable. A larger number of such routes would potentially yield more dependable results, as they would be less susceptible to external factors and biases. Finally, another limitation of this work

was the absence of consideration for capacity constraints and time windows. These factors would have greatly enhanced the quality of the solutions presented, aligning them more closely with real-world route creation behavior and thereby increasing the relevance of the findings obtained.

Regarding future works, if a proposed variant were applied to a daily model, the results obtained with the proposed variants would show a significant increase in the number of drivers. This would happen because of the current model's assumption that each driver should be associated with one micro-district.

However, this causes inefficiencies, as a particular driver may be assigned to a micro-district even if they have only one stop on a single day, resulting in unnecessarily low service time. Another inefficiency would be the requirement for two trips from and to the depot associated with each micro-district (each route starts and finishes in the depot).

To address these inefficiencies, a precursor model to the proposed model should be developed. This addition would be responsible for aggregating multiple micro-districts for a given day, allowing these to be visited by one single driver. So, in an ideal situation, the following topics should be taken into account in the aggregation process:

1. **Proximity between micro-districts**: Micro-districts should be grouped to minimize the total distance traveled between them.

2. **Number of drivers**: The model should aim to aggregate as many micro-districts as possible to minimize the number of drivers needed to meet the demand, considering constraints such as:

    - Vehicle capacity regarding the total weight possible to transport in one trip
    - Vehicle capacity regarding the number of packages possible to transport in one trip
    - Maximum and minimum driver service time per day

3. **Assignment of micro-districts based on knowledge**: Considering the learning curves mentioned by Zhong et al. (2007), the model should aim to assign districts to experienced drivers to ensure their performance improves progressively.

Although not mandatory for its main objective of micro-district aggregation, the model could also benefit from taking into account:

- Client Time Windows;

- The possibility for drivers to visit the depot more than once in a given day to align their service times more closely with a functional work schedule;
- The ability to assign certain nano-districts within a single micro-district to different drivers;
- The consideration of the problem as a Soft Clustered VRP, meaning a driver can depart from a micro or nano-district but later return to it to potentially reduce total distances.

If this model is developed and applied before route creation, a decrease in total distance traveled would be expected in comparison to the same variant without the precursor model, as the number of trips between depots and micro-districts should be reduced. A significant reduction in the number of drivers would also be expected.

Variant 9 is recommended for route creation following the development of this model; however, analyzing the performance of other variants such as variant 11, or variants that use different distance types, such as variants 5 or 15 may also yield interesting insights.

For further analysis, after the inclusion of the said predecessor model, the best-performing variant or variants should be submitted to a daily analysis, not comparing the variants with only 1-Micro District routes as done throughout this work, but analyzing the model's performance in creating all the routes necessary to supply demand in a day-to-day basis. This analysis should not only take into consideration factors such as traveled distances, but also the number of necessary drivers, total service times, and many other relevant factors.

Following its development, and if constraints such as vehicle weight capacities, number of packages per vehicle, and time windows are considered, a direct comparison with the current model will provide more conclusive results regarding the effectiveness of the proposed variants throughout this work.

Appendix A

Declaration of Integrity

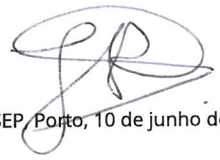
DECLARAÇÃO DE INTEGRIDADE

DECLARAÇÃO DE INTEGRIDADE

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.



ISEP, Porto, 10 de junho de 2024

Bibliography

Agus, A. (2013). The importance of supply chain management on financial optimization. *Jurnal Teknik Industri*, 15(2):77–84. Copyright - © 2013. This work is published under http://creativecommons.org/licenses/by/3.0/ (the "License"). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Última atualização em - 2020-04-10.

Alshamlan, H. and Menai, M. (2012). Solving shortest hamiltonion path problem using dna computing. In *Proceedings of the International Conference on Bioinformatics and Computational Biology*, pages 76–82, Venice, Italy. IARIA. Publication date: June 24, 2012.

Altinel, I., Aras, N., and Oommen, B. (2000). Fast, efficient and accurate solutions to the hamiltonian path problem using neural approaches. *Computers & Operations Research*, 27(5):461–494.

Barreto, S., Ferreira, C., Paixão, J., and Santos, B. S. (2007). Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research*, 179(3):968–977.

Barthélémy, T., Rossi, A., Sevaux, M., and Sörensen, K. (2010). Metaheuristic approach for the clustered VRP. In *EU/ME 2010 - 10th anniversary of the metaheuristic community*, Lorient, France.

Battarra, M., Erdoğan, G., and Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Operations Research*, 62(1):58–71.

Bellmore, M. and Nemhauser, G. L. (1968). The traveling salesman problem: a survey. *Operations Research*, 16(3):538–558.

Bender, M., Kalcsics, J., and Meyer, A. (2020). Districting for parcel delivery services – a two-stage solution approach and a real-world case study. *Omega*, 96:102283.

Boeing, G. (2017). Osmnx: New methods for acquiring, constructing, analyzing, and

visualizing complex street networks. *Computers Environment and Urban Systems*, 65:126–139.

Bouazzi, K., Hammami, M., and Bouamama, S. (2021). Application of an improved genetic algorithm to hamiltonian circuit problem. *Procedia Computer Science*, 192:4337–4347. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021.

Bowerman, R. L., Calamai, P. H., and Brent Hall, G. (1994). The spacefilling curve with optimal partitioning heuristic for the vehicle routing problem. *European Journal of Operational Research*, 76(1):128–142.

Boysen, N., Fedtke, S., and Schwerdfeger, S. (2021). Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum*, 43:1–58.

Brajevic, I. (2011). Artificial bee colony algorithm for the capacitated vehicle routing problem. In *Proceedings of the European computing conference (ECC'11)*, pages 239–244.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers & Operations Research*, 2(2):115–119.

Christofides, N. (1976). The vehicle routing problem. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V1):55–70.

Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

Croci, D. (2020). Districting for routing with consistency. Laurea magistrale, Politecnico di Milano.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Defryn, C. and Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers & Operations Research*, 83:78–94.

Dimitrovski, F. (2023). *elkai: Solving Traveling Salesman Problems using LKH*. LKH and elkai Community. Released on May 9, 2023. Retrieved from the URL on June 13, 2023.

Ding, C., Cheng, Y., and He, M. (2007). Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale tsps. *Tsinghua Science & Technology*, 12(4):459–465.

Expósito-Izquierdo, C., Rossi, A., and Sevaux, M. (2016). A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Computers & Industrial Engineering*, 91:274–289.

Fleischmann, B. and Paraschis, J. N. (1988). Solving a large scale districting problem: a case report. *Computers & Operations Research*, 15(6):521–533.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75.

Goyal, S. (2010). A survey on travelling salesman problem. In *Midwest instruction and computing symposium*, pages 1–9.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA.

Hartl, R. F., Hasle, G., and Janssens, G. K. (2006). Special issue on rich vehicle routing problems. *Central European Journal of Operations Research*, 14(2):103.

Haugland, D., Ho, S. C., and Laporte, G. (2007). Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 180(3):997–1010.

Helsgaun, K. (2018). *Using POPMUSIC for Candidate Set Generation in the Lin-Kernighan-Helsgaun TSP Solver*. Roskilde Universitet.

Helsgaun, K. (2022). *LKH - The Lin-Kernighan Heuristic*. Roskilde University. Released in November 2022. Available at http://akira.ruc.dk/~keld/research/LKH/. The code is distributed for academic and non-commercial use. The author reserves all rights to the code.

Hintsch, T. and Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European Journal of Operational Research*, 270(1):118–131.

Höner zu Siederdissen, C., Prohaska, S. J., and Stadler, P. F. (2014). Dynamic programming for set data types. In Campos, S., editor, *Advances in Bioinformatics and Computational Biology*, pages 57–64, Cham. Springer International Publishing.

Islam, M. A., Gajpal, Y., and ElMekkawy, T. Y. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. *Applied Soft Computing*, 110:107655.

Izquierdo, C., Rossi, A., and Sevaux, M. (2013). Modeling and solving the clustered capacitated vehicle routing problem. In *Proceedings of the 14th EU/ME Workshop*, pages 110–115.

Junior Mele, U., Maria Gambardella, L., and Montemanni, R. (2021). Machine learning approaches for the traveling salesman problem: A survey. In *2021 The 8th International Conference on Industrial Engineering and Applications(Europe)*, ICIEA 2021-Europe, page 182–186, New York, NY, USA. Association for Computing Machinery.

Jünger, M., Reinelt, G., and Rinaldi, G. (1995). Chapter 4 the traveling salesman problem. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 225–330. Elsevier.

Kalcsics, J., Nickel, S., and Schröder, M. (2005). Towards a unified territorial design approach—applications, algorithms and gis integration. *Top*, 13:1–56.

Kawashima, A. and Sugai, Y. (2014). A theoretical framework to solve the tsps as classification problems and shortest hamiltonian path problems. *American Journal of Intelligent Systems*, 4:1–8.

Le, C., Nguyen, D. D., Oláh, J., and Miklós, P. (2022). Clustering algorithm for a vehicle routing problem with time windows. *Transport*, 37:17–27.

Lei, H., Laporte, G., and Guo, B. (2012). Districting for routing with stochastic customers. *EURO Journal on Transportation and Logistics*, 1(1):67–85.

Lei, H., Wang, R., and Laporte, G. (2016). Solving a multi-objective dynamic stochastic districting and routing problem with a co-evolutionary algorithm. *Computers & Operations Research*, 67:12–24.

Li, F., Golden, B., and Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179.

Lu, Y., Hao, J.-K., and Wu, Q. (2022). Solving the clustered traveling salesman problem via tsp methods.

Mehrotra, A., Johnson, E. L., and Nemhauser, G. L. (1998). An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1114.

Miranda, P. A., Gonzalez-Ramirez, R. G., and Smith, N. R. (2011). Districting and customer clustering within supply chain planning: A review of modeling and solution approaches. In Renko, S., editor, *Supply Chain Management - New Perspectives*. IntechOpen. https://ideas.repec.org/s/ito/pchaps.html.

Munari, P., Dollevoet, T., and Spliet, R. (2017). A generalized formulation for vehicle routing problems.

Muyldermans, L., Cattrysse, D., and Oudheusden, D. V. (2003). District design for arc-routing applications. *Journal of the Operational Research Society*, 54(11):1209–1221.

Nanda Kumar, S. and Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 04.

Omran, M. G., Engelbrecht, A. P., and Salman, A. (2007). An overview of clustering methods. *Intelligent Data Analysis*, 11(6):583–605.

Pop, P. C., Fuksz, L., Marc, A. H., and Sabo, C. (2018). A novel two-level optimization approach for clustered vehicle routing problem. *Computers & Industrial Engineering*, 115:304–318.

Prischink, M., Kloimüllner, C., Biesinger, B., and Raidl, G. R. (2016). Districting and routing for security control. In Blesa, M. J., Blum, C., Cangelosi, A., Cutello, V., NUOVO, A. D., Pavone, M., and Talbi, E.-G., editors, *Hybrid Metaheuristics*, pages 87–103, Cham. Springer International Publishing.

Rahman, M. S. and Kaykobad, M. (2005). On hamiltonian cycles and hamiltonian paths. *Information Processing Letters*, 94(1):37–41.

Santos, B. B. d. (2023). Using districting and a data driven tsp to improve last mile delivery. masterthesis, Mestrado em Engenharia e Gestão Industrial, [Instituto Superior de Engenharia do Porto].

Sarikyriakidis, S., Goulianas, K., and Margaris, A. (2023). Using self-organizing maps to solve the travelling salesman problem: A review. *WSEAS TRANSACTIONS ON SYSTEMS*, 22:131–159.

Sevaux, M. and Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME*.

Singh, K., Bedi, S. K., and Gaur, P. (2020). Identification of the most efficient algorithm to find hamiltonian path in practical conditions. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 38–44.

Sisodia, D., Singh, L., Sisodia, S., and Saxena, K. (2012). Clustering techniques: a brief survey of different clustering algorithms. *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, 1(3):82–87.

Tamilarasi, M. and Dhenakaran, S. S. (2018). Hamiltonian approach for finding shortest path. *International Journal of Scientific Research in Science and Technology*, 4(8):484.

Tavares-Pereira, F., Figueira, J. R., Mousseau, V., and Roy, B. (2007). Multiple criteria districting problems: The public transportation network pricing system of the paris region. *Annals of Operations Research*, 154:69–92.

Vidal, T., Battarra, M., Subramanian, A., and Erdog˘an, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research*, 58:87–99.

Vidal, T., Laporte, G., and Matl, P. (2020). A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2):401–416.

Wøhlk, S. and Laporte, G. (2019). A districting-based heuristic for the coordinated capacitated arc routing problem. *Computers & Operations Research*, 111:271–284.

Xu, H. and Lan, H. (2023). An adaptive layered clustering framework with improved genetic algorithm for solving large-scale traveling salesman problems. *Electronics*, 12(7).

Zheng, J., He, K., Zhou, J., Jin, Y., and Li, C.-M. (2023). Reinforced lin–kernighan–helsgaun algorithms for the traveling salesman problems. *Knowledge-Based Systems*, 260:110144.

Zhong, H., Hall, R., and Dessouky, M. (2007). Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41:74–89.