Rethinking transport
27–30 April 2020

Proceedings of 8th Transport Research Arena TRA 2020, April 27-30, 2020, Helsinki, Finland

A tool to improve the efficiency of waste collection: development and application to a case study

Jesús Aransay[a]*, Víctor Galilea[a], Inés Hernández[a],

[a]*Universidad de La Rioja , Departamento de Matemáticas y Computación, c/ Madre de Dios 53, Logroño 26004, Spain*

Abstract

In this work, we present a real case study of optimization of waste collection routes. In our particular case, the collection area is La Rioja (a region of Spain with an area of 5.045 km^2 and a population of ca. 300.000 people). The starting point of our study was some unstructured data that The Circular Lab (https://www.thecircularlab.com/en/) handed us from the routes that were being used, as well as some particular instances of these routes that had been completed by trucks in their daily routine. From that particular routes we generated our own data model, a middleware and a connection to specialised computing services that allowed us to optimise the routes. These optimisations gave place in some particular cases to savings of up to 40% (either in the distance, the time, or the CO_2 emissions required to complete the routes).

Keywords: waste collection, route planning, data processing, linear programming, high performance computing.

* Corresponding author. Tel.: +34941299438;
E-mail address: jesus-maria.aransay@unirioja.es

1. Introduction

Waste collection requires heavy investments in time, fuel and human resources. It is also a challenging task since its optimisation is known to be an NP-hard problem, Lenstra and Kan (1981), which means that for 'big' instances (in other words, real world cases) its optimal solution cannot be computed in a reasonable amount of time.

The nature and modelisation of the problem varies among particular cases. Some modelisations require the starting point and the depot to be different points (in some others they coincide), some other admit several depots or intermediary collection points, and some other have various trucks with different capacities; a daily periodicity can be also included in the problem; then some other modelisations include periodicities in a time range (for instance, a collection point has to be visited twice every week). These differences, together with the computational complexity of the problem, lead to particular solutions being proposed for particular instances of the problem.

In our particular case, the region of interest is La Rioja, a region in the north of Spain of 5.045km^2 and ca. 300.000 habitants. The population is distributed in some cities and in a relevant number of small villages and rural regions. In this project, that we develop in collaboration with The Circular Lab (an innovation center located in La Rioja for best practices about packaging and recycling), we will focus in paper and plastic collection. There are more than 2300 collection points distributed along 130 villages or towns, and three trucks to perform the collection. The starting point and the depot are located at the same place. The input information of our project was some raw data (in csv format) about the routes being performed at that moment, as well as a small number of routes completed that have been obtained from some GPS located in the trucks in a few particular days.

These data provided us information about the routes being completed and also about the time and distance that was being completed at that time. Our proposal was developing a general framework that could be used for similar problems in waste collection; the framework consists in a set of libraries to retrieve information about the distances among collection points and villages, a data model (that we implemented in MongoDB) to store data about the routes, geographical information and also our computations; a middleware that is used to invoke in a unified way different computations tools (such as OR-Tools, Gurobi and CPLEX) and that can be easily adapted to communicate with new tools, and also a server that uses geovisualisation tools and APIs to present the results and comparisons among routes.

The utility and scalability of the framework is illustrated with our case study, where more than 30GB of data has been generated (both from converting the original data and storing the relevant parts of our computations), and are stored in our database, and time and distance reductions of 30% (mean value along the different routes) have been obtained in the routes that were being completed before our study.

The paper will be divided in four sections. In Section 2 we present the application that we have developed to store the original data as well as the rest of the information generated by the system. In Section 3 we present the different modelisations of the problem that can be used to optimise route computation, as well as their implementation. Then, in Section 4 we present a comparison of the total times from the original routes and from the routes computed in our work.

2. Data model and architecture of the application.

This section will be divided in two parts. In Subsection 2.1 we present the data model that we generated from the information that we got from our costumer, and that shall be applicable to similar problems. In Subsection 2.2 we present the architecture of the application that we have developed to retrieve geographical information, store it, compute routes, and present the routes to the user.

2.1. Data model.

The raw data for this project, provided by The Circular Lab, was in csv format. For better processing, we transformed it to MongoDB, which allows to include flexibility in the data structure (avoiding having to start with an initial data structure that limits changes in the database) and increase efficiency when viewing and editing data (comparing with relational databases, such as those that use SQL). The data structure that we designed and

implemented is shown in Fig. 1.

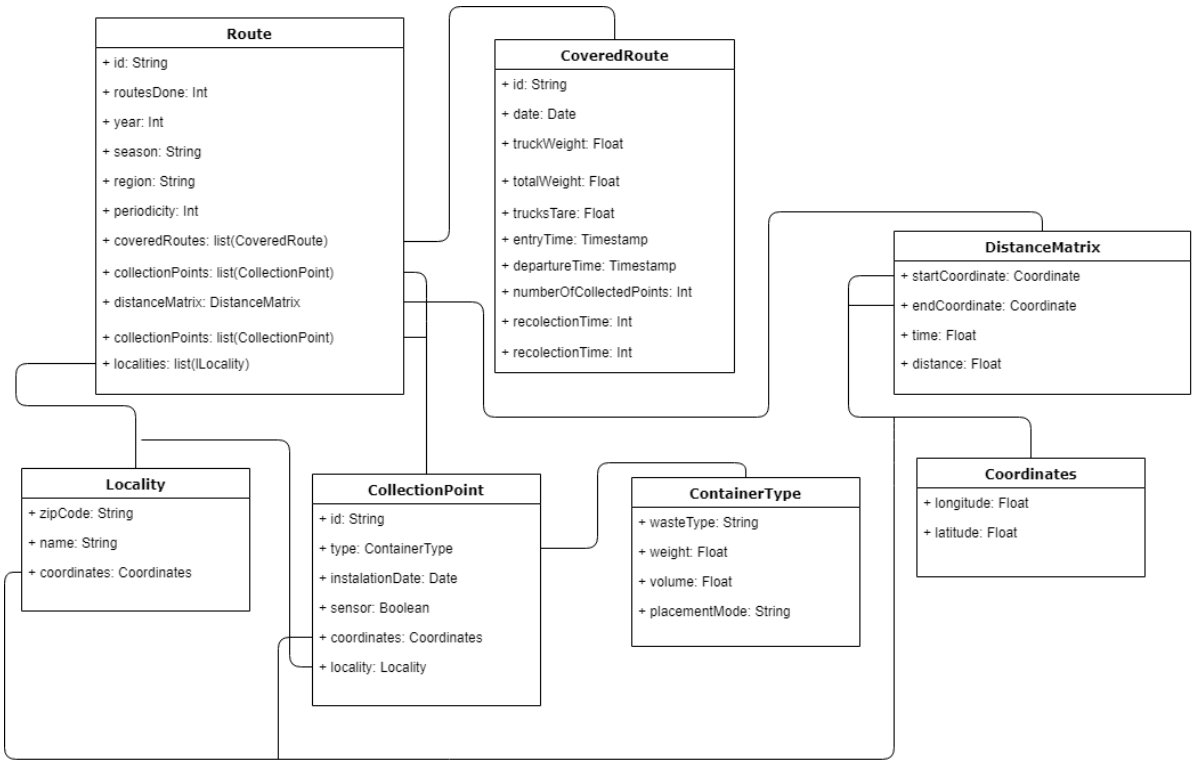

Figure 1. Data model of the system.

In this data model we can see several data collections that provide different information to the framework. Some of these collections are mainly descriptive, like “ContainerType”, “CollectionPoint” or “Locality”, where we save information about the technical features of the container and its installation or about the region we are considering. The rest of the collections are necessary for the computations, like route optimization and the comparison of the results of these optimizations with the original routes, or for the geographical presentation of the information, such as “Route”, “CoveredRoute”, “DistanceMatrix” and “Coordinates”.

The input data is from La Rioja waste collection routes made with trucks, but we could use this model with other regions waste collection routes or even with another type of routes. The data model is general enough to be also useful in different case studies with routes performed with any type of vehicle that has to collect something at certain points.

2.2. Architecture of the system.

We obtained information about the routes, coordinates of each container, days and periodicity when a collection point is visited, and so on, from the company that provides this service in La Rioja. In order to obtain the distance and time matrices for the coordinates, to use them in the optimization of the time or distance of the routes, we needed to know the travelling time and distance between each available coordinate. To get this information we used both the Google Maps and the Open Street Map information services; since there were significant differences between both services we had to make some assumptions (in general, Google Maps seemed to contain more accurate information than Open Street Map, but Open Street Map is less restrictive, without limitations on the amount of API queries per day).

In order to use these services to obtain new data we use the appropriate APIs for each case, OSRM (Open Source Routing Machine) for Open Street Map and Google Maps Directions API for Google Maps (official API). With these services, besides getting travelling time and distance between coordinates, we obtained the driving directions to go from one coordinate to other, useful to later visualize the routes on a map in greater detail, rather than with

straight lines between coordinates.

The technologies that we have used in the different parts of our project (see the image below) are:

- Python, as the general programming language for the framework;
- JavaScript, used for route visualization within a map;
- MongoDB, used to generate the data model;
- Some optimization solvers used with Python like OR-Tools (a Google tool for linear programming) and Gurobi (one of the best-known mathematical optimization solvers).

The mathematical models that we implemented followed the formulations presented in the seminal work by Öncan et al (2009). These formulations were implemented by us in the optimized software OR-Tools and Gurobi. We present some more details in Sect. 3 about these formulations. We present the architecture of our framework in Fig. 2:

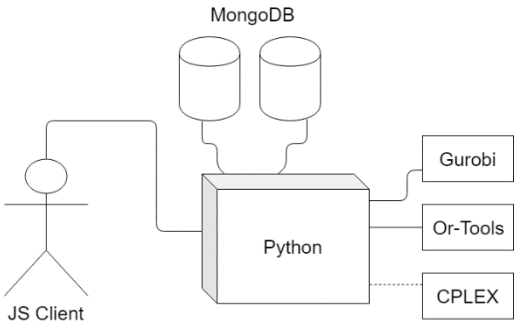

Figure 2. Architecture of the application.

As we can see in the previous figure Python is the application's main language. It is the one in charge of communicating the rest of the modules, the database, the JavaScript client and the optimisation solvers.

In addition to optimize the routes, we can show the original and optimized routes in a map and compare them. For this, we have used the open source JS libraries for geovisualization "Leaflet" and "IDERioja" to display the routes in a web browser.

Apart from that, the development is general enough to support other Linear Programming modules (in addition to OR-Tools and Gurobi solvers) such as the well-known IBM solution CPLEX or also particular solutions. It should be emphasized that when using different solvers for the same optimization problem we do not need to change anything in the data model or for routes visualization, the only thing that changes is the service that optimizes the route.

As we commented in the previous subsection, this architecture has been proven with a specific case, but we can use it with similar transport problems. For example, to optimize or view the route that a motorbike makes to deliver some orders.

3. Modelisation of the problem and computation of the solutions.

This section is divided into two parts. In Subsection 3.1 we describe the different mathematical modelisations that can be used for the problem of route optimisation. In Subsection 3.2 we present the implementation of the modelisations presented in Subsection 3.1 as well as the tools that we have used to solve them.

3.1. Mathematical modelisation of the problem.

The computational part of the project is the most challenging one, partly because of the complexity of the optimisation problem being tackled.

Our approach is as follows. In a first stage, we decided to study individually each one of the routes that we were handed. This amounts to 51 routes with 150 to 300 collection points each, and collection times ranging from 4 to 9 hours. The routes vary in three different ways:

- There are different collection routes for the different geographical subregions of our region;
- There are different collection routes for each day of the week (Monday to Saturday, some with periodicity equal to one week and some with periodicity equal to two weeks);
- There are different collection routes during the summer ("high season") and during the rest of the year.

Therefore, in order to approach the optimisation problem, we first considered each route individually and tried to optimize it by means of an asymmetric Travelling Salesman Problem (ATSP). The asymmetricity is originated by some points whose distances depend on the particular order they are traversed. There are different formulations of the ATSP; we closely followed the formulations surveyed by Öncan et al (2009). We present some details about the implementation in the next subsection. The results obtained by means of this formulation are presented in Sect. 4.

In a second stage, we considered all the collection points that are visited in a stretch of two weeks, their individual periodicity, and all the available vehicles (three in our particular case study) and formulated the problem as a Periodic Vehicle Routing Problem (PVRP). Our modellisation of the problem follows the one by Christofides and Beasley (1984). We present its implementation in the following subsection and the results achieved in Section 4.

The previous formulation proved to be a hard-computational problem, and the results obtained were far from optimal (and also worse than the ones obtained by means of the ATSP). In that situation, we decided to step back and formulate the problem as a Vehicle Routing Problem (VRP), where the traversing of the collection points is to be computed, but the day of the week in which each collection point must be visited is decided beforehand. In fact, this is a well-known strategy -- see, for instance, Carravilla et al. (2011) -- to tackle the PVRP when computational cost is too high.

Our formulation of the Vehicle Routing Problem follows the one presented by Bettinnelli (2010) which is originally the one labelled as "VRP4" (or *three-index vehicle flow formulation*) in the seminal survey by Toth and Vigo (2001), and is presented in the next subsection, and whose results are presented in Section 4.

3.2. Implementation of the modelisations.

As we have stated in Subsection 3.1, we propose three different modelisations of the collection process. The first one, as a Travelling Salesman Problem (TSP); the second one as a Vehicle Routing Problem (VRP) and the third one as a PVRP. In this subsection we present the mathematical formulations that we have used of each one of the problems, and that we later implemented in Gurobi (a specialised software to optimise Linear Programming problems).

The ATSP admits several modelisations that have been surveyed in various works. We closely followed the survey by Öncan et al. (2009) and implemented some of the modelisations presented there. We chose the formulations DFJ (Dantzig et al. (1954)), MTZ (Miller et al. (1960)), SCF (Gavish and Graves (1978)), DL (Desrochers and Laporte (1991)), SD (Sherali and Driscoll (2002)), MCF (Wong (1980)), CLAUS (Claus (1984)) and CLAUS GP (Gouveia and Pires 2001). There are two parameters that have to be taken into account when comparing different formalisations: the deviation between the relaxation bounds and the optimal solution, and the CPU times to obtain the optimisations. The reason to perform the comparison among the modelisations and the computations by ourselves was to find the modelisation that was best suited for our particular solver (Gurobi) and hardware. As a matter of example, CLAUS is known to obtain good deviations from the optimal solution, but its computing time is slower than that of DL or SD.

In our particular setting, and with our example instances (obtained from TSPLIB), DFJ proved to be the best performing (summing up deviations and computing times) modelisation. Its mathematical formulation follows:

Minimize: $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

subject to: $\sum_{j=1}^{n} x_{ij} = 1,\ i = 1 \dots n,$

$\sum_{i=1}^{n} x_{ij} = 1,\ j = 1 \dots n,$

$0 \leq\ x_{ij}\ \leq 1,\ i,j = 1 \dots n,$

$x_{ij} = 0, 1, i,j = 1 \dots n,$

$\sum_{i,j\,\in\,S} x_{ij} \leq\ |S| - 1,\ S\ \subseteq \{2, \dots., n\},\ 2\ \leq |S| \leq n - 1.$

The last constraint is used to avoid subtours in the solution. In practice, we only add them gradually, each time a solution with subtours is reached. Our computational tests in Gurobi with the examples of TSPLIB showed that DFJ performed better than the other formulations, especially when the number of collection points grew (the number of collection points ranging from 33 to 443 in our tests; in practice, DJF even solved to optimality some of the real routes containing more than 200 collection points).

In order to solve the Periodical Vehicle Routing Problem, we implemented the formulation by Christofides and Beasly (1984). As they already announced, and our experiments confirmed, the formulation has a huge number of variables (do note that which truck has to visit each collection point in each day of a 14 days period has to be computed, in contrast to the TSP where only the order in which the collection points are to be visited was computed), and can be solved to optimality only in trivial examples.

Therefore, we followed a two-step methodology. In a first stage, we assign collection points to days in the 14 days period. In a second stage, the Vehicle Routing Problem is computed for each day. This strategy is well-known in the literature (see, for instance, the work by Bianchi-Aguiar et al. (2011)). The modelisation of the VRP that we followed is the three-index model, but using the subtour elimination constraints by Fisher and Jaikurmar (1981):

Minimize: $\sum_{i\in N} \sum_{j\in N} c_{ij} \sum_{k\in K} x_{ijk}$

subject to: $\sum_{k\in K} y_{ik} = 1,\ \forall i \in N_c,$

$\sum_{k\,\in\,K} y_{0k} = K,$

$\sum_{j\in N} x_{ijk} =\ \sum_{j\in N} x_{jik} = y_{ik}\ \ \forall i \in N; k \in K$

$\sum_{i\in N} W_i y_{ik} \leq C, \forall k \in K$

$\sum_{\mathrm{i}\in\,\mathrm{Q}} \sum_{j\in Q} x_{ijk} \leq\ |Q| - 1, \forall Q \subseteq N, |Q| \geq 2, k \in K$

$0 \leq\ x_{ij}\ \leq 1,\ i,j = 1 \dots n,$

$x_{ijk} \in \{0, 1\},\ \forall i,j \in N,\ k \in K$

$y_{ik} \in \{0, 1\}, \forall i \in N,\ k \in K.$

The Boolean variable x_{ijk} is 1 if vehicle k traverses the arc(i,j). The Boolean variable y_{ik} is 1 if vehicle k goes through node i. N_c is the set of collecting points, and N the set of collecting points and the depot. The set of trucks is K, the capacity of the trucks is C and d_i represents the collection at point i.

It is worth noting that this last optimization was performed not over collection points but over municipalities (in order to reduce the number of vertices in the graph) and then, once the optimized traversal among municipalities is known, the best sorting to traverse the collection points of each municipality has to be computed (by means of a small and solvable to optimality instance of the ATSP).

4. Results

In Fig. 3 we present the comparison of the routing times (in minutes) required to complete the routes without optimisation (i.e., the original routes, shown in blue in the image) and after the optimisation process by means of the ATSP formulation (using the DFJ modelisation presented in Sect. 3, shown in orange in the image). The horizontal axis shows the 51 routes that are performed along the year.

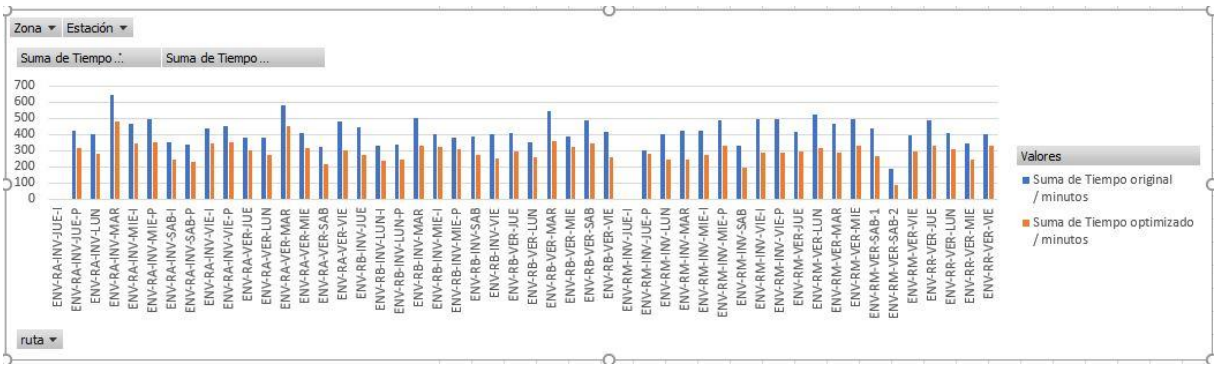

Figure 3. Comparison of the original and optimised routes, using ATSP

All the routes that we were handed admitted optimisation. The average improvement by means of the ATSP is 31%, with some routes admitting optimisation close to 50%. A visual inspection of the original and optimised routes shows that the original routes usually traverse every village and collection point when heading to the most distant point of the depot, and then return directly to the depot (without performing any collection in their way back from the most distant collection point / village). On the other hand, the optimised routes do take advantage of their travel in both directions, when heading to the most distant point of the route, but also in their way to the depot. Some routes even do not collect every container in a village in their way to the most distant point, but divide the collection points into two subsets, one of them being collected when heading to the most distant point (with respect to the depot), and the other one being collected in their way back to the depot. Probably further optimisations are taking place that we have not detected by means of a visual inspection.

We find particularly interesting the fact that 2136 hours of the 6875 hours required yearly can be saved by means of the routes that we have computed. This result shows that human intuition sometimes does not perform very well in optimisation problems (specially when the size of the problems grows). Moreover, this strengthens the idea that the VRP or PVRP probably will produce more interesting results (and more savings) than the ATSP.

We have not presented here the experiments that we have performed with the PVRP by assigning first some dates to each collection point (without changing their periodicity), and then optimising the traversal of the collection points selected for each day. Due to the size of the problem, we had to group the containers in each village or town in “clusters” (the original 2300 collection points were reduced to 130 clusters) and compute the solutions with respect to these 130 clusters. Even so, the results obtained still do not improve on the presented results obtained by means of the ATSP, and thus we decided to not introduce them here.

The previous results obtained with the ATSP show that human intuition sometimes does not always work properly in optimisation problems. Therefore, we think that the PVRP will produce further improvements with respect to the results obtained, once we obtain a sensible simplification of the problem, or enough computational power to reduce the gap among the results obtained with respect to the optimal solution.

Further work

Our first and foremost goal is to obtain advantage of the VRP optimisation with respect to the ATSP optimisation. It seems clear that this will require better computing resources than the ones we are using now. We are considering using a High-Performance Computing (HPC) facility. A sensible strategy to obtain the results has been presented in Sect 3.

For the next framework version, and in order to obtain a secure record of the routes being completed and increase the integrity of the data being collected, so that they cannot be modified or deleted, we also plan to implement a Solidity Smart Contract that stores the information in the Ethereum Blockchain.

We would also like to obtain data about the collection routes from some other regions with a size similar to our region (from Spain or from other countries) from which we can optimise routes being performed and that help to reduce traffic congestion and CO_2 emissions.

Another source of optimisation could be to organise the collection routes considering information about container fill rate, in such a way that containers are collected just in case their fill rate is above 2/3. This would require information (or predictions) about the container fill rate, that nowadays is not available. In any case, the tool presented in this work could be easily applied to the set of containers that satisfies a particular condition (such as their fill rate being above 2/3).

Acknowledgements

This work has been partially supported by the project “IRIS 4.0”, 2017 – I – IDD – 00057 from the “Agency for the economic development” (ADER) of La Rioja, and the FEDER program (EU) for La Rioja, 2014 – 2020 and by the project REGI18-52 of the Universidad de La Rioja.

References

Bettinelli, A., 2010. Mathematical Programming Algorithms for Transportation Problems. Ph. D. Università Degli Studi di Milano. 2010.

Bianchi-Aguiar, T, Carravilla, M. A. and Oliveira J. F., 2011. “Vehicle Routing for Mixed Solid Waste Collection,” VII ALIO-EURO Work. Appl. Comb. Optim., pp. 137–140.

Christofides, N. and Beasley, J. E., 1984. The period routing problem. Networks 14 (2), 237–256.

Claus, A., 1984. A new formulation for the travelling salesman problem. SIAM Journal on Algebraic and Discrete Methods 5, 21 – 5.

Dantzig G. B., Fulkerson D. R. and Johnson S.M., 1954. Solutions of a large-scale traveling-salesman problem. Operations Research, 2, 363 – 410.

Desrochers M. and Laporte G., 1991. Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. Operations Research Letters, 10, 27 – 36.

Fisher M.L. and Jaikumar, R., 1981. A generalized assignment heuristic for the vehicle routing problem. Networks, 11, 109 – 124.

Gavish, B. and Graves, S.C., 1978. The travelling salesman problem and related problems. Working Paper GR-078-78, Operations Research Center, Massachusetts Institute of Technology.

Gouveia, L., Pires, J.M., 2001. The asymmetric travelling salesman problem: on generalizations of disaggregated Miller – Tucker – Zemlin constraints. Discrete Applied Mathematics.

Lenstra, J. K., and Kan, A. H. G. R., 1981. Complexity of vehicle routing and scheduling problems. Networks, 11(2), 221–227.

Miller, C.E., Tucker A.W. and Zemlin, R.A., 1960. Integer programming formulations and travelling salesman problems. Journal of the Association for Computing Machinery 7, 326 – 9.

Öncan, T., Altinel K. and Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations, Computers & Operations Research, 36 (3): 637 – 654.

Sherali H. D. and Driscoll P.J., 2002. On tightening the relaxations of Miller–Tucker–Zemlin formulations for asymmetric traveling salesman problems. Operations Research, 50, 656 – 69.

Toth, P. and Vigo, D., 2001. The Vehicle Routing Problem. SIAM.

Wong R.T., 1980. Integer programming formulations of the traveling salesman problem. Proceedings of the IEEE international conference of circuits and computers, 149 – 52.

Gurobi documentation: https://www-proxy.gurobi.com/documentation

OR-Tools documentation: https://developers.google.com/optimization/

OSRM documentation: http://project-osrm.org/

Google Maps API Directions documentation: https://developers.google.com/maps/documentation/directions/start