NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

PEDRO ALEXANDRE PARREIRA PRIMOR

BSc in Computer Science

OPTIMIZATION PROBLEMS IN THE POSTAL SECTOR

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
September, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

OPTIMIZATION PROBLEMS IN THE POSTAL SECTOR

PEDRO ALEXANDRE PARREIRA PRIMOR

BSc in Computer Science

Advisers: Paula Alexandra Amaral
Associate Professor, NOVA University Lisbon
João Carlos Gomes Moura Pires
Associate Professor, NOVA University Lisbon

Examination Committee

Chair: Vasco Miguel Moreira do Amaral
Associate Professor, NOVA University Lisbon

Members: Isabel Cristina Silva Correia
Associate Professor, NOVA University Lisbon
João Carlos Gomes Moura Pires
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
September, 2022

Optimization problems in the postal sector

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.6) [22].

Acknowledgements

It fills me with joy to finally be writing these paragraphs of gratitude. This project has proven itself to be the most demanding, the most complex and the most consequential I've encountered thus far, and because of this I'd like to make some honourable mentions about each and every person that has contributed to this personal, academic achievement of mine.

Firstly, I'd like to thank my advisors, Paula and João, for lighting up the path that has brought me to this place. For their guidance and utmost support during this work's execution, I am extremely grateful.

I'd also like to thank the Department of Computer Science (DI) and NOVA School of Science and Technology (FCT), as a whole, the institutions that were my second home throughout these last five years and that gave me the opportunities to grow intellectually, professionally and emotionally.

To CTT – Correios de Portugal, S.A. (CTT), the company that confided to me this fascinating task and that helped me achieve it, I'd like to express my sincere appreciation.

Finally, a special recognition to my family and friends, that have continuously supported me during this last year and during my academic path. They gave me tools that I needed to reach this goal and have made the journey much more enjoyable.

Abstract

Supply chain optimization is a widely studied field of operations research. Nevertheless, adapting the existing solutions to the specifications of each company is an interesting and stimulating challenge. With this in mind, the project described herein, developed in partnership with CTT, looks to provide the company with precious tools to more efficiently manage the labour allocated to mail delivery and increase the productivity of the workforce as a whole.

To achieve these objectives, it follows up on a previous work by Pereira[26], where an extension of the Vehicle Routing Problem (VRP) was proposed to optimize the last-mile delivery step of the mail distribution procedure, but this time giving particular relevance to the adequacy of the model developed to the intricacies imposed by the company and exploring suitable adaptations. One of the requirements, for standardization purposes, is the creation of segments, composed of sets of postal codes that serve as input to the optimization model. Finally, it was necessary to merge this work with the company's workflow by integrating the model with SISMA, a productivity assessment tool already used by CTT.

Keywords: combinatorial optimization, last-mile, operations research, postal service, vehicle routing problem

Resumo

A otimização de uma cadeia de abastecimento é um campo vastamente estudado no âmbito da investigação operacional. Contudo, adaptar as soluções existentes aos critérios de cada empresa é um desafio bastante interessante e estimulante. Tendo isto em consideração, este projeto, desenvolvido em parceria com os CTT – Correios de Portugal, S.A. (CTT), procura fornecer à empresa ferramentas que permitam uma gestão eficiente da força de trabalho afeta à distribuição de correio.

Para atingir este propósito, este trabalho teve como ponto de partida uma proposta desenvolvida por Pereira[26], onde uma adaptação do Vehicle Routing Problem (VRP) foi desenvolvida para otimizar a etapa last-mile do processo de distribuição. No presente trabalho, dá-se uma atenção redobrada à compatibilidade do modelo desenvolvido com as complexidades impostas pela empresa e explora-se algumas melhorias consideradas apropriadas. Um dos requisitos, para manter alguma estibilidade nos resultados, é a introdução de segmentos, compostos por conjuntos contíguos de códigos postais, que alimentam o modelo. Finalmente, para combinar este trabalho com o fluxo de trabalho da empresa, fez-se a integração do modelo de otimização com o SISMA, uma ferramenta de avalição de produtividade já utilizada pelos CTT.

Palavras-chave: investigação operacional, last-mile, otimização combinatória, problema de roteamento de veículos, serviço postal

Contents

List of Figures

List of Tables

Glossary

assignment problem An assignment problem is a combinatorial optimization problem that attempts to assign a set of agents to a set of tasks, while minimizing the total cost of the assignment. Any agent can perform any task but there is a cost associated with each assignment pair. (**p. 58**)

complete digraph In graph theory, a complete graph is a graph in which every pair of vertices is connected by a distinct edge. Likewise, a complete digraph is a graph in which every pair of vertices is connected by two unique edges (one in each direction). (**p. 45**)

last-mile Last-mile, in supply chain management and transportation planning, is the last leg of a journey comprising the movement of goods from a transportation hub to a final destination. (**pp. iv, v, 2, 24, 52, 58**)

local search Local search is a heuristic method for solving computationally hard optimization problems. These algorithms move from solution to solution in the space of candidate solutions by applying local changes, until a solution deemed optimal is found or a time limit is elapsed. This means that a local search is prone to converge into local optima. (**pp. 16, 17**)

set partitioning problem A set partitioning problem is the task of deciding whether a given set of items S can be partitioned into smaller subsets so that a predetermined condition is valid for all generated partitions. All the items in S must belong to only one subset. Example: In a two-way number partitioning problem the goal is to divide S into two subsets so that the sum of the numbers in each subset are as nearly equal as possible [20]. **(p. 15)**

Acronyms

***k*-NN**	*k*-Nearest Neighbours (**p. 66**)
ACO	Ant Colony Optimization (**p. 17**)
ANACOM	Autoridade Nacional de Comunicações (**pp. ix, 1, 2**)
B&B	Branch and Bound (**pp. 3, 10, 13**)
BFS	Best-first Search (**pp. 11, 12**)
BOOP	Boolean Optimization Problem (**p. 14**)
BrFS	Breadth-first Search (**p. 11**)
CBFS	Cyclic Best-first Search (**pp. 11, 12**)
CDP	Postal Distribution Center (**pp. 23–25, 27, 29, 30, 32–35, 48, 49, 58, 62–64**)
CP	Constraint Programming (**p. 15**)
CPL	Production Logistics Center (**p. 23**)
CTT	CTT – Correios de Portugal, S.A. (**pp. iii–v, ix, 1–4, 21–25, 27–30, 33, 41, 50, 52, 54–58, 63**)
CVRP	Capacitated Vehicle Routing Problem (**p. 7**)
DFS	Depth-first Search (**p. 11**)
DI	Department of Computer Science (**p. iii**)
FCT	NOVA School of Science and Technology (**pp. iii, 2, 48**)
GA	Genetic Algorithm (**pp. 3, 16**)
GUI	Graphical User Interface (**p. 28**)
HTTP	Hypertext Transfer Protocol (**p. 55**)
IP	Integer Programming (**pp. 10, 14, 58**)

LP Linear Programming (**p.** 14)

MIP Mixed-integer Programming (**pp.** 10, 14)
MMS Mixed Mail Sorting (**pp.** 23, 32)
MTZ Miller-Tucker-Zemlin (**pp.** 9, 41, 43)

PAM Partitioning Around Medoids (**pp.** 19, 20)

QP Quadratic Programming (**p.** 14)

RMS Rest Mail Sorting (**pp.** 23, 32)

TDP Truck Dispatching Problem (**p.** 6)
TS Tabu Search (**pp.** 3, 16)
TSP Traveling Salesman Problem (**pp.** ix, 2, 15, 16, 33, 34)

UEC Unidade Equivalente de Correio (**p.** 25)

VBA Visual Basic for Applications (**pp.** 55, 56)
VNS Variable Neighborhood Search (**pp.** 3, 16)
VRP Vehicle Routing Problem (**pp.** iv, v, 2, 4, 6, 7, 9, 10, 14, 15, 17, 25, 27, 28, 33, 41, 44, 46–50, 52, 54, 57, 58)

1 Introduction

1.1 Motivation

For the last few years, there has been a paradigm shift in the field of postal services. Digitization keeps on growing and mail delivery is steadily being replaced by e-mail services. Numbers show that letter deliveries have shrunk substantially in the last decade. In 2010 there were a total of approximately 1 135 million postal objects delivered in Portugal. By 2020 that number had shrunk by 46.8% to an approximate total of 604 million postal objects [3].

According to the latest data published by ANACOM, in Portugal, there was a 5.1% decrease in postal traffic in the last trimester of 2021 alone when compared to the same period in 2020. On the other hand, with the rise of e-commerce, exacerbated in part by the pandemic, parcel deliveries are picking up strength. In the last trimester of 2021, an increase of 4.0% was verified when compared to the homologous period in 2020. The trends are clear, and they show no sign of reversing, as can be seen in Figure 1.1.

The reduction in traffic has as consequence the cutback in human resources and in distribution centres, which saw a decline of 5.7% and 9.4% respectively in the last decade [3].

These changes have opened up space for much competition to emerge. This is why it's important for delivery/postal service companies to strive to increase their productivity in the hopes of lowering the cost of their services and consequently attracting a greater number of clients.

The optimization of the entire logistical process from the moment when a letter is first dropped in a post box; is processed by a postal distribution centre; sometimes transported hundreds of kilometres and finally dropped in a mailbox by a postman is of extreme importance, not only to the companies involved in this service, which increase the efficiency of their resource utilization, but also to society itself.

For the company CTT, this phenomenon has brought about a lot of change for a business that is historically tied to the exchange of correspondence in all of Portugal.

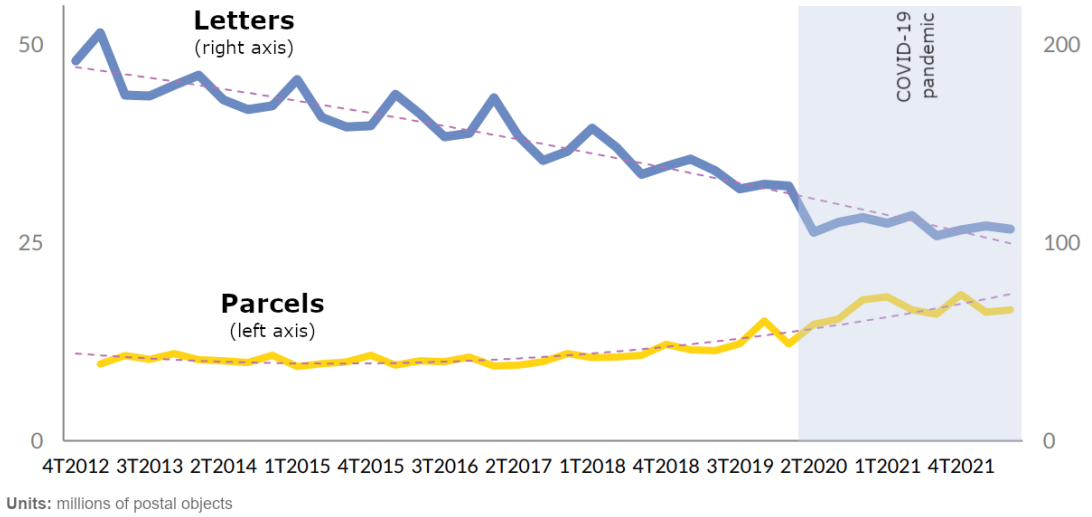


Figure 1.1: Quarterly evolution of the number of letters and parcels delivered in Portugal. (ANACOM)

1.2 Context

As mentioned before, this dissertation follows the work of Pedro Pereira, a former student at FCT. In his work, developed in partnership with CTT, the goal was to optimize the routes made by postmen in the last-mile delivery step of the distribution process. This optimization consisted of producing a model based on one of the most important problems in the fields of supply chain management and distribution logistics: the Vehicle Routing Problem (VRP). The VRP was introduced in 1959 by Dantzig and Ramser [10] and is considered a generalization of the TSP. In its simplest form, the TSP asks the question of how a vehicle can, starting and finishing at a specified location, visit every location once at a minimal cost. The VRP generalizes this problem in that it asks the question of what is the optimal set of routes for a fleet of vehicles to deliver goods to a set of locations at a minimum cost. A VRP instance can be represented using a graph, in which vertices are the locations that must be visited and edges are the paths between locations. Modelling a VRP instance requires using a mathematical formulation to define the problem. It is important to note that VRPs are a wide-ranging class of problems with many variants that adapt to specific requirements.

The model proposed by Pereira in [26], referred to as MVRP, used a three-index vehicle flow formulation (see Section 2.1.1), had the goal of minimizing the number of vehicles or postmen (assuming that each postman is assigned to a single vehicle), and used some additional constraints, such as limiting the amount of time a postman is allowed to work.

A VRP is an NP-hard problem, and solving relatively large VRPs can be a computational challenge [30]. This is why the decision of selecting the best strategy to solve these

problems must be carefully considered. Firstly, if the solution returned should be accompanied by an optimality certificate, then an exact method must be employed. The most interesting, widely used method is the Branch and Bound (B&B) algorithm. The B&B is based on two principles: "branching"(splitting the search space into smaller domains) and "bounding"(using pruning rules to eliminate suboptimal search spaces). Even if the algorithm stops because a pre-defined time limit is exceeded, an incumbent solution is in general returned with the indication of a bound on the optimality gap, as a measure on how far away it is from an optimal solution. However, if it suffices to find acceptable solutions without an optimality certificate, then the use of heuristics is a good alternative. The class of heuristic methods available is large, including well-known algorithms like Greedy Search, Variable Neighborhood Search (VNS), Tabu Search (TS), and Genetic Algorithms (GAs).

In the previous work, an empirical comparison was made between the two approaches using the distribution centre of Alverca. The first one used an optimization package (CPLEX) that uses an exact method to produce a solution with an optimality certificate. The second one used a greedy search heuristic to reach an acceptable solution, but without the guarantee of optimality. The exact method approach was more successful because the results obtained in Alverca showed the greedy search method saw a 6.3% increase in total routing time [26].

1.3 Key contributions

Although the previous work provided a valuable contribution to solving this problem, it was just a first step towards a tool capable of being applied in real-world situations for CTT. In this thesis, three objectives were outlined.

The first was to improve the existing model and work in unison with the CTT to reach a definitive and more realistic model which can be applied to every distribution centre operated by the company in Portugal. To do this, it was important to understand how the routes created by the previous work's optimization process relate to the ideal routes the company is looking for. It was necessary to alter the previous model by changing the objective function.

It is also important to make sure that a problem's instance can be solved in a reasonable time frame. To guarantee this, some optimizations to the model are proposed, that attempt to reduce the size of the problem and, as a consequence, the time needed to solve it.

The second, put forth by CTT, was to devise a way to aggregate postal codes into segments. A segment, is a collection of sequential postal codes that must be visited jointly, one after another, by the same postman. Since the traffic destined to postal codes changes daily, the routes generated by the optimization model will also change, from one day to the next, in accordance to this. Using individual postal codes in the routing procedure would completely change the routes from one day to another and cause unwanted disturbance

to operations. Using segments for creating the routes reduces the problem's granularity, and these daily alterations become more manageable.

The introduction of segments also allows for a reduction in the problem size, because the routing procedure is now done with dozens or hundreds of segments, instead of with thousands of postal codes.

The last objective is the integration of the optimization model with SISMA. SISMA is a spreadsheet tool, used by CTT, that contains a standardized time for each task and subtask available to postmen and workers in the supply chain. These times are then manipulated and aggregated to find coefficients associated to a distribution flow. A flow is, in turn, a distribution procedure of a single postal object. For example, a valid flow would be the procedure of delivering an untraceable small letter, to an apartment block using a moped. After obtaining these estimates, SISMA is able to output an estimate into how many hours of work are needed for a certain amount of objects. It does this by aggregating the time needed to handle the objects in each kind of flow.

1.4 Document structure

This document is organized in the following manner:

1. Introduction

This chapter presents the reader with a motivation behind the project, a brief description of the problem and the goals that must be successfully achieved.

2. Literature review

This second chapter discusses some approaches proposed in the literature to deal with the problem at hand, focusing primarily on VRPs and how to solve them.

3. Problem statement

In this chapter, several aspects of the problem are described, namely how the internal operation of CTT functions and the specifications that must be taken into account when modelling the VRP. It also presents the previous work and SISMA.

4. Segmentation

This chapter describes the segmentation process, its motivation, how the data used to implement it was retrieved and the two algorithms developed. It also presents several visualizations to analyse segmentation results.

5. VRP formulation

This chapter discusses the formulation used to model the VRP and its ability to correctly represent the *modus operandi* of the company. It presents the optimizations and implications it has in reducing the problem size and how it affects the optimality of the solution.

6. Results and solution analysis

This is the chapter where analyses are made comparing the proposed solution with the company's current practice and with the previous work's results.

7. Integration with SISMA

Here, the integration of the routing model with SISMA is explained, together with how it can be incorporated in the company's operational system.

8. Conclusions

This last section contains some final remarks and makes some comments regarding the necessary follow up on this work.

2

Literature Review

This chapter will provide some insights into the problems in question and the solutions proposed in the literature to solve them. It will focus primarily on the VRP, its variants, and methods for solving it.

2.1 Vehicle Routing Problem

According to Irnich, Toth, and Vigo [30], a Vehicle Routing Problem (VRP), in its simplest form, is an optimization problem that given a set of transportation requests and a fleet of vehicles attempts to determine a set of vehicle routes to perform all (or some) transportation at a minimum cost.

This set of problems was first introduced in 1959 by Dantzig and Ramser, referred then as the Truck Dispatching Problem (TDP). They proposed a way to find the optimal routing of a fleet of delivery trucks between a terminal and a series of service stations that are to be supplied by the terminal [10]. Since then, a plethora of variants and generalizations of the problem have emerged to deal with specific requirements like:

1) vehicles having a limited carrying capacity (CVRP),
2) having multiple terminals or depots (MDVRP),
3) not imposing the return of the vehicles to the terminal (OVRP),
4) having both deliveries and pickup requests (VRPPD),
5) vehicles having to make all the deliveries before doing any pickups (backhauling) (VRPB),
6) having predefined time windows for the deliveries (VRPTW).

It is important to realize that some of these characteristics can be combined to create more complex formulations. In principle, we could have a VRP with limited carrying capacity, multiple depots, time windows, and backhauling, for example.

A VRP is an NP-hard problem [30] and as a consequence finding an exact solution of some instances can often be computationally expensive, especially if the instance has a

sufficiently large number of variables. This is why the use of some heuristics is often an alternative when it suffices to find acceptable but not optimal solutions in a reasonable amount of time.

2.1.1 Problem formulations

To solve a specific Vehicle Routing Problem (VRP)s, one needs to create a mathematical model that can characterize and rigorously define the problem. Three approaches exist that attempt to do this.

Vehicle flow formulations

This kind of formulation uses binary variables to indicate whether a vehicle travels the path between two locations. These formulations can be further subdivided into **two-index** formulations and **three-index** formulations. In two-index formulations, only the two locations are specified because it is assumed that all vehicles are identical and that one needs not to differentiate between vehicles. Three-index formulations use a vehicle index to indicate if a certain vehicle traverses a certain path. The three-index formulation of a pure Capacitated Vehicle Routing Problem (CVRP) is the following:

Given:

V: the set of locations, with the depot at index 0;

K: the set of vehicles;

t_{ij}: the time taken to go from location i to location j, $\quad i \neq j, i \in V, j \in V$;

c_i: the time taken to service location i, $\quad i \in V$;

v_i: the amount of goods to deliver in location i, $\quad i \in V$;

M_k: the maximum capacity of vehicle k, $\quad k \in K$;

T: the maximum time a vehicle is allowed to work.

And defining the variables:

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle k visits location j after i} \\ 0 & \text{otherwise} \end{cases}, k \in K, i \in V, j \in V, i \neq j.$$

u_i: rank order in which location i is visited, $\quad i \in V$

The objective function is to:

$$minimize \sum_{i \in V} \sum_{j \in V \setminus i} \sum_{k \in K} x_{ijk}(t_{ij} + c_i) \tag{2.1}$$

subject to the following constraints

$$\sum_{i \in V \setminus \{0\}} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in V \tag{2.2}$$

$$\sum_{j \in V \setminus \{0\}} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V \tag{2.3}$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad \forall k \in K, \forall h \in V \setminus \{0\} \tag{2.4}$$

$$\sum_{i \in V \setminus \{0\}} \sum_{j \in V} x_{ijk} v_i \leq M_k \quad \forall k \in K \tag{2.5}$$

$$\sum_{i \in V} \sum_{j \in V} x_{ijk} (c_i + t_{ij}) \leq T \quad \forall k \in K \tag{2.6}$$

$$\sum_{i \in V \setminus \{0\}} x_{0jk} \leq 1 \quad \forall k \in K \tag{2.7}$$

$$\sum_{i \in V \setminus \{0\}} x_{i0k} \leq 1 \quad \forall k \in K \tag{2.8}$$

$$\text{subtours elimination constraint} \tag{2.9}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in V, \forall j \in V, \forall k \in K \tag{2.10}$$

The objective function 2.1 states that the goal is to minimize the total working time of all vehicles. Constraints 2.2 and 2.3 state that each location is visited only once. Constraint 2.4 guarantees that if a vehicle enters a location, it must also exit that same location. Constraint 2.5 guarantees that the maximum capacity of the vehicles is not exceeded, and likewise constraint 2.6 guarantees that the maximum time a vehicle is allowed to work is not exceeded. Constraints 2.7 and 2.8 guarantee that each vehicle can depart from and arrive at the depot only once.

Constraint 2.9 is used to prevent the occurrence of subtours in the solution. Subtours are infeasible routes because the nature of the problem implies that routes must start and end at the depot. A few approaches have been proposed to tackle this problem, but the most interesting one will be discussed further in Section 2.1.2.

Constraint 2.10 sets the flow variables of the problem to be boolean values.

Commodity flow formulations

Commodity flow formulations add a new set of continuous variables to account for commodity flow between locations. Given a set of locations V and locations $i, j, l \in V$ the flow variables indicate the amount of demand that flows through the arc between i and j, and that is destined to a third location l.

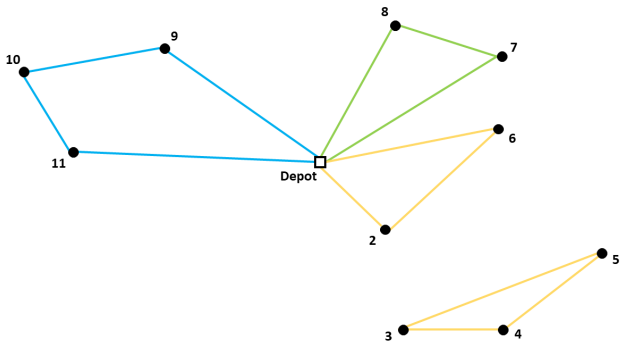


Figure 2.1: Example of a subtour generated within a route (AIMMS)

Set partitioning formulations

Set partitioning formulations use a possibly exponential number of binary variables to represent every single feasible route. Every feasible route has an associated cost, and the objective is to minimize the sum of all the costs to find the optimal solution. A major downside of this type of formulation is that, since the number of feasible routes can be exponentially large and there is a binary variable for each route, the number of variables can easily surpass millions in most problems.

2.1.2 Subtours elimination

A subtour is defined as a route that doesn't start nor end at the depot. It is infeasible because by definition, in the context of VRPs, all routes must have the depot as a starting point. The constraints defined in 2.9 prevent the creation of subtours and the way they are formulated is an interesting problem because the number of constraints generated can be very large and influence the efficiency of solving algorithms. The approach that better solves this problem is the Miller-Tucker-Zemlin (MTZ) formulation [23]. This formulation uses $n-1$ extra variables (one for each location, except the depot) to represent the order in which a location is visited and approximately $n^2/2$ extra constraints. The formulation is the following:

$$u_i - u_j + n * x_{ijk} \leq n-1 \quad \forall i \in V \setminus \{0\}, \forall j \in V \setminus \{0\}, i \neq j, \forall k \in K \tag{2.9}$$

The basic idea is to have a u_i value for every location i except the depot. Every time a location is visited the value of u increases, and this formulation makes it impossible for a vehicle to travel to a location it has already visited because that location would have a smaller u value. Using Figure 2.1 as an example, the subtour at the bottom right wouldn't be generated because a vehicle wouldn't be able to travel from location 5 to location 3. Because of this, there is a guarantee that the only way a vehicle drives in a circle is if it starts and ends at the depot.

2.2 VRP solving approaches

Determining the solution of a VRP is NP-hard [30], and because of this when it suffices to produce approximate solutions without the need for an optimality certificate, heuristics are often employed. The range of heuristics available is vast and some concrete examples are discussed in Section 2.2.3.

Heuristics can provide good solutions, but they are unable to provide an optimality certificate for the solution returned. This means that when this certificate is desired, exact methods have to be used. The most common exact algorithm is the Branch and Bound (B&B) method.

2.2.1 **Exact Methods** (Branch & Bound)

The Branch and Bound (B&B) algorithm is a widely used framework used for producing exact solutions to NP-hard problems [5]. As the name implies, it is built upon two basic principles: "branching" (splitting the search space into smaller domains) and "bounding" (using rules to prune off suboptimal search spaces). The "bounding" step, also called "pruning", is activated when a subproblem generated by the branching mechanism is impossible to solve or, even if it contains a feasible solution, there is the guarantee that from that node no solution better than the incumbent can be found

The behaviour of the algorithm is illustrated in Figure 2.2 [5]. Starting from the complete solution space, the algorithm iteratively splits and filters the search spaces until there are no unexplored branches of the decision tree left.

If one applies the Branch and Bound method to an Integer Programming (IP) or Mixed-integer Programming (MIP) problem, the algorithm starts by computing a solution to a linear relaxation of the problem. If the solution found respects the integrality constraints, the algorithm ends and the result is the solution found. Otherwise, the value of that solution is considered a lower bound or an upper bound, depending on whether one is dealing with a minimization or maximization problem, respectively. In this case, the algorithm selects a variable that doesn't respect the integrality constraints and branches the problem into two subproblems: one in which a condition is added so that the variable is less than or equal to $\lfloor x_i \rfloor$ (integer part of the value of the variable x_i in the solution); and one in which a condition is added so that the variable is greater than or equal to $\lceil x_i \rceil$ (integer ceiling value of the variable x_i in the solution). A solution is then computed for both subproblems. If any of the solutions are better than the best found so far, then that solution becomes the incumbent solution. If it can be proven that no solution can be found to improve the incumbent solution by expanding a certain branch, then the search space is pruned, and the algorithm considers that subproblem as terminal [24], meaning all subproblems in that search space are suboptimal. The pruning step prevents unnecessary computation and lowers the time needed for the algorithm to produce a result. This procedure is repeated iteratively until there are no unexplored branches of

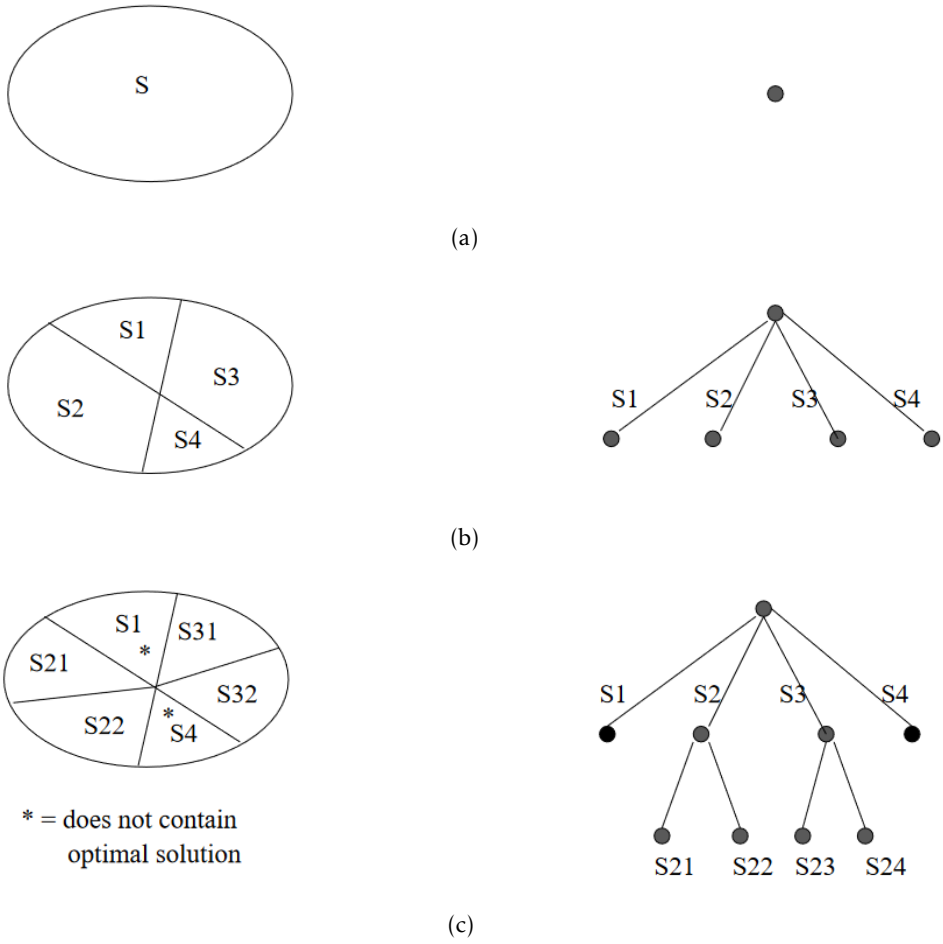
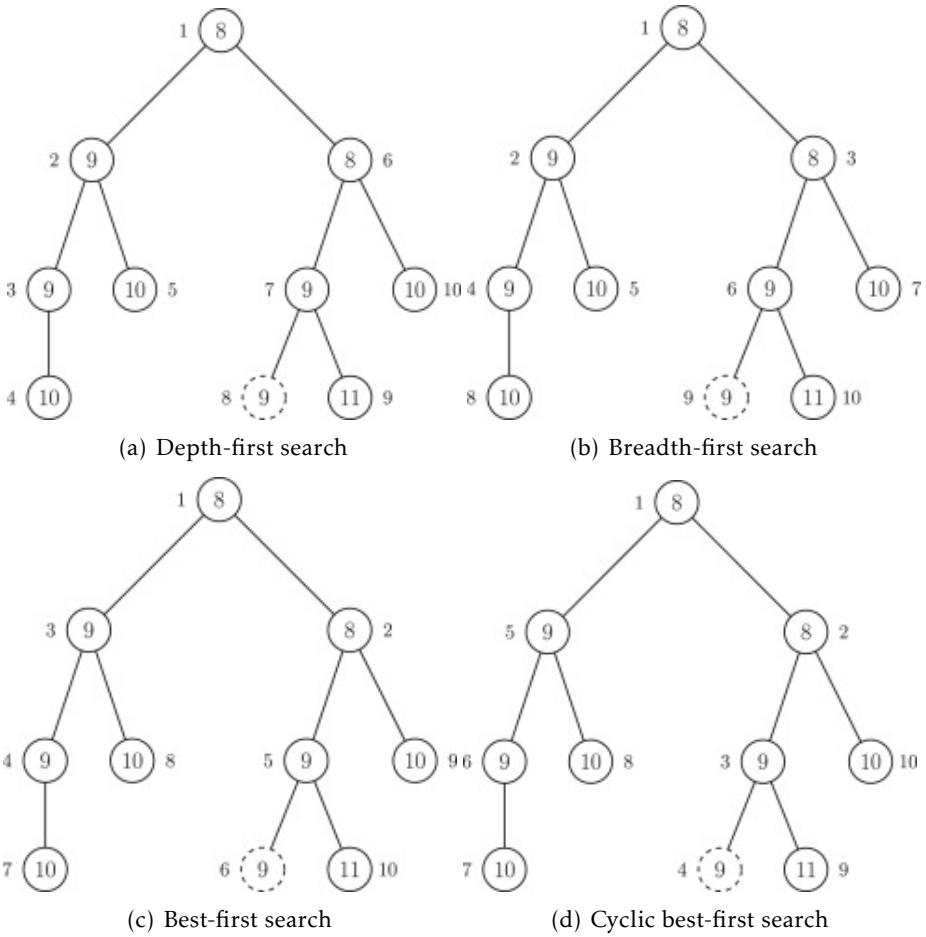


Figure 2.2: Illustration of the search space of Branch & Bound (Jens Clausen)

the decision tree or until they are all provably suboptimal. To improve the performance of the algorithm, three aspects are of utmost importance.

Search strategy

Firstly, the search strategy has to be selected. The most common approaches used for search strategies are: Depth-first Search (DFS), Breadth-first Search (BrFS), Best-first Search (BFS) and Cyclic Best-first Search (CBFS) [24]. With a DFS approach, branches are explored as far as possible before backtracking, and its implementation usually relies on a stack data structure. In a BrFS, all nodes at a certain depth are explored before continuing to the next depth level. It is usually implemented with a queue data structure. Using a best-first search strategy, the next node to be explored is selected using a measure-of-best function. The function typically used is the lower/upper bound on the value of the best solution in the subproblem. Because BFS is not tied to exploring one specific branch before any other, it often finds good solutions earlier in the process. A more recent approach that has seen good results is a Cyclic Best-first Search (CBFS). The difference



(a) Depth-first search (b) Breadth-first search

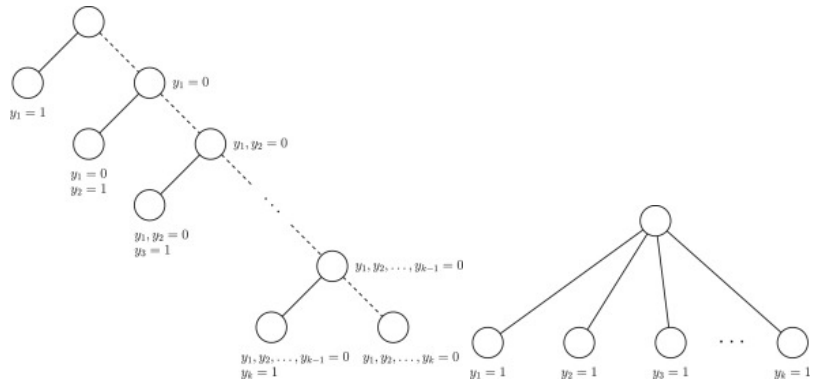
(c) Best-first search (d) Cyclic best-first search

Figure 2.3: Branch & Bound search strategies (David R. Morrison)

between CBFS and BFS is the introduction of contours. BFS uses a single heap data structure to store all unexplored subproblems, while CBFS divides them into multiple heaps or contours. The rule by which contours are created varie, butt an example could be having each contour be associated with a depth level. This behaviour is illustrated in Figure 2.3. The number inside each node is the value of the solution to that subproblem and the number to the left of each node is the order by which nodes are visited.

Branching strategy

The second aspect to consider is the branching strategy. The branching strategy determines how children are generated from a subproblem and the approaches can be categorized into two groups: **binary** or non-binary, also known as **wide** strategies. Binary strategies divide a problem into two mutually exclusive subproblems and are the most common in integer programming problems. Wide branching, on the other hand, divide problems into n different subproblems. Wide branching methods can greatly alter the size of the search tree because it avoids having to create long sequences of subproblems



(a) Binary branching (b) Wide branching

Figure 2.4: Branch & Bound branching strategies (David R. Morrison)

as be seen in Figure 2.4. Since the problem this thesis aims to solve uses mostly binary variables and binary branching is the most employed for solving integer programming problems, this review will emphasize this type of strategy.

The greatest impact on performance in a binary branching strategy boils down to the choice of the branching variable. For choosing the best variable, a lot of rules have been proposed. The easiest rule to implement is the **lexicographic ordering** rule, which selects the variable according to the order by which they were defined (variable x_k selected before x_{k+1}). Another commonly used and easy to implement is the **most fractional** rule, which selects the variable whose fractional part is closest to 0.5. This approach, however, has been shown in [2] to be generally no better than selecting a variable at random. Another widely regarded method is **strong branching** which selects the variable that induces the largest change in the objective function. It involves performing a look-ahead before branching and testing which of the candidate variables, if selected, improves the objective function the most. A third approach worth mentioning is **pseudocost branching** which uses past experience to predict the change in the objective function for each candidate variable. Using the change induced in the objective function to select the branching variable is beneficial to performance because it is more likely that the generated subproblems can be pruned.

Pruning mechanism

The last aspect which affects performance in a B&B setting is the pruning rules used. It is the key component of any B&B algorithm because it is the only way to reduce the size of the search tree and greatly limit the computational effort needed to solve a certain problem. The most basic and common way to prune is to use the lower bound (or upper bound for maximization problems) on the objective function at each subproblem. The subproblems which have a worse lower/upper bound than the incumbent solution are

pruned. Another option is to use **dominance relations**. Dominance relations allow subproblems to be pruned if it can be proven that they are dominated by another subproblem. Subproblem S_1 dominates subproblem S_2 if for any solution descending from S_2 there is a solution descending from S_1 that is at least as good, and so it suffices to explore just S_1. Another great idea proposed by Gomory was the notion of **cutting planes** [16]. A cutting plane is a constraint that can be added to an IP problem to tighten the feasible region without leaving out any integer solution. Cutting planes led to the development of a **branch-and-cut** algorithm by Padberg and Rinaldi[25]. At every subproblem generated, new cutting planes are added to the Linear Programming (LP) relaxation. The question of when to generate cutting planes and when to branch is an interesting problem when designing branch-and-cut algorithms. Finally, another pruning methodology called **branch-and-price** was introduced in [27]. It combines elements of the branch-and-bound method with an approach called column generation. In this method, instead of adding new constraints to the generated subproblems, sets of variables (columns) are left out and are added back as needed. This approach is especially helpful in problems where the majority of columns are not essential for solving the problem.

2.2.2 Optimization solvers

Optimization solvers are software packages specialized in solving NP-hard problems to optimality. Since competition is fierce in this field, commercial solvers have long striven to provide the fastest tool using the best state-of-the-art algorithms. While free and open-source software is often favoured in the programming community, in the case of optimization software, open-source solvers are lagging behind commercial ones. The most successful commercial optimization solvers in the industry nowadays are CPLEX by IBM, Gurobi and FICO Xpress. To select the best one, it's important to measure their performance when faced with a problem like the one discussed. The authors of [18] compared the performance of these three solvers using a Boolean Optimization Problem (BOOP). A BOOP is an IP problem in which all decision variables are of the boolean type. This problem is similar to ours because in a three-index VRP formulation most variables are binary flow variables. The results from [18] suggest that XPRESS finds worse solutions than the two other packages. Between the two, Gurobi is the one in which the optimal solution appears first, but CPLEX is slightly faster in providing that solution with an optimality certificate. Because of this last statement and because CPLEX is the most complete, it was selected as the best to tackle this problem.

CPLEX

CPLEX is one of the most advanced and accepted optimization software packages for solving LP, MIP, and Quadratic Programming (QP) problems [8]. It uses a **branch-and-cut** method to solve IP problems like the VRP. The optimization package has been under constant development and improvement since its conception in 1988 and, along with

XPRESS by FICO, it is one of the few solvers capable of handling Constraint Programming (CP) problems [4].

CPLEX allows users to parameterize its optimizer to increase performance depending on the problem's characteristics. For example, one can control the branching direction the solver should take first at each node or set a time limit for finding the optimal solution. If that time limit is reached and the algorithm still hasn't produced the solution with the optimality certificate, it will return the solution (if it was found), i.e. the best it has found up to that moment. It also offers APIs for using the solver with programming languages such as C, C++, Java, and Python.

2.2.3 Heuristics

For very large problems, when exact methods are not able to find exact solutions in a reasonable amount of time or when enough memory is not available, it is common to use heuristics to find acceptable solutions. Heuristic methods implement search strategies to find good solutions in promising regions of the search space, but without guaranteeing the optimality of the result. The class of VRP heuristics expanded greatly when the computational resources available were incapable of handling even average-sized problems with hundreds of locations. At a macro-level, VRP heuristics can be classified into four components [30]: constructive heuristics, improvement heuristics, population mechanisms, and learning mechanisms [7].

Constructive heuristics

The constructive heuristics class defines heuristics that start off with an empty solution and iteratively extend the incumbent solution until it is complete. In the case of VRPs, this would mean that all the locations have their supply met by a delivery vehicle. These heuristics are commonly employed as a starting point for other heuristics, as is the case of petal algorithms. Petal algorithms consist of generating a set S of feasible routes and combining them by solving a set partitioning problem.

Another constructive heuristic, immensely popular for its simplicity, is the **greedy search** algorithm. A greedy search algorithm is that which makes the locally optimal choice at each step. In the context of a TSP, for example, the procedure starts with an empty graph and selects the edge with the least cost. Afterwards, it will keep selecting edges of the least cost while making sure that subtours are not generated. The algorithm stops when a cycle passing through every vertex has been found. Figure 2.5 illustrates the behaviour of this method in a TSP instance.

Improvement heuristics

In most classical improvement heuristics, the solution is refined until a local optimum is reached. In metaheuristics, the same mechanism is applied with the addition of more sophisticated search structures that allow the algorithm to escape local optimum values.

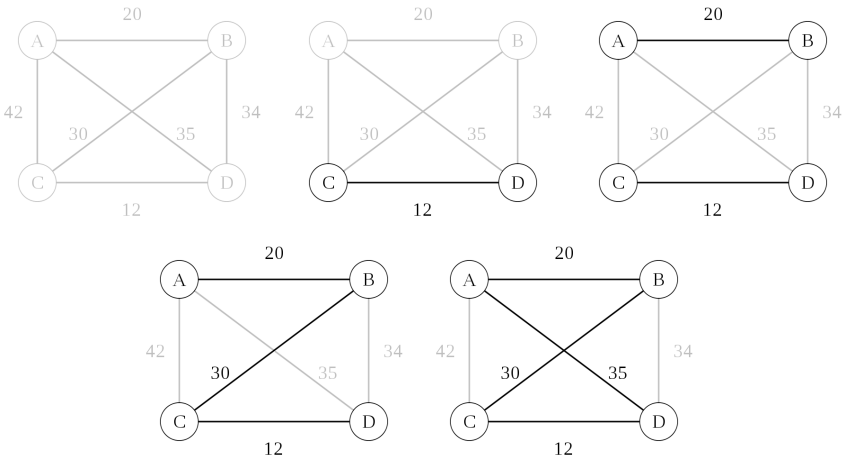


Figure 2.5: Greedy search iterations in a general TSP example

Variable Neighborhood Search (VNS) methods, for example, build on the idea that the local optimum incumbent solution might not be the global optimum solution. When it finds a local minimum, it proceeds to explore other neighbourhoods of this solution, and it moves to a new neighbourhood if it contains an improved solution. Following this, it reapplies a local search method to arrive at a new local minimum. If it can't find an improvement or if the computational effort to find one is too much, the algorithm stops and outputs the incumbent solution.

Another example, the Tabu Search (TS) method, starts with an initial feasible solution, and it inspects solutions in its neighbourhood, choosing the best that is not "tabu". It uses memory structures to conceive a tabu (forbidden) list, which consists, in essence, of a set of solutions that have been recently visited, or moves related to those solutions. A move is a rule incorporated in the definition of the neighbourhood. These memory structures prevent the algorithm to fall into the same local optimum it had already found and prevents the cycling behaviour present in classical improvement heuristics.

Population mechanisms

Population-based algorithms take inspiration from natural concepts like the evolution of species. These methods implement a high-level guidance strategy using complex data structures. The most famous class of population mechanisms is the Genetic Algorithm (GA) class. In a GA candidate, solutions are known as individuals and their properties are called genotypes. The decision on whether an individual survives into the next generation is dependent on its fitness level (usually the value of the objective function in the optimization problem being solved). A selection procedure is responsible for picking a set of individuals with the best fitness values to "breed". Breeding, also called the generation procedure, consists of creating a new generation of individuals based on the

population resulting from the selection procedure using crossover and mutation operators. The selection and generation procedures are repeated until a stopping condition is reached.

Learning mechanisms

Two learning mechanisms have been proposed to deal with VRP problems: neural networks and Ant algorithms [7]. Neural networks proved themselves unable to compete with other VRP heuristics and are now only applied in hybrid approaches. Ant Colony Optimization (ACO) algorithms are population-based algorithms with a learning paradigm derived from an analogy observed in the behaviour of ants, which lay pheromones on their trail as they forage for food. Paths for the best food sources are identified by a larger amount of pheromones. The algorithm can be divided into three major components: a **ConstructAntSolutions** phase, where a set of artificial ants constructs solutions, an optional **ApplyLocalSearch** phase where a local search is performed to improve the solutions obtained by the ants, and an **UpdatePheromones** phase where the aim is to increase the pheromone values associated with good or promising solutions and decrease those that are associated with bad ones [12].

2.3 Clustering methods

In this section, some clustering methods are discussed because they will be used in the construction of segments that is described in more detail in Chapter 4. The segmentation procedure included in this work firmly resembles a cluster analysis because it consists of grouping sets of CP7s that are close to each other into segments. However, we consider applying such a clustering method when the proximity between CP7s is an important and decisive factor.

There is a significant number of clustering methods available that serve multiple purposes and can be applied in various scenarios. These methods can usually be categorized into one of the following clustering models: connectivity-based, centroid-based, distribution-based, and density-based.

Connectivity-based models rely on the idea that objects are more related to other objects close to them than to those further away. These models group objects based on the distance or similarity between them. In **centroid-based** models, each cluster has a representative, also called centroid, that does not need to be part of the dataset. The k-means method is an example of a centroid-based model. In distribution-based clustering, objects are assigned to clusters given a probability distribution model and each object typically has a probability of belonging to a particular cluster. In **density-based** clustering, regions with higher density of data points are aggregated, while objects in sparse areas are usually considered noise.

2.3.1 k-means

The k-means clustering method is one of the most popular centroid-based clustering algorithms. It attempts to cluster data points in k groups by minimizing the **inertia criterion** (also known as within-cluster sum-of-squares criterion) [6]. In k-means, each cluster is assigned a centroid, which is calculated using the mean of the data points in the cluster. The algorithm requires the number of clusters k, to be provided in advance, which can be seen as a disadvantage when this number can't be estimated.

The criterion is formalized in Equation 2.3.1, where C is the set of clusters and u_j is the centroid of cluster j. The minimization factor is the variance of each cluster.

$$\sum_{i=0}^{n} \min_{u_j \in C} (\|x_i - u_j\|^2)$$

Since the problem is NP-hard, the implementation itself often resorts to an approximate method called **Lloyd's algorithm** [21]. Starting from an initial partition set, it repeatedly finds the centroid of each set and then re-partitions the data points according to which of these centroids is closest to each point. The algorithm can be decomposed into four steps:

1. Start with k points as initial centroids
2. Assign each data point to the closest centroid
3. Recalculate each centroid as the mean of the data points belonging to its cluster
4. Repeat steps 2 and 3, until convergence or a stopping criterion is met

For selecting the initial centroids, there are many alternatives. For example, the **Forgy** method chooses k random observations from the data set and uses these as the initial medoids. Another option, by the name of **Random Partition** method, starts with a random assignment of data points to k clusters and then computes the initial medoids as the mean of these random clusters.

2.3.2 k-medoids

The k-medoids problem is a clustering problem analogous to k-means, first described by Kaufman and Rousseeuw in [19]. In similarity to k-means, it attempts to minimize the distance between points belonging to a cluster and the cluster's centroid, but in this case the centroid (or medoid) is an actual data point. This means it can be used with arbitrary dissimilarity measures between every pair of data points, which makes it more robust to noise and outliers than its counterpart. A medoid is simply the object whose average dissimilarity to all objects in the cluster is minimal, and it can be viewed as the most centrally located point.

The work in [19] proposed a heuristic solution to the problem, based on a greedy search approach: the Partitioning Around Medoids (PAM) algorithm. The algorithm's goal is to minimize the average dissimilarity of objects to their medoid.

The PAM algorithm can be broken down into two phases (adapted from [29]):

1. BUILD

In this first phase, a collection of k objects are select to be initial medoids. The selection process entails the following steps:

1. Initialize the set of initial medoids S, by adding to it the object for which the sum of the distances to all other objects is minimal (i.e. the most centrally located object).
2. Consider an object $i \in U$ (U is the set of unselected objects) as a candidate for inclusion into S;
3. For an object $j \in U \setminus \{i\}$ compute D_j, the dissimilarity between j and the closest object in S;
4. If $D_j > d(i,i)$ (the dissimilarity between i and j) object j will contribute to the decision to select object i; let $C_{ij} = max(D_j - d(i,j), 0)$;
5. Compute g_i, the total gain obtained by adding i to S as:

$$g_i = \sum_{j \in U} C_{ij}$$

6. Choose the object i that maximizes g_i and add it to S.

These steps are performed until all k initial medoids have been selected.

2. SWAP

This second phase attempts to improve upon the set of initially selected medoids. It consists of comparing all pairs $(i,h) \in S \times U$ and computing the effect T_{ih} that transferring i from S to U and, inversely, transferring h from U to S has on the clustering.

To compute T_{ih}, one has to take into account the contribution that each object $j \in U \setminus \{h\}$ has on the swap between i and h. This contribution will be referred to as K_{jih}.

The SWAP phase is composed of the following steps:

1. K_{jih} is computed, taking into account the following cases:
   a) if $d(j,i) > D_j$, then:
      i. if $d(j,h) \geq D_j$, then $K_{jih} = 0$;
      ii. if $d(j,h) < D_j$, then $K_{jih} = d(j,h) - D_j$;

   In both subcases, $K_{jih} = \min\{d(j,h) - D_j, 0)\}$.

b) if $d(j,i) = D_j$, then:

i. if $d(j,h) < E_j$ where E_j is the dissimilarity between j and the second-closest selected object, then $K_{jih} = d(j,h) - D_j$ (K_{jih} can be either positive or negative);

ii. if $d(j,h) \geq E_j$, then $K_{jih} = E_j - D_j$ (in this case $K_{jih} > 0$);

In both subcases $K_{jih} = \min\{d(j,h), E_j\} - D_j$.

2. Compute T_{ih}, the total effect of the swap as:

$$T_{ih} = \sum_{j \in U} K_{jih}$$

3. Select a pair $(i,h) \in S \times U$ that minimizes T_{ih};

4. If $T_{ih} < 0$ the swap is carried out and the process restarts from step 1 of the SWAP phase. If $\min T_{ih} > 0$, the value of the objective function cannot be decreased further and the algorithm halts.

The main drawback of this original PAM algorithm is its complexity for the BUILD phase has a runtime complexity of $O(kn^2)$ and the SWAP phase one of $O(k(n-k)^2)$ [28]. For this reason, some alternatives that trade quality for runtime have been proposed, like the CLARA and CLARANS algorithms.

3

Problem statement

This chapter will present some technical aspects of the operations undertaken by CTT. It will showcase the characteristics of the problem as a whole and describe the previous work (that served as a starting point for this thesis).

3.1 CTT's distribution process

CTT is, by far, the largest postal delivery service in Portugal by market share as it accounts for 84.6% of the total postal traffic [3]. For this reason, there are many aspects of the whole operation that must be taken into account for improving the efficiency of the process.

3.1.1 Postal codes

The first aspect to consider is how CTT groups postal addresses. The Portuguese territory is divided into postal codes. A postal code is a series of digits that represent a certain geographical area, and they are arranged hierarchically. The first subdivision is called CP4. It comprises four digits, and it represents a certain region. The second subdivision is CP7, which is obtained by adding three digits to a CP4, usually with a hyphen separating the two sets of digits. A CP7 is commonly used to represent some street or a set of delivery locations that are supplied jointly. The last and smallest division is called CP10, and it is used to uniquely identify a delivery location and postal address. Figure 3.1 illustrates the areas encompassed by each CP4 in the municipality of Lisbon.

3.1.2 Postal objects

The second aspect to consider is the fact that each delivery is unique and has a set of attributes that characterize it. Some examples of such attributes are:

Traceability: the capability to trace a delivery object from source to destination.

Format: for categorizing a delivery object's type and how it should be handled internally. The five existing categories are LLE (letters), SP (small packages), MP (medium packages), P (packages), and LNP (large packages).

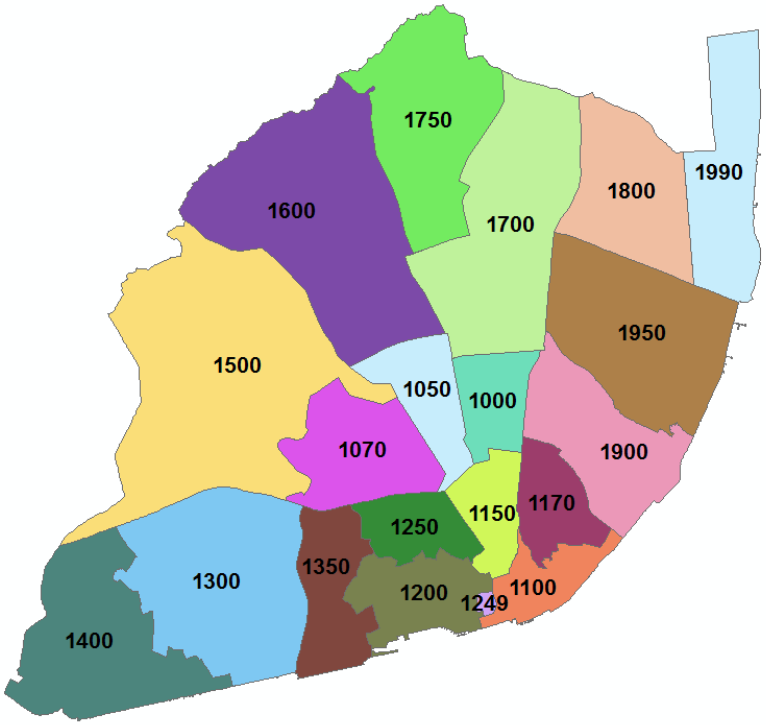


Figure 3.1: CP4 regions in the municipality of Lisbon (Rosa Melo Félix)

Delivery standard: the maximum time limit an object has to be in the supply chain, from when it arrives at one of CTT's facilities to when it reaches its final destination. The available standards are D+0, D+1, D+2, D+3, and D+5 and the number indicates how many days the object must be delivered in.

Upstream division level: how the postal objects are organized upon arriving at a distribution centre. The existing levels are: CP10, CP7 and CP4.

Delivery point: drop points can be either be personal homes (both stand-alone houses and apartment blocks), commercial or functional addresses, post office boxes, or clients.

Route type: which type of vehicle is responsible for the delivery.

Proof of delivery: whether the service requires the recipient to sign a proof of delivery.

These attributes and a permutation of the different values each can have are what is defined as a postal object flow. Objects are grouped into flows because, in theory, two objects that have the exact same distribution flow should contribute the same amount of time to the total working time in a distribution centre.

3.1.3 Distribution process

The third aspect to take into account is how the distribution process is streamlined. To explain this, some nomenclature, used internally by CTT, has to be defined. For instance, a Postal Distribution Center (CDP) is the depot of the last mile delivery step. It is assigned a geographic range and is responsible for supplying CP7s within that range. Every morning the postmen that work in that CDP are assigned to a route, formed by a set of CP7s, and deliver the mail assigned to them in order. The routes are static and are only updated occasionally with the help of human resources. A CDP is, in turn, supplied by a Production Logistics Center (CPL). CPLs are large postal centres responsible for handling a huge amount of postal objects of every kind. Inside CPLs, there are several mail sorting machines, which are responsible for automatically sorting and organizing incoming mail. There are two kinds of such machines: Mixed Mail Sorting (MMS) machines (for letters and envelopes) and Rest Mail Sorting (RMS) machines (for parcels and larger packages).

The distribution steps of a postal object consists of the following phases (illustrated in Figure 3.2):

1. Acceptance:

On this first step, mail is delivered to collection points by individual customers and then forwarded onwards to treatment centres. However, since most traffic nowadays comes from corporate clients, they are usually the ones responsible for delivering the mail from the printing facilities to treatment centres.

2. Treatment:

Treatment centres, known as CPLs, have the role of handling and preparing mail to be transported downstream to CDPs. Incoming mail first gets separated by format and standard. Whenever possible, LLE objects get fed into mail processing machines, that group mail by final destination. These machines also print a bar code into each letter to mark the date, time, and machine that processed it. After this step, workers feed batches of objects to mail sorting machines. These machines are responsible for arranging the mail in a specified order. This order is given by the routes to where the objects are destined. Finally, the sorted objects are placed in containers and loaded into trucks to be transported to their corresponding CDP.

3. Transportation:

On this step, the mail is delivered by truck from a CPL to the multiple CDPs spread across the country.

4. Dispatch

Postal objects that arrive at CDPs can be sequenced by CP10, grouped by CP7, or completely unorganized. This step deals with merging incoming mail and assigning each object to a route, dependent on the object's destination.

5. Distribution (last-mile):

This step, commonly known as last-mile, consists of transporting mail from the CDP to its final destination. Each postman is assigned a route, a vehicle and postal objects at the start of each day and is responsible for delivering mail to every location with traffic and return to the depot.

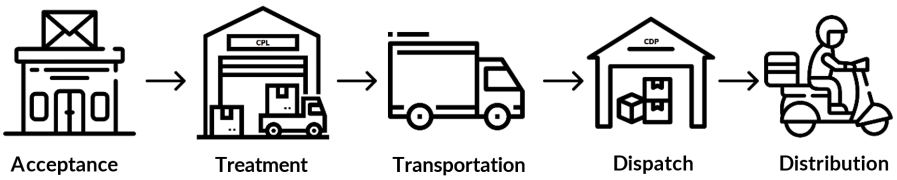


Figure 3.2: Distribution flow of CTT

3.1.4 Last-mile routing

,

Finally, it is worth discussing the way routes are set up and the two models that are in place in the last-mile routing phase. Firstly, routes are split into an active part and an idle part. Throughout the active part, the postmen are diligently delivering postal objects to their addresses. Alternatively, idle parts exist at the start and at the end of the route, where postmen are simply travelling from and to the CDP. Additionally, there is a separation between priority mail and non-priority mail. Priority mail is mail that must be delivered to its recipient in less than 3 business days upon being acknowledged by CTT. Non-priority mail, on the other hand, can take 3 or more business days to reach its final destination. It is worth mentioning that non-priority mail accounts for 82.4% of the total traffic and priority mail accounts for 17.6% [9]. To increase efficiency, the company has assembled two distribution approaches: the **XY** model and the ***in-trevo*** model.

In the XY model, every route is split into two: a fast half and a slow half. During a certain day, the locations in the fast half only receive priority mail and the locations present in the slow half of the route receive both kinds of mail. On the following day, the two halves swap roles, i.e., the fast half becomes a slow half and vice versa.

Likewise, the *in-trevo* model follows the same mechanism, but instead of splitting a route into two halves, a three-way division is made: with two fast sections and a single slow section. The slow section alternates between the three in a circular fashion.

Deciding on whether to use one model or another in a CDP is done manually and typically relates to the amount of traffic in that CDP. The simpler model, XY, is more prevalent in rural regions and the *in-trevo* model is more established in urban and suburban ones.

The rationale behind this process is the fact that non-priority mail needs not to be delivered every day and so by keeping this mail in the supply chain for a longer period

of time the company can save transportation costs and labour time, as locations with just non-priority mail are not visited if not required.

3.2 Problem specifications

To reach a feasible solution that can be applied in a real-world context, some characteristics were identified that uniquely characterize this problem.

3.2.1 Routing model

Firstly, the base of operations of the last-mile distribution step is a CDP. This means that each CDP operates independently of all others in its vicinity and is responsible for its own unique locations and its own VRP instance.

Secondly, the CTT has put forward the requirement that the routing solutions should be conservative. Routes are static and, for that reason, postmen are familiarized with the current routes and prepared to deal with unexpected problems. This means there should be some stability in the way routes are created. The solution to this problem, as suggested by the company, was the introduction of **segments**. A segment is, simply put, an indivisible section of a route and is composed of a set of CP7s. This means that routes are now defined as a sequence of segments instead of as a sequence of CP7s. Inevitably, with this requirement, arise some questions of implementation like: how to best create these segments, how large should they be and how many.

Another requirement is the balancing of workload between postmen. At the end of the day, a postman's working time should not differ too much from the working time of its colleagues. Since the majority of a postman's job is done in distribution, outside the CDP, routes assigned to postmen should all have a similar time to completion. Furthermore, there is a limit into how much time a postman can work, and it is set to 7.8 working hours a day.

In addition, since the goal of this dissertation is to propose a pilot project of how to apply the system to a CDP it focuses solely on the specific implementation in Alverca, for which the postal code is 2615.

Finally, the model should be flexible enough to adapt to future changes imposed both globally by the company and locally by each CDP.

3.2.2 SISMA

Beyond improving the model formulation, this work also focuses on integrating it with SISMA.

SISMA is a recent project carried out by CTT to estimate execution times of certain tasks and calculate productivity coefficients associated with the day-to-day operations at the CDP level. This tool arises from a necessity to replace the obsolete model, Unidade Equivalente de Correio (UEC), previously used by the company, which was outdated and

unable to correctly reflect the changes imposed by the paradigm shift verified during the last years in postal distribution.

In essence, SISMA is a dataset, stored in a spreadsheet format, that comprises several sheets with information about various metrics of operations and time standards. The most useful characteristic about it is the way it can output the labour time needed for all processes of the supply chain, given the number of postal objects. For example, by varying the number of objects of a certain kind, with SISMA, the company can analyse the impact this variation will have in labour time and adjust its workforce accordingly.

The way it does this is by having a correspondence between a flow and a task, and assigning a boolean value signifying whether a certain task is carried out during a certain flow. These boolean values are disposed in a matrix format inside the document (see A.5). In this context, a flow is a possible delivery procedure of a postal object. For example, the delivery of a small package using a D+1 standard to a PO box is a viable flow. These attributes are characteristics of a certain delivery and are described in further detail in Section 3.1.2 By aggregating the time taken by each task during a flow it can evaluate a reference time, which is estimated based on benchmarks defined in the postal deliveries' industry, and it signifies the amount of time that each delivery flow needs. This time includes the tasks partaken during the entire process, including both the processing and transportation steps. The way flows relate to each other can be viewed as a tree-like structure. A leaf of this tree corresponds to a flow with all attributes explicitly specified. Each leaf has a "parent" flow, which is obtained by omitting one of the attributes. The reference time of this "parent" flow is calculated by doing a weighted average using the weight of each "child" flow. These weights are based on the volume of traffic in each flow. After aggregating all flows, one can analyse the productivity of the deliveries made by each distribution centre. The productivity indices are obtained from the division between the real-time measured empirically, and the theoretical time needed based on reference studies.

Despite being a major help, SISMA is not foolproof. Its main disadvantage resides in the way it estimates the expected transportation time (the time spent travelling between delivery points), which the company considers somehow unrealistic. The reason for this is the way SISMA estimates travel times. It uses assumptions to indicate the average speed of vehicles, distance estimates for the routes and address density values, all the while disregarding important conditions like the terrain, speed limits, and the type of road a postman uses.

This shortcoming cannot be taken lightly, as transportation time occupies a large part of the labour time. In Alverca, for example, it accounts for 37.7% [9]. That's why solving this problem and integrating the proposed model with SISMA is one of the most important contributions of this dissertation.

3.3 Previous work

As mentioned previously, this dissertation will have as a starting point the work previously developed by Pedro Pereira, a former M.Sc. student. In his master thesis, the goal was to create a framework for simulating and calculating the optimal routes for postmen in different scenarios [26]. To do this, a variant of the VRP was created, using the three-index vehicle flow formulation, which was referred to as MVRP.

In this model, the goal was to minimize the number of postmen needed to cover all locations. It is a specialization of a regular VRP in two aspects. It added constraints to ensure that each route's duration doesn't fail to meet the minimum time a postman is allowed to work and that it doesn't exceed the maximum time a postman is allowed to work. In addition, to meet the conservative solution specification, constraints were put in place to make sure that each generated route had at least a percentage of the locations present in the routes already used by CTT.

For testing purposes, the proposed solution the CDP of Alverca and the CP7s that are in its distribution range were used.

Data extraction

To solve the optimization problem, some data needs to serve as input to define the model's parameters and constraints.

Firstly, the delivery locations are very important. These locations were provided by CTT and include 1018 CP7s and the depot. A problem arises due to the very large number of locations and the exponentially large number of variables and constraints generated. CPLEX, despite being one of the best solvers available, is unable to produce optimal solutions in less than 3 days of computation time using so many data points. This issue is dealt with in Section 3.3.

Secondly, the travel time between locations had to be estimated. For this, the best approach was to use a routing service to find the fastest path between two locations. Many services exist, but the one used was **Openrouteservice** [15].

Another essential datum is the time spent by a postman in each location. This data was calculated using some data provided by the CTT. The actual data provided was the daily average amount of mail delivered during February 2020, the percentage of total mail delivered in each CP7 during the same month, and the routes of each postman. In [26], these times were calculated using the number of daily objects delivered to each CP7 by multiplying the total amount of objects by the percentage delivered to it. These values were used to estimate a productivity coefficient for each route by dividing the sum of objects delivered per route by the average time a postman works. All the coefficients were averaged to produce a global productivity coefficient. Finally, to get the time taken to attend to each location, one just needs to divide the number of objects destined to each CP7 by the productivity coefficient.

Lastly, the model needs also the currently used routes, which were provided directly by the CTT.

Clustering

As mentioned in Section 3.3, a way to aggregate locations was needed for CPLEX to produce optimal solutions in a reasonable amount of time. As such, a clustering technique was applied to reduce the number of locations. A bisecting clustering algorithm using three different methods was chosen (K-Medoids, Agglomerative clustering, and K-Means). The K-Means approach ended up being the chosen option, for it generated the least number of locations (52) and because of that CPLEX was able to produce a result faster, using only 24 postmen.

Route optimization

The optimization process is portrayed in Figure 3.3. Starting with the data retrieval step, where the data provided by the CTT is manipulated and prepared for the next step. Next, the locations are clustered using a bisecting K-Means algorithm. After clustering, data has to be modified. The number of locations becomes the number of clusters plus the depot. The time taken to deliver mail in each cluster becomes the sum of the time taken to deliver mail in all locations belonging to that cluster. For the time taken to travel between each pair of clusters, the maximum time between any pair of locations in both clusters was used. In the third step, the solver comes into play, generating optimal routes and assigning them to a postman. These routes are an ordered set of clusters, not locations. To finalize the process, the MVRP is applied again, for each route generated and using the actual locations instead of the clusters. At the end of this process, the result is an optimized set of routes through every CP7 in the dataset.

Graphical user interface

Besides optimizing the routes created by way of solving a VRP, the previous work also focused on building a Graphical User Interface (GUI) to serve as a support tool for the company. It employed the **tkinter** [14] Python module, which allows for an easy way to build functional and lightweight GUIs.

The application developed required the user to input the locations file with the coordinates of every location and the duration matrix file with the time taken between every pair of locations. In addition, it required the user to parameterize the number of postmen available, the number of locations, and the minimum and maximum time a postman is allowed to work. It also had some configuration options. The user could choose to solve the problem using CPLEX (exact method) or a greedy search approach, using the K-Means or K-Medoids clustering methods and set limits on the time taken to produce solutions.

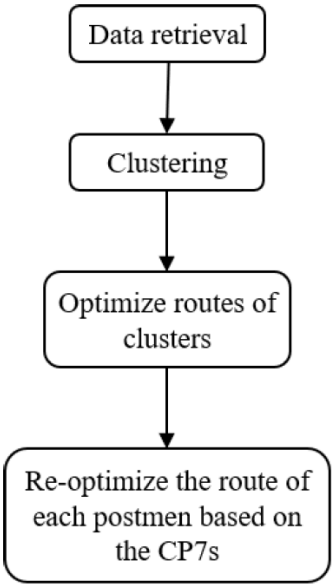


Figure 3.3: Optimization process steps (Pedro Pereira)

Results

To examine the results of the MVRP, a comparison was made between these and the current routes used by the CTT in Alverca's distribution centre. Alverca's CDP has 25 routes, therefore needing at least 25 postmen. However, when applying the MVRP the solution found needed only 24 postmen, which despite being only a 4% decrease translates to a considerable amount of capital saved in the long run. Other than the slight decrease in the number of postmen, the total routing time was also reduced by 11% [26]. Since the parameters provided to the model can greatly alter the final result, an analysis was made to compare the results by varying the time taken to travel between locations and the coefficient of objects delivered per hour.

4

Segmentation

The routing model is meant to be applied daily and generate routes taking into account the real traffic of each day. Routing individual CP7s would generate completely different routes from one day to the next and be too disruptive for postmen and CDP managers.

As such, the use of segments to compose routes, instead of CP7s, was a suggestion put forth by CTT to simplify the routing model without causing too much disturbance to the current state of operations. This chapter will describe, as thoroughly as possible, the steps undertaken and the decisions made to arrive at a feasible segmentation that can be applied in a real-world context. Furthermore, the chapter will focus on explaining how the expected times to complete a CP7 and a segment were reached, the algorithms and parameters used in the segmentation and the reasoning behind them. Finally, it will present the results and the visualizations made to analyse them.

4.1 Data retrieval

A segment is, in essence, an ordered set of CP7s that a postman will always complete uninterruptedly. How long a postman will take to complete a segment is not known *a priori*, and this variable depends on the amount of mail destined to that segment, the number of addresses that will be visited, and also the vehicle used. Since the first two variables are constantly changing and segments are not supposed to, it was decided to use reasonable estimates to determine the average time to complete a segment. This means that this segment time may change from one day to the next, but the routing step will make sure that the total length of a route doesn't surpass the legal and established limit.

The time per segment can be split into time travelling between CP7s and time spent delivering mail in a CP7. Furthermore, since a CP7 consists of a set of addresses, the time taken delivering mail in a CP7 can be divided into time travelling between addresses and time spent delivering mail in each address.

This hierarchical description can be formalized in the following manner (using Table 4.1 for guidance).

Equation 4.1 calculates the time taken to delivery mail in the CP7 p and this time

Table 4.1: Notation for describing the decomposition of times per segments and CP7s

A	Set of addresses (or CP10s)
R	Set of edges connecting individual addresses
P	Set of postal codes (or CP7s)
S	Set of segments
Q	Set of edges connecting CP7s
A^p	Set of addresses belonging to postal code p
R^p	Set of edges in the route that visits every address in p
P^s	Set of postal codes in segment s
Q^s	Set of edges in the route that visits every CP7 in s
T_d	Delivery time
T_t	Travel time

results from the addition of the time taken to delivery mail in each of p's addresses and the time taken to travel between these addresses.

$$T_d^p = \sum_{i \in A^p} T_d^i + \sum_{(i,j) \in R^p} T_t(i,j) \tag{4.1}$$

Likewise, equation 4.2 calculates the time taken to deliver mail in segment s. This time is obtained by adding the summation of the time taken to deliver mail in each CP7 in the segment (given by 4.1) and the total time spent travelling between CP7s.

$$T_d^s = \sum_{p \in P^s} T_d^p + \sum_{(k,l) \in Q^s} T_t(k,l) \tag{4.2}$$

The time travelling between two CP7s was calculated using the two closest addresses in both CP7s. Furthermore, as will become evident in Section 4.1.1, the time taken to deliver mail in the set of addresses in a CP7 p (first summation in Equation 4.1) was extracted directly from SISMA as there wasn't data regarding this time for individual addresses.

4.1.1 Estimating delivery times

SISMA has information about the delivery times of postal objects. The summary sheet (see A.6) contains information about the delivery times of each flow. The goal, however, is to estimate a delivery time for every CP7 and for this the following procedure was followed:

1. Use the traffic by CP7 sheet (A.4) to produce weightings for traffic in each CP7

SISMA has traffic data per CP7 that passed through the automated processing machines. This traffic, however, which does not constitute the entirety of traffic destined to each CP7 and can't be used directly. For this reason, this traffic data was only used to produce weightings for each CP7 by dividing the CP7's traffic value by

Table 4.2: Traffic data by CP7

CP7	MMS			RMS			Total
	Non-priority	Priority	Traceable	Non-priority	Priority	Traceable	
2600002	168	20	17	1	1	0	207
2600004	53	7	12	1	0	0	73
2600005	10	1	2	1	0	0	14
2600006	23	2	3	2	1	2	33
2600007	4	0	3	0	0	0	7

the CDP's total. The data is divided into six columns: three for the MMS machines and three for the RMS machines (for priority, non-priority and traceable mail).

This process basically translates Table 4.2 to 4.3, which have both been cropped for visualization. In Table 4.3, each column's percentages must add up to 100%.

Table 4.3: CP7 traffic weights

CP7	MMS			RMS		
	Non-priority	Priority	Traceable	Non-priority	Priority	Traceable
2600002	0.04%	0.04%	0.02%	0.01%	0.01%	0.00%
2600004	0.01%	0.01%	0.01%	0.01%	0.00%	0.00%
2600005	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%
2600006	0.01%	0.00%	0.00%	0.01%	0.01%	0.03%
2600007	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

2. Apply the mapping in Table 4.4 to correlate flows into one of the weightings

Using the percentages from the previous step, the real traffic data from the traffic sheet (see A.1) and the mapping in Table 4.4 it is possible to estimate how many objects of each flow a CP7 will receive on average.

Table 4.4: Mapping from flow to object scheme

Flow			Object scheme
Format	Traceability	Delivery standard	
LLE	w/o trace	D+1	MMS Priority
LLE	w/ trace	D+1, D+2	MMS Traceable
LLE	w/o trace	D+3, D+5	MMS Non-priority
SP, MP, P	w/o trace	D+1	RMS Priority
SP, MP, P	w/ trace	D+1, D+2	RMS Traceable
SP, MP, P	w/o trace	D+3, D+5	RMS Non-priority

3. Multiply SISMA's coefficients to the volume of traffic and add up the results

SISMA's summary sheet (see A.6) contains productivity coefficients for each flow in seconds per object. By multiplying the delivery time coefficient by the number of objects in each flow, the time needed to deliver the objects is obtained. Finally, to calculate the delivery time for a CP7, we just add up the delivery time values for every flow.

Table 4.5 shows the final result of this process. In particular, the last column contains an estimate of the delivery time for each CP7. This table contains only the first five CP7s and some columns were omitted.

Table 4.5: Delivery times table

CP7	w/o trace						w/ trace				Total	Delivery time [min]
	LLE			SP			MP		P			
	d+1	d+3	d+5	d+1	d+3	d+5	d+1	d+2	d+1	d+2		
Total	261.564	8501.518	773.225	155.481	180.016	16.373	151.879	48.348	111.372	25.488	11,848	4,406.58
2600002	0.099	3.594	0.327	0.021	0.012	0.001	0.000	0.000	0.000	0.000	4.23	0.83
2600004	0.035	1.134	0.103	0.000	0.012	0.001	0.000	0.000	0.000	0.000	1.41	0.30
2600005	0.005	0.214	0.019	0.000	0.012	0.001	0.000	0.000	0.000	0.000	0.27	0.07
2600006	0.010	0.492	0.045	0.021	0.024	0.002	0.042	0.013	0.031	0.007	0.85	0.65
2600007	0.000	0.086	0.008	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.12	0.03

4.1.2 Estimating travel times

For estimating the travel times in each CP7 CTT provided data at the CP10 level for the objects that passed through the automated processing machines in the month of January 2022. This data is important because for estimating the travel time in each CP7, one has to know which of its addresses will be visited and by what order a postman will visit them.

Once an object is recognized by the machine, we can assume that it will arrive in a CDP before dawn so that it is distributed the following day. This doesn't happen every time, but it's a reasonable simplification. Considering every object has a timestamp, we can aggregate this data and devise how much traffic the CDP had on a given date.

The simulation procedure for arriving at a travel time for a given date and CP7 was the following:

1. Get which CP10s had some traffic

Filtering the data by date and CP7 results in a list of objects with their destination CP10. Aggregating the objects by CP10 results in a collection of CP10 with its respective object count.

2. Estimating the route a postman will make in each CP7

After step 1, follows a simple TSP routing problem to determine the optimal path a postman should take, passing through every CP10 in the CP7. A TSP was employed instead of a VRP because the goal is to simply draw the best path passing through every CP10 and constraints like how much time these paths should incur should not modelled. In addition, since it is expected postmen will be rational to try and minimize their travel time in a CP7 without requiring optimality, the decision was made to solve this routing problem using a greedy search approach. Furthermore, the use of a greedy method is less computationally intensive.

The actual implementation required the ascertainment of several duration times between CP10s. To determine these, the **Openrouteservice** [15] tool was employed. In addition, to calculate the duration between two CP10s of the same CP7, the assumption was made to use the currently established vehicle responsible for the CP7.

3. Extracting the routing time required

Following the determination of the greedy optimal path, follows the summation of all its constituents, which results in an estimated travel time for each CP7. The TSP greedy search method finds a cycle passing through every CP10 but since after completing the CP7 a postman does not need to return to the first CP10 visited the largest constituent time in the cycle was subtracted from the total travel time.

This procedure was repeated for every day in January 2022 that had an overall traffic value of more than half the median traffic value for the month. The results were averaged to obtain a realistic estimate on the average travel time for the month. Table 4.6 shows these estimates, in seconds, for the first business week of January 2022.

Table 4.6: Travel times estimates for the first business week of January 2022

CP7	TravelTime (1/3/2022)	TravelTime (1/4/2022)	TravelTime (1/5/2022)	TravelTime (1/6/2022)	TravelTime (1/7/2022)	AverageTime
2600010		81.73	36.8	12.2	81.74	61.4445
2600011	551.8	648.69	643.65	640.8	712.16	569.2114762
2600014	49.59	67.61	60.44	62.23	65.92	64.13333333
2600015	19.79	40.13	25.89	36.34	42.72	29.16352381
2600016		17.71	18.6	16.82	10.27	14.93380952
2600017	48.77	53.84	60.02	60.02	54.11	51.30809524
2600018	92.68	82.04	105.99	115.55	87.51	83.61238095
2600019	24.35	48.09	48.84	26.55	52.84	42.4252381

4.2 Segmentation scenarios

Having delivery and travel time estimates for each CP7 facilitates the generation of segments because the only restriction is that they can be completed in a reasonable amount of time by a single postman.

This work focused on implementing two different scenarios for segment generation. The first scenario was to build a hierarchical clustering algorithm that utilizes the complete set of CP7s in the CDP and recursively bisect it into smaller and smaller segments until every segment can be completed in no more than a certain time limit. The second, more conservative, was to utilize the currently established routes and partition them into smaller subsets of CP7s until, again, no segment takes more than a certain amount of time to traverse. The second alternative is more conservative because it implies less radical changes to distribution. For example, because CP7s are currently assigned to a route, they are visited using that route's vehicle. If two CP7s of different routes were to be merged in the same segment, a decision would have to be made as to which vehicle would supply them.

In both algorithms, the distance metric used between data points (CP7s) was the duration time to go from one CP7 to another. Likewise, they both receive a parameter which indicates the maximum segment size (i.e. the maximum time a segment is allowed to be completed in). The values used for this parameter are: 60, 90, 120, 150 and 180 minutes.

4.2.1 Bisecting *k*-medoids

The idea behind this algorithm is to provide the optimal segmentation, disregarding the current routes. The algorithm's starting point is the complete set of CP7s in the CDP. From here it selects a segment with a time to completion greater than the given parameter (the collection of CP7s in this case), and it applies a bisecting k-medoids clustering to it. From this step result two distinct segments in terms of CP7s. The forementioned procedure is repeated for each of these segments until no segment for which the estimated time to completion is larger than the parameterized limit exists. Algorithm 1 is a pseudocode implementation of this scenario.

The inner k-medoids algorithm receives a precomputed travel time matrix between CP7s, and at each step, the selection of the initial medoids is made using the two data points furthest in travel time from each other.

The decision to use a hierarchical algorithm instead of a simple clustering method was because most clustering methods take as parameters the number of clusters to create and don't allow for a custom duration matrix between data points to be provided. The inner clustering method used was k-medoids instead of k-means because in k-medoids centroids of clusters are original data points, and this allows the use of a precomputed distance matrix (durations between CP7s).

Algorithm 1 Bisecting k-medoids

```
segments ← BisectingKMedoids([CP7sCollection])
procedure BisectingKMedoids(S)
    if exists some segment larger than time limit and with more than 1 location then
        find that segment s ∈ S
        s1, s2 ← k-medoids(data = s, clusters = 2)
        return BisectingKMedoids(S \ {s} ∪ {s1, s2})
    end if
    return S
end procedure
```

4.2.2 Bisecting routes

Alternatively, an algorithm to extract segments from the current routes is proposed in this subsection. In this case, the procedure starts from a set of routes and selects the first that surpasses the parameterized time limit. It lays out the sequence of CP7s on a cutting board and determines the largest duration jump from one CP7 to the next. The sequence is split where this largest arc appears, and two subsegments are created. With them, the procedure repeats recursively until every segment's time to completion is valid. Algorithm 2 provides a pseudocode implementation of this method.

Algorithm 2 Bisecting current routes

segments ← *BisectingRoutes*(*CurrentRoutes*)
procedure BISECTINGROUTES(S)
  if exists some segment larger than time limit **and** with more than 1 location **then**
    find that segment $s \in S$ and get its locations $\{p_0, ..., p_i, p_{i+1}, ..., p_{|S|-1}\}$
    find i such that $distance(i, i+1)$ is maximum
    split s into $s1 = \{p_0, ..., p_i\}$ and $s2 = \{p_{i+1}, ..., p_{|S|-1}\}$
    return BISECTINGROUTES($S \setminus \{s\} \cup \{s1, s2\}$)
  end if
  return S
end procedure

4.3 Results

From the implementation of these scenarios resulted two tabular data structures stored in a spreadsheet: one for the specification of the segments created (which CP7 compose them) and one with information about each segment. Tables 4.7 and 4.8 are snapshots of each of these two files, respectively.

In Table 4.7, each row corresponds to a CP7. The first two columns identify the segment in which the CP7 was inserted and the order by which it will be visited. The third column has information about the time taken to travel to the next CP7 in the segment. Columns 4 and 5 have information about the routes the CP7s are currently a part of. In particular, the route identifier and the sequence number of the CP7 in the route. The second to last column is the algorithm used in the segmentation, and the last one is the maximum segment size parameter.

A segment is done uninterruptedly by a single postman using only one vehicle. In the case where a segment is composed of CP7s that are assigned to different types of vehicles, there is a decision to be made as to which vehicle should be used in this segment. In these cases, the vehicle used in the most CP7s seems the logical choice.

Table 4.7: Segments specification results file

SegId	SegSeq	TimeToNextLoc	RouteId	RouteSeq	CP7	Lat	Lon	DeliveryTime	TravelTime	Time	Algorithm	MaxSegmentSize
0	0	13.52	MC100	21	2615686	38.91796	-9.02806	185.6625	15.9005	201.563	BisectingKMedoids	60
0	1	94.14	MC100	22	2615683	38.91783	-9.02866	61.59787	18.13053	79.7284	BisectingKMedoids	60
0	2		MC100	24	2615706	38.9188	-9.03006	18.07985	1.8182	19.89805	BisectingKMedoids	60
1	0	6.07	UV070	30	2615737	38.91689	-9.02635	294.6412	18.09048	312.7317	BisectingKMedoids	60
1	1	24.42	UV070	31	2615687	38.91686	-9.02674	60.26986	3.763333	64.03319	BisectingKMedoids	60
1	2	6.72	UV070	35	2615704	38.91731	-9.02704	301.6779	24.5919	326.2698	BisectingKMedoids	60
1	3	8.32	UV070	34	2615690	38.9175	-9.02678	34.18931	0.66	34.84931	BisectingKMedoids	60
1	4	22.62	UV070	33	2615691	38.91733	-9.02639	66.34737	0.715	67.06237	BisectingKMedoids	60
1	5	67.59	UV070	32	2615663	38.91783	-9.02557	70.82164	9.3565	80.17814	BisectingKMedoids	60
1	6		UV070	44	2615675	38.91872	-9.02497	807.7401	53.16333	860.9035	BisectingKMedoids	60

In Table 4.8, each row relates to a segment. The first column is the segment identifier, the second is the number of CP7s it contains, and the last two are, like the previous file, the algorithm and maximum segment size parameter. The time attributes are in seconds.

The results of all test runs, using the two algorithms and the five parameters, were aggregated in a single file for simplicity when importing the results to Tableau and a data visualization tool that was used to analyse the segmentation solutions.

Table 4.8: Segments metrics results file

SegId	Size	DeliveryTime	TravelTime	TotalTime	Algorithm	MaxSegmentSize
0	3	265.3403	143.5092	408.8495	BisectingKMedoids	60
1	7	1635.687	246.0805	1881.768	BisectingKMedoids	60
2	2	1660.136	157.4425	1817.579	BisectingKMedoids	60
3	2	161.7971	25.60371	187.4008	BisectingKMedoids	60
4	8	1880.085	523.4459	2403.531	BisectingKMedoids	60
5	5	1221.157	362.3598	1583.517	BisectingKMedoids	60
6	9	2377.876	877.1209	3254.997	BisectingKMedoids	60
7	2	109.0339	64.70519	173.7391	BisectingKMedoids	60
8	6	1563.675	576.8415	2140.517	BisectingKMedoids	60
9	10	3046.527	457.6	3504.127	BisectingKMedoids	60
10	19	1378.935	1307.988	2686.923	BisectingKMedoids	60

4.3.1 Segmentation visualization workbook

After a successful segmentation follows a detailed analysis of the results, methods, and parameters. For this analysis, the suggested approach was to use Tableau for data visualization. Tableau allows its users to import data stored in a variety of formats and intuitively create helpful visualizations to review the data and reach fitting conclusions. This subsection attempts to showcase the main visualizations in the workbook created for this effect.

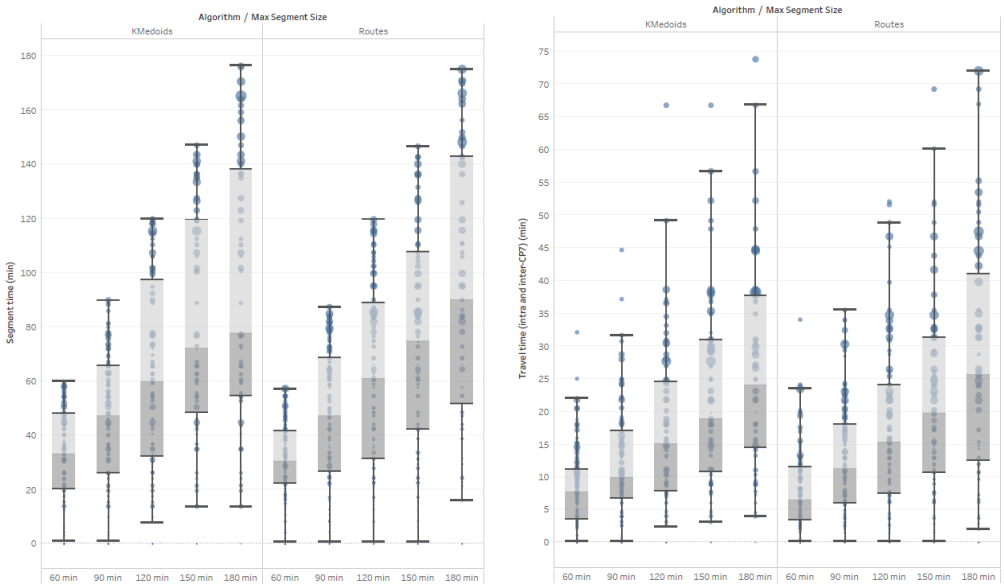
The first visualization, shown in Figure 4.1, is a simple overview of the segments created by each method and parameter. From it, we can directly observe that the larger the maximum time limit per segment, the fewer segments are created and the larger the average time and number of CP7s per segment. It is also the case that the first algorithm (4.2.1) outputs fewer segments than its counterpart. This can be explained by the fact that it better optimizes segments. Using the second method results in more segments because the starting point is already a collection of segments and the procedure has to bisect them further to make sure they all fall under the parameterized limit.

	BisectingKMedoids					BisectingRoutes				
	60 min	90 min	120 min	150 min	180 min	60 min	90 min	120 min	150 min	180 min
Average time per segment	30.1	42.5	59.3	76.8	91.0	27.5	43.2	58.3	69.8	88.2
Maximum time per segment	62.9	90.0	119.9	147.1	176.5	62.9	87.5	119.6	146.7	175.2
Number of segments	163.0	118.0	86.0	67.0	57.0	180.0	118.0	89.0	75.0	60.0
Average number of CP7s per segment	4.8	6.6	9.0	11.6	13.6	4.3	6.6	8.7	10.4	13.0

Figure 4.1: Segmentation methods and parameters overview

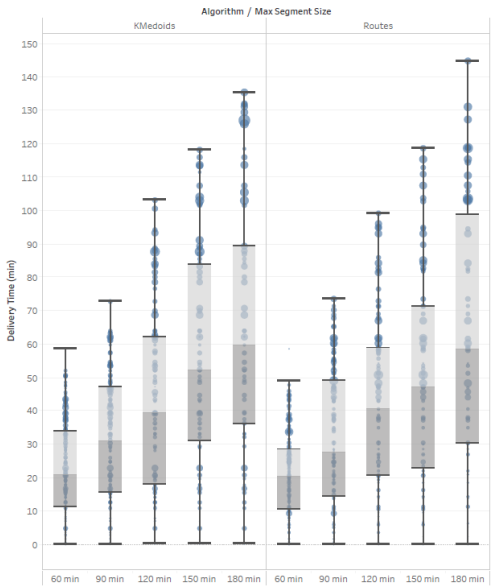
The next visualization is a collection of hybrid dot and box plots applied to each segmentation scenario. In Figure 4.2, each scenario contains a clickable dot for every segment that shows that segment's information. The left axis reflects the respective time dimension.

More important than a simple table is the ability to visualize segments in a geographic map. This realization led to the addition of another visualization, seen on the right side of Figure 4.3, which can interactively show a solution, selecting the algorithm and the parameter, with all CP7s' geographical positions overlaid on the region of Alverca. Each colour corresponds to a segment, and so CP7s in the same segment have the same colour



(a) Total segment time plot

(b) Travel time plot



(c) Delivery time plot

Figure 4.2: Times per segment plots

and a path connecting them. The left-hand side of Figure 4.3 shows a bar chart of the estimated completion time of each segment and a floating container for selecting the solution. The bars are decomposed into delivery time (in blue), intra-CP7 travel time (in orange), and travel time between CP7s (in red).

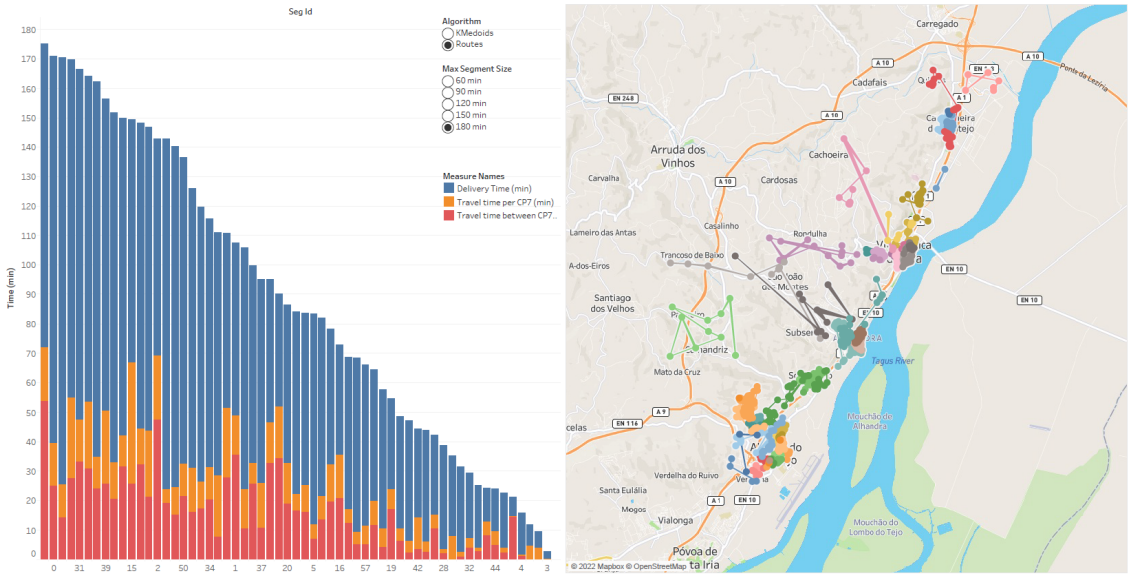


Figure 4.3: Segmentation solution analysis dashboard

Other than inspecting individual solutions, it would be interesting to know the total segmentation time of each solution. This is what is shown in Figure 4.4. Analysing it, clearly smaller values for the maximum segment time parameter generate smaller working times. While the intra-CP7 travel and delivery times remain constant independently of the scenario, the travel time between CP7s increases as a function of the maximum segment size. This is because smaller segments have less CP7s, and so this travel time diminishes.

In principle, a maximum segment size of 0 would result in a segmentation equal to the entire collection of CP7s. These results would completely defeat the purpose of segmentation, and so it is undesirable. Because of this and because the routing time between segments contributes a lot more to the total working time of postmen, evaluating segments in this way is not a rational approach.

Finally, Figure 4.5 is a helpful side-by-side comparison between the segments generated using the Bisecting Routes algorithm (on the left) and the current routes (on the right). This visualization only makes sense using the second method because the first does not use the current routes as a starting point.

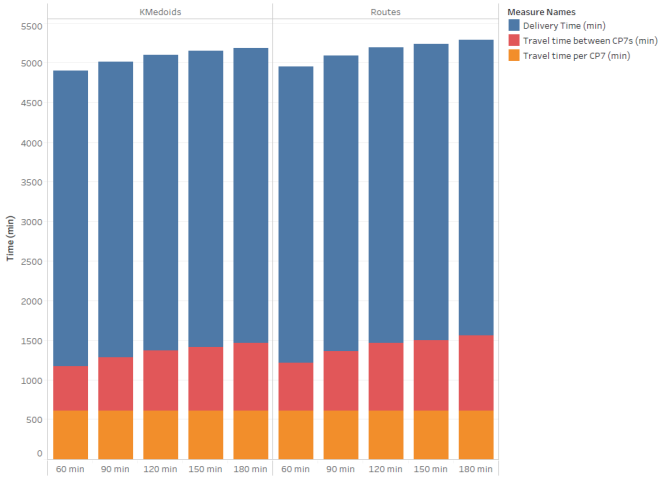


Figure 4.4: Total segmentation times chart

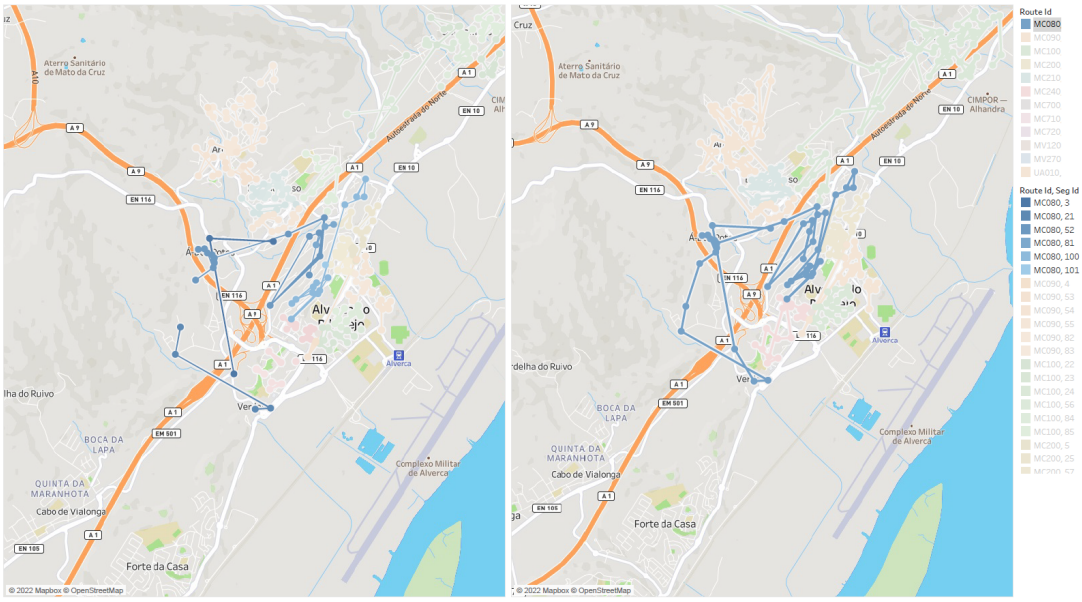


Figure 4.5: Segmentation comparison to current routes' dashboard

5

VRP FORMULATION

The mathematical modelling of a VRP and its translation into machine-readable code is what allows a problem as complex as this one to be solved by optimization packages like CPLEX. This task, of adapting and combining the many formulations that already exist in the literature to arrive at a model that can reflect the process of the last-mile routing at CTT cannot be taken lightly.

The work that precedes this project is of major help to the problem because it already brought to light what a working formulation would look like. Like the previous one, this model is also built on top of a three-index vehicle flow formulation (see Section 2.1.1) using the MTZ formulation for the subtour elimination problem (see Section 2.1.2). This time, however, some adaptations have been made to refine it and decrease the computational effort required to solve it.

The adapted formulation, with its parameters and variables specified, for this improved model, which will be referred to as CTTVRP, is the following:

Parameters

n: total number of locations including depot

p: total number of available postmen

t_{min}: minimum working time of a postman

t_{max}: maximum working time of a postman

t_{ij}: time taken to go from location i to location j

w_i: time taken to deliver mail in location i

Sets

K: set of postmen, $K = \{1, ..., p\}$

N: set of locations, $N = \{1, ..., n\}$

N_c: subset of locations without the depot, $N_c = \{1,...,n\}$

A: set of admissible arcs, $A = \{(0,1),...,(n-1,n)\}$

Variables

$$z_k = \begin{cases} 1 & \text{if postman k is used} \\ 0 & \text{otherwise} \end{cases}, k \in K$$

$$x_{ijk} = \begin{cases} 1 & \text{if postman k visits location j after i} \\ 0 & \text{otherwise} \end{cases}, i,j \in N, k \in K$$

u_i: rank order in which location i is visited, $i \in N$

The formulation is as follows

$$minimize \sum_{i \in N} \sum_{j \in N \setminus \{i\}} \sum_{k \in K} t_{ij} x_{ijk} \tag{5.1}$$

subject to the following constraints

$$\sum_{k \in K} \sum_{j \in N_c} x_{1jk} \leq p \tag{5.2}$$

$$\sum_{j \in N_c} x_{1jk} = z_k \quad \forall k \in K \tag{5.3}$$

$$\sum_{i \in N_c} x_{i1k} = z_k \quad \forall k \in K \tag{5.4}$$

$$\sum_{j \in N_c \setminus \{i\}} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in N_c \tag{5.5}$$

$$\sum_{i \in N_c \setminus \{j\}} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in N_c \tag{5.6}$$

$$\sum_{i \in N} \sum_{j \in N \setminus \{i\}} (t_{ij} + w_i) x_{ijk} \leq t_{max} \quad \forall k \in K \tag{5.7}$$

$$\sum_{i \in N} \sum_{j \in N \setminus \{i\}} (t_{ij} + w_i) x_{ijk} \geq t_{min} z_k \quad \forall k \in K \tag{5.8}$$

$$\sum_{i \in N \setminus \{j\}} x_{ijk} - \sum_{r \in N} x_{jrk} = 0 \quad \forall k \in K, \forall j \in N_c \tag{5.9}$$

$$u_i - u_j + n * x_{ijk} \leq n-1 \quad \forall i \in N_c, \forall j \in Nc, i \neq j, \forall k \in K \tag{5.10}$$

$$x_{ijk} = 0 \quad \text{if } (i,j) \notin A \quad \forall i \in N_c, \forall j \in N_c, \forall k \in K \tag{5.11}$$

$$x_{ijk} \in \{0,1\}, z_k \in \{0,1\}, i \in N, j \in N, k \in K \tag{5.12}$$

Unlike previously, the objective function 5.1 now focuses on minimizing the travel time needed for postmen to cover all the segments. Constraint 5.2 ensures that the number of paths leaving the depot is not greater than the number of postmen (because each postman has to leave the depot only once in the morning and return during the afternoon). Constraints 5.3 and 5.4 guarantee that each working postman leaves from and arrives at the depot only once. Constraints 5.5 and 5.6 ensure that each segment is visited only once. Constraint 5.7 guarantees that the routing time of a postman (from the moment it leaves to when it arrives at the depot) doesn't exceed a maximum time limit. Similarly, constraint 5.8 guarantees that a postman's routing time surpasses a minimum time limit. Constraint 5.9 makes sure that each segment visited by a postman k has to be connected to some other segment visited by k. Constraint 5.10 is the MTZ subtour elimination constraint. Constraint 5.11 is the constraint that guarantees that the solution only contains the arc between two segments i and j if that arc is an element of A, the set of admissible arcs. The inclusion of this constraint greatly diminishes the problem size and the computational time needed to reach a solution, as further explained in Section 5.1.

Objective function

The change introduced to the objective function comes from a weakness identified in the previous solutions, which was the fact that postmen would often have to make irrational and unnecessary journeys from one end of the map to another opposite to it, only to go back to where they just were. The problem with using the number of postmen as the optimization variable is that solutions will disregard the distances travelled and the time spent by these working postmen. For example, suppose the problem we are solving contains 10 segments and 5 postmen. Additionally, suppose that all 10 segments can be visited in a single day by just one postman. By having the objective function be the minimization of the number of postmen, the solver will output any route that this postman can make, as long as it visits every segment. This is obviously a problem because there is a multitude of routes passing through all 10 segments, but not all of them are optimal in saving costs. The same logic applies when we increase the problem size. As such, the proposed approach is to change the objective function to minimize the total travel time of the entire workforce.

Tables 5.1 and 5.2 present the results obtained before and after this change to the objective function. The tests were performed with the same exact segments in both scenarios, and the only difference was the objective function. The tables show the average results over five consecutive test runs. Analysing these results, it is clear that there was a decrease in the routing time of the solutions. In addition, there does not appear to be a large increase in the number of postmen used in the solutions.

Table 5.1: VRP solution details minimizing the number of postmen

Segments count	5	10	15	20
Routing time (minutes)	2650.8280	4082.7600	5445.556	6358.6760
Solve time (seconds)	0.0376	0.3282	0.500	6.0124
Postmen used	1	2	2	3

Table 5.2: VRP solution details minimizing the total travel time

Segments count	5	10	15	20
Routing time (minutes)	2609.5240	3920.1420	5080.7600	5717.3900
Solve time (seconds)	0.0406	0.1998	1.0312	5.3342
Postmen used	1	2	2	3

5.1 Improvements

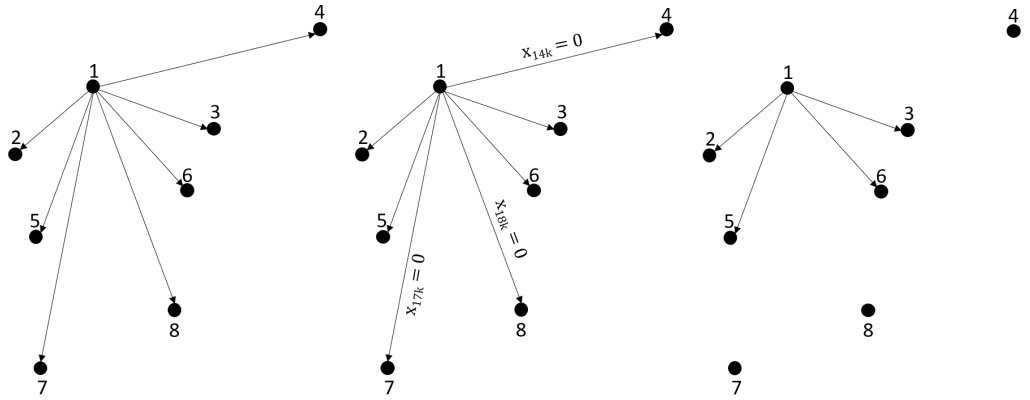
The main problem that comes with solving a large VRP to optimality is the computational effort required to do so. Even top-level solvers like CPLEX, using state-of-the-art branch-and-cut techniques, often struggle to find the optimal solution to problems with many locations in a reasonable amount of time. This is why, for example in the previous work, a clustering method had to be applied to reduce the number of locations used and, in turn, the problem size. The problem with this approach is that solutions are worsened by this clustering step because it affects the data granularity. To avoid this, two kinds of changes to the model are discussed in this section that intend to greatly reduce the problem size without affecting the quality of the solutions. The motivation behind both approaches is the realization that a postman will probably never need to travel from one segment on the map to another diametrically opposed to it without passing through an intermediate one. This means that the arc between the two opposite segments shouldn't need to be considered in the solution space.

1. Using constraints to limit the x_{ijk} variables admissible in solutions

This first method introduces a new constraint to the model (Constraint 5.11). This constraint forces all the values of x_{ijk} variables to become 0 if the arc (i, j) is not a part of the admissible arcs set A. The problem of defining this set is discussed in Section 5.1.1.

2. Limit the x_{ijk} variables generated by the model

This alternative limits the x_{ijk} variables that are generated when the model is defined by only creating the variables in which the arc (i, j) is an element of A.



(a) No optimization (b) Improvement method 1. (c) Improvement method 2.

Figure 5.1: Approaches taken to limit problem size applied to a single segment (k=4)

5.1.1 Defining the admissible arcs set

The approach taken to define A is the following: for each segment i other than the depot, consider only its k-nearest neighbours to define the set of arcs (i, j). There is obviously only one exception to this rule, which is the case of the depot. Every arc that originates from or arrives at the depot must be considered by the model. When i is 1, all arcs $(i, 2), ..., (i, n)$ have to be elements of A (equivalently, when j is 1).

This simplification greatly tightens the solution space because before with n segments we can imagine a set A that would contain all arcs in the complete digraph of N (the set of all segments). This set would have a size of $n(n-1)$. By defining A in this way, its cardinality becomes $2(n-1)+k(n-1)$. Since k can be expected to be much smaller than, n this results in a massive reduction in the number of x_{ijk} variables inside the solution space.

Figure 5.1 illustrates what this behaviour would look like for a segment i that is not the depot and $k = 4$. In Subfigure 5.1(a) all arcs are defined. In Subfigure 5.1(b) all x_{ijk} variables are created but half of them are preconditioned to be 0. In the last, Subfigure 5.1(c), only the arcs $(1, j)$ where j is one of the 4 closest neighbours of i are rendered.

Finding the best value for k

The decision on which value of k should be used in a particular instance is not one to be taken lightly. If k is too small, the ultimate optimal solution might not fall in the diminished solution space. If it is too large, then the benefits of the optimizations are not reaped as much.

It seems intuitive that the more segments exists, the larger k should be. This is why, the heuristic chosen to determine k was $k = \lfloor \sqrt{n} \rceil$, where n is the cardinality of N (the set of segments). For example, if N were to contain 150 segments, the value for k would be set to 12. While this approach is not a theoretical one, it was employed because its use

has seen good results in practice and because it satisfies the strong consistency condition for selecting k [1].

5.1.2 Improvement techniques comparison

For comparing the two forementioned techniques and evaluating the increase in performance relative to the previous model, it is important to analyse the results on two fronts: the solving time needed for CPLEX to output the optimality certificate and the impact these optimizations have on the value of the solution.

With this in mind, an experiment was put forward to see how the three methods compare as the problem size (number of segments) increases. Several VRP instances were defined and solved using CPLEX on a personal machine. The locations, i.e. segments, used in each VRP were sampled from the segmentation process.

Figure 5.2 shows the time (in seconds) needed to solve an instance of a problem using each solving method.

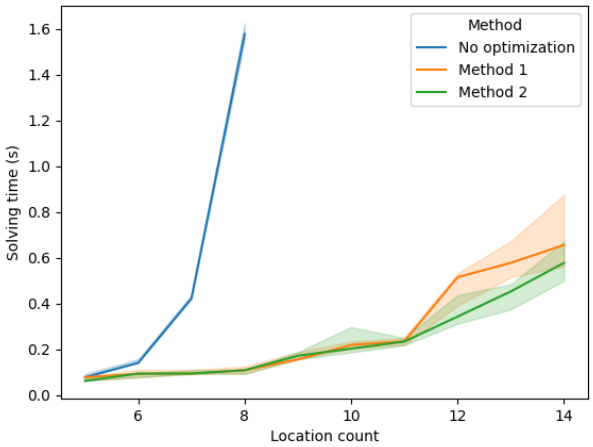


Figure 5.2: Improvements solving time comparison

For this experiment, the set of locations used was the same for each location count and five consecutive tests were performed for each location count and method to produce a median time. The shaded region corresponds to the 95% confidence interval around the median. The approach without reducing the problem size was stopped after a location count of just 8 because the solving time rapidly exploded upwards.

From 5.2, we can conclude that there was a large decrease in the time needed to solve the VRP after applying the improvements in 5.1. Furthermore, the second method, i.e. generating only the variables x_{ijk} that belong to the k-neighbourhood of i, saw the best results and produced solutions faster than the first method.

Other than evaluating the performance improvement, it is also important to evaluate the decrease in the quality of the solution after making these improvements.

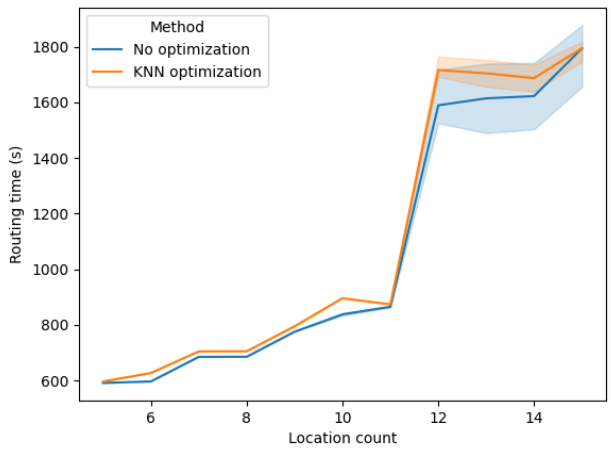


Figure 5.3: Optimization routing time comparison

Figure 5.3 shows how the quality of the solution is affected by the use of this k-nearest neighbour optimization. The experiment was run up to 15 locations only because solving a VRP with more than this number of locations became computationally too expensive.

As expected, there is an increase in routing time with the problem size pruning. However, the argument made herein is that the decline in quality is compensated by the increase in performance. This decline is not significant enough and there is reason to believe, although it cannot be proven empirically, that for even larger problems the solution quality decrease is even less relevant.

6

Experimentation and solution analysis

The analyses and comparisons made in this chapter attempt to reach definitive conclusions on the optimal segmentation scenario and routing model. For each analysis, there is a detailed explanation of how the results were attained.

In addition to this, the solutions that were produced by this work are also compared to the currently established routes and the routing solution of the previous work.

6.1 Experiments overview

The experiments in this chapter were conducted by having CPLEX (v20.1.0) solve the model with the segments created by each scenario. The CDP used in these experiments was Alverca, for which the CP4 is 2615.

The execution itself was performed in Turing [1], a workstation used by the Department of Mathematics at FCT for solving computationally difficult problems, like this one.

The model requires one to define the time taken to go from any segment to every other segment. Determining this time was performed by calculating the minimum of time required to go from every CP7 in one segment to any other CP7 in the other segment, using the vehicle assigned to both segments. Due to the fact that a segment is supplied by only one vehicle and that these vehicles have to be determined before solving the VRP, it should not be possible to connect two segments that are supplied by a different vehicle type in the same route. This realization leads to the question of how to best model this scenario.

On one hand, solving a single VRP using all vehicles means that in the duration matrix between every pair of segments, the time required to go from two segments supplied by the same vehicle type should be calculated using the shared vehicle type. Additionally, whenever two segments are supplied by a different vehicle type, the time between them was set to be infinite, for they should never be connected. There is an exception in case

[1] 2×CPU: AMD EPYC™ 7702 (64 Cores 256MB Cache, 2.0GHz to 3.35GHz GHz), 512GB RAM @ 3,200GHz

of the depot (or CDP). The depot isn't assigned a vehicle, and so it has to be possible for a postman to go from the depot to any other segment. The time between the depot and any other segment, and vice-versa, is estimated using the vehicle profile of this other segment.

Alternatively, the VRP with the total set of segments can be divided into smaller VRPs, one corresponding to each vehicle type. In the case of Alverca, three sets of routes exist: walking routes, scooter routes and driving routes. This would lead to three distinct VRPs with its own exclusive set of segments to supply. The only segment in common between the three would be the depot, which is the starting point for all three.

At first glance, it is not clear which approach is better, so this chapter contemplates both approaches.

The estimated time between CP7s was determined with Openrouteservice, the routing service used throughout this project.

There is one limitation common to these approaches, which is the fact that postmen in walking routes can use public transportation to go from the depot to their starting segment. Up-to-date public transit services are very limited outside large metropolitan areas and while some data sources exist that include routes by Rodoviária de Lisboa, the public transport bus service inside Alverca, there wasn't any data regarding Boa Viagem, which has routes passing through Vila Franca de Xira, also within the Alverca's delivery zone.

Finally, the parameterization of the model was set to use a minimum time of 2 hours and a maximum of 6.5 hours for each route. The number of postmen corresponds to the actual size of the workforce in Alverca (28 postmen). The solutions were obtained with the k-nearest neighbours optimization that saw best results in Section 5.1.2. Every segmentation scenario (combination of algorithm and maximum segment size) was tested for comparison.

6.2 Segmentation solutions

Given these assumptions, the first experiment was meant to compare routing solutions using the segments created by the segmentation step without partitioning the VRP into three subproblems (one for each transportation mode). Table 6.1 shows some metrics about the routing solutions found by CPLEX after one hour of solving time. In just one hour, CPLEX was unable to find any solution for problems with more than 75 to 80 segments, and so for segmentation scenarios that surpass this number of segments the solution metrics are empty.

For those it was able to find routing solutions, the best performing actor (using the working time metric) was the segmentation using the bisecting k-medoids algorithm (see 4.2.1) and a maximum time per segment of 150 minutes. On the other hand, there was a segmentation scenario (bisecting k-medoids with a maximum segment size of 180 minutes) in which a less number of routes were produced. The decision on whether it is

Table 6.1: Routing results by segmentation scenario after 1h of solving time

Algorithm	Maximum segment size (min)	Segments	Routing time (s)	Working time (s)	Inter-segment distance (m)	Routes	Gap
Bisecting k-medoids	60	163	-	-	-	-	-
	90	118	-	-	-	-	-
	120	86	-	-	-	-	-
	150	67	56 276	364 844	220 527	18	46.6%
	180	57	55 709	367 017	211 642	17	36.7%
Bisecting Routes	60	180	-	-	-	-	-
	90	118	-	-	-	-	-
	120	89	-	-	-	-	-
	150	75	57 245	371 302	221 333	18	80.9%
	180	60	56 984	374 357	216 360	18	74.3%

preferable to have more postmen working fewer hours or fewer postmen working more hours in total has to be made by CTT. Another useful observation to make is the fact that the bisecting k-medoids algorithm performs better than the bisecting routes. This has to do with the fact that in the former, segments are more optimized.

The routing time metric is the time travelled by postmen between segments and including the depot. The working time metric is the summation of all segments' times with the solution routing time between them. The distance metric is the length of road travelled by postmen between segments (it does not include the intra-segment distance) and the gap values indicate the difference between the incumbent solution found after one hour and the best bound (the best objective value a solution could have).

As an alternative to using all the segments in the same VRP, the decision was made to partition the problem into three distinct subproblems, one for each transportation mode. In this case, CPLEX was given just 20 minutes for each subproblem to output the best solution it finds. As shown in 6.2, using this approach, CPLEX is able to solve almost all the smaller subproblems. However, if we compare the solutions produced in table 6.1 with these results, we see that the total working time of the three VRPs is greater than the single working time of the VRP containing all segments and that the number of routes created is also greater.

Regarding Table 6.3 (bisecting routes), the results were very similar to Table 6.2. There is a tendency for the working time to increase as the number of segments decreases and they become more dissimilar. However, with a maximum segment size of 180 minutes, the solution returned by CPLEX was better than with smaller maximum segment sizes.

Globally, the best segmentation scenario is still up to debate. On one hand, the working time of the workforce should be minimized. On the other, if some solution is found that requires fewer postmen than the minimal working time solution, should it be applied instead?

Furthermore, comparing the two solutions of the bisecting k-medoids method from Table 6.1 we can suppose that since a workforce of 17 postmen can work 367 017 seconds in a day (averaging 6 hours per postman), then these same postmen should also be able to work only 364 844 seconds. This means one could try and reduce the maximum number

Table 6.2: Routing results with partition by mode of transport (bisecting k-medoids)

Maximum segment size (min)	Transportation mode	Segments	Routing time (s)	Working time (s)	Routes	CPLEX gap
60	Walking	52	20 570	116 334	6	59.7%
	Scooter	-	-	-	-	-
	Van	19	7 036	46 178	3	0%
	Total	163	-	-	-	-
90	Walking	32	14 944	106 410	6	21.8%
	Scooter	68	37 880	209 434	11	70.9%
	Van	18	6 747	44 544	3	0%
	Total	118	59 571	360 388	20	-
120	Walking	23	16 313	111 627	6	0%
	Scooter	52	44 170	209 768	11	73.5%
	Van	11	6 906	52 106	4	0%
	Total	86	67 389	373 501	21	-
150	Walking	17	15 633	114 272	7	0%
	Scooter	42	43 058	210 356	11	26.9%
	Van	8	6 105	48 735	3	0%
	Total	67	64 796	373 363	21	-
180	Walking	14	15 068	112 531	6	0%
	Scooter	36	45 009	225 918	11	30.3%
	Van	7	5 270	38 206	3	0%
	Total	57	65 347	376 655	20	-

Table 6.3: Routing results with partition by mode of transport (bisecting routes)

Maximum segment size (min)	Transportation mode	Segments	Routing time (s)	Working time (s)	Routes	CPLEX gap
60	Walking	57	30 258	135 207	7	18.5%
	Scooter	-	-	-	-	-
	Van	28	13 205	54 599	4	0%
	Total	180	-	-	-	-
90	Walking	37	19 346	127 429	7	14.7%
	Scooter	64	34 180	189 173	9	30.6%
	Van	17	10 150	52 630	4	0%
	Total	118	63 676	369 232	20	-
120	Walking	29	26 818	136 663	8	6.0%
	Scooter	46	34 682	193 014	10	33.7%
	Van	13	9 389	52 637	3	0%
	Total	88	70 889	382 314	21	-
150	Walking	25	26 119	136 668	7	5.7%
	Scooter	38	30 291	190 129	10	29.4%
	Van	11	4 920	48 591	3	0%
	Total	74	61 330	375 388	20	-
180	Walking	22	16 104	127 283	6	0%
	Scooter	28	28 740	190 796	10	10.5%
	Van	10	4 100	48 236	3	0%
	Total	60	48 944	366 315	19	-

of postmen and execute the VRP a second time changing this parameter or assess whether two routes from the optimized solution can be joined into a single route.

These metrics are useful for evaluating segmentation scenarios, but tell us nothing about the effect the model can have in improving the current state of operations.

6.3 Comparison to the current routes

For comparing the solutions obtained by way of the routing procedure, some metrics about the current routes were estimated. These metrics were calculated using the CP7 times estimated by the segmentation step and the order by which they are traversed in the current routes. The data used for the segments' times was the same throughout the experiments.

Table 6.4 shows the total working time and distance travelled by the postmen in a day using the average delivery and travel times per CP7 in Alverca.

Table 6.4: Total metrics of current routes

Routes	25
Total working time (s)	380889.16
Total distance travelled (m)	638088.8

Comparing these results with Table 6.1, we can see that there is a clear decrease in the working time by postmen during the last-mile stage. In the best case scenario, using the segmentation from the bisecting k-medoids algorithm with a maximum segment size of 150 minutes, there is a 4.21% decrease in labour time. A 4.21% decrease in labour time saves 16 045 seconds or approximately 4.46 hours every day divided by all postmen. Saving 4.46 hours every day in Alverca means a considerable amount of labour time and money saved in the long run.

If instead we analyse the number of routes created, the optimized scenario performs even better. It shows that CTT would need not 25 but 17 or 18 postmen to supply every segment in this hypothetical day using the average time per segment.

6.4 Comparison with previous work

Other than comparing the results with the current routes, it is important to analyse the effect of the optimizations proposed. To do this, the previous model was executed using the times estimated in this work. Table 6.5 shows the results obtained using the segments generated in this work, but applying the model of the previous work.

Also in this case, after 1 hour of computation, CPLEX was unable to find solutions for larger problems. Furthermore, because the number of flow variables in the previous model is much larger, the solver can't reduce the gap between the incumbent solution and the best bound. This means that, even though there are solutions that use 18 and

even 17 postmen to supply the given segments (as proven in 6.2), with this formulation CPLEX wasn't able to find them because of to its inefficiency.

Table 6.5: Routing results with the previous work's model after 1h of solving time

Algorithm	Maximum segment size (min)	Segments	Routing time (s)	Working time (s)	Routes	CPLEX gap
Bisecting k-medoids	60	163	-	-	-	-
	90	118	-	-	-	-
	120	86	-	-	-	-
	150	67	97122	405690	19	100%
	180	57	86821	398129	19	100%
Bisecting Routes	60	180	-	-	-	-
	90	118	-	-	-	-
	120	89	-	-	-	-
	150	75	111365	425422	21	100%
	180	60	91939	409311	19	100%

In every segmentation scenario, the working time of the new formulation was less than that of the previous model. This is because the routing time between segments is much larger when minimizing for postmen, instead of for routing time.

Using the best segmentation of the new formulation (bisecting k-medoids with a maximum time per segment of 150 minutes), there was a 10.1% decrease in working time, when comparing to the previous model. In absolute terms, this is a difference of 11.35 hours every day.

7

Integration with SISMA

Finally, this work wouldn't be complete if it didn't merge itself onto the company's workflow, i.e., the tools already used by CTT to assess productivity levels. The intent of this chapter is to explain how the routing model can be integrated into SISMA and showcase the advancements made in this regard.

For the integration itself, there is a clear need to separate the machine where the model is solved from SISMA. CPLEX needs a computationally powerful machine in order to solve large real-life problems like these and SISMA is merely a spreadsheet tool, meant to be used by anyone at the company, without any hardware requirements.

This is why the decision was made, early on, to have CPLEX running on a centralized server and it is this server that receives requests from SISMA via an API call to solve a particular routing problem. At the SISMA level, there needs to be a VBA macro that makes these requests and prints the solution in a tabular format after receiving the response.

The integration architecture is shown very simply in Figure 7.1. SISMA clients are the end-users. They simply use their SISMA spreadsheet client to make requests to the VRP application server about the routing problem they wish to solve and get back results containing, among other information, the solution to the problem. The VRP application server is the component responsible for receiving routing problems, solving them, and forwarding the results to a database. It is also responsible for retrieving these results whenever a client asks for them. The database is where routing problems are stored for later examination. Since a VRP takes a long time to solve with CPLEX it would be impractical to solve them on demand. A simple document-oriented database system for storing solution files for later review is indispensable.

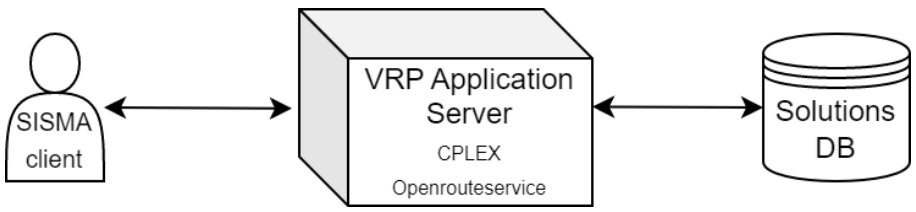


Figure 7.1: Integration architecture diagram

7.1 Integration components

7.1.1 Application server

The application server is the component which does all the hard work. It is responsible for two separate tasks: handling incoming routing requests which trigger the solving procedure and returning routing results to clients. As such, it needs at least two endpoints:

1. **POST**: Create routing problem job
   To not overload the system, this has to be a reserved endpoint, in which the job is only executed if the client that called it is authenticated and is allowed to create routing jobs. It receives the data intrinsic to the routing problem, i.e., the segments and model parameters in the request's body. If the application server has enough resources to handle multiple solving processes, it can be parallelized and could, in theory, run multiple jobs at the same time. If not, then there would have to be a queue data structure for saving routing requests. A successful call to this endpoint would just return an OK response, indicating the job was added to the solving queue.

2. **GET**: Get routing solution
   This endpoint is available to all SISMA clients in CTT. It is responsible for communicating with the database and retrieving the requested routing solution when available. If a client asks for a problem that wasn't yet solved, a file not found error is returned.

3. **GET**: Check system status
   Since solving a routing job can take a long time, there should be a way for clients to check the status of the problems being solved and also which are still pending.

This server needs to communicate with a container hosting Openrouteservice to calculate a variety of directions between locations (for calculating the travel time matrix between every two locations, for example). As such, the container needs to be created and hosted either on the machine hosting the application server or in a second system, which doesn't need to have the hardware requirements of the first.

7.1.2 SISMA client

SISMA is a spreadsheet tool built with Microsoft Excel. This means that one is forced to use a Visual Basic for Applications (VBA) macro to establish a connection to the application server. This connection is made through Hypertext Transfer Protocol (HTTP) requests using the Microsoft WinHTTP Services reference built into Excel.

The macro should also be responsible for parsing the server's response and write it to a new spreadsheet with the solution details and metrics. The first piece of information that should be displayed is the ordered sequence of segments and/or CP7s in each route.

Then, it is also important to show metrics on the routes generated like how much time it will take for a postman to complete it, the distance travelled, the vehicle assigned to it and how many segments and/or CP7s are in it.

7.2 Integration prototype

Given the time frame, it was not possible to produce a fully-fledged integration system and build all working components. What did, however, get built was a sample project showing how it is possible to make this integration and have the routing solutions incorporated into SISMA.

With regard to the application server, a simple application using flask [17], a lightweight Python web framework, was created to accommodate the routing model. The server was hosted on a personal machine, just like the Openrouteservice container. The two endpoints were created, but the solution files were saved locally instead of in the CTT's database.

In SISMA two macros were created: one for triggering the solving of a VRP and one for getting the results. The first just uses the WinHTTP service to call the first endpoint's URL and returns a message simply stating that the problem was added to the list of jobs being solved. The second macro makes a call to the second endpoint and, if a solution is received successfully, creates a table containing information about which segments compose each route. Table 7.1 shows what such a table looks like in SISMA.

Microsoft Excel and VBA, in particular, do not have an inbuilt tool for handling JSON data like the API's response content type. VBA-JSON, a community tool for JSON conversion and parsing, had to be used for this.

Table 7.1: Routing solutions displayed in SISMA

RouteId	Segments
2	0, 52, 5, 47, 0
3	0, 17, 28, 0
5	0, 21, 38, 0
6	0, 8, 63, 44, 51, 0
8	0, 48, 43, 56, 0
11	0, 2, 15, 27, 33, 55, 16, 0
...	...
21	0, 49, 20, 37, 67, 0
22	0, 39, 0
24	0, 46, 54, 42, 45, 0
25	0, 41, 30, 0
27	0, 12, 19, 23, 0
28	0, 58, 25, 50, 1, 10, 60, 0

8

Conclusions

The work described in this document aimed to give CTT a new routing mechanism that they could use to optimize the routes done by postmen on any given day. This is an extremely complex task, but it can bring immense rewards for the company and its clients.

The optimization model in Chapter 5 based itself on the work of Pedro Pereira in [26], but this time the objective function was changed to attempt to minimize labour time instead of assigned postmen. In addition, an optimization was made to the formulation using a k-nearest neighbours approach to minimize the number of flow variables and the computational effort required for CPLEX to solve the model.

Moreover, CTT set out the challenge of creating segments to be used in the routing model. This was put forth for stability purposes, for it would be a struggle for postmen to completely revamp their routes from one day to the next. A segment is a collection of CP7s, preferably close to each other, attended to uninterruptedly, and the routing procedure uses these segments for composing routes. The use of segments greatly helps in solving the VRP because using all the CP7s for routing would be infeasible, from a computational standpoint.

The combination of the segmentation and routing procedures experimented led to a significant decrease in the labour time when compared to the routes currently at play and to the routes created using the previous model in Pereira's work, as exposed in Chapter 6.

Lastly, the integration of the optimization model with SISMA, the principal productivity assessment tool at CTT, is important to make sure the landing of this model goes smoothly. In Chapter 7, a proposal is made on how to make this integration and the advancements made in this regard. For time constraints, it was not possible to completely build the system, but hopefully it can be implemented in the future.

8.1 Future work

Given the importance behind the development of this project and the implications that it can have in the long run to improve the company's service and customer satisfaction, while lowering operating costs, it is worth advancing the research and development to

find even better solutions to the problem of optimizing last-mile routing at CTT.

At first glance, there should be more work put into the conception of several Tableau visualization sheets that can better analyse the routing solutions.

Secondly, there is a research opportunity to find a heuristic method that is capable of finding as optimal solutions as the ones found by the exact method used in this project. This is not an easy task, in any way, because the heuristic must also come to terms with the routing requirements, and it should be flexible enough to allow for changes to these requirements with minimal effort.

In the routing step, there is an opportunity to define an assignment problem to delegate each postman to a route, given the postman's familiarity with the locations visited in said route. This would result in a decrease in working time and an increase in worker's satisfaction.

As for the solving of the VRP model, CPLEX allows for a great deal of customization and parametrization to its IP solver in order to adapt it to this problem's characteristics and improve the time needed to reach a solution and produce an optimality certificate. This parametrization can be a major help in improving the system's performance.

For the integration of this system at CTT there is still much work to be done. Building the components which make the integration functional from a regular user's perspective is a full project in itself. The SISMA client spreadsheet needs a lot of work put in to produce useful reports for analyzing routes. The VRP application server needs to expand to become the hub for solving the company's routing problems, and the database implementation needs a go-ahead.

Finally, once the system is able to provide meaningful routing insight in Alverca it is time to expand and apply it incrementally to all CDPs in Portugal.

Bibliography

[1] microhaus (https://stats.stackexchange.com/users/294515/microhaus). **Why is $k = \sqrt{N}$ a good solution of the number of neighbors to consider?** Cross Validated. URL: https://stats.stackexchange.com/q/535051 (cit. on p. 46).

[2] T. Achterberg, T. Koch, and A. Martin. "Branching rules revisited". In: **Operations Research Letters** 33.1 (2005), pp. 42–54. ISSN: 0167-6377. DOI: https://doi.org/10.1016/j.orl.2004.04.002. URL: https://www.sciencedirect.com/science/article/pii/S0167637704000501 (cit. on p. 13).

[3] ANACOM. **Serviços postais - 3.º trimestre de 2021**. 2021-11. URL: https://www.anacom.pt/render.jsp?contentId=1710432 (visited on 2022-01-17) (cit. on pp. 1, 21).

[4] R. Anand, D. Aggarwal, and V. Kumar. "A comparative analysis of optimization solvers". In: **Journal of Statistics and Management Systems** 20.4 (2017), pp. 623–635 (cit. on p. 15).

[5] J. Clausen. "Branch and bound algorithms-principles and examples". In: **Department of Computer Science, University of Copenhagen** (1999), pp. 1–30 (cit. on p. 10).

[6] **Clustering - K-means**. 2022-09. URL: https://scikit-learn.org/stable/modules/clustering.html#k-means (visited on 2022-09-28) (cit. on p. 18).

[7] J.-F. Cordeau et al. "New heuristics for the vehicle routing problem". In: **Logistics systems: design and optimization** (2005), pp. 279–297 (cit. on pp. 15, 17).

[8] I. I. Cplex. **IBM ILOG CPLEX Optimization Studio CPLEX User's Manual**. English. Version Version 12.8. IBM. 2011-11-16. 570 pp. (cit. on p. 14).

[9] CTT – Correios de Portugal, S.A. **Sistema_Matricial_CDP_v20_11052022**. Version 20. 2022-05-11 (cit. on pp. 24, 26).

[10] G. B. Dantzig and J. H. Ramser. "The truck dispatching problem". In: **Management science** 6.1 (1959), pp. 80–91 (cit. on pp. 2, 6).

[11] L. Devroye, L. Györfi, and G. Lugosi. **A probabilistic theory of pattern recognition**. Vol. 31. Springer Science & Business Media, 2013, pp. 169–174 (cit. on p. 66).

[12] M. Dorigo, M. Birattari, and T. Stutzle. "Ant colony optimization". In: **IEEE computational intelligence magazine** 1.4 (2006), pp. 28–39 (cit. on p. 17).

[13] E. Fix and J. L. Hodges. "Discriminatory analysis. Nonparametric discrimination: Consistency properties". In: **International Statistical Review/Revue Internationale de Statistique** 57.3 (1989), pp. 238–247 (cit. on p. 66).

[14] P. S. Foundation. **tkinter — Python interface to Tcl/Tk**. 2022-02. URL: https://docs.python.org/3/library/tkinter.html (visited on 2022-02-17) (cit. on p. 28).

[15] H. gGmbH. **Openrouteservice**. 2021. URL: https://openrouteservice.org/ (visited on 2022-02-01) (cit. on pp. 27, 33).

[16] R. E. Gomory. "An algorithm for integer solutions to linear programs". In: **Recent advances in mathematical programming** 64.260-302 (1963), p. 14 (cit. on p. 14).

[17] M. Grinberg. **Flask web development: developing web applications with python**. O'Reilly Media, Inc., 2018 (cit. on p. 56).

[18] L. Hvattum, A. Løkketangen, and F. Glover. "Comparisons of commercial MIP solvers and an adaptive memory (tabu search) procedure for a class of 0–1 integer programming problems". In: **Algorithmic Operations Research** 7.1 (2012), pp. 13–20 (cit. on p. 14).

[19] L. Kaufman and P. J. Rousseeuw. **Finding groups in data: an introduction to cluster analysis**. John Wiley & Sons, 2009. Chap. 2 (cit. on pp. 18, 19).

[20] R. E. Korf. "A complete anytime algorithm for number partitioning". In: **Artificial Intelligence** 106.2 (1998), pp. 181–203. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00086-1. URL: https://www.sciencedirect.com/science/article/pii/S0004370298000861 (cit. on p. xiii).

[21] S. Lloyd. "Least squares quantization in PCM". In: **IEEE Transactions on Information Theory** 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 18).

[22] J. M. Lourenço. **The NOVAthesis LaTeX Template User's Manual**. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[23] C. E. Miller, A. W. Tucker, and R. A. Zemlin. "Integer Programming Formulation of Traveling Salesman Problems". In: **J. ACM** 7.4 (1960-10), pp. 326–329. ISSN: 0004-5411. DOI: 10.1145/321043.321046. URL: https://doi.org/10.1145/321043.321046 (cit. on p. 9).

[24] D. R. Morrison et al. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: **Discrete Optimization** 19 (2016), pp. 79–102 (cit. on pp. 10, 11).

[25] M. Padberg and G. Rinaldi. "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems". In: **SIAM review** 33.1 (1991), pp. 60–100 (cit. on p. 14).

[26] P. Pereira. "Modelation and Optimization in Mail Delivery". MA thesis. NOVA University Lisbon, 2020 (cit. on pp. iv, v, 2, 3, 27, 29, 57).

[27] M. Savelsbergh. "A branch-and-price algorithm for the generalized assignment problem". In: **Operations research** 45.6 (1997), pp. 831–841 (cit. on p. 14).

[28] E. Schubert and P. J. Rousseeuw. "Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms". In: **Information Systems** 101 (2021), p. 101804. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2021.101804. URL: https://www.sciencedirect.com/science/article/pii/S0306437921000557 (cit. on p. 20).

[29] **The PAM Clustering Algorithm**. 2011. URL: https://www.cs.umb.edu/cs738/pam1.pdf (visited on 2022-11-03) (cit. on p. 19).

[30] P. Toth and D. Vigo. **Vehicle routing: problems, methods, and applications**. SIAM, 2014 (cit. on pp. 2, 6, 10, 15).

A

SISMA showcase

The aim of this appendix is to bring clarity to the reader about the SISMA spreadsheet tool. It is intended to present the sheets which were used at some point during this project and give them a visual representation. All the images are purely snapshots taken at version 20 of SISMA, released in May 2022, and do not convey the entire dataset.

A.1 Traffic overview sheet

The traffic overview sheet shows the traffic of each CDP discriminated by traceability, object format and delivery standard, the three main attributes of a postal object. Each row corresponds to a CDP and each cell represents the average amount of postal objects for each flow (columns) on any given day.

			s/ trace											c/ trace								
			LLE			SP			MP			P			LLE		SP		MP		P		
			d+1	d+3	d+5	d+1	d+3	d+5	d+1	d+3	d+5	d+1	d+3	d+5	d+1	d+2	d+1	d+2	d+1	d+2	d+1	d+2	Total
1000	URBANO	AODS1	696	17,351	1,578	278	325	30	32	5	0	32	5	0	3,665.2	25.8	1,121.0	151.6	431.2	64.9	309.1	34.2	26,135
1300	URBANO	AODS1	1,211	29,955	2,724	699	1,191	108	95	19	2	95	19	2	3,265.4	27.9	980.0	164.0	383.0	70.2	297.5	37.0	41,346
1500	URBANO	AODS1	62	8,462	770	66	144	13	9	2	0	9	2	0	1,215.6	23.7	366.8	139.5	149.5	59.7	107.7	31.5	11,633
1600	URBANO	AODS1	245	11,630	1,058	156	237	22	30	5	0	30	5	0	2,086.1	16.9	586.5	99.6	224.8	42.6	171.1	22.5	16,669
1700	URBANO	AODS1	1,047	17,879	1,626	412	513	47	27	4	0	27	4	0	1,624.9	14.6	504.3	86.2	196.2	36.9	147.4	19.4	24,217
1800	URBANO	AODS1	145	13,217	1,202	158	202	18	12	3	0	12	3	0	2,127.0	20.4	582.9	120.2	228.4	51.4	175.7	27.1	18,306
2000	SUBURBAN	AODS5	301	12,371	1,125	186	188	17	16	3	0	16	3	0	1,407.1	3.3	406.6	19.7	167.5	8.4	128.0	4.4	16,372
2040	RURAL	AODS5	51	4,986	454	37	45	4	5	1	0	5	1	0	441.9	3.5	255.7	20.6	107.4	8.8	68.4	4.7	6,499
2070	SUBURBAN	AODS5	61	4,186	381	47	52	5	4	1	0	4	1	0	427.5	2.9	208.6	17.2	87.6	7.3	57.1	3.9	5,554
2100	RURAL	AODS3	99	2,896	263	45	41	4	7	1	0	7	1	0	310.7	1.7	157.1	10.2	64.0	4.4	45.6	2.3	3,961
2125	RURAL	AODS3	124	7,473	680	78	88	8	8	2	0	8	2	0	958.6	7.9	543.2	46.7	229.5	20.0	146.6	10.5	10,433
2200	RURAL	AODS5	156	7,418	675	151	180	16	15	4	0	15	4	0	711.0	5.8	420.4	34.4	177.1	14.7	111.7	7.8	10,118
2300	SUBURBAN	AODS5	352	8,786	799	122	134	12	12	3	0	12	3	0	705.1	5.4	387.1	31.9	163.0	13.6	103.6	7.2	11,652

Figure A.1: Traffic overview

A.2 Productivity coefficients sheet

The productivity coefficients sheet is the first step for estimating productivity levels. It contains the benchmarked reference times for each task and subtask when available.

Atividade	Tarefa/Micro Tarefa	Nome proposto	Unidade	Tempo modelo (seg)	Obsevações
Descarga	Tarefa	**Descarga da viatura R2**	Contentor	**41.488**	
	Micro tarefa	Retirar contentor/objeto de camião e arrumar na plataforma/zona Stockagem		21.180	
	Micro tarefa	Descer plataforma		6.215	
	Micro tarefa	Transporte de contentor da Plataforma até Preparação		14.093	
	Tarefa	**Descarga da viatura R2**	Palete	**137.228**	
	Micro tarefa	Transporte de transpalete até viatura		6.380	
	Micro tarefa	Colocar transpalete na plataforma		7.540	
	Micro tarefa	Subir plataforma		4.060	
	Micro tarefa	Posicionar palete no transpalete e deslocar para a plataforma		86.768	
	Micro tarefa	Descer plataforma		5.220	
	Micro tarefa	Transporte de Palete até Preparação		27.260	
	Tarefa	**Descarga da viatura R2**	Pally	**41.488**	Utilizar o tempo de descarga de 1 contentor
	Micro tarefa	Retirar pally de camião e arrumar na plataforma/zona Stockagem		**0.000**	
	Micro tarefa	Descer plataforma		**0.000**	
	Micro tarefa	Transporte de pally da Plataforma até Preparação		**0.000**	
	Tarefa	**Descarga da viatura R2**	Objetos /K7 avulso	**41.488**	Utilizar o tempo de descarga de 1 contentor
	Micro tarefa	Retirarobjeto de camião e arrumar em áreas de trabalho		**0.000**	
Desconsolidação	Tarefa	**Desconsolidar K7 de contentor e preparar SG e SV**	k7	**10.464**	**v2**
	Micro tarefa	Retirar K7 de contentor e arrumar em áreas de trabalho		10.464	
	Tarefa	**Arrumar contentor vazio após desconsolidação**	Contentor	**27.690**	
	Micro tarefa	Arrumar contentor vazio		27.690	
	Tarefa	**Desconsolidar box de pally e preparar SG**	Box	**9.045**	
	Micro tarefa	Retirar box de pally e arrumar em áreas de trabalho		9.045	
	Tarefa	**Arrumar pally vazia após desconsolidação**	Pally	**12.997**	**v2**
	Micro tarefa	Arrumar pally vazia		12.997	
	Tarefa	**Desconsolidar objetos/K7 de palete e preparar SG**	Objetos /K7 avulso	**9.737**	
	Micro tarefa	Retirar objetos/K7 de palete e arrumar em áreas de trabalho		9.737	
	Tarefa	**Arrumar palete vazia após desconsolidação**	Palete	**51.922**	
	Micro tarefa	Arrumar palete vazia dentro de um contentor		27.271	
	Micro tarefa	Arrumar contentor com palete		24.651	

Figure A.2: Productivity coefficients

A.3 Traffic in routes sheet

This sheet showcases the routes used by CTT and how much traffic they have on an average day. It does not list which specific postal codes make up the routes. The table is filtered to show only the routes used in the CDP of Alverca.

CDP Orgânico	Designação	CP4 operacional	Giro	PD Servidos	Tráfego Média Dia Nov 21 [obj]								Percurso Ativo [metros]	Percurso Morto [metros]	Tempo de Percurso Transportes Publicos [minutos]
					Sequenciado (CP10)	Dividido Giro (CP7)		Separação Geral (CP4)			Registos	EMS			
					F	F	M	F	M	V					
2615	ALVERCA	2615	DV110		0	0	0	53	5	7	76	34	27000	0	0
2615	ALVERCA	2615	DV120		0	0	0	75	36	6	40	23	48000	0	0
2615	ALVERCA	2615	DV130		0	0	0	32	9	4	8	34	40000	0	0
2615	ALVERCA	2615	DV190		0	0	0	0	0	0	5	53	35000	0	0
2615	ALVERCA	2615	DV270		0	0	0	0	0	9	7	26	34000	0	0
2615	ALVERCA	2615	DV280		0	0	0	80	23	6	36	28	57000	0	0
2615	ALVERCA	2615	MC080	414	238	107	0	54	62	5	58	14	12650	9350	0
2615	ALVERCA	2615	MC090	351	226	111	0	57	60	7	63	13	11900	16100	0
2615	ALVERCA	2615	MC100	372	191	85	0	63	57	7	38	11	23500	25500	0
2615	ALVERCA	2600	MC200	188	239	98	0	47	35	4	57	13	16850	25150	0
2615	ALVERCA	2600	MC210	232	193	51	0	43	31	3	45	15	16200	19800	0
2615	ALVERCA	2600	MC240	274	240	134	0	52	26	6	66	15	19750	32250	0
2615	ALVERCA	2600	MC700	259	123	140	0	58	33	3	56	14	53100	17900	0
2615	ALVERCA	2615	MC710	294	115	103	0	54	33	3	32	6	43650	22350	0
2615	ALVERCA	2615	MC720	263	107	83	0	57	34	3	22	5	46100	16900	0
2615	ALVERCA	2615	MV110	26	0	78	0	52	17	3	0	0	3200	10000	0
2615	ALVERCA	2615	MV120	68	15	86	0	58	22	7	0	0	7025	0	0
2615	ALVERCA	2600	MV270	18	5	14	0	34	13	0	22	30	18000	0	0
2615	ALVERCA	2615	UA010	274	197	54	0	55	35	3	49	11	6750	1000	20
2615	ALVERCA	2615	UA030	242	348	50	0	48	92	8	56	15	8350	1000	30
2615	ALVERCA	2615	UA040	371	318	64	0	54	49	5	59	14	8750	200	30
2615	ALVERCA	2615	UA050	281	322	52	0	49	59	7	64	16	7800	1000	20
2615	ALVERCA	2615	UA060	196	277	85	0	49	77	5	66	22	6950	800	0
2615	ALVERCA	2615	UA160	536	310	47	0	51	37	8	71	30	6545	600	0
2615	ALVERCA	2615	UC170	324	214	28	0	40	26	5	37	11	7300	12700	0
2615	ALVERCA	2615	UC180	204	300	38	0	37	49	4	58	14	6870	13130	0
2615	ALVERCA	2615	UC220	291	204	32	0	60	32	4	42	14	8650	8350	0
2615	ALVERCA	2600	UC250	293	270	49	0	50	49	6	48	20	8820	23180	0
2615	ALVERCA	2615	UV020	111	107	29	0	39	25	2	25	4	4000	1000	0
2615	ALVERCA	2615	UV070	319	285	74	0	70	62	7	65	16	16050	7950	0
2615	ALVERCA	2615	UV190	116	196	29	0	53	37	2	37	14	1650	13500	0
2615	ALVERCA	2600	UV230	256	222	54	0	36	45	5	47	25	9700	19300	0

Figure A.3: Traffic in routes

A.4 Traffic by CP7 sheet

This sheet uses the data extracted from the automated processing machines to produce a view into how many objects each CP7 receives. Since not all traffic passes through the machines, the data does not reflect the entire traffic of the CDP. In addition, the values are a compilation for the month of November 2021.

CDP	CP4_base	CP4	CP7	Tipo	Finos e Médios [MTT/IBS/MMS]			Volumosos [RMS]			Total_Trafego
					Tráfego Acumulado [Não Prioritário]	Tráfego Acumulado [Prioritário]	Tráfego Acumulado [Registado]	Tráfego Acumulado [Não Prioritário]	Tráfego Acumulado [Prioritário]	Tráfego Acumulado [Registado]	
2615	2600	2600	2600002	PE	81	3	4	0	1	0	89
2615	2600	2600	2600004	PE	28	1	1	1	0	0	31
2615	2600	2600	2600005	PE	5	0	0	1	0	0	6
2615	2600	2600	2600006	PE	8	0	1	0	0	0	9
2615	2600	2600	2600007	PE	1	0	1	0	0	0	2
2615	2600	2600	2600009	PE	1112	142	110	11	4	7	1386
2615	2600	2600	2600010	PE	323	13	36	4	1	3	380
2615	2600	2600	2600011	PE	797	33	76	12	5	3	926
2615	2600	2600	2600012	PE	58	2	5	0	0	0	65
2615	2600	2600	2600013	PE	270	12	24	1	1	1	309
2615	2600	2600	2600014	PE	814	31	122	3	3	1	974
2615	2600	2600	2600015	PE	148	6	25	0	4	1	184
2615	2600	2600	2600016	PE	190	8	33	4	2	1	238
2615	2600	2600	2600017	PE	367	11	51	2	1	1	433
2615	2600	2600	2600018	PE	296	13	24	2	1	2	338
2615	2600	2600	2600019	PE	180	12	29	1	0	5	227
2615	2600	2600	2600020	PE	171	7	19	0	0	0	197

Figure A.4: Traffic by CP7

A.5 Flows sheet

This sheet specifies the base flows and which tasks are associated to each one. Some flow attributes were omitted for convenience purposes.

2615							descarga	descarga	descarga	descarga	descarga	DESCARGA
							D_1	D_2	D_3	D_3	D_4	
						Descrição Tarefa	descarga da viatura R2 [contentor]	descarga da viatura R2 [palete]	descarga da viatura R2 [pally]		descarga da viatura R2 [obj]	
						Tempo norma [min]	0.691	2.287	0.691	0.691	0.691	
						unidade convertida	obj	obj	obj [K7]	obj [box]	obj	
						Tempo norma /obj [min/obj]	0.009	0.044	0.000	0.002	0.691	
Rastreabil	Formato	Padrão	Nivel (	Ponto Entrega	Avisado	Fluxo						
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	
s/ trace	LNP	d+0	CP10	Apartados	s	0.0000%	0	0	0	0	1	

Figure A.5: Flows sheet

A.6 Summary sheet

The summary sheet is the main sheet in SISMA. Each row corresponds to a flow with only the three principal attributes specified (traceability, format and delivery standard).

In addition, each flow has a reference time for each category of tasks. These times are calculated by aggregating the times from the flows sheet (see A.5)

Sistema Matricial de Cálculo da Produtividade | CDP

CDP 2615 ALVERCA
Tráfego Total 11848

1. Tempo de 1 Objeto [min]

Fluxo	Rastreab	Formato	Padrão	descarg	desconsoli	separação_ger	T&T	sequenciame	percurs	entrega	prestação_con	entrega_avi	expedição	consoli	carga	TOTAL
dispersão	s/ trace	LLE	d+1	0.0001	0.0006	0.0438	0.0000	0.0715	0.8215	0.1772	0.0000	0.0000	0.0000	0.0000	0.0000	1.1147
dispersão	s/ trace	LLE	d+3	0.0001	0.0006	0.0106	0.0000	0.0602	0.2113	0.1703	0.0000	0.0000	0.0000	0.0000	0.0000	0.4532
dispersão	s/ trace	LLE	d+5	0.0001	0.0006	0.0106	0.0000	0.0602	0.2113	0.1703	0.0000	0.0000	0.0000	0.0000	0.0000	0.4532
dispersão	s/ trace	SP	d+1	0.0018	0.0037	0.1494	0.0000	0.2163	0.8215	1.4629	0.1301	0.1227	0.0000	0.0000	0.0000	2.9074
dispersão	s/ trace	SP	d+3	0.0018	0.0037	0.1494	0.0000	0.2162	0.2113	1.4629	0.1301	0.1227	0.0000	0.0000	0.0000	2.2979
dispersão	s/ trace	SP	d+5	0.0018	0.0037	0.1494	0.0000	0.2162	0.2113	1.4629	0.1301	0.1227	0.0000	0.0000	0.0000	2.2979
dispersão	s/ trace	MP	d+1	0.0092	0.1806	0.2321	0.0000	0.1210	0.7227	2.2204	0.1171	0.1161	0.0000	0.0000	0.0000	3.7192
dispersão	s/ trace	MP	d+3	0.0092	0.1806	0.2321	0.0000	0.1210	0.4693	2.2220	0.1171	0.1161	0.0000	0.0000	0.0000	3.4934
dispersão	s/ trace	MP	d+5	0.0092	0.1806	0.2321	0.0000	0.1210	0.4693	2.2220	0.1171	0.1161	0.0000	0.0000	0.0000	3.4934
dispersão	s/ trace	P	d+1	0.0440	0.1789	0.2106	0.0000	0.1210	0.7227	2.2204	0.1171	0.1161	0.0000	0.0000	0.0000	3.7307
dispersão	s/ trace	P	d+3	0.0440	0.1789	0.2106	0.0000	0.1210	0.4693	2.2220	0.1171	0.1161	0.0000	0.0000	0.0000	3.4950
dispersão	s/ trace	P	d+5	0.0440	0.1789	0.2106	0.0000	0.1210	0.4693	2.2220	0.1171	0.1161	0.0000	0.0000	0.0000	3.4950

% Fluxo	ráfego Flux	Tempo [h]
2.2%	262	8
71.8%	8,502	64
6.8%	773	6
1.8%	155	3
1.8%	180	7
0.1%	16	1
0.1%	13	1
0.0%	3	0
0.0%	0	0
0.1%	13	1
0.0%	3	0
0.0%	0	0

Tempo Modelo UEC [h]
2
45
4
3
3
0
0
0
0
0
0
0

Figure A.6: Summary sheet

I

k-NEAREST NEIGHBOURS ALGORITHM

The k-Nearest Neighbours (k-NN) paradigm was first introduced in 1951 by Fix and Hodges in [13]. It can be used for classification or regression, and the main principle behind it is to consider only the k closest neighbours when computing some property about a single object. In the case of classification, the output is a class membership and for regression it is a property value.

In detail, the algorithm is divided into a training phase and a classification phase. The training phase consists of only storing the features and the class labels of the training sample. In the classification phase, an unlabelled test point is classified by assigning to it the label which is most frequent among the k training samples nearest to this point. This rule for classification is simply called the k-nearest neighbour rule, and it is the most popular among researchers. Formally, it can be defined by Equation I.1 [11], where $w_{ni} = 1/k$ if X_i is among the k-nearest neighbours of x and $w_{ni} = 0$ elsewhere.

$$g_n(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_{ni} I_{\{Y_i=1\}} > \sum_{i=1}^{n} w_{ni} I_{\{Y_i=0\}} \\ 0 & \text{otherwise,} \end{cases} \tag{I.1}$$

X_i is said to be among the k-nearest neighbors of x if its distance to x is among the k smallest distances.

The distance metric commonly used for continuous variables is the Euclidean distance, but any metric can be employed in practice. For discrete variables, the overlap metric is often used.

Taking Figure I.1 as an example, let's consider we have the data set displayed. The blue squares belong to class 1 and the red triangles to class 2. Additionally, let's suppose we want to know to which class would the green circle be most closely related and most likely to belong to. Using a neighbourhood of $k = 3$ (region inside the solid circle), there are two objects belonging to class 2 and only to class 1. Thus, using the simple k-nearest neighbour rule as defined in [13], the object would be assigned to class 2. The rule is essentially a majority vote between the k neighbours.

Alternatively, let's now consider a neighbourhood of $k = 5$ (region inside the dashed circle). In this scenario, there are now three objects belonging to class 1 and only two

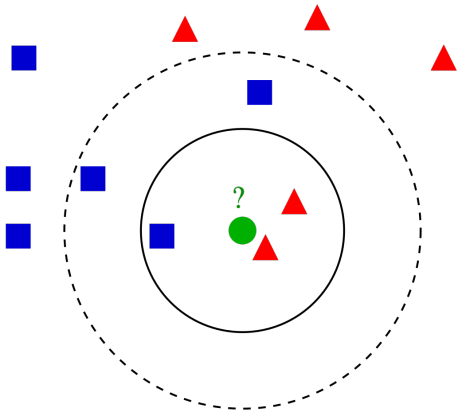


Figure I.1: k-nearest neighbours example (Antti Ajanki)

belonging to class 2. As such, the object in question would be assigned to class 1. This example serves to demonstrate how the selection of an appropriate value for k is of great importance.