Route optimization for non-stationary air pollution sensors in the City of Antwerp

David Parrilla

Promotor: Prof. Filip De Turck, dr. ir. Gregory Van Seghbroeck,

Supervisor: Leandro Ordoñez Ante

Master's dissertation from an exchange student of the

Master of Science in industrial engineering and operational research

Department of Information Technology

Chairman: Prof. Gregory Van Seghbroeck

Faculty of Engineering and Architecture

Academic year 2017-2018

GHENT UNIVERSITY

GHENT UNIVERSITY
GLOBAL CAMPUS

Route optimization for non-stationary air pollution sensors in the City of Antwerp

David Parrilla

Promotor: Prof. Filip De Turck, dr. ir. Gregory Van Seghbroeck,

Supervisor: Leandro Ordoñez Ante

Master's dissertation from an exchange student of the

Master of Science in industrial engineering and operational research

Department of Information Technology

Chairman: Prof. Gregory Van Seghbroeck

Faculty of Engineering and Architecture

Academic year 2017-2018

GHENT
UNIVERSITY

PREFACE

This work has been undertaken during my stay at Ghent University as an exchange student. I chose the department of Information Technology (INTEC) to do this work because it gave me the opportunity to expand my knowledge of computer science, specifically "Big data", technology currently booming.

The work was initially framed around the topic of anomaly detection but then, while approaching the problem and the data available, the focus was shifting towards route optimization for non-stationary sensors in an IoT environment.

I thank my promotor, Gregory Van Seghbroeck, for giving me the opportunity to do this work, for his understanding and help when changing the subject. I thank him and Leandro Ordoñez for their help with the work, for the suggestions and the feedback received. Finally, I appreciate the help provided by my colleague Jan De Geest at the beginning of this project.

Keywords: Simulation, Route optimization, Combinatorial problem, City of Things, mobile sensors.

ABSTRACT

A whole system of mobile sensors that report real-time air quality data has been put in place in the city of Antwerp. This is part of one of the projects launched in this city as part of the City of Things program.

The issues detected in the data collected by this system have led to questioning the convenience of changing the routes made by these sensors throughout the city. This is the point where the present project fits.

Specifically, this project involves the implementation of a computer program that determines an optimal set of routes in a reasonable time. These routes must provide enough data for the appropriate analysis of air quality in Antwerp.

Though this project contemplates a series of simplifications and approximations that distance it from making its deployment on a real-world use case, it is considered as a first approach to the problem that facilitates the work of future research on the topic.

The result obtained is a computer program that determines the routes according to the number of available vans. These routes are generated so that, first, the spatial distance between sensor measurements is minimum, and second, the distance between non-stationary sensors at any moment in time is maximum: sensors should always be as far as possible from each other.

In this work it is verified that the created program effectively generates optimal routes for non-stationary air-pollution sensors to perform measurements around the city of Antwerp, even though the routes it comes up with might not be realistic. This issue can be fixed by enabling the program to make queries to Google maps. This way routes would be totally adapted to the streets and traffic of Antwerp. Other improvements to the proposed approach have been discussed in cases where processing capacity is a less pressing constraint.

This program could be applied in other scenarios adding the necessary specific restrictions. Some examples are determining the routes of a set of drones (for applications such as environmental sensing and traffic control) or the optimal routes for garbage trucks (complementing the existing project of using sensors to determine which containers are full).

CONTENTS

Preface 5
Abstract 6
List of figures and tables 9
List of figures 9
List of tables 10
1 Introduction 1
1.1 context 1
1.1.1 City of things (CoT) 1
1.1.2 Air quality project 3
1.1.3 Mobile sensors 5
1.1.4 Antwerp air quality data 6
1.2 Problem statement 8
1.3 Proposal 9
2 Literature review 10
2.1 Chapter introduction 10
2.2 Combinatorial problem 10
2.2.1 Definition 11
2.2.2 Kind of algorithms 11
2.2.3 Specific problems 12
3 Implementation 16
3.1 Methodology 16
3.2 Preliminary approach 17
3.2.1 Proposal 18
3.2.2 Problem found 19
3.2.3 Possible alternative approaches 22
3.3 Definitive solution 24
3.3.1 Program description 24
3.3.2 Decisions and approximations 24
3.3.3 Problem formulation 26
3.3.4 Algorithms explanation 28
3.3.5 Finding the set of good solutions 30
3.3.6 Post-processing 32

4 Results 34
4.1 Chapter introduction 34
4.2 Generation of solutions 34
4.2.1 Squared and rectangular discretization regions 34
4.2.2 Full and half routes 37
4.2.3 Algorithm randomization 38
4.3 Post-processing analysis 40
4.3.1 Preselection of solutions 41
4.3.2 Space distribution 43
4.3.3 Time distribution 47
4.3.4 Best solution 48
4.4 Chapter summary 49
5 Conclusion 50
5.1 Review 50
5.2 Future work 51
5.2.1 Improve time distribution 52
5.2.2 Make queries to Google Map 52
5.2.3 Re-discretize when creating route 52
5.2.4 Variable mesh 53
5.2.5 Half routes improvement 54
5.3 Personal conclusion 55
6 References 56
Annexes 57
Annex A Results obtained while analysing the generation of solutions 57
Annex B Results obtained while analysing the post-process 58

LIST OF FIGURES AND TABLES

LIST OF FIGURES

Figure 1 Context schema[1] 1
Figure 2 Antwerp, “City of Things" 2
Figure 3 Internet of Things[4] 3
Figure 4 Low emission zone in Antwerp 5
Figure 5 Air Quality sensor mounted on a Bpost car 6
Figure 6 PM 2.5 Time series plot (05/02/2018) 7
Figure 7 PM 10 time series plot (10/02/2018) 7
Figure 8 Average of NO2 by sensor and hour (3 hours bin) 8
Figure 9 VRP example 14
Figure 10 Generic resolution scheme for real combinatorial problems 16
Figure 11 Map with an extracted route[25] 17
Figure 12 PM2.5 time series plot (04/02/2018) 20
Figure 13 Travel map of a wrong route extracted from the data collected. 20
Figure 14 Travel map of a long route outside Antwerp 21
Figure 15 Antwerp map divided into zones: covered (blue) and uncovered (red) 21
Figure 16 Point by point route creation algorithm - example 22
Figure 17 Program architecture 24
Figure 18 Map of the studied area. 25
Figure 19 Possible discretization of the studied area 26
Figure 20 First solution with Clarke and Wright algorithm 29
Figure 21 Discretization of the new studied area [34] 29
Figure 22 Map of the new studied area 29
Figure 23 Solution with the new study area 30
Figure 24 Plot of a solution re-discretization 32
Figure 25 Deviation of routes to improve the distribution of the points 32
Figure 26 Graphic of the distance between measurement depending on the number of vans 36
Figure 27 Solution example for 15 vans with 72% of coverage ratio 45

Figure 28 Solution example for 15 vans with 85% of coverage ratio 45
Figure 29 Example of the changes produced in one solution (graphic of the solution before and after applying the post-process) 47
Figure 30 Sensor in a garbage container 51
Figure 31 Graphic of the routes of one solution whit the zones where a more accurate mesh would be useful 54

LIST OF TABLES

Table 1 Methods for the generation of solutions 31
Table 2 Possible combinations for 3 routes 33
Table 3 Maximum distance between measurements by number of vans 35
Table 4 Maximum distance between measurements by number of vans using rectangular-shaped regions 36
Table 5 Number of vans needed by discretization and kind of routes (half or full routes).....38
Table 6 Percentage of improvement and degradation because of the randomization of the algorithm 39
Table 7 Improvement and degradation percentage because of the randomization of the algorithm by a variant of the program 40
Table 8 Total percentage of improvement and degradation of the solution by the configuration of the randomization.....40
Table 9 Number of solutions found by a variant of the program and the number of vans needed. 41
Table 10 number of solutions that need 15 vans by number of divisions 42
Table 11 Best coverage rate of the solutions by program variant.....44
Table 12 Best coverage ratio of the solutions by program variant after changing the routes...46
Table 13 Distribution rate by a variant of the program 47
Table 14 Distribution rate by a variant of the program after choosing the direction of the routes 48
Table 15 Best solutions summary 48

1 INTRODUCTION

This section starts by elaborating on the context around which this project was developed, then the problem statement is presented, and finally the proposal for this work is outlined.

1.1 CONTEXT

A brief overview of the "City of Things" program is presented next, focusing on the subsystem subject of the present study: the air quality mobile sensors.

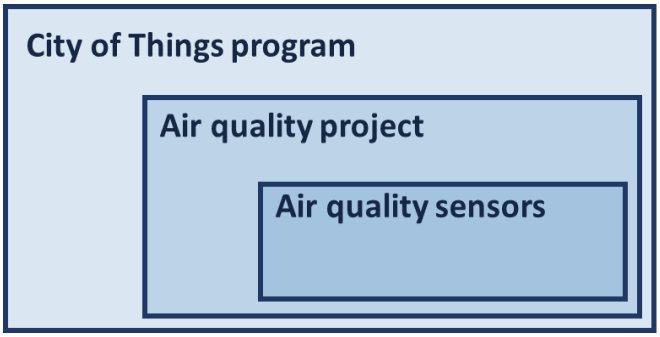

Figure 1 Context schema[1]

1.1.1 City of things (CoT)

City of things (CoT)[2] is a program launched in Antwerp, the second largest city in Belgium. Antwerp is a multicultural city where around 500,000 people live and it houses the second largest port in Europe.

The CoT program promotes the development of technology, mainly that related to the internet of things aiming to devise solutions for societal challenges such as mobility, safety and sustainability, in an attempt to turn Antwerp into a Smart City. CoT is about establishing platforms for the experimentation of pioneering projects, moving from paper to reality.

[1] Source : own elaboration

[2] www.imec-int.com/en/cityofthings

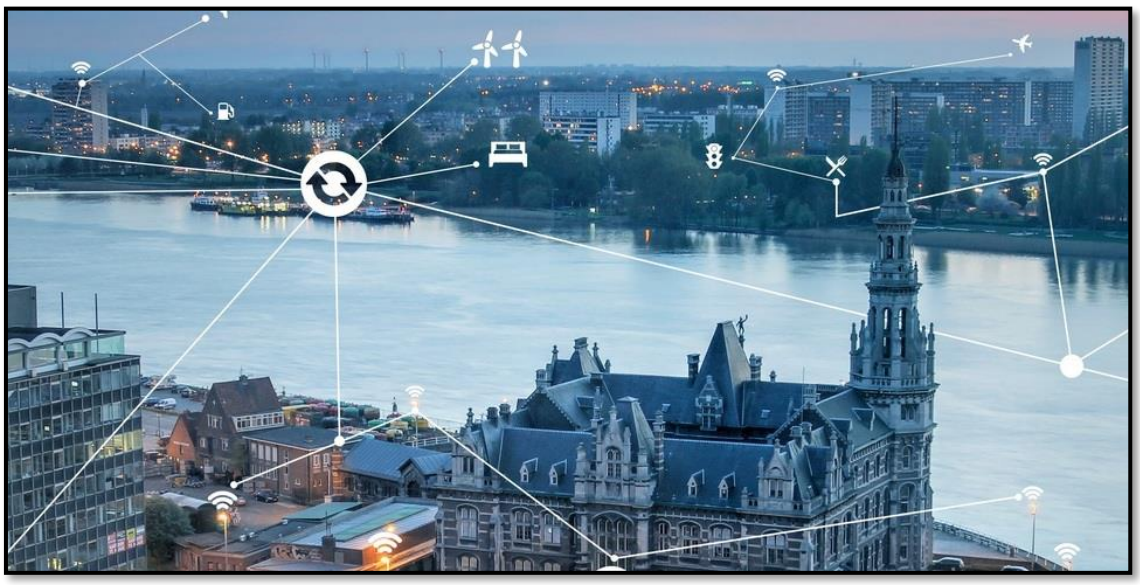

Figure 2 Antwerp, "City of Things"[3]

1.1.1.1 Beneficiaries

The proposals developed inside the CoT program aim to make life and work in the city more comfortable, this is why citizens are the end users of all this technology. From the development of the idea to the validation of the final product, the comfort of the citizens is the main objective of this project.

Other beneficiaries of this program include research centers such as universities and companies, and also other cities.

For companies, the environment provided by this project, the large-scale living laboratory, allows the development of new business models, Smart apps services or products which can be tested within this program.

The goal is to advance the state of the art of Smart City technology. Universities and other institutions are offered access to the testbeds of the City of Things. The idea is to combine the knowledge of all partners and push innovation to a new level.

[3] Source: www.flandersinvestmentandtrade.com/invest/en/news/antwerp-flanders-aspires-be-%E2%80%98capital-things%E2%80%99

For other cities, both local and international, this project serves as a guide to the challenges and problems that logistically, technologically and legally suppose the development of a Smart City.

1.1.1.2 Projects in development

The CoT program is installing sensors that collect data and connect everything to the Internet. This is exactly what the internet of things consists of, connecting everyday objects to the internet, gathering data from those objects and taking advantage of the data collected.

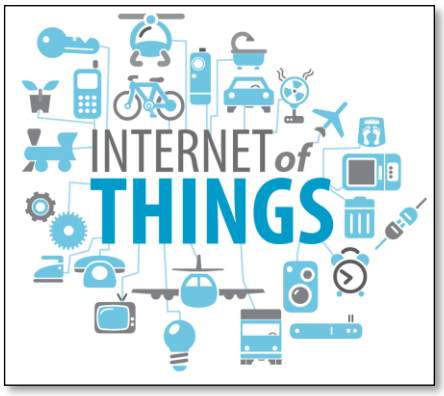

Figure 3 Internet of Things[4]

Some of the projects[5] would be:

- Sensors in public rubbish bins to monitor when it is time for the garbage truck to empty them.
- Chips in the water mains and sewage system to analyze water quality and to warn about potential floodings.
- Sensors on Bpost's delivery vans to measure air quality.
- Smart traffic lights, which can switch to green or red depending on the traffic situation.

1.1.2 Air quality project

As previously mentioned as part of the CoT program, a sensor system has been installed to monitor air quality in the city of Antwerp. This responds to the growing concern about the harmful effects of continuous exposure to pollution on the citizens of large cities. Different conditions ranging from eye irritation and all kinds of respiratory problems, to even cardiac diseases can occur.[6]

[4] Source: https://www.informationsecuritybuzz.com/articles/laser-tools-next-insecurity-internet-things/

[5] businessinantwerp.eu/news/city-antwerp-transforming-smart-city

[6] www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

If the air quality falls (or, due to atmospheric conditions, it is expected to fall) below some limits, the authorities can take measures such as prohibiting the circulation of all vehicles that emit pollution, prohibit the practice of sports outdoors or recommend the use of masks.

The fight against air pollution is developed in the following fronts:

• In the control of anthropogenic contamination sources and setting of adequate standards for emissions.

• In the monitoring of air quality and determination of minimum standards, from which exceptional measures are put in place to limit emissions.

In Antwerp, measures have been applied to address this problem. One of them involves the restriction of traffic in the center of the city[7]. The zone of low emissions restricts the transit of vehicles of internal combustion of 4 or more wheels, as those are deemed to be the ones emitting most of the polluting gases. Restrictions will be made more stringent over the coming years. At the time of writing this report, the circulation of polluting vehicles in the city center it is only allowed with a special permission, which also involves paying a fee and fulfilling a series of requirements.

This system has been implemented in cities throughout Europe such as Barcelona or Paris. The following figure shows the low emission area of Antwerp.[8]

[7] www.slimnaarantwerpen.be/en/LEZ

[8] businessinantwerp.eu/low-emission-zone

Figure 4 Low emission zone in Antwerp[9]

1.1.3 Mobile sensors

A System that collects information about the quality of the air has been implemented in Antwerp as part of the CoT program. The system consists of sensors mounted on the roof of Belgian postal services "Bpost" vans that drive around the city. This sensors measure different air quality indicators and record the location where each measurement is taken.

The main goal of this system is to detect high amounts of organic compounds in the atmosphere and then alert citizens of air pollution in real-time. It has been implemented throughout 2017 and as of the beginning of 2018 fifteen vans have already sensors installed and working. Figure 5 shows a photo of the sensor mounted in one van.

These sensors measure the particulate matter (PM), i.e. the concentration of solid and liquid particles of organic and inorganic substances suspended in the air. These sensors measure:

- PM1, PM2.5, PM10 (concentration of particles with a diameter smaller than 1, 2.5 and 10 respectively)
- VOC (concentration of volatile organic matter) and the
- NO2 (concentration of nitrogen oxides).

[9] Source: businessinantwerp.eu/low-emission-zone

Particles less than 2.5 are especially harmful because they can cross the pulmonary barrier and enter the blood system. [10]

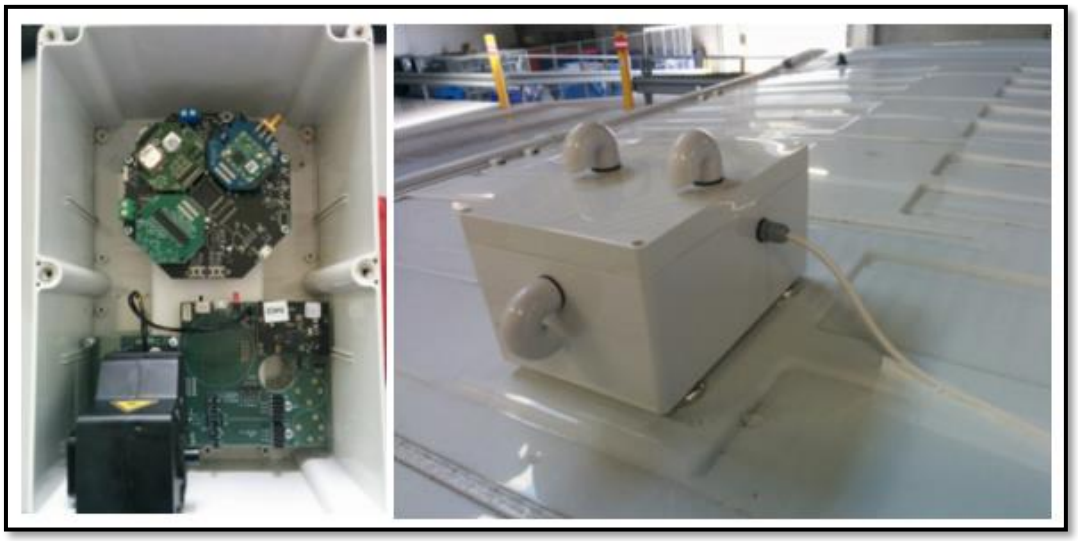

Figure 5 Air Quality sensor mounted on a Bpost car [1]

1.1.4 Antwerp air quality data

As mentioned, the mobile air quality sensor system is still under development, being constantly studied and analyzed. One of the institutions that is developing different projects around the data collected by these sensors is Ghent University (UGent)[11].

At the beginning of this work an analysis of the collected data was conducted in order to validate how correct it was, if the data were accurate enough and if enough information was collected. The intention behind this was developing an algorithm to stop anomalies in real time using all the information available from the system to clean missing or faulty data.

Some generic anomaly detection techniques for temporary data series were also evaluated. No significant results were obtained due to the high variance of the values and the noise present in the measures.

[10] www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

[11] www.ugent.be/ea/idlab/en/research/research-infrastructure/city-of-things-r-d-infrastructure.htm

In the following graph you can see an example of the PM 2.5 data collected by a sensor for 5 hours.

Figure 6 PM 2.5 Time series plot (05/02/2018)[12]

The big amplitude and the high frequency of the oscillations of the values, as shown in figure 6, diminish the effectiveness of anomaly detection techniques.

At this point, with the analysis of the data collected, some discrepancies and faulty data were observed. Some of the problems found in the sets are listed below:

- Some sensors are down during different periods of time. This is frequently happening while a van is traveling around the city. Usually after a while the sensor starts working again in another location (since the van has been moving). Therefore there is no data of any of the measurements made during such period.
- In the graphic below notice that the sensor was down between 7 and 8 AM.

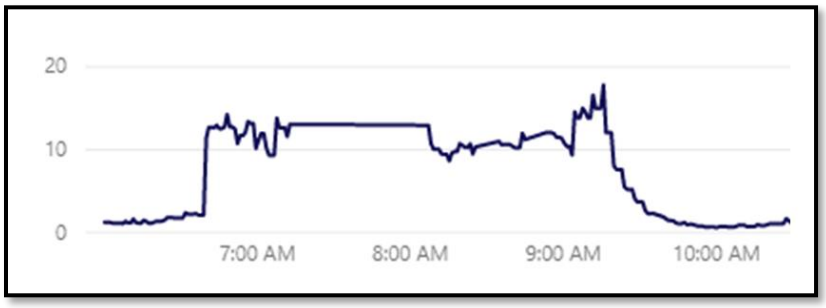

Figure 7 PM 10 time series plot (10/02/2018)[13]

[12] Source: own elaboration – Power Bi

[13] Source: own elaboration – Power Bi

- High variability between the measurements from different sensors. In some cases it was found that the values obtained by some of the sensors were totally inconsistent. It was observed that some sensors always obtain higher values than others on average in different periods, which leads to a poor pattern detection on the available data. The latter is not directly considered an error since it may be due to the variance of the measures, but it is a problem because it means that more data has to be collected in order to be sure that the measures are correct.
- The following graph shows the average NO2 measurement over the collected data every 3 hours. Notice that even when the vans are together in the parking lot from 0 to 6 A.M. the values are not similar between these sensors.

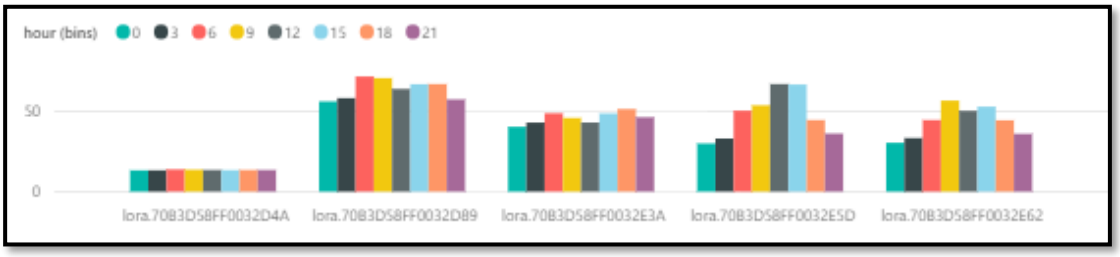

Figure 8 Average of NO2 by sensor and hour (3 hours bin)[14]

- It was observed that the values between sensors are widely spread in time and space so correlations cannot be made. Also, the collected data show that different zones of the city are covered once a month or not covered at all: hence more data is needed in order to be able to get a complete picture of the air quality in Antwerp.

1.2 PROBLEM STATEMENT

Once the issues stated above regarding the datasets collected from the CoT setup were identified, different approaches were considered. In this document, the problem studied is the one related to the distance within the measures of different sensors in both time and space.

Reducing the distance within the measures of different sensors (both in time and space) would help to have enough data to identify and easily solve other problems, as better correlations

[14] Source: own elaboration – Power Bi

within the data can be derived, and in the context of the present project it will also be useful to get the complete picture of the air quality in Antwerp.

1.3 PROPOSAL

The chosen approach consists in generating the routes that the vans should follow each day to cover the city as thoroughly as possible. These routes must be distributed around the whole city, the more distributed the better, so the measurements are closer to one another and all the zones get covered.

The routes must be distributed on time too. A simple example to explain this concept would be: given two routes that cover the same two zones, A and B, then it would be appropriate for them to be scheduled so that they cover zones A and B in an alternating order: for instance during the morning one route does zone A while the other is doing zone B and in the afternoon the first route is scheduled to do route B and while the second does zone A. By doing so overlapping measurements of the air quality at any time would be avoided, this way providing a better picture of the Antwerp Air quality. With more non-stationary sensors this concept is more complex, but the purpose behind it remains the same: to avoid as much as possible overlapping measurements.

These routes do not have to be the same each day, but they could be adapted daily according to different conditions such as the traffic expected or the existence of more interesting areas due to a specific event, or to exceptional values detected the previous day.

In this sense, a computer program that generates routes meeting said features in an affordable time was implemented. In this document different variants of the devised system are explained, some of them implemented (Section 3.2.5 Post-processing) and others have been deferred to future work (Section 5.2).

2 LITERATURE REVIEW

2.1 CHAPTER INTRODUCTION

For addressing the problem stated in the previous chapter, it was necessary first to get to know different methodologies and concepts related to route optimization. In this chapter, the fundamental concepts for understanding the work carried out are detailed. In the first place, it is explained what kind of problem has been identified and why: combinatorial type problem. Some fundamental concepts are also defined.

A more detailed explanation of this type of problems is provided, along with the existing mechanism for solving them, as well as some illustrative examples. Some of the examples are described in more detail as they have been used for solving the problem considered in this project.

2.2 COMBINATORIAL PROBLEM

The problem described in the first chapter of this report consists in determining the routes a fleet of vans should use to travel the city. In order to find the best solution regarding a series of criteria, in this case spatial and temporal distribution of the routes, the system should explore all possible options to be able to find the optimal one. This because it is not possible to characterize the optimal solution so the number of possible combinations is reduced.

This way, all possible routes in Antwerp would have to be determined (taking into account that completing a route cannot take more than one working day) and then these routes would have to be assigned and scheduled to the available vans. The number of possible combinations in this case is finite but to explore them all in an affordable amount of time is infeasible. This is a combinatorial optimization problem.

2.2.1 Definition

In applied mathematics and theoretical computer science, combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects[15]. Specifically, combinational problems are those optimization problems in which all or some variables only admit a finite number of values (so there is a finite number of solutions) or those where there is always an optimal solution in a finite subset of feasible solutions to the problem.

The combinatorial optimization algorithms solve these problems by reducing the effective size of the solution space (usually very large), and by exploring the space efficiently.

Some examples of this type of problem can be in the decision making in the selection of investments, in the cutting of materials, in the packaging or in the plant distribution.

2.2.2 Kind of algorithms

There are two types of algorithms to deal with this type of problem: exact and heuristic[2].

Heuristic or approximate methods are defined by Diaz et al (1996) as "A procedure to solve a well-defined optimisation problem through an intuitive approach, in which the structure of the problem is used intelligently to obtain a good solution". They are effective procedures to find good solutions, although not optimal, in which the speed of the process is as important as the quality of the solution[16].

On the other hand, the exact algorithms guarantee to find the solution sought (optimal), if it exists, in a finite time. But the time invested can be of an order of magnitude much higher than that of the heuristic (and inapplicable).

Different heuristic methods can be used and combined depending on the problem[17]:

- Constructive (greedy): they construct a solution by selecting, according to an evaluator, the best candidate from an ordered list.

[15] www.math.ethz.ch/ifor/research/combinatorial-optimization.html

[16] www8.cs.umu.se/kurser/TDBAfl/VT06/algorithms/BOOK/BOOK2/NODE91.HTM

[17]www.springer.com/cda/content/document/cda_downloaddocument/9783642167287-c1.pdf?SGWID=0-0-45-1036146-p174061358

- Destructive: all the elements are in an infeasible solution and the worst ones are eliminated until obtaining a feasible one.
- Random (from Monte Carlo): select solutions randomly and keep the best.
- Relaxation: you get a solution, sometimes infeasible, by relaxing some restriction; later it is modified until it reaches feasibility.
- Reduction: identifies some property that "seem" to meet the good solutions and is added as a restriction of the problem.
- Decomposition: the original problem is divided into smaller subproblems.
- Local search: start with an initial solution and generate the neighboring solutions: if there is a better one, it is saved and iterated again, else the process ends.

2.2.3 Specific problems

There are some extensively studied semi-theoretical problems, so there are vast literature and algorithms developed for them. When facing a new problem it is useful to identify some particular structures that appear in these semi-theoretical problems. This allows to use the research done for those and perhaps to be able to adapt some of the algorithms. Some examples are listed below:

- Chromatic number. Given an unoriented graph, assign the minimum number of colors to the vertices, so that two adjacent vertices have different colors.

  It can be applied for example to the problem of scheduling exams. The colors in this case would be the days, the subjects would be the vertices, and the arcs the existence of students who attend the two subjects.
- Problem of the backpack. Given n objects, with weight w_j and value v_j; select those to be included in a backpack without exceeding the **W** weight, and maximizing the total value of the included objects. It can be applied for example for deciding how to cut a wooden board in pieces.
- Set cover problem. Given a set of objects $S = \{1, 2, ..., m\}$ and the subsets of S, $H = \{H1, H2, ..., Hn\}$ where each Hi have an associated cost c_i; cover, at minimum cost, all the elements of S with subsets H_i. Used for establishing the allocation of services, or deciding the assignment of crews to flights.

Some of these semi-theoretical problems have been used in solving the problem addressed in this work. These problems are explained in more detail in the following sub-sections.

2.2.3.1 Travelling salesman problem

The Traveling Salesman Problem (*TSP*)[18] seeks to determine the shortest possible route (or the route with minimum cost) that a traveler who has to visit *n* cities has to follow. The route starts and ends in a certain city and has to pass only once through each one of the other cities.

The TSP problem is one of the most studied combinatorial problems, so for cases between 100 and 1000 cities there are many algorithms and methodologies that get a good solution instantly (heuristic methods) or even guarantee the optimum (exact methods).

This problem, even in its simplest form, has many applications. Some examples are industrial planning, logistics or design of electronic circuits.

This problem is usually represented by a graph where the vertices are the cities and the arcs the routes between cities. Each of these arcs has an associated cost, usually depending on the time or distance that each route involves.

2.2.3.2 Vehicle Routing Problem (VRP)[19]

This semi-theoretical problem is a generalization of the TSP explained above. It consists in establishing the distribution routes given a finite number of cargo trucks, a warehouse, and some customers (with their demand and location). An example is shown in the figure below:

[18] www.math.uwaterloo.ca/tsp/

[19] neo.lcc.uma.es/vrp/vehicle-routing-problem/

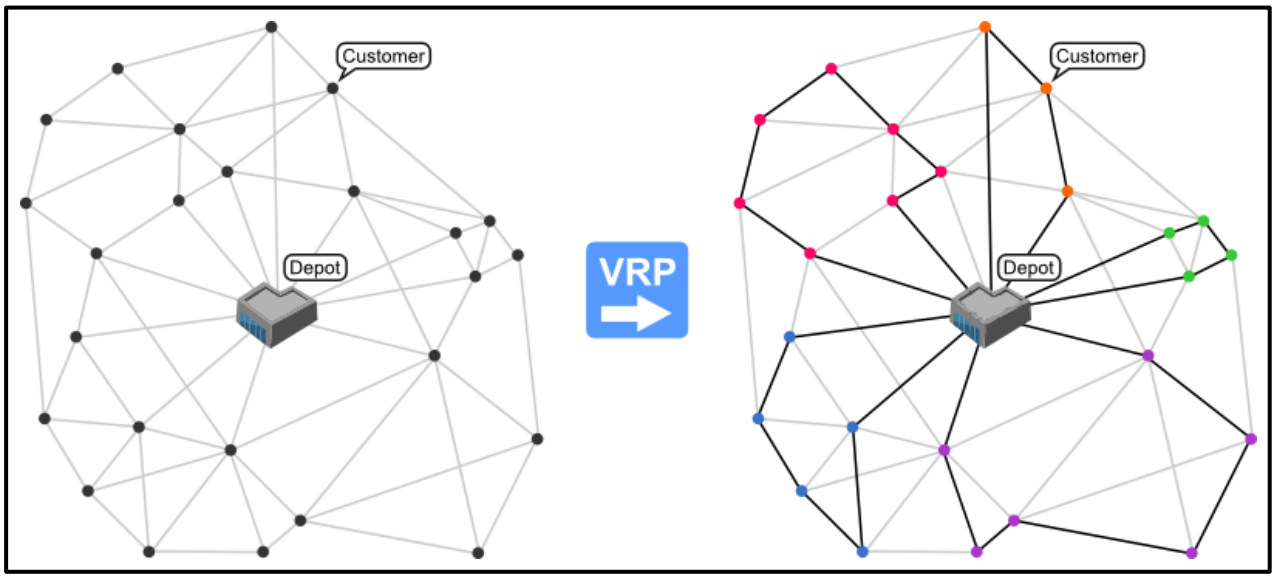

Figure 9 VRP example[20]

The most widely used implementations to solve the problem are based on heuristics because they produce fairly acceptable results for large instances of it.

One of the best-known heuristic algorithms for the resolution of this type of problem is the *Clarke and Wright Algorithm*[21] also called Savings Algorithm, which is the one that has been used in the system implemented in the context of this work. As a heuristic methodology it does not guarantee to come up with an optimal solution.

This methodology starts considering that for each point there is one different route, starting at the origin, reaching such point and then coming back to the origin. Then the program calculates for each pair of points how much choosing to go from one point to the other will reduce the total distance travelled. The inner workings of this algorithm are introduced in the next chapter (section 3.3.3).

[20] Source: neo.lcc.uma.es/vrp/vehicle-routing-problem/

[21] neo.lcc.uma.es/vrp/solution-methods/heuristics/savings-algorithms/

2.2.3.3 Bin Packing Problem

The Bin packing problem (BPP)[22] consists in given a set of elements of different volumes, packing these elements in Bins so that the minimum Bins possible is used. It can be presented in one, two or three dimensions.

This last problem, although it is not as well known as the previous two, also has some of its own heuristics that allow finding good solutions in a reasonable time.

It can be applied for example in:

- The industrial sector for cutting material. In an effort to reduce costs, the objective is to minimize the amount of material lost in the process.
- The computer sector when loading files into computer storage, or when trying to assign tasks to processors.

[22] www.bernabe.dorronsoro.es/vrp/index.html?/Problem_Descriptions/BinPacking.html

3 IMPLEMENTATION

This chapter is divided into 3 sections. The first section explains the general methodology used to solve a combinatorial problem, while the second one explains the preliminary approach to address the problem stated in the first chapter of this report. Finally, the third section elaborates on the definitive solution: the different steps that were taken along with the different decisions are also explained and justified in this section.

3.1 METHODOLOGY

To solve a specific combinatorial problem as the one addressed in the work reported herein, the first thing to do is to observe reality and create a mathematical model by considering the necessary restrictions and simplifications. A second stage involves applying different algorithms to obtain analytical results that allow decision making and choosing a solution that should be applied in the system under study. This sequence of stages is shown next in figure 9. This figure also reflects structure in which the main sections of this report has been written.

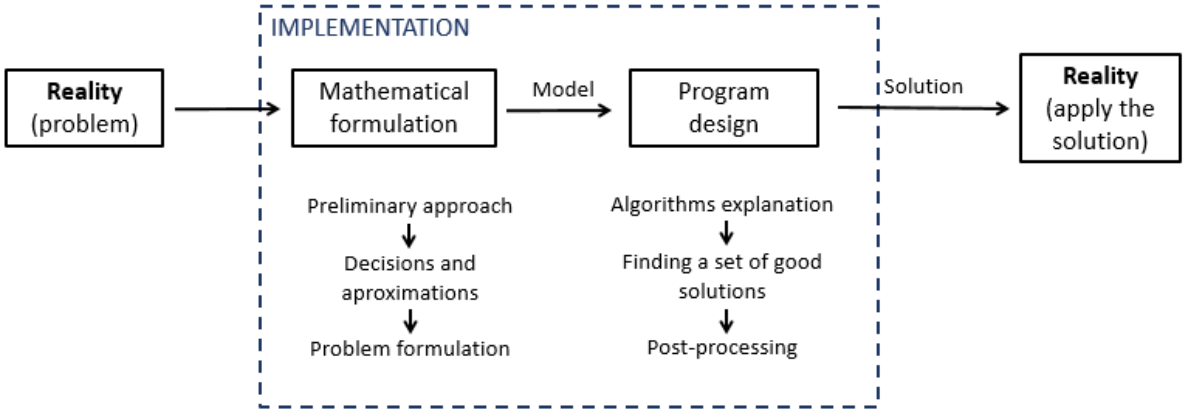

Figure 10 Generic resolution scheme for real combinatorial problems[23]

There are a series of techniques and procedures that can be useful to find good solutions depending on the problem. It is essential to take into account all the available information (e.g. the structure of the problem, the form of the solution, etcetera).

[23] Source: own elaboration

Before applying any algorithm, all those combinations regarded as suboptimal solutions should be removed: applying some preprocessing to reduce the number of combinations to explore (search space).

Another useful technique is the identification of structures the problem has in common with some of the already extensively studied semi-theoretical problems. In this way, specific heuristic algorithms can be used to come up with near-to-optimal solutions for the problem at hand.

3.2 PRELIMINARY APPROACH

The first step before starting with the implementation of the simulation was to study the data available.

The routes of the vans were extracted from the data collected during the two first months of 2018. These routes comprise points of latitude and longitude of the van in the locations where measures were taken (sensor on the vans take one measurement every 5 seconds).

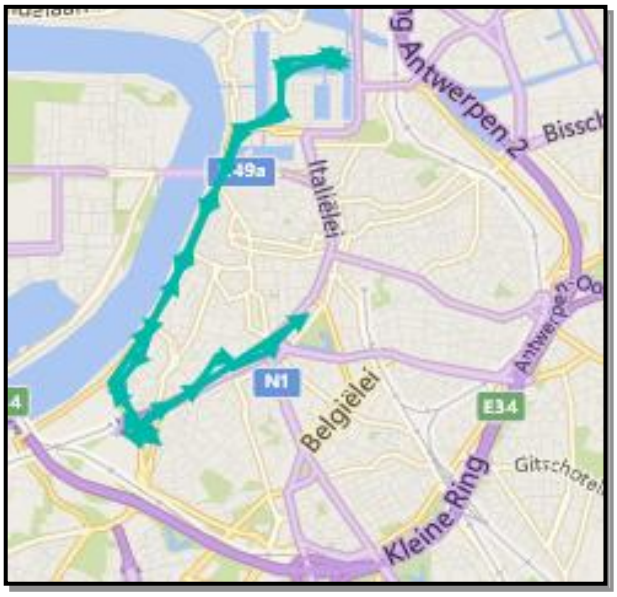

Figure 11 Map with an extracted route[25]

Looking at the map (figure 11) with the points plotted on it, it is possible to see that the routes follow the streets of Antwerp, as expected.

[24] Source: own elaboration – Power Bi

3.2.1 Proposal

It is possible to propose the following implementation leveraging on the data of the routes mentioned earlier: consider the problem as a restricted version of the bin packing problem explained in the literature review.

The idea was to use the routes extracted as a set of elements that can be packed. The vans would be then the bins where the routes can be "packed": if we consider that the vans can only do a specific number of kilometres per day (depending on the number of hours of work and the mean velocity), this would be equivalent to the capacity of the bins (in this case vans) and the length of the routes would be the size of the elements. Given:

- a set of vans $S_1, S_2 \ldots$ that can travel at most the same distance V,
- a list of **P** points that must be reached,
- a list of n routes with lengths $a_1, \ldots, a_n$
- and the points that each route reach $r_{11}, \ldots, r_{1p}, \ldots, r_{n1}, \ldots r_{np}$.

Find:

- an integer number of vans B
- and a B partition $S_1 \cup \ldots \cup S_B$ of the set $\{1, \ldots, n\}$

such that:

- all points are reached
- and the maximum distance of each van is not exceeded.

A solution is optimal if it has minimal B.

A possible Integer Linear Programming formulation of the problem is:

Minimize $B = \sum_{i=1}^{n} y_i$

subject to $B \geq 1,$

$$\sum_{j=1}^{n} a_j x_{ij} \leq V y_i, \qquad \forall i \in \{1, \ldots, n\}$$ *: Limitation of distance*

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1, \ldots, n\}$$ *: The routes can be chosen only one time*
: All points must be reached

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}r_{ik} \geq 1, \qquad \forall k \in \{1,\dots,p\}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \{1,\dots,n\} \forall j \in \{1,\dots,n\}$$

$$y_i \in \{0,1\}, \qquad \forall i \in \{1,\dots,n\}$$

where $y_i = 1$ if van ***i*** is used and $x_{ij} = 1$ if the route ***j*** is chosen for the van ***i***.

With this formulation, it is possible to minimize the number of vans, but after this, it would be necessary to optimize the distribution in space and time given a specific number of vans. The restriction that makes each route to be chosen only one time could be removed and the objective function should be changed. The best option would be to make the function with two addends, multiplying each one by a weight (***w*** and ***1-w***) to optimize both the distribution and the number of vans. Then the weight can be tuned between 0 and 100% to obtain different solutions.

3.2.2 Problem found

At first, the routes extracted seemed good to be used in the optimization of the system but further analysis revealed that they were not. The problem found in the data was that routes usually were wrong due to sensor's location misreading for periods where the sensor was not working. Additionally, there are cases in which the Bpost vans appear to be driving outside of the city render those routes obsolete. The number of routes available after removing all the useless ones was very small and it was also observed that there were zones that were not covered by any of these routes.

The mean velocity of the vans was calculated at first using the data collected but the ill-measured locations, and the stops during the travels made this velocity not accurate. In consequence, the mean velocity was calculated using google maps, searching for different routes that cross the city in different periods of time. This velocity was used along with the working time of the vans to calculate the maximum distance that the vans could drive daily.

3.2.2.1 Data analysis

In this section some examples of the problems explained are shown in some graphics.

Figure 12 and figure 13 show examples of routes extracted from the data collected where the sensor was closed and opened again after some time. In the first figure, the values measured are shown in a time series plot. Two red squares, one around 14:00h and another between 15:00 and 16:00h, delimit the periods in which the sensor was off.

The second figure (figure 13) shows the path followed for one of the vans in Antwerp. Between the point A and B the sensor has not made any measurements: the sensor was off. That is why this part of the route is not following any street.

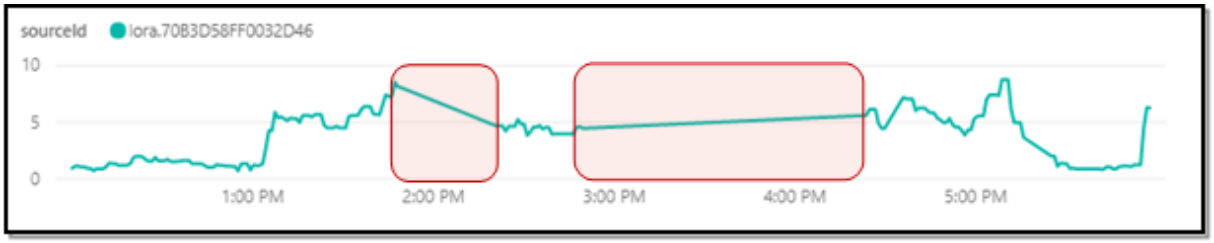

Figure 12 PM2.5 time series plot (04/02/2018)[25]

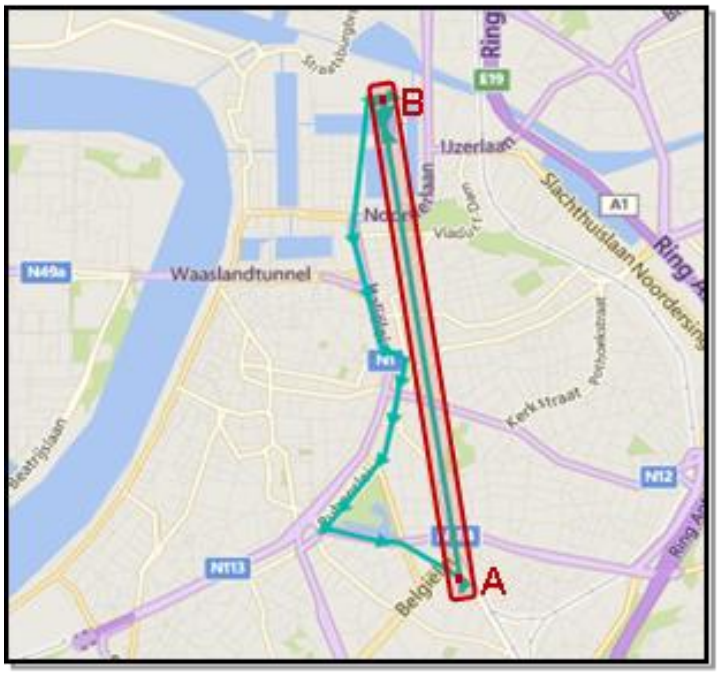

Figure 13 Travel map of a wrong route extracted from the data collected.[26]

[25] Source: own elaboration – Power Bi

[26] Source: own elaboration – Power Bi

An example of a route that cannot be used for the optimization because it goes out of Antwerp is shown in Figure 14.

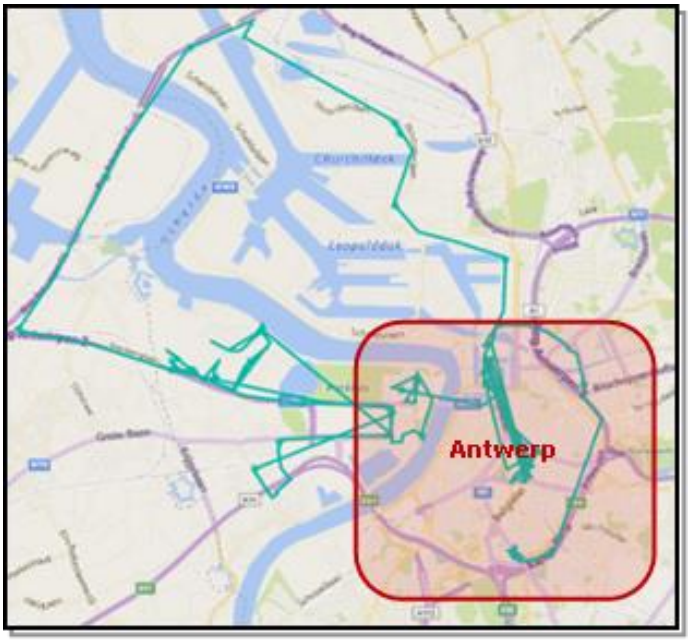

Figure 14 Travel map of a long route outside Antwerp[27]

Finally, figure 15 shows a map of Antwerp divided into zones: the blue zones are covered by at least one route and the red ones are not covered but are important for the optimization.

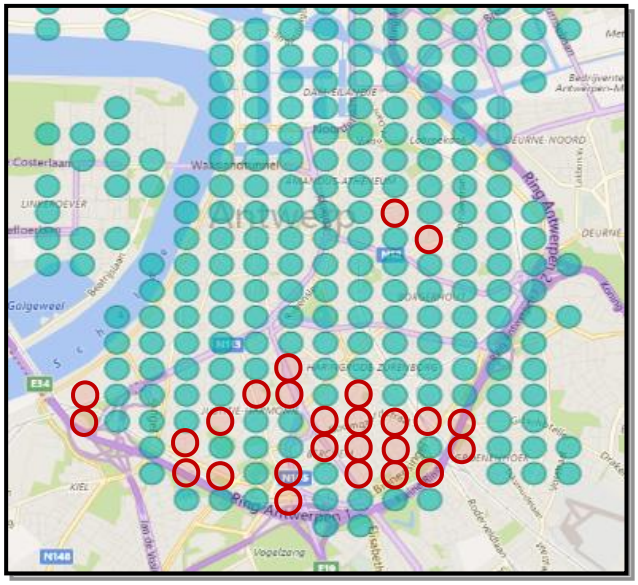

Figure 15 Antwerp map divided into zones: covered (blue) and uncovered (red)

[27] Source: own elaboration – Power Bi

3.2.3 Possible alternative approaches

3.2.3.1 Approaches formulation

Three possible approaches to tackle the problem were considered at this point:

The first one was to generate the routes point by point. The idea behind it is that the algorithm would evaluate a function on each of the possible points (points close to the current position of the van) and the best point gets selected. This kind of algorithm is called “Greedy”. An example is given in figure 16: in this example, there are 2 vans, and the routes are starting from the point "o". The red points are the ones that are being evaluated, so in the next stage the best one is selected (plot in green) and new points are evaluated for the other van.

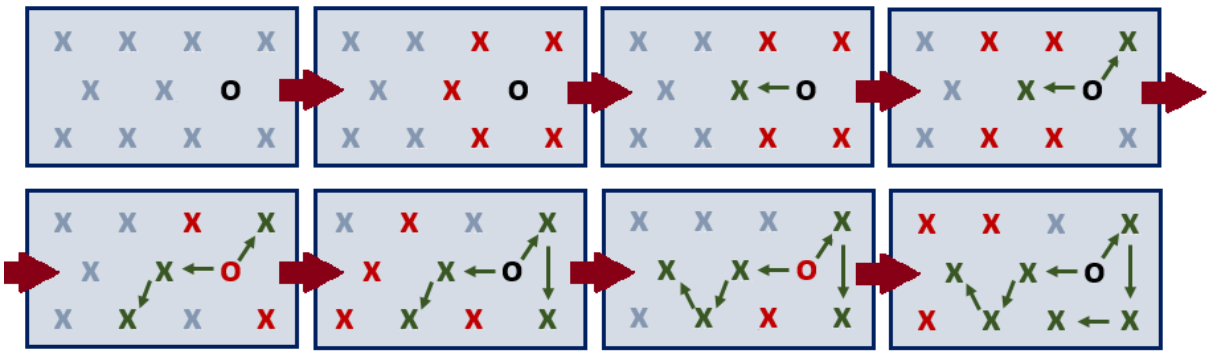

Figure 16 Point by point route creation algorithm - example[28]

The function that has to be evaluated should calculate the distance to the points already chosen, should consider also the number of points that will be available after selecting the point and other parameters that allow the algorithm to get routes as distributed as possible in time and space and to cover all the possible points.

The second approach was to generate a huge amount of random routes, covering different points and zones. All zones should be covered for more than one route, then the problem would be reduced to a Bin Packing problem, as explained before (section *3.1.1 Proposal*): choosing between the routes created, which one are good combinations for each van.

Finally, the third considered approach involved dividing the problem into smaller problems, doing some approximations. First, the routes are created minimizing the maximum distance

[28] Source: own elaboration

between measures and the total distance travelled. Then having already a good solution, it is modified and adapted to improve other aspects of the solution such as the distribution in time of the measurements

3.2.3.2 Justification of the decision

The three options were studied and tested, each reaching a different level of implementation.

The problem found while implementing the first approach was the function to evaluate each one of the available points. Only considering the parameters of the point under evaluation it was not possible to come up with a good solution: it is required to know first if the point that would be chosen after choosing the evaluated point is a good one. To know if this last point is good, at the same time it is needed to know if the next point that would be chosen is good or not, and so on. The number of steps that the program has to look forward has to be determined, a low number would mean a bad solution and a big number would make the program very slow due to all the possible combinations that have to be evaluated, which is computationally expensive. So finally, this approach was rejected because it cannot guarantee to find a good solution in a reasonable time.

The second approach was more reasonable but still far from reachable. First, the set of routes that have to be created need to satisfy some requirements. Requirements such as reaching all the points and that every route does not exceed the van capacity. The more the routes created the better the solution would be, at the same time the processing time will grow factorially with the number of routes due to the increase of the combinations. In short, either we have a bad solution calculated in a relatively short time or a good solution that takes ages to be calculated.

Finally, the third option was chosen and implemented successfully. It is possible to find a good solution in a reasonable time as the different approximations allow to reduce the execution time of the program considerably. The different stages of the implementation are explained in the next section.

3.3 DEFINITIVE SOLUTION

3.3.1 Program description

Finally, the developed program follows the structure shown in the diagram below. The different steps that were taken along with the different decisions are explained and justified in the following sections.

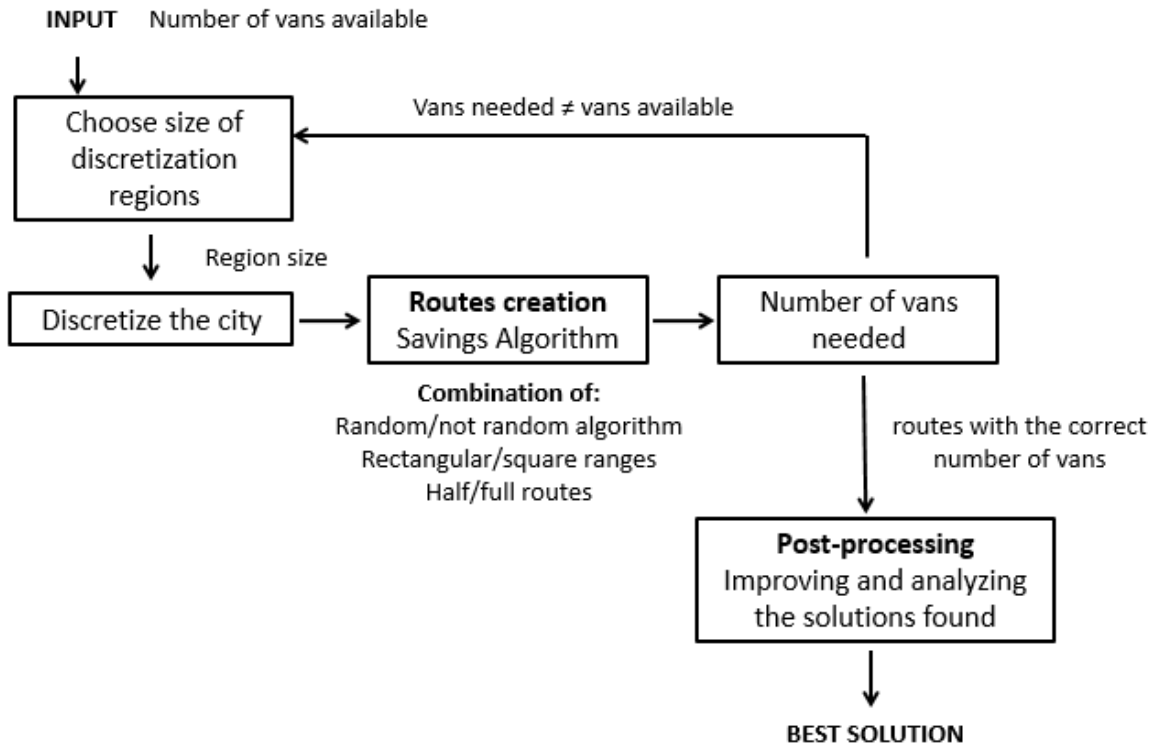

Figure 17 Program architecture[29]

3.3.2 Decisions and approximations

Before starting with the implementation some decisions and approximations were done. First, the range that the vans should cover had to be determined. It was decided to focus on the city center because it allows the simplification of the algorithm: it allows the computed speed to

[29] Source: own elaboration

be closer to reality and the discretization of the city in location regions. The map in figure 18 shows the zone under study.

Figure 18 Map of the studied area.[30]

Secondly, the city was discretized in location regions in order to get a specific number of points that need to be reached for the vans. The size of the regions is not set from the beginning but it is left as a variable. This variable is closely related with the maximum distance allowed between two different measures of the sensors and it is one of the factors that is aimed to be studied. Figure 18 shows a possible example of discretization of the studied area.

[30] Source: own elaboration – Power Bi

Figure 19 Possible discretization of the studied area[31]

The routes extracted from the data were analysed using a function that gives the points that each route travels through, given different sizes of the regions for the discretization of the locations. This was used to check the zones where any route goes through.

3.3.3 Problem formulation

The problem now is approachable as a Vehicle routing problem. As explained in the literature, this problem is based on finding the best combination of routes serving all the clients (points extracted from the discretization of the city) from a given origin (in this case the parking), without exceeding a capacity (number of kilometres each van can do), while minimizing the total distance travelled.

Given:

- a set of vans $S_1, S_2 \ldots$ that can travel at most the same distance V,
- a list of **n** points that must be reached,
- and the distance between each pair of points $C_{12}, C_{13}, \ldots, C_{n(n-1)}$

Find:

- an integer number of vans K

[31] Source: own elaboration

- and the route that each van does **d**

such that:

- all points are reached
- and the maximum distance of each van is not exceeded.

A solution is optimal if the distance total travelled is minimal.

A possible Integer Linear Programming formulation of the problem is:

Minimize

$$Distance = \sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{i=1}^{n} C_{ij}X_{ijk}$$

subject to,

$$\sum_{k=1}^{n}\sum_{i=0}^{n} X_{ijk} = 1, \qquad \forall j \in \{1,\dots,n\}$$

$$\sum_{k=1}^{n}\sum_{j=0}^{n} X_{ijk} = 1, \qquad \forall i \in \{1,\dots,n\}$$

} *: All points must be reached and only one*

$$\sum_{k=1}^{n}\sum_{i=1}^{n} X_{i0k} = K,$$

$$\sum_{k=1}^{n}\sum_{j=1}^{n} X_{0jk} = K,$$

} *: All the routes that start at the origin return to it*

$\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ijk}C_{ij} \leq V, \qquad \forall k \in \{1,\dots,p\}$ *: Limitation of the distance that each van can travel*

$x_{ijk} \in \{0,1\}, \qquad \forall i \in \{1,\dots,n\}\ \forall j \in \{1,\dots,n\}\ \forall k \in \{1,\dots,n\}$

where $x_{ijk} = 1$ if the route done by the van ***k*** is chosen to travel from the point ***i*** to the point ***j***.

3.3.4 Algorithms explanation

One of the most useful algorithms that provide good solutions to the classic VRP problem is the algorithm of Clarke & Wright. This methodology starts by considering that for each point there is one different route, starting at the origin, reaching such point and then coming back to the origin. Then the program calculates for each pair of points how much choosing to go from one point to the other will reduce the total distance travelled.

It is stored usually in one matrix (***S***). Then the program chooses one pair of points in decreasing order, according to the value calculated, and then it decides, according to some rules, to join the routes of this points or not. The process is described next:

1. Make ***n*** routes: from the origin to each point and back to the origin.
2. Compute the savings for merging delivery locations ***i*** and ***j***, which is given by
$$s_{ij} = C_{io} + C_{0j} - C_{ij}, \qquad \text{for all } i, j \geq 1 \text{ and } i \neq j$$
3. Sort the savings in descending order;
4. Starting at the top of the (remaining) list of savings, merge the two routes associated with the largest (remaining) savings, provided that:
    a. The two delivery locations are not already on the same route;
    b. Neither delivery location is interior to its route, meaning that both points are still directly connected to the depot on their respective routes;
    c. The distance constraints ***V*** is not violated by the merged route.
5. Iterate from step *3*. until no additional savings can be achieved.

The approximation explained until now gives good results. The graphic below (figure X) shows a solution to one of the tests done. Each color is one different van, the origin in this test was chosen to be in the coordinates *(4,6).*

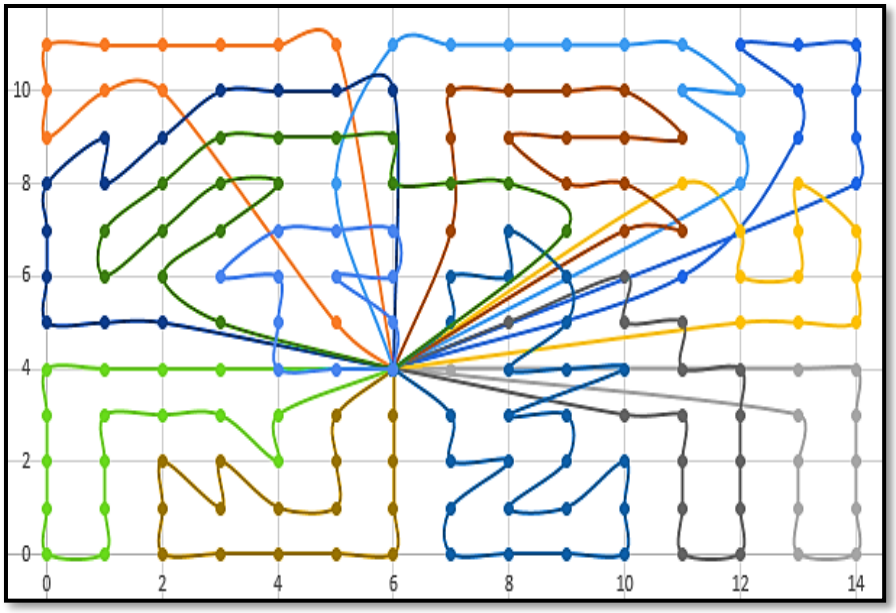

Figure 20 First solution with Clarke and Wright algorithm[32]

The next step was adding some restrictions to the program to make it more close to the reality. One of the improvements was to take the river traversing the city into account, so that the routes avoid going through it if it is not by crossing the bridge. This changes the way how the values are calculated within the Clarke and Wright algorithm. The algorithm also considers the fact that some points do not need to be covered because an air quality measurement station is placed nearby.

. Figure 22 Map of the new studied area[33]

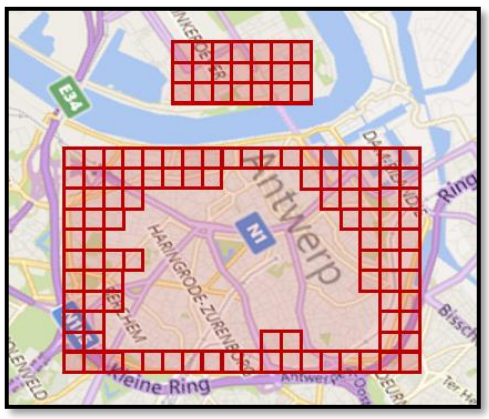

Figure 21 Discretization of the new studied area [34]

[32] Source: own elaboration

[33] Source: own elaboration – Power Bi

The two figures above (20 and 21) show the new studied area and a possible discretization of it. In the figure below (figure X) the solution of the application of the new restrictions to the Clarke and Wright algorithm is shown.

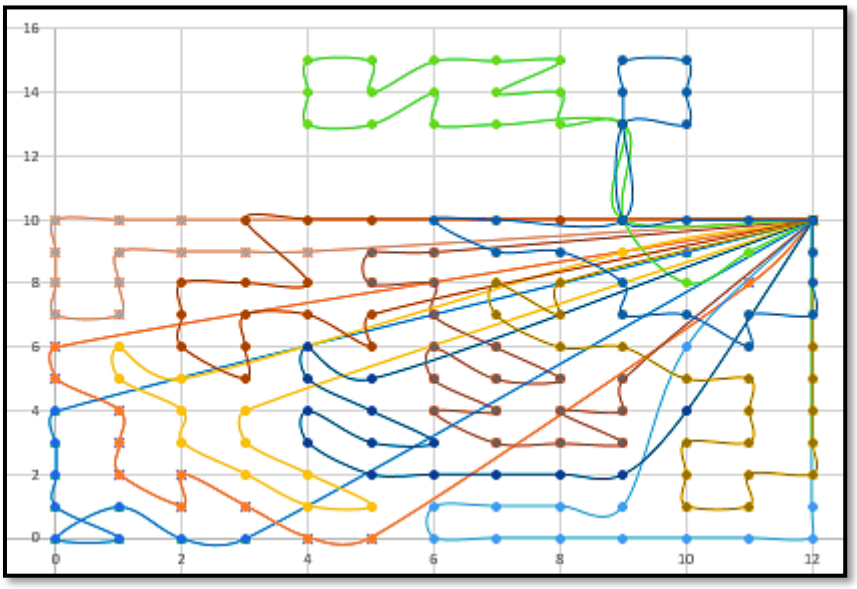

Figure 23 Solution with the new study area[34]

The algorithm obtained at this point calculates the minimum number of routes given a specific size of the regions for discretizing the city. This program was used in a new function that given the number of vans available, calculates the minimum size of the regions reachable.

3.3.5 Finding the set of good solutions

As explained earlier in this report (literature review section *2.2.2 Kind of algorithms*), a heuristic methodology such as the one proposed here is capable of finding good solutions but does not ensure the optimum. For this reason, it is interesting to find not only a solution but also a set of solutions: when applying different post-processing methods to these solutions, a solution that initially is not good could outperform the best one. Different methodologies have been described in this section.

The default program makes square regions for the discretization but a variant was created that allow rectangular regions, always making all the rectangles of the same size. This variant is not just useful for minimizing the maximum distance between measurements even more

[34] Source: own elaboration

(which does not always apply), but for making different solutions that can be used in the following stages.

Another variant was developed which calculates solutions with half routes (this means that the vans would do two routes each day) or using a mixture of both full and half routes. A priori this change would make the routes better distributed in time, as long as it keeps the same size of the regions of discretization.

More good routes were obtained modifying the savings algorithm (Clarke and Wright algorithm). As explained before, the algorithm merges the two routes associated with the largest (remaining) savings starting at the top of the (remaining) list of savings. But what happens when two options with similar savings are available? By randomizing the choice between similar savings the program could generate more good solutions that can be useful later.

Generate more solutions - method	Number of achievable solutions (approx)
Randomize algorithm C&W	10
Rectangle discretization regions	2
A mix of half and full routes	15

Table 1 Methods for the generation of solutions

3.3.6 Post-processing

Once the different solutions are obtained the routes are transformed to be continuous instead of being formed by points, and then they are discretized again but changing the size of the regions to the half. With this process, it is possible to determine which route is better distributed in space. It will be the one covering as many points as possible.

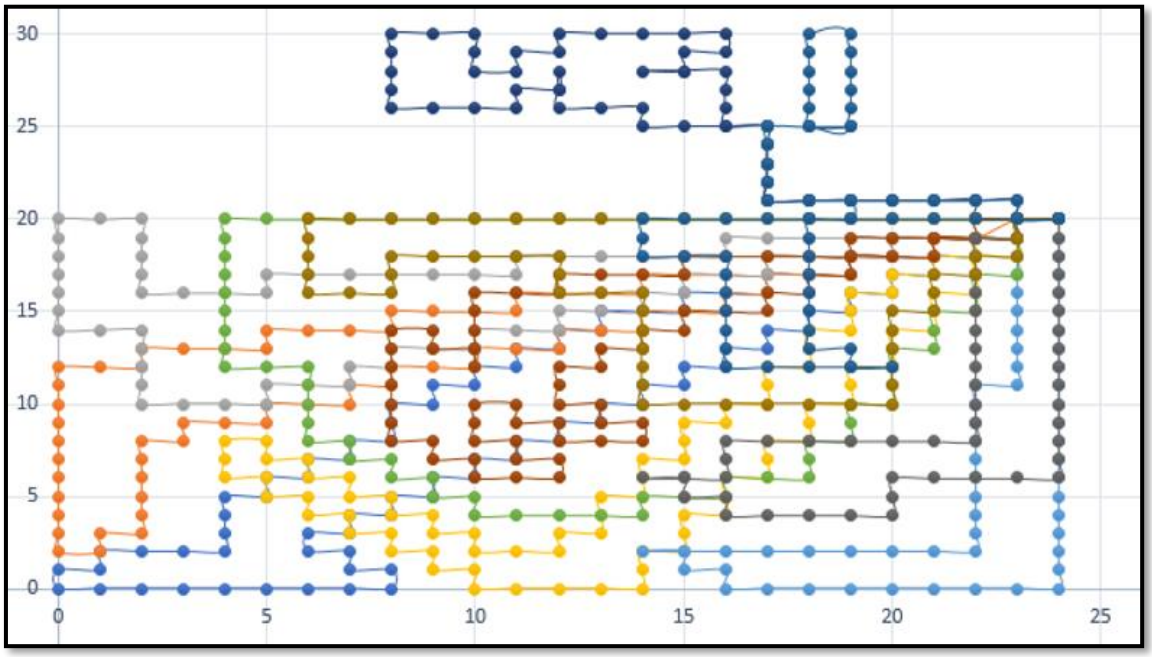

Figure 24 Plot of a solution re-discretization[35]

If the route has not reached the maximum capacity, that is to say, that the van still has time to make some small deviation. The program calculates where this deviation should be done in order to improve the number of points reached.

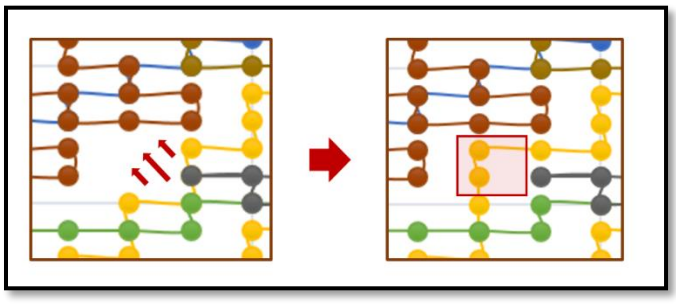

Figure 25 Deviation of routes to improve the distribution of the points[36]

[35] Source: own elaboration

[36] Source: own elaboration

Finally, to improve the distribution on time of the vans the strategy adopted was choosing the direction of the routes, then the sum of the distance between the vans for each moment of time is calculated for each combination of the routes in the two possible directions, and the one that maximizes this summation is chosen.

The table below shows an example of 3 different routes (*A, B* and *C*) that could be chosen in two directions (*0* or *1*). The eight possible combinations are evaluated to choose the best one. At first, it is possible to consider getting rid of those combinations that are conjugated with another one already evaluated: *110* will give the same results than *001* for example. This is only true when all the routes reach the maximum capacity, and since usually at least one does not reach the maximum capacity, it was decided not to apply such simplification.

	route		
comb.	A	B	C
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Table 2 Possible combinations for 3 routes

The execution time of this function is relatively long due to all the combinations possible – 12 routes implies 2^{12} combinations. In cases where increasing the number of routes in large numbers is required, some heuristic techniques should be implemented in order to reduce the execution time of the program.

4 RESULTS

4.1 CHAPTER INTRODUCTION

In this chapter the results obtained with the execution of the program are analysed.

First, the different variants of the program to generate solutions are compared in relation to the maximum distance between measurements obtained (chapter 1.2). This was done for a different number of vans available.

Secondly, the impact of post-processes on the solutions created is analysed. The application of post-processes allows the analysis of other aspects and the improvement of the solution.

4.2 GENERATION OF SOLUTIONS

In this first section of the chapter, the solutions generated by the different variants of the program are compared.

4.2.1 Squared and rectangular discretization regions

The minimum distance between measurements achievable has been studied depending on the number of available vans. The results are presented in the table below (Table 3)[37], showing the maximum distance between measures achievable for each number of sensors, ranging between one and thirty.

[37] A full version of this data is available in the annex (table A1)

Number of vans	Number of divisions		Maximum distance (m)		
	X	Y	X	Y	Total
1	4	5	1050.00	1060.00	1060.00
4	7	9	600.00	588.89	600.00
5	8	10	525.00	530.00	530.00
7	10	12	420.00	441.67	441.67
8	11	14	381.82	378.57	381.82
11	13	16	323.08	331.25	331.25
13	13	17	323.08	311.76	323.08
15	15	19	280.00	278.95	280.00
17	16	20	262.50	265.00	265.00
20	17	22	247.06	240.91	247.06
23	18	23	233.33	230.43	233.33
26	20	25	210.00	212.00	212.00
29	21	26	200.00	203.85	203.85

Table 3 Maximum distance between measurements by number of vans

For the correct analysis it is important to bear in mind that, as mentioned above, the program divides the entire city into rectangular regions. Having more regions implies that the measurements will be closer to each other but also that more sensors will be needed. When dividing the city into regions it is considered that all their dimensions are the same, which implies that the length of each region is exactly the result of dividing the full length of the studied area by an integer.

In this case, the number of divisions in X and Y (perpendicular and parallel to the river respectively) is set so that the rectangular regions have approximately the same length in the X and Y axis, i.e. the regions are as squared as possible. These results have been compared with those derived from relaxing this restriction by allowing non-squared regions. Table 4 (complete in the annex as table A1) shows the results for this second case. A column with the results obtained previously has been added to show the improvement of using rectangular-shaped regions.

Number of vans	Number of divisions		Maximum distance (m)			
	X	Y	X	Y	Total	Before
1	4	6	1050.00	883.33	1050.00	1060.00
4	8	11	525.00	481.82	525.00	600.00
5	8	10	525.00	530.00	530.00	530.00
7	10	13	420.00	407.69	420.00	441.67
8	12	14	350.00	378.57	378.57	381.82
11	13	16	323.08	331.25	331.25	331.25
13	13	17	323.08	311.76	323.08	323.08
15	16	19	262.50	278.95	278.95	280.00
17	16	21	262.50	252.38	262.50	265.00
20	17	22	247.06	240.91	247.06	247.06
23	19	23	221.05	230.43	230.43	233.33
26	20	25	210.00	212.00	212.00	212.00
29	21	26	200.00	203.85	203.85	203.85

Table 4 Maximum distance between measurements by number of vans using rectangular-shaped regions

As expected, the result never gets worse in this new variant, but the improvement is not significant after 8 vans. This is also evidenced in Figure 25 where the two solutions are compared for a different number of vans ranging from 2 to 30.

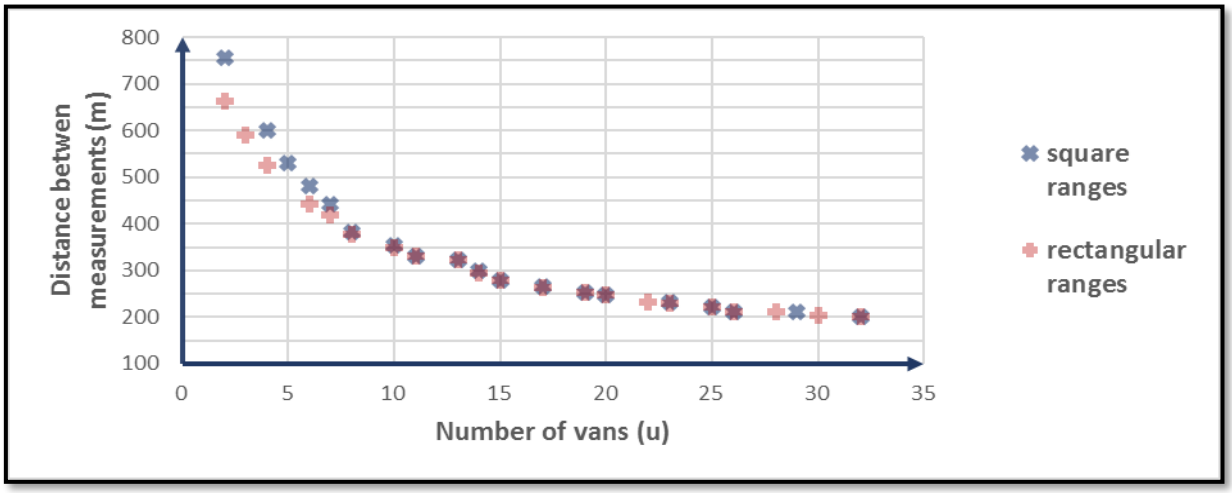

Figure 26 Graphic of the distance between measurement depending on the number of vans

In this last figure, it is also possible to see the behaviour of the distance depending on the number of vans. The distance between the measures decreases rapidly while increasing the number of vans from 1 to 10 and very slowly after the 25th. In consequence it is possible to say that there is no appreciable benefit, in terms of distance between measurements, on putting one more van in circulation after reaching 25 vans.

4.2.2 Full and half routes

Solutions have been obtained in the two ways mentioned above (allowing or not rectangular regions) while reducing the travel distance of the vans by half. In this way, twice as many routes are obtained approximately but each van can make two routes each day. Other cases have also been analysed where part of the vans make half routes and the other part complete routes.

Initially, the number of vans required should be increased because each van has to return to the parking lot twice and therefore the distance to travel will be greater. It has been observed instead that, in some cases, it is possible to reduce the number of vans required to cover the city while maintaining the same discretization.

The problem is the Clarke and Wright algorithm that covers the same zone more than once, increasing the distance total travelled, requiring more vans in consequence. In these cases, it would be possible to reduce the number of vans needed for complete routes with an improvement of the program. This improvement is explained in section *5.2.3 "Re-discretize when creating route"*.

The following table shows the number of necessary vans according to the regions used to discretize the city with the option of half routes as well as complete routes. The complete table, with all the analysed regions, is shown in Annex A1.

Type of discretization	Number of divisions		Number of vans needed	
	X	Y	full routes	half routes
Square	12	15	9	9
Rect.	13	15	10	9
Rect.	12	16	9	9
Square	13	16	10	10
Rect.	14	16	12	11
Square	13	17	11	11
Rect.	14	17	14	13
Rect.	13	18	14	13
Square	14	18	15	14
Rect.	15	18	15	14
Rect.	14	19	16	15

Table 5 Number of vans needed by discretization and kind of routes (half or full routes)

It is observed that this variant of the program manages to reduce the number of vans needed for different sizes of discretization in most cases. It means that given a specific number of vans available it is possible to discretize the city in smaller regions, so the new variant will improve the solution in terms of the distance between measurements.

4.2.3 Algorithm randomization

The last of the variants contemplated in this work consists in the randomization of the Clarke and Wright algorithm. So far the algorithm has been generating the routes choosing point by point as explained in the implementation chapter. The new variant instead of choosing the best candidate (point) each time to join one of the routes created will choose randomly between the best 5 candidates. In this way it is possible to generate a different solution each time the program is executed, retrieving good solutions and potentially bad solutions as well.

The algorithm randomization variant was executed along with the previous variants, getting the results shown in the following tables:

Type	Number of divisions		Full routes		Half routes	
	X	Y	improvement	degradation	improvement	degradation
Square	12	15	0%	0%	80%	0%
Rect.	13	15	40%	20%	0%	40%
Rect.	12	16	0%	0%	0%	0%
Square	13	16	0%	20%	0%	40%
Rect.	14	16	60%	0%	0%	0%
Square	13	17	0%	0%	0%	0%
Rect.	14	17	0%	0%	0%	0%
Rect.	13	18	40%	0%	0%	0%
Square	14	18	0%	0%	0%	0%
Rect.	15	18	80%	0%	0%	40%
Rect.	14	19	40%	0%	80%	0%

Table 6 Percentage of improvement and degradation because of the randomization of the algorithm

If the random solution manages to reduce the number of vans needed for one combination it means that the solution is better: with the same number of vans now it is possible to discretize the city in smaller regions, reducing the distance between measurements. It is considered then, that the solution improves if the number of vans needed is reduced for the same discretization and that the solution is degraded if the number of vans is increased. For each combination, the program with the randomization has been executed 5 times and compared with the solution obtained without this modification. The value shown is the percentage of cases where there was an improvement/degradation from the total cases evaluated for that combination.

Table 7 shows the summary of how randomization affects each variant of the program. This table shows the same results as the previous table but aggregated to show a more generic view. The values are calculated in the same way (percentage of cases where there was an improvement/degradation from the total cases evaluated) Finally, of all the analysed cases, it has been obtained that in 17% of the cases the solution is improved while at 8% it is degraded.

	Improvement			Degradation		
Type \ Routes	**Full**	**Half**	**Total**	**Full**	**Half**	**Total**
square	0%	20%	**10%**	5%	10%	**8%**
rectangular	33%	10%	**22%**	3%	13%	**8%**
Total	**22%**	**13%**	**17%**	**3%**	**12%**	**8%**

Table 7 Improvement and degradation percentage because of the randomization of the algorithm by a variant of the program

For the results shown up to this moment in this section, the program is supposed to choose randomly between the 5 best candidates of the algorithm Clarke and Wright. Alternatives such as randomly choosing from the first 3 and 8 top candidates have also been analysed. The results obtained are the following:

Choose between	Improvement	Degradation
1	0%	0%
3	7%	4%
5	17%	8%
8	14%	16%

Table 8 Total percentage of improvement and degradation of the solution by the configuration of the randomization

According to this results, the option of providing the best results is to randomize among the first five.

4.3 POST-PROCESSING ANALYSIS

This section addresses the post-processing applied to the solutions found through the different methodologies. The goal of the program is to create the best routes possible given a specific number of vans. For this reason, in this section it has been decided to set the number of sensors, specifically to 15 (that was the number of sensors available at the beginning of 2018, when the analysis of the system was done). In this way it is possible to analyse how the post-processing improves the solution that the program finds given a specific number of vans.

4.3.1 Preselection of solutions

The first step is to generate solutions and remove those that are not useful. The analysis carried out up to this point allows us to know around which discretization value we will find solutions with 15 vans, as specified below in table 9. The program is executed for different dimensions of discretization obtaining solutions with the different variants of the program.

A non-random solution and 10 random ones have been generated for each of the possible combinations of the program variants. The table below shows the number of solutions obtained for each configuration according to the number of vans that this solution need (14, 15 or 16).

	Type	Number of divisions		Solutions found		
		X	Y	14 Vans	15 Vans	16 Vans
Full routes	Square	14	18	0	11	0
	Rect.	14	19	7	4	0
	Rect.	15	18	0	4	7
	Square	15	19	0	9	2
	Rect.	15	20	0	0	11
	Rect.	16	19	0	0	11
Half routes	Square	14	18	11	0	0
	Rect.	14	19	11	0	0
	Rect.	15	18	0	11	0
	Square	15	19	0	11	0
	Rect.	15	20	0	0	11
	Rect.	16	19	0	0	11
			Total	29	49	52

Table 9 Number of solutions found by a variant of the program and the number of vans needed.

Notice that from specific cases of discretization solutions with only 15 sensors are not obtained anymore, and that equivalently if the number of divisions is reduced then solutions with fewer vans are obtained. This is clearly shown in the following table:

Number of divisions		Solutions with 15 vans
X	Y	
smaller regions		0
14	17	0
14	18	11
14	19	3
15	18	15
15	19	20
15	20	0
bigger regions		0
	Total	**49**

Table 10 number of solutions that need 15 vans by number of divisions

Out of all the solutions generated only the ones containing 15 routes (i.e. 49 solutions) are studied below. At this point, it might be possible to dismiss the solutions with fewer divisions since a priori these would be worse. Removing these solutions at this point means to reduce the number of times the post-processing is executed, reducing the overall execution time this way. However, these solutions were kept considering the following:

- To be able to contrast the different derived solutions.
- These solutions might be better in aspects other than the distance between measurements.
- As mentioned above, in a combinatorial problem like this one, when applying certain post-processing to each solution, one that is not as good at first may turn out to be the best one.

4.3.2 Space distribution

The dimensions of the discretization region guarantee a maximum distance between measurements but do not guarantee that the routes are optimally distributed in space or time. In order to determine which route is best distributed in space, the post-processing discussed earlier in section *3.2.5* is applied.

As it was mentioned before, each route of the different solutions consists of a set of ordered points. These points depend on the dimensions of the discretization regions, so solutions are notably different from one another.

With the space-distribution post-processing, all the routes are transformed to continuous and then discretized again with the same dimensions of the discretization regions. for this procedure 27 divisions were made on the X-axis and 35 on the Y axis (parallel and perpendicular to the river).

The initial solutions meet a series of restrictions that the new solutions do not comply with. These restrictions are specific to the Clarke and Wright algorithm:

- Each point belongs to only one route except the origin.
- All points must be reached.

Therefore, this methodology allows calculating the coverage ratio, which consists in dividing the number of points reached by the solution among the total points in which the city is divided.

The results obtained are shown in the following table (Table X). Out of the different random solutions for the same configuration, similar ratios are obtained. The best ratios are shown in the table below. Note also the column with the maximum distance between measurements that have been added to see both aspects of the solutions.

	Type	Number of divisions		Best solution - Rate		Distance between measurements
		X	Y	Not random	Random	
Full routes	**Square**	14	18	72.3%	76.6%	300
	Rect.	14	19	76.3%	76.2%	300
	Rect.	15	18	no solution	75.6%	294
	Square	15	19	77.9%	77.8%	280
Half routes	**Rect.**	15	18	82.6%	84.9%	294
	Square	15	19	83.9%	83.9%	280

Table 11 Best coverage rate of the solutions by program variant

It can be seen that half-route solutions allow improving the coverage ratio by more than 5%. Also, the random solutions manage to improve the ratio in some cases. It is also possible to observe that the best solution in terms of coverage ratio does not match the best in terms of the distance between measurements. One would have to assess in which cases one is more favourable over the other.

At this point, we could choose to eliminate all those solutions that do not exceed an 80% coverage ratio.

The following two graphs show the routes of one of the solutions with 72% coverage ratio and one of 85%. From these figures it is possible to notice a substantial difference between the two of them: It is observed that the second figure has many more points leaving less and smaller blank areas.

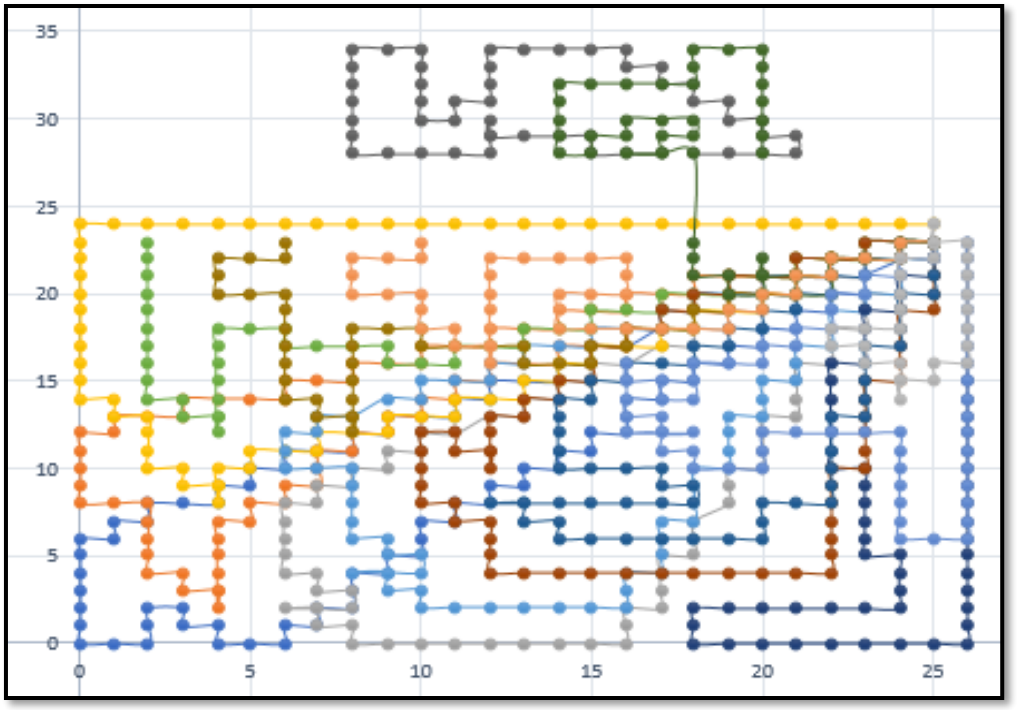

Figure 27 Solution example for 15 vans with 72% of coverage ratio[38]

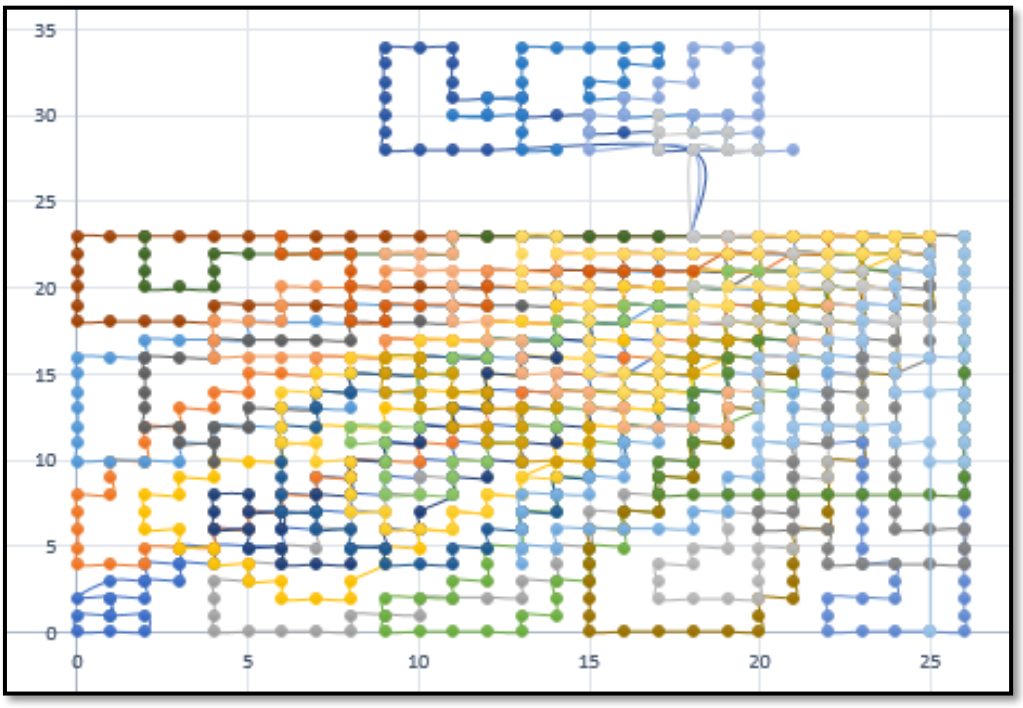

Figure 28 Solution example for 15 vans with 85% of coverage ratio[39]

[38] Source: own elaboration

[39] Source: own elaboration

Next, an improvement processing has been applied to each solution that diverts the routes in order to increase the coverage ratio, always bearing in mind that the route cannot exceed a certain distance. This type of methodology is known as "neighbourhood exploration" and has been explained in the literature review (section *2.2.3 Useful methodologies*)

This last process improves the solution gradually but is relatively slow. In some cases, it could take days to find that it cannot improve the solution any further. To avoid this, the execution time of the program has been fixed to 60 seconds. More time would mean for the program to come up with a better solution. In this sense, for cases where there are no strict constraints in terms of execution time a running this post-processing can be useful.

Finally, by applying the process above it is possible to increase the coverage ratio of the solutions by 4.5% on average. The results obtained are shown in the table below:

		Number of divisions		Best solution - Rate	
	type	X	Y	no aleat	aleat
Full routes	**Square**	14	18	75.9%	82.4%
	Rect.	14	19	81.2%	80.8%
	Rect.	15	18	no solution	79.4%
	Square	15	19	80.9%	83.1%
Half routes	**Rect.**	15	18	88.4%	90.7%
	Square	15	19	87.6%	89.0%

Table 12 Best coverage ratio of the solutions by program variant after changing the routes

A figure with the effect of the improvement processing applied on the routes of one of the solutions introduced earlier is also presented below in figure 28. It is possible to observe that such a solution covers many more points but still some remain yet unvisited.

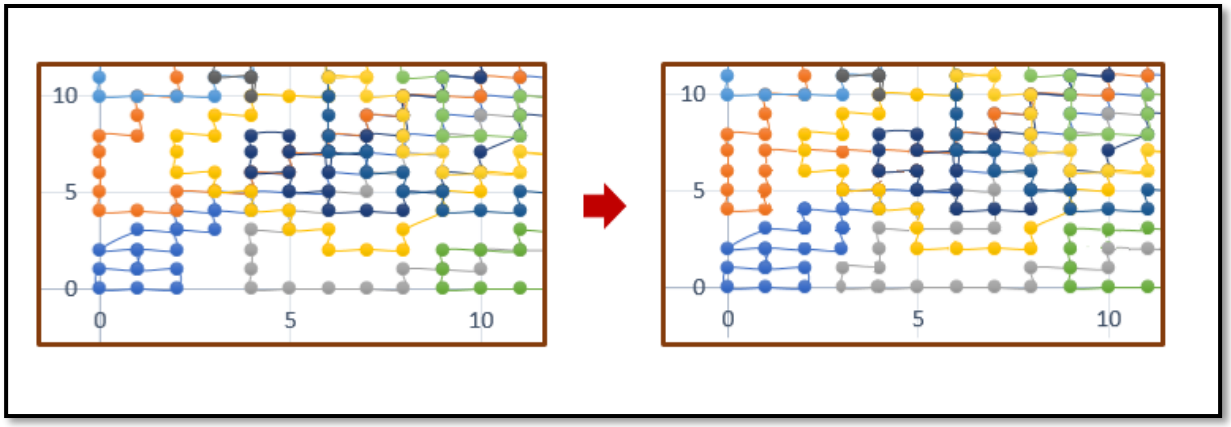

Figure 29 Example of the changes produced in one solution (graphic of the solution before and after applying the post-process)[40]

4.3.3 Time distribution

The sum of the distances between the vans at any given point in time has been calculated for each solution. Although this computation does not convey any useful information, it allows us to compare the different solutions among each other. The solution whose sum is higher will be the best distributed in time.

The table below shows how each solution is distributed in relation to the most distributed solution obtained. This allows knowing how close each solution is to the best solution achievable in terms of time distribution.

		Number of divisions		Best solution - Rate	
	type	X	Y	no aleat	aleat
Full routes	Square	14	18	51.9%	60.6%
	Rect.	14	19	49.0%	56.6%
	Rect.	15	18	no solution	61.0%
	Square	15	19	53.7%	56.2%
Half routes	Rect.	15	18	90.2%	95.3%
	Square	15	19	86.1%	92.7%

Table 13 Distribution rate by a variant of the program

[40] Source: own elaboration

The program described in section *3.2.5* has been used to choose the direction (forward or backwards) of the routes over all the obtained solutions. All the possibilities of choosing different directions for each of the routes of one solution are combinations that should be explored. In this way, solutions with better distribution in time have been obtained.

Only some of the combinations were further explored (by arbitrarily selecting part of them) since studying all of the 46 different solutions would take a prohibitive amount of time. The results are shown in the following table.

		Number of divisions		Best solution - Rate	
	type	X	Y	Non-random	random
Full routes	**Square**	14	18	56.5%	61.9%
	Rect.	14	19	53.4%	57.8%
	Rect.	15	18	no solution	63.0%
	Square	15	19	58.7%	57.0%
Half routes	**Rect.**	15	18	94.9%	100.0%
	Square	15	19	91.1%	94.9%

Table 14 Distribution rate by a variant of the program after choosing the direction of the routes

The best solution again comes from using half routes. Choosing the order and direction of these routes allows you to improve the solution by up to 5%.

4.3.4 Best solution

To conclude this section, the following table shows the configuration of the best solution for each of the important aspects of it:

		Configuration			Score		
		Rectangular	Half route	Random	Distance measures	Space distrib.	Time distrib.
Best solution in	**distance**	**no**	**yes**	**yes**	280	89.0%	94.9%
	space	**yes**	**yes**	**yes**	294	90.7%	100.0%
	time	**yes**	**yes**	**yes**	294	90.7%	100.0%

Table 15 Best solutions summary

This is the result found in the case of having 15 vans, in other cases a solution with another type of configuration may outperform this one. That is why, it is always appropriate to work with all the solutions without discarding them until the end.

4.4 CHAPTER SUMMARY

Finally, in this chapter it has been seen that all the variants that generate solutions are useful as they allow to generate a set of good solutions with minimum distance between measurements. This was achieved by changing the size of the discretization regions, by creating two smaller routes for each van and by randomizing the method to create the routes.

It was possible to notice that by applying the described post-processing it is possible to compare and improve the different solutions generated in aspects different from the distance between measurements. Specifically, these aspects are the space and time distribution.

5 CONCLUSION

5.1 REVIEW

In this work the problem of optimizing the routes of the non-stationary air pollution sensors in the City of Antwerp was analysed. The system behind this problem was simulated, along with a program that optimizes it in the following aspects:

- Minimizes the distance between measurements.
- Minimizes the total distance travelled.
- Distribute the routes in the space.
- Distribute the routes in time.

Another goal this project went after was to maintain an affordable execution time of the optimization procedure. In this sense, the implemented program should be able to find solve the optimization problem in a reasonable time, without compromising too much the quality of the solutions it comes up with.

As previously mentioned in a combinatorial program, to find the ultimate optimal solution and involves evaluating all the possibilities. The limiting factor then is time, more time means applying more methods, evaluating more possibilities and therefore finding a better solution.

Considering the stated time constraint, some of the proposed approaches and variants of the program have not been actually implemented because their execution would take a prohibitively long span of time. However, since these variants might be useful to improve the solution in some of the aspects or bring the system closer to reality, the following section briefly elaborates on them.

The program obtained determines the routes that each vehicle from a fleet of n delivery vans has to follow to cover a certain area inside the city of Antwerp. This program could be adapted to other types of systems by applying the necessary restrictions.

In the context of Antwerp's "City of Things" program, this program could be used to determine the routes that garbage trucks should travel every day.

Currently, a project is being carried out that studies the possibility of placing a sensor in garbage containers to know their status always. The program carried out in this work could use the data collected by these sensors to make a list of the points that need to be covered and then determine which truck should empty each container.

Figure 30 Sensor in a garbage container[41]

Another possible application could be in the field of drones, technology which is booming right now. It could be used to determine the routes to be covered by these drones to measure different parameters of air quality, or even sensing traffic conditions.

5.2 FUTURE WORK

This chapter summarizes some possible improvements that can be applied to the program implemented in this project to enhance both the quality of the solution and the simulation of the problem conditions (making it closer to real-world conditions). These variants are:

- Improve time distribution
- Make queries to Google Maps
- Re-discretize when creating routes
- Variable mesh
- Half routes improvement

[41] Source : www.reuters.com/brandfeatures/venture-capital/article?id=37038

5.2.1 Improve time distribution

The implemented program obtains the best temporal distribution among the solutions studied. That is, it creates the solutions by optimizing the distance between measurements and the spatial distribution of the routes, then it sets the direction of the routes for each solution for which the best the temporal distribution of routes is achieved, to finally choose the best solution.

The temporal distribution becomes a secondary criterion to derive the solution. It would be interesting to change this procedure to obtain better-distributed solutions in time and, perhaps, less distributed in space.

Specifically, an option would be diverting the routes maximizing the temporal distribution, once the routes are created with the Clarke and Wright algorithm.

5.2.2 Make queries to Google Map

This is the most interesting improvement, since it would bring the simulation very close to reality. The first step for the Clarke and Wright algorithm is to create a matrix where the cost of going from each point of the city to any other point is saved. Currently this cost is a function of the distance between these two points. If replaced by real-time, taking into account the roads available between the two points, the routes would change completely adapting to the actual road infrastructure of the city.

To implement this the only thing that would be needed when creating the said matrix, is to send queries to Google maps and save the traveling times. Google should authorize these queries and the response of said queries should be negligible so it does not compromise the processing time too much.

5.2.3 Re-discretize when creating a route

This improvement was not implemented because it would mean to increase the processing time too much. It consists in re-discretize when creating the routes. When the Clarke and Wright algorithm chooses a route to go from one point to another, it does not take into account that during this journey it may be going through other points in between.

If the routes were first transformed to continuous paths and then discretized again as they are created, it could be determined which points have actually been covered. This would further optimize the allocation of routes to vans, by avoiding going more than twice through certain points in the coverage area.

The algorithm generates the routes point by point, modifying the routes every time, therefore it would be needed to re-discretize the routes every time: 100 points would mean that all the routes have to be re-discretized at least 100 times. For this reason, this change would imply a significant increase in the program's execution time.

5.2.4 Variable mesh

This improvement could reduce the distance between measurements of the initial solution. It's about not dividing the city into regions of the same size but making them smaller as they get far from the parking of the fleet of sensing vehicles.

When plotting one of the re-discretized solutions of section *4.3.2* it is observed that covering the points that are far from the origin entails more cost in contrast to those closer to the origin. A smaller mesh in these farther areas would allow increasing the coverage ratio and reducing the minimum distance between measurements. The following chart shows the areas where this variable mesh approach would be beneficial.

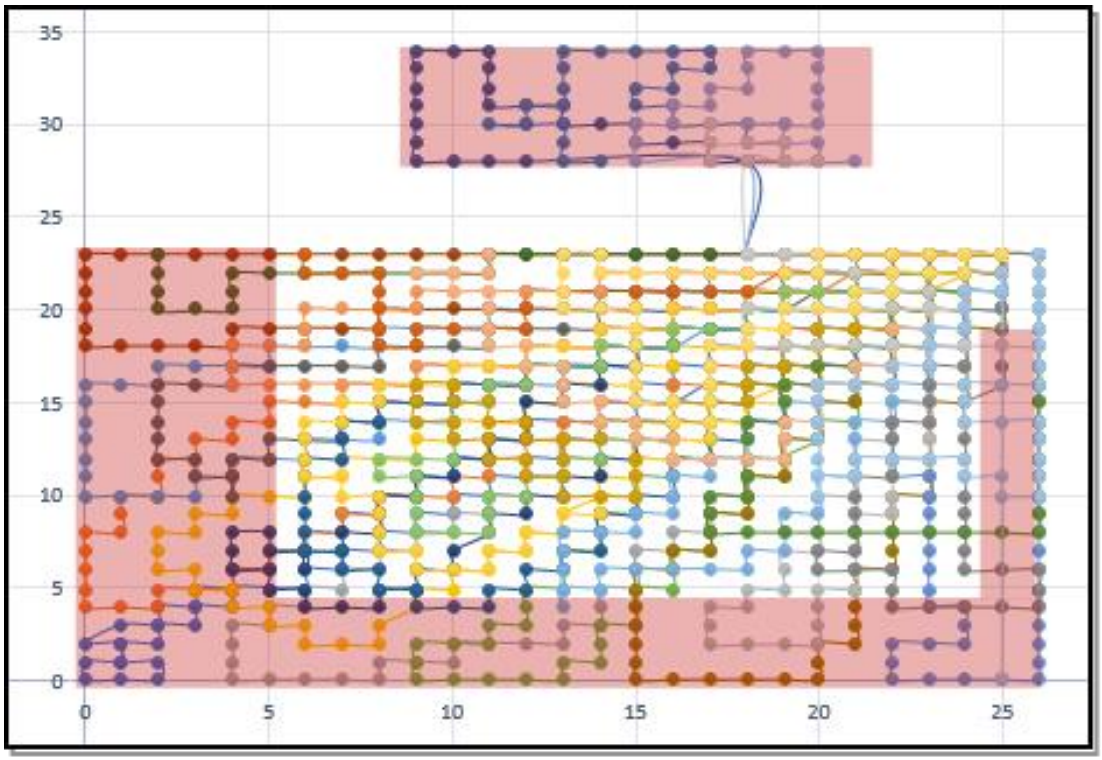

Figure 31 Graphic of the routes of one solution whit the zones where a more accurate mesh would be useful[42]

5.2.5 Half routes improvement

According to the results obtained when using 15 non-stationary sensors, the variant of the problem involving the use of half-routes provides the best performance. There are more alternatives to be considered that can potentially improve the solution. Some examples could be a van making a part of its journey with one route and the remaining part with a different one (the option studied only contemplates 50% -50%) or perhaps combine long routes with short ones.

[42] Source: own elaboration

5.3 PERSONAL CONCLUSION

As a personal conclusion this work has allowed me to apply different theoretical methodologies that I have been learning throughout my studies, to a real-world problem.

When initially tackling the problem, my approach was very broad as I wanted to take into account every possible variable involved in the system's behaviour, without considering any simplification. I think it has been especially interesting to overcome through logic the different challenges I faced, and to solve the problems and realizing the limits that I have found, looking for new alternatives when running into an issue.

After undertaking this project I am attracted to the idea of continuing to learn about simulation and optimization of systems, even considering to further develop my professional career in this field.

6 REFERENCES

[1] Santos J., Wauters T., Volckaert B., De Turck F. (2018) "Anomaly detection for Smart City applications over 5G Low Power Wide Area Networks", *NOMS2018, the IEEE/IFIP Network Operations and Management Symposium* 1-9.

[2] Martí R., Reinelt G. (2011) "Exact and Heuristic Methods in Combinatorial Optimization" 1-172.

ANNEXES

Annex A Results obtained while analysing the generation of solutions

A.1 Table of the number of vans needed and the maximum distance between measurements depending on the dimensions of the discretization regions.

Shape of regions	Number of divisions		Number of vans	Distance between measurements (m)		
	X	Y		X	Y	Total
Square	4	5	1	1050	1060	1060
Rectangular	4	6	1	1050	883	1050
Square	5	6	2	840	883	883
Rectangular	6	7	2	700	757	757
Rectangular	7	8	2	600	663	663
Square	7	9	3	600	589	600
Rectangular	8	9	3	525	589	589
Square	8	10	4	525	530	530
Rectangular	8	11	4	525	482	525
Square	9	11	6	467	482	482
Rectangular	9	12	6	467	442	467
Square	10	12	6	420	442	442
Square	10	13	7	420	408	420
Rectangular	11	14	8	382	379	382
Rectangular	12	14	8	350	379	379
Square	12	15	10	350	353	353
Rectangular	12	16	10	350	331	350
Square	13	16	11	323	331	331
Rectangular	13	17	13	323	312	323
Rectangular	14	18	14	300	294	300
Rectangular	15	18	14	280	294	294
Square	15	19	15	280	279	280
Rectangular	16	19	15	263	279	279
Square	16	20	17	263	265	265
Rectangular	16	21	17	263	252	263
Square	17	21	19	247	252	252
Rectangular	17	22	20	247	241	247
Rectangular	18	23	23	233	230	233
Rectangular	19	23	23	221	230	230
Square	19	24	25	221	221	221
Rectangular	20	24	25	210	221	221
Square	20	25	26	210	212	212
Rectangular	20	26	28	210	204	210
Square	21	26	30	200	204	204

Annex B Results obtained while analysing the post-process

The tables of this annex show the results obtained depending on the variant of the program. Notice that for each row the N column is the result of executing the program without the randomization of the algorithm. The results obtained each time the program with the randomization algorithm was executed are presented from the column A to the J.

B.1 Table of the number of vans needed depending on the size of the discretization regions

Number of divisions			Number of vans needed									
X	Y	Half routes	N	A	B	C	D	E	F	G	H	I	J
12	15	No	9	9	9	9	9	9	9	9	9	9	9
12	15	Yes	9	8	8	8	8	9	8	8	8	8	9
13	15	No	10	11	10	9	9	10	11	10	9	9	10
13	15	Yes	9	10	10	9	9	9	10	10	9	9	9
12	16	No	9	9	9	9	9	9	9	9	9	9	9
12	16	Yes	9	9	9	9	9	9	9	9	9	9	9
13	16	No	10	10	11	10	10	11	10	11	10	10	11
13	16	Yes	10	10	10	10	10	10	10	10	10	10	10
14	16	No	12	11	11	12	12	11	11	11	12	12	11
14	16	Yes	11	11	11	11	11	11	11	11	11	11	11
13	17	No	13	13	13	13	13	13	13	13	13	13	13
13	17	Yes	12	12	12	12	12	12	12	12	12	12	12
14	17	No	14	14	14	14	14	14	14	14	14	14	14
14	17	Yes	13	14	13	13	13	13	14	13	13	13	13
13	18	No	14	14	14	14	13	13	14	14	14	13	13
13	18	Yes	13	13	13	13	13	13	13	13	13	13	13
14	18	No	15	15	15	15	15	15	15	15	15	15	15
14	18	Yes	14	14	14	14	14	14	14	14	14	14	14
14	19	No	15	14	14	14	15	14	14	14	14	15	14
14	19	Yes	14	14	14	14	14	14	14	14	14	14	14
15	18	No	16	16	15	16	15	16	16	15	16	15	16
15	18	Yes	15	15	15	15	15	15	15	15	15	15	15
15	19	No	15	15	15	16	15	15	15	15	16	15	15
15	19	Yes	15	15	15	15	15	15	15	15	15	15	15
15	20	No	16	16	16	17	16	16	16	16	17	16	16
15	20	Yes	16	16	16	16	16	16	16	16	16	16	16
16	19	No	16	16	16	16	16	16	16	16	16	16	16
16	19	Yes	16	16	16	16	16	16	16	16	16	16	16

B.2 Table of the coverage ratio of each solution that needs 15 vans

Number of divisions			Coverage ratio (%)										
X	Y	Half routes	No	A	B	C	D	E	F	G	H	I	J
14	18	No	72.3	74.7	75.5	74.8	75.9	75.2	74.7	76.4	75.4	76.6	74.8
14	19	No	76.3				76.0					76.2	
15	18	No			75.6		75.1			75.6		74.3	
15	18	Yes	82.6	83.2	83.1	84.9	84.8	83.6	83.4	82.4	84.9	84.5	83.7
15	19	No	77.9	75.7	74.8		76.5	77.7	75.8	75.0		77.8	76.6
15	19	Yes	83.9	83.8	83.4	83.1	83.4	83.0	83.9	83.8	82.9	82.7	83.3

B.3 Table of the coverage ratio of each solution that needs 15 vans after changing the routes

Number of divisions		Half	Coverage ratio (%)										
X	Y	routes	No	A	B	C	D	E	F	G	H	I	J
14	18	No	75.9	78.6	79.9	80.0	80.1	78.5	78.5	80.2	78.9	82.4	79.3
14	19	No	81.2				79.9					80.8	
15	18	No			78.9		78.2			79.4		77.6	
15	18	Yes	88.4	88.8	88.7	88.6	89.4	88.1	88.3	86.9	90.7	90.5	87.5
15	19	No	80.9	80.8	79.5		79.7	83.1	79.0	78.5		82.7	80.7
15	19	Yes	87.6	87.0	87.3	88.7	88.4	86.3	89.0	87.6	85.9	86.9	88.4

B.4 Table of the value of distribution found for each solution with 15 vans

N. of divisions		Half	Distribution value										
X	Y	routes	No	A	B	C	D	E	F	G	H	I	J
14	18	No	3616	3819	4125	4098	3959	4218	3819	4125	4098	3959	4218
14	19	No	3409				3941					3941	
15	18	No			4250		4116			4250		4116	
15	18	Yes	6281	6634	6504	6337	6468	6203	6634	6504	6337	6468	6203
15	19	No	3739	3755	3779		3634	3913	3755	3779		3634	3913
15	19	Yes	5992	5831	6328	5954	6456	6278	5831	6328	5954	6456	6278

B.5 Table of the value of distribution found for each solution with 15 vans after choosing the direction of the routes

N. of divisions		Half	Distribution value										
X	Y	routes	No	A	B	C	D	E	F	G	H	I	J
14	18	No	3932	3849	4307	4248	4178	4297	3849	4307	4248	4178	4297
14	19	No	3715				4024					4024	
15	18	No			4388		4168			4388		4168	
15	18	Yes	6610	6842	6963	6545	6636	6679	6842	6963	6545	6636	6679
15	19	No	4088	3785	3923		3969	3953	3785	3923		3969	3953
15	19	Yes	6342	6196	6608	6369	6595	6567	6196	6608	6369	6595	6567

B.6 Table of the distribution rate of each solution that needs 15 vans

N. of divisions			Distribution rate										
X	Y	Half routes	No	A	B	C	D	E	F	G	H	I	J
14	18	No	52%	55%	59%	59%	57%	61%	55%	59%	59%	57%	61%
14	19	No	49%				57%					57%	
15	18	No			61%		59%			61%		59%	
15	18	Yes	90%	95%	93%	91%	93%	89%	95%	93%	91%	93%	89%
15	19	No	54%	54%	54%		52%	56%	54%	54%		52%	56%
15	19	Yes	86%	84%	91%	86%	93%	90%	84%	91%	86%	93%	90%

B.7 Table of the distribution rate of each solution that needs 15 vans after choosing the direction of the routes

N. of divisions		Half	Distribution Rate										
X	Y	routes	NO	1	2	3	4	5	6	7	8	9	10
14	18	No	56%	55%	62%	61%	60%	62%	55%	62%	61%	60%	62%
14	19	No	53%				58%					58%	
15	18	No			63%		60%			63%		60%	
15	18	Yes	95%	98%	100%	94%	95%	96%	98%	100%	94%	95%	96%
15	19	No	59%	54%	56%		57%	57%	54%	56%		57%	57%
15	19	Yes	91%	89%	95%	91%	95%	94%	89%	95%	91%	95%	94%

Route optimization for non-stationary air pollution sensors in the City of Antwerp

David Parrilla

Promotor: Prof. Filip De Turck, dr. ir. Gregory Van Seghbroeck,

Supervisor: Leandro Ordoñez Ante

Master's dissertation from an exchange student of the

Master of Science in industrial engineering and operational research

Department of Information Technology

Chairman: Prof. Gregory Van Seghbroeck

Faculty of Engineering and Architecture

Academic year 2017-2018

GHENT UNIVERSITY