Genetic Algorithm for Waste Collection in Smart Cities

Case of Campolide

Evandro da Silva Mendonça

Project Work presented as requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

2018

Title: Genetic Algorithm for Waste Management in Smart Cities

Subtitle: Case of Campolide

Evandro da Silva Mendonça

MGI

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

GENETIC ALGORITHM FOR WASTE COLLECTION IN SMART CITIES

by

Evandro da Silva Mendonça

Project Work presented as requirement for obtaining the Master's degree in Information Management, with a specialization in Knowledge Management and Business Intelligence

Advisor: Professor Miguel de Castro Neto, PhD

October 2018

ABSTRACT

Smarts cities are becoming an important concept in the cities, it tries to discover methods to interact with the environment in sustainable ways inside urban areas. This concept emerged to deal with the growing urbanization faced by the cities around the globe. Within problems brought by the urbanization, waste management is one of the hardest and most impactful. The collection stage of the waste management is the costliest and the route planning for the garbage trucks is a well-known hard problem. In this project, a genetic algorithm is proposed to deal with the waste collection routing problem using a heterogeneous fleet of trucks. As the population in the city is expected to grow over the years, the project adapts to the current state of the city, because it uses the concept of open data from the municipality to feed itself with the garbage collection information and generate its results. Multiple runs were performed to define its parameters. The algorithm was tested in the simplified real case of Campolide, in the municipality of Lisbon, and proved to be feasible for application on real-world scenarios relying only on actual data of the cities' waste collection.

KEYWORDS

Arc Routing Problem; Genetic Algorithm; Smart Cities; Waste Management

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ARP	Arc routing problem
CARP	Capacitated arc routing problem
GA	Genetic Algorithm
GVR	Genetic Vehicle Representation
ICT	Information and communication technology
OSM	OpenStreetMap
TSP	Traveling Salesman Problem
VRP	Vehicle routing problem

1. INTRODUCTION

1.1. Cities urbanization and waste management problem

Human activity has been pushing environmental changes. Global warming, air pollution, and biodiversity decrease are some of the examples of these changes that can be observed (Bătăgan, 2011). Urban areas are the principal responsible that drive these changes at multiple scales. Being centers of production, consumption and waste disposal, the impacts on the environment can be repeatedly observed among the cities, especially those located in the developed world (Grimm et al., 2008).

The issues generated by the urbanization are even more worrying given that from the 1950s to 2014 the urban population went from 30 percent to more than half of the world's population with 54 percent. Furthermore, in the coming decades, the change on the size and distribution of the urban area will be more expressive, projected to have 66 percent of the entire world's population living in the cities by 2050 (United Nations, 2014). Megacities, the ones that by convention have more than 10 million inhabitants are emerging mostly in the developing world, and economic growth will follow the urban growth, demanding more services and resources (Grimm et al., 2008). Although the urbanization process brings opportunities for development, at the same time challenges arise, namely on social equity, environmental sustainability and government (United Nations, 2014).

One of the major environmental and socio-economic challenges that come with urbanization is waste management (Fujdiak, Masek, Mlynek, Misurec, & Olshannikova, 2016; Karadimas, Papatzelou, & Loumos, 2007). The amount of waste is increasing over time in the urban society. Data from the 2012 World Bank's report shows that the cities were generating about 1.3 billion tons of solid waste per year, costing $205.4 billion. By 2025 it is expected to increase this generation by 2.2 billion tons with the management cost of $375.5 billion, mainly in lower-income countries (Hoornweg & Bhada-Tata, 2012). The projections of urban growth and waste generation for 2025 by region can be seen in table 1.1, the table shows that the regions that currently generates more waste are the one where most of the developed countries are. However, a shifting trend can be observed where the regions with developing countries will take the lead in the waste generation. As an example, the OECD (Organisation for Economic Co-operation and Development), currently the bigger generator, will increase their waste generation by 11%, and EAP (East Asia and pacific) will more than double their waste production.

Waste management is particularly impactful in the short-term to the citizens and the environment, while compared with other problems that massive urbanization may cause (Hoornweg & Bhada-Tata, 2012). The idea of waste management involves many cycles, these can be listed as the collection, transport, processing, recycling, and monitoring. More steps can be presented depending on the cities' waste management scenario, although the waste management aim a common goal in every place it is applied, different cities have their own particularities and need to be addressed in own specific ways. The most important of these cycles naturally is the collection as it directly impacts people living in those urban areas. The collection is also the step that has more costs involved in referring economic terms because it requires intensive labor work and massive use of trucks to be able to deliver the service to the entire city (Beliën, De Boeck, & Van Ackere, 2011). From the total amount of money spent on waste management, 60 to 80 percent is distributed over the collection, transportation, and disposal of solid waste (Karadimas et al., 2007).

Region	Current Available Data			Projections for 2025			
	Total Urban Population (millions)	Urban Waste Generation		Projected Population		Projected Urban Waste	
		Per Capita (kg/capita/day)	Total (tons/day)	Total Population (millions)	Urban Population (millions)	Per Capita (kg/capita/day)	Total (tons/day)
Africa	260	0.65	169,119	1,152	518	0.85	441,840
East Asia and pacific	777	0.95	738,958	2,124	1,229	1.5	1,865,379
Europe and Central Asia	227	1.1	254,389	339	239	1.5	354,810
Latin America and Caribbean	399	1.1	437,545	681	466	1.6	728,392
Middle East and North Africa	162	1.1	173,545	379	257	1.43	369,320
Organisation for Economic Co-operation and Development	729	2.2	1,566,286	1,031	842	2.1	1,742,417
South Asia	426	0.45	192,410	1,938	734	0.77	567,545
Total	**2,980**	**1.2**	**3,532,252**	**7,644**	**4,285**	**1.4**	**6,069,703**

Table 1.1 – Waste generation projection for 2025 by region. Adapted from Hoornweg, D., & Bhada-Tata, P. (2012)

Uncollected waste can be harmful to the environment and consequently bring a variety of health issues to the population. Also, poorly waste management have economic impact on the city because the costs can be higher than it would be to properly address the problem. Manage the waste collection of the households is a hard problem that is faced by cities' government across the globe (Hoornweg & Bhada-Tata, 2012).

Ongoing urbanization stresses the importance of efficiency waste collection, cities must find ways to maximize the acceptance of collection solution (Beliën et al., 2011). Waste collection is about the collection, transportation, and disposal of solid waste from residences, commerce, industry and any other agent that produces solid waste. The collection can be done house-to-house (or door-to-door), via community bins, self-delivered, among others (Hoornweg & Bhada-Tata, 2012). Waste collection is a hard problem that must be aware of many factors that influence the collection, making this step efficient is difficult since this kind of problems does not have an exact solution in a feasible time.

1.2. SMART CITIES ROLE IN WASTE MANAGEMENT

Cities' main challenge have become be able to manage the ecosystem services dependence, which exhausts the biodiversity and natural resources although prioritizing public health and quality of life (Science for Environment Policy, 2015). With such changes and challenges arising, keeping livable conditions within this context demands a deeper understanding of a smart city, and how it can support cities among the world to deal with these emerging problems (Chourabi et al., 2012).

The smart city concept heavily bases itself on the environmental aspect of the cities and the engagement of people and government in environmental activities (Giffinger, 2007). There is a special motivation on the preservation of natural resources and related infrastructure (Chourabi et al., 2012), and as discussed before, waste management is one of the most important problems with socio-economic impact in the city. Indeed, smart cities itself came to face the challenges that urban areas are facing today and probably the ones that they will face in a near future (Nam & Pardo, 2011).

As others fresh and controversial concepts, the smart city one is not different in the fact that there is no standard definition or template of framing (Nam & Pardo, 2011). In the policy arena in the past years, this concept has been greatly quoted. It seems that the focus approached on this area is about the role of the ICT, an acronym for Information and Communication Technology. The ICT-driven development is believed to be the path to follow for many countries in the EU for example (Caragliu, del Bo, & Nijkamp, 2011).

Obviously, ICT has transformed for better many urban areas economics, social and environment. But laying only in the technology and communication would not benefit the whole city, in some cases, these smart cities need to deal with a problem brought by this form of approaching the concept, like social polarization that create bigger social divisions over the population. The educated and technology included society, mostly middle class, that are attracted by this kind of policy can produce highly gentrified neighborhoods while excluding traditional and poorer residents of the city (Hollands, 2008).

Smarter solutions are arising to deal with the problem of waste management within the smart cities, either using new techniques, data, ICT components, or even a combination of those concepts (Fujdiak et al., 2016). Multiple solutions proposed make heavy use of ICT, as sensors in recycling bins, and brings huge benefits to the waste management saving up time and money spent in the waste collection step (Catania & Ventura, 2014; Fujdiak et al., 2016). Mendes, A. (2017) describe an example of this technology applied in the municipality of Cascais in Portugal. Underground containers are equipped with sensors that indicate data of their level of load in real time. This approach reduced the number of trips that the trucks were used to make to collect the garbage, in consequence, economic saving where attained in the process, the carbon emissions and kilometers traveled also decreased (Mendes, 2017).

The usage of ICT in waste management works well when the garbage is disposed of in fixed bins along the streets, where the truck can make the collect at any time. The door-to-door waste collection has more issues on adopting ICT to improve the collection phrase, an example of a difficulty would be a building with some apartments, in most cases, the residents share the same bins that are collected by a truck in pre-specified days. However, door-to-door type of collection can be addressed in a different way by the smart cities, where the focus is not about using ICT with sensors and technological chips, but using available open data with information of the cities' waste collection generated by the collections routes in the past, and trying to optimize the routes using new techniques to reduce the economic and environmental impact caused by the collection step of the waste management.

1.3. GARBAGE TRUCK ROUTES PLANNING

Intelligent collection management is vital to ensure cost reduction, improve coverage and efficiency of the waste collection process (Buenrostro-Delgado, Ortega-Rodriguez, Clemitshaw, González-Razo, & Hernández-Paniagua, 2015). While focusing on ICT components, one can use many techniques to ensure that the collection is done efficiently, but as stated before, apply this approach when dealing with households' waste collection can be harder than having chips placed on static bins over the streets. Therefore, the collection of waste at each door using ICT is not being considered. Instead, this project’s aim is about optimizing the routing of the collection trucks in order to provide a better route for each truck available, while respecting the known limitations.

The collection process is negatively affected by a variety of factors, including poor route planning and the number of vehicles available (Guerrero, Maas, & Hogland, 2013). Optimize the routes that trucks perform in order to serve the households in a city would be beneficial for both the people and the government. Since planning better routes is a well-known hard problem, some algorithms exist in order to give a good enough solution for this issue.

The most common algorithms used in order to deal with the routing problem are the Vehicle Routing Problem (VRP) (Mohammed et al., 2017) and the Capacitated Arc Routing Problem (CARP) (Arakaki & Usberti, 2018), with their variations. The CARP is the key problem inside the Arc Routing Problem (ARP) and is the counterpart to the VRP (Wøhlk, 2008). Both problems are considered hard combinatorial optimization problems (Arakaki & Usberti, 2018; Fadzli, Najwa, & Luis, 2015; HAN & Cueto, 2015; Wøhlk, 2008). These algorithms run upon graphs, that in the waste collection case represents the streets and collection spots. The difference between these algorithms is that the VRP, the most studied routing problem between both (Fadzli et al., 2015; Ramdane-Cherif, 2006), consist in to process the demands of the nodes in a graph, while the CARP focus on serving the edges instead of the nodes (Ramdane-Cherif, 2006). Relating the two algorithms with the waste management problem can be stated that the VRP can be applied to deal with community bins, where each bin is a node in the graph and the edges represent the streets between them. In the CARP, the edges still are the streets and the nodes are intersections between the streets. The CARP approach is more suitable for door-to-door collections since the garbage truck must collect the garbage from a street instead of a specific bin (Fadzli et al., 2015). Indeed it is one of the applications that this algorithm tries to solve (Willemse & Joubert, 2016) because serving the edges instead of the nodes fits better in this problem.

Methods that deal with this kind of collection problem, like vehicle allocation and route designation, can be applied using traditional mathematical methods like linear methods, but these can rapidly suppress the computational resources even with medium sized instances (de Oliveira Simonetto & Borenstein, 2007). This set of problems are known as NP-hard (non-deterministic polynomial-time) problems and until nowadays is only viable to use exact methods for very small instances because of their complexity (Lacomme, Prins, & Ramdane-Cherif, 2001; Pereira, Tavares, Machado, & Costa, 2002). Therefore, heuristics and metaheuristics are used to approximate solutions. Although these approaches don’t guarantee optimal solutions to the problems, they generally provide good solutions that can be used in real life applications (HAN & Cueto, 2015).

Metaheuristics has been the path followed by many researchers in the past years, and its result appears to be more promising than heuristics in many cases (HAN & Cueto, 2015). Genetic algorithms have been broadly studied over the literature and seem to allow a more thorough search over the solution space. This approach can lose track of good solutions when jumps to spaces far from optimal solutions, but it also allows moving away from local and not global optimum solutions with some techniques including recombination and random mutations. Some authors are applying genetic algorithms to solve the VRP and ARP with success.

1.4. PROJECT GOALS

With the rise of smarts cities, every aspect of its operation is been rethought. Waste management is one of these aspects that has a huge impact on the city's environment and citizens life. In order to reduce the economic and environmental impact of waste in the cities, better ways to deal with these problems must be found. In fact, many researchers are dealing with these issues, in the academic and corporate world. Waste management by itself is not a sole activity, as stated before, it includes some activities that must be accomplished. Among these activities, this research aims to optimize the routes and usage of trucks in the household waste collection.

The routing problem is a known hard problem and it is contained in the NP-hard problem set. This brings the issue of not been able, until this date, to have an exact algorithm to deal with this problem in a feasible time. Leading the researchers to develop heuristics and metaheuristics to deal with the problem is a smart way.

The project aims to use GA to optimize the door-to-door collection routes and truck usage with multiple garbage trucks of different capacities, taking into consideration past data from the municipality management to estimate the amount of garbage that each street produces. As stated before, the door-to-door collection problem can be seen as an ARP problem, most specifically the CARP. This algorithm concern is about serving the edges, differently from the usual node serving system. Street information is needed in order to calculate some measures like path length, street direction, and its connections.

To this project, data from the waste collection of several past months is needed. Governments are the largest creator and holder of data within the city (Janssen, Charalabidis, & Zuiderwijk, 2012), and most of the time, they have this data through cities' managers or companies that are responsible for the waste collection. In the case of this project, Câmara Municipal de Lisboa shared the data of Campolide's garbage collection, completely without personal data and making sure not to compromise any personal privacy.

OpenStreetMap (OSM) will be used to access the streets information. Each piece of route that the garbage truck can travel need to be mapped and stored in a data structure. A usage of this information is to match the data from past collection with each piece of route to estimate the amount of waste generally produced by that place. Also, it allows generating a graph with the connections to measure the distance between two edges.

Having past collection data, and a data structure representing the streets of Campolide will be possible to use a genetic algorithm to calculate multiple routes that the trucks can travel. Choosing the right operators to be applied in the algorithm is a tough step of the genetic algorithm development. Pereira, Tavares, Machado & Costa (2002) created a genetic representation for the VRP algorithm, this representation will be used in this project's genetic operations. With the offspring generation and rules on when to stop the algorithm, the best result of the last offspring will be considered the best solution.

2. LITERATURE REVIEW

2.1. OPEN DATA

The concept of open data is important in the scope of this project as it makes use of government data and collaborative data to reach its objectives. It is important to address this subject while it gives greats benefits to the city as a whole. This section has the objective of introducing the concept of open data and enumerate its possible benefits and barriers. Collaborative projects will be addressed in order to explain how and why this can bring great benefits to the knowledge sharing and social improvement.

Open data is a relatively new but promising topic, with the concept that collected data held by organizations, should be available in electronic form granting its access for the population (Gurstein, 2011). Governments from all over the world are committed to implementing open data strategies with the promise of more openly available data, more transparency, efficiency (Molloy, 2011) and population participation in the cities' management (Huijboom & Broek, 2011).

In the era of Smart Cities, sharing data helps the generation of knowledge over the city by individuals, academy or even the industry, leading to stimulate innovation and promote economic growth. Besides the stated, there are some motivational drivers that lead the government to share their data, like increase democratic control and political participation, foster service and innovation and strength law enforcement (Huijboom & Broek, 2011). However, there is no method on how to predict the return on investment on sharing data, because open data by itself has no value and only aggregate value when used for some purpose (Janssen et al., 2012). Added with some barriers this helps to slow down its adoption, even with the known incentives and benefits, these barriers must be overcome in order to spread its adoption.

The barriers found are commonly interrelated (Janssen et al., 2012). Task complexity, information quality and participation, privacy legislation and closed government culture (Huijboom & Broek, 2011; Janssen et al., 2012), the last two been mentioned by most policymakers and experts (Huijboom & Broek, 2011). Huijboom & Broek (2011) also realized that whereas the motivational drivers stand most outside the government, the barriers lie principally within government organizations. And individual government agencies are often reluctant to implement the measures to provide open data, even if the strategies were already defined by government higher hierarchy.

To successfully implement open data strategies, require more than just providing the information. Janssen et al. (2012) defined some requirements that must be presented in the open data strategy. First, the data's quality must be improved with better measurement and storage. A swift of the organization culture upon this concept needs to take place. And instruments and tools to access the data must be provided, as internet portals for example, where the population can easily access relevant information (Janssen et al., 2012).

Besides having data only held by organizations, some projects called collaborative projects also gather data from the population, individual person with local knowledge, and share it with others, including companies and government. The greater example of a collaborative project is Wikipedia, but the collaborative project that is used by this research is OpenStreetMap (OSM), at OSM, volunteer mappers edit the map, that is then freely available for use ("History of OpenStreetMap," 2018). This kind of project ensure the share of data overall levels and, as open data initiatives from government, motivates innovation, because more people will have access to data once in few hands.

2.2. CAPACITATED ARC ROUTING PROBLEM

The routing problem addressed by this project matches the issues that the CARP algorithm proposes to deal with. The CARP is about serving a set of streets with a fleet with limited capacity starting and ending at a deposit, this problem has been proved to be an NP-hard problem (Wøhlk, 2008). This algorithm can and actually integrate many more limitations, like the vehicle size for a street, the total amount of distance allowed on a trip, the time windows. For each one of these nuances, the researchers develop variations of the CARP with different limitation.

Formally, Lacomme et al. (2001) define the Capacitated Arc Routing Problem as: An undirected network $G = (V, E)$ where V is a set of n nodes and E a set of m edges. A fleet with k identical vehicles of capacity Q based at the depot node s. A subset $R \subset E$ of edges that must be served by exactly one vehicle. All edges can be traversed any number of times. Each edge $(i, j) \in E$ has a traversal cost $c_{ij} \geq 0$ and a demand $e_{ij} \geq 0$. The CARP consists in determining a set of vehicles routes with minimal total cost, such as each trip start and ends at the depot s, each required edge $(i, j) \in R$ is serviced by one single trip, and the total served edges by any vehicle does not exceed the capacity Q.

This undirected version can be used on roads that can be traveled by a vehicle in any direction, these cases occur mostly in low-density areas with very low-traffic (Lacomme et al., 2001). In bigger cities, as the case of this project that deals with Lisbon city, the directed version must be applied. In a directed version of the CARP, each edge represents a street or one side of a street, in the case of it allow traveling using both directions, with mandatory direction for servicing it (Lacomme et al., 2001).

The CARP problem can be represented in a variety of ways. There are CARP with intermediate facilities (CARPIF) where the graph has recharging nodes, CARP with vehicle-site dependency (CARPVSD) in which only a certain class of vehicles are allowed to transverse some roads because of some limitation, there are also a few studies where external factors are considered (Fadzli et al., 2015). The problem of waste collection in Lisbon using the CARP on mixed graphs called MCARP using a heuristic method was also studied (Mourão, Nunes, & Prins, 2009).

According to Wøhlk (2008), besides heuristics, researchers are also applying metaheuristics to deal with it the CARP problem. Simulated annealing, Tabu Search, and Genetic Algorithm are some of the metaheuristic's algorithms used to find a solution. These methods have been producing good results, being on the most performance algorithms for the CARP (Wøhlk, 2008).

2.3. OVERVIEW OVER GENETIC ALGORITHMS

Genetic algorithm is a metaheuristic that imitates the biological process of reproduction and natural selection (Carr, 2014). It was described by John Holland in the early 1960s (Mitchell, 1995) and belongs to the class of evolutionary algorithms (Carr, 2014; Whitley, 1994). These algorithms are commonly used as functions optimizer, and it has been applied in a broad range of known problems (Whitley, 1994). One of the greatest barriers of software design, that is to fully understand the structure of complex problems can be solved mimicking natural selection, the specification of every feature of the problems and how to deal with them are not an impediment to search for a solution using this approach (Holland, 1992).

Given its nature, genetic algorithms have been used to find solution for hard problems like the Travelling Salesman Problem (TSP), VRP (Pereira et al., 2002), CARP (Deng et al., 2007) and many other problems that due to their complexity don't have an algorithm that produces exact solutions. This is possible because these algorithms tend to explore a far greater range of potential solutions in the search space (Holland, 1992).

Genetic algorithms for the CARP was created, tested, and compared with real case scenarios over the world. To the problem of sprinkler cars routing in Chongqing City in China, taking 37 vertexes and 1 deposit into consideration, the GA reduced the travel distance by 33%, also giving a better result than other algorithms (Deng et al., 2007). CARP with time windows GA where developed and compared with heuristics (Ramdane-Cherif, 2006). This research also concluded a superior performance of the evolutionary algorithm when compared with the heuristics.

Because genetic algorithms are based on biological evolution, the terminology used is the same as the one used in biology, although representing fairly simpler concepts than their biological counterpart. Most GA share commons elements, like populations of solutions, selection, crossover, and mutation (Mitchell, 1995). To move forward on understanding genetic algorithms, the concepts attached with their nomenclature must be defined, these common elements are described in table 2.1.

Concept	**Definition**
Gene	A variable (parameter) of the chromosome
Chromosome	Set of genes, is a candidate solution for the problem, the representation of the phenotype on a data structure that can be understood by the algorithm
Fitness function	A function to measure the fitness of a solution compared with others, this is the function that must be maximized or minimized depending on the algorithm objective
Population	Set of chromosomes with possibility to be selected to breed the next generation
Crossover	Combination of chromosomes to generate the offspring for the next generation
Mutation	Random changes of genes in the chromosome

Table 2.1 – Genetic Algorithm concepts

Summarizing the steps followed by a common genetic algorithm as stated by Jebari & Madiafi (2013), at the principle, a population of chromosomes are generated at random or by some criteria, this stage is called initialization. Then each individual in the population is evaluated by the fitness function and receive a fitness value. A selection mechanism takes place and select, using specifics techniques, parents that will be crossed to generate an offspring for the next generation of chromosomes that can also be mutated. These steps, excluding the initialization, are repeated until some condition is reached (Jebari & Madiafi, 2013). Must be noticed that some parameters are required to be defined prior the running of the algorithm, they are the population size, mutation, and crossover rates, the GA makes use of these variables in its operations.

This whole process will be discussed in deeper details in the next sections. Each step of a common GA discussed in this summary will be addressed in more details. Different techniques will also be addressed to give an overview of the variety of approach each operation in a GA can have.

2.3.1. Fitness function

The fitness function is one of the most important parts of the genetic algorithm approach as it is the only one method of evaluating the quality of the solution and measure the improvement through the generations. Because of its importance, the fitness function must be addressed before going through every step that composes the GA.

A number called fitness is assigned to each chromosome in the population using the fitness function. The fitness of the chromosome depends on how well it solves the problem that the GA is supposed to solve (Mitchell, 1995). The fitness function must be more sensitive than just measuring good or bad results, it needs to be able to define where the chromosomes stand in the fitness range and compare it with other solutions presented in the population (Carr, 2014).

Fitness function can be the most limiting factor to a genetic algorithm. As addressed above, the fitness function must translate how the solution performs, and this, in most cases, is not straightforward. Generating complex and expansive fitness functions that are not computational efficient and require hours to complete, as cases of real-world simulations, can be prohibitive in the development of a genetic algorithm. Need to be considered that the fitness function will be evaluated for each chromosome of the population for every new generation produced.

2.3.2. Initialization

The GA is based on evolving every population until some condition is reached. The initial population must be generated to let the evolution iteration begins. There are some different ways of generating the first population, it is normally generated randomly (Whitley, 1994), but some knowledge can be applied to the generation.

The size of the population depends on a previously defined parameter. This size is preserved through the entire life of the algorithm. The initialization can be done totally random or applying some previous knowledge of the problem, in this case, some chromosomes can be included with known genes that makes sense to the problem (M. Kumar, Husian, Upreti, & Gupta, 2010), this can lead the algorithm to converge faster to areas where optimal solutions are more likely to be found. From this early step, the evolutionary process begins.

2.3.3. Selection

A subset of the population is then selected and used to breed a new generation, that said, this step is critical since it needs to select good individuals trying to keep the diversity of the selected chromosomes. The subset size is also a parameter that must be set into the algorithm.

The selection step can take place using a variety of techniques. Some methods focus on the fitness of the individual, where chromosomes with the best fitness are the one to be selected. Other methods are based on randomness selection or combination of these techniques. No method is guaranteeing to be the best one, and the choice must be problem specific.

There are many selection methods, the most used are the roulette wheel and tournament selection (Saini, 2017). But other methods like Stochastic Universal Sampling, Rank Selection and Random Selection can be found in the literature.

The tournament selection and roulette wheel will be addressed in this research. Both methods provide good and diverse parents in most cases because they give the possibility of poorer fit chromosomes to be chosen and still rely on the fitness value to decide on which individual to choose in their deterministic steps.

In the tournament selection, the selection process that is used in this project, K different individuals are randomly selected from the population as shown in figure 2.1. Within this set, the chromosome with the best fitness is then selected to reproduce. This process is done once more to select the next parent.

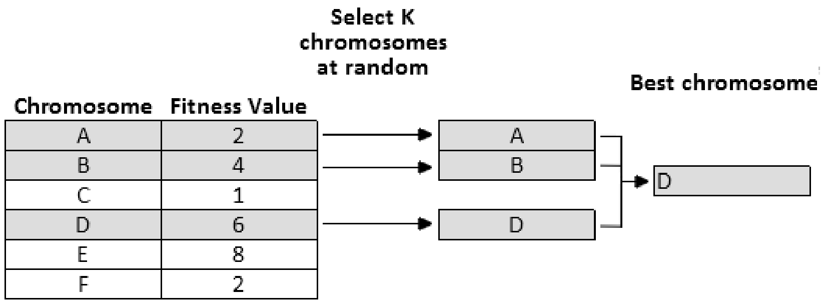


Figure 2.1 – Tournament selection

The roulette wheel selection gives each chromosome in the population a probability of being selected. This probability is proportional to its fitness. To select one individual, a random number is generated, simulating a roulette, and the generated number will define which chromosome will be chosen to produce the offspring. Again, this process is repeated to select the next parent.

2.3.4. Reproduction

Once the parents are selected, the reproduction step takes place. The parents are combined using a crossover to generate the offspring. Then, the generated chromosomes can have its genes randomly mutated by the mutation process at a certain rate, this helps the algorithm to run away from local optimum and have a broader exploratory range in the search space. These steps and their probability rates are also problem specifics, giving that each problem will use the crossover and mutation methods that make sense.

2.3.4.1. Crossover

Crossover is a vital process in the generation of new chromosomes. It exchanges genetic material (genes) from two or more chromosomes hoping that can generate individuals with better fitness in the next population (V. Kumar & Panneerselvam, 2017). Usually, the crossover is applied with a high probability, it means that in most cases the genetic material of the parents will be recombined to generate the children, less likely, they will just be copied to the next generation as they are. Using a crossover rate of 100% means that every chromosome in the offspring was generated using at least the crossover.

As the selection phrase, there are multiple methods to apply crossover, some of the most known and generic are one-point crossover and two-point crossover, among others. This project uses a different type of crossover that does not share the behavior of these generic methods and will be further explained in chapter 4. Because of that, only these two crossover techniques will be explained. These crossover methods will be addressed here to give an overview of how this process takes place in the majority of the cases and illustrate the crossover operation.

The simplest crossover operator is the one-point crossover. In this type of crossover according to Kumar & Panneerselvam (2017), a random point is selected within the limits of the parent, this point is called the cut point. Every point possible to be selected have an equal chance of being selected. To illustrate, in figure 2.2, two parents represented by an array of characters with a total size of 6 elements each, the cut point would be any number between 0 and 4, and in this case, the point 3 was selected. The cut point splits the parents into two halves each. The first part is every array element which its index in the array is less or equal the cut point, the second part obeys the opposite rule where the elements with an index greater than the point are considered. To generate the children, the algorithm copies the first part of the parent one and insert in the offspring, then get the second part of the other parent and insert in the offspring. Changing the order of the parents and doing the same operation generates the second child (V. Kumar & Panneerselvam, 2017).

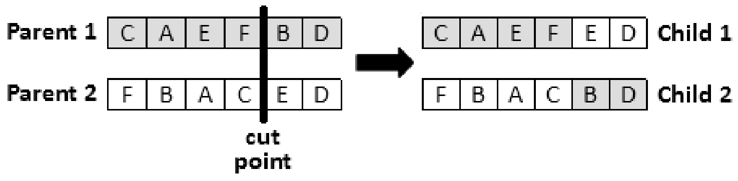


Figure 2.2 – One-point crossover

There are cases where elements within a chromosome cannot be repeated, as routing problems where a city must be visited only once, and all cities must be visited. In these cases, while applying the one-point crossover, instead of just blindly copy the half of the second parents, the algorithm must copy the genes one by one in order, avoiding the elements that are already in the child until it is fulfilled.

The two-point crossover is a generalization of one-point crossover. The difference between them is that the two-point crossover chooses two cut point instead of just one, this will split the parents into 3 parts. As a reference, multi-point crossover also exists, everything depends on the number of cut points selected.

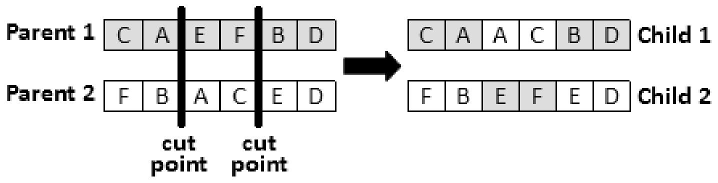


Figure 2.3 – Two-point crossover

Two-point crossover mixes the parts of each parent in the child. Each part is alternately selected and inserted into the child following a specific order. Using an example illustrated in figure 2.3 that is similar to the one previously saw in the one-point, the first part of the parent 1 goes first in the child, then the second part of the parent 2 is inserted, finally, the last part of the parent 1 is inserted. Repeating this operation interchanging the order of the parents generate the second offspring.

2.3.4.2. Mutation

Mutations are small random changes in the genetic material of a chromosome. Mutations itself is not supposed to carry the solution to a better fitness in purpose, but they provide an insurance policy against the development of uniform populations that are less likely to improve themselves in the next iteration (Holland, 1992). Typically, the mutation rate is applied with a low probability of 1% or less in many cases (Whitley, 1994), because with very right probability, the algorithm could be reduced to a random search over the space.

Common mutation methods are bit flip, swap, inversion, among many others that can be found in the literature. In the example below in figure 2.4, the swap mutation is shown. In this mutation technique, two genes of the chromosome are randomly selected and swapped between them, these cases are useful when repeating a gene is not allowed, as the case of the TSP. In the bit flip mutation, for each gene in the solution, there is a chance of change the data of the gene to other random data with the possibility to be inserted in the chromosome.

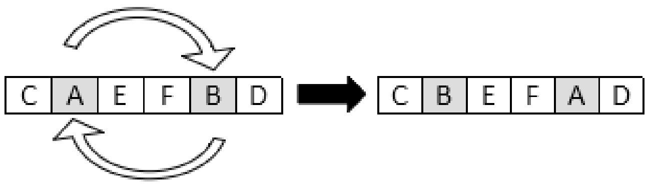


Figure 2.4 – Swap mutation

2.3.5. Termination

Genetic algorithms create population after population iterating until some condition, or conditions, have been reached. Conditions must be pre-defined by the developer of the algorithm, usually, they rely on time, the number of iterations, minimum criteria found. Kumar et al. (2010) described some techniques used as stop conditions listed below.

- Found a solution that satisfies a minimum criterion;
- Number of iterations reached;
- Computational time reached (budget);
- The algorithm has reached the highest fitness solution and no longer is producing a better solution for a number of iterations;
- Manual inspection;
- Combination of the previous methods or any other method created.

Once the termination condition is reached, the chromosome with the best fitness of the last population, or the chromosome with the best fitness found overall iteration is selected as the best solution found by the GA.

3. METHODOLOGY

This project will follow the design science research methodology to accomplish its final goal. This research methodology requires that the motivation, problem, and objectives of the project must be clearly defined. Then with those steps accomplished, the development of the project is described based on the theory previously analyzed. With the completion of the project test cases take place to measure the performance and feasibility or the solution reached. In the case of this project, the test case was created using data from the Campolide parish's waste collection, in the municipality of Lisbon, Portugal. The next paragraphs describe the steps of the methodology applied in more details, relating where each piece of the process can be found in this written research.

Following this methodology, the first chapter presents the motivation and the problem of the study, specifically the subchapters 1.1 and 1.2. These subsections give a broad contextualization in the inherent nature of the population growth problem in the urban areas. Relating it to the sustainability concern in these cities and the waste management problem. Then it shows how the emerging concept of smarts cities came to deal with the overpopulation and overgeneration of waste, focusing on the use of ICT in approaches to deal with the waste collection, and describing the problems and limitation of basing every solution upon technological equipment.

Still, in the subsection 1.3, the problem that this project aims to solve is presented, on how to optimize waste collection routes on the door-to-door collection without relying purely on ICT. This subsection explains the importance and challenges of the garbage truck routing problem giving an overview of the currently available solutions found in the literature that approach this problem, that is defined as a CARP. In the latest paragraphs, meta-heuristics are introduced as a possible way to deal with these kinds of problem, mostly the genetic algorithm, also describing the benefits of the application of these methods.

On the subchapter 1.4, the objectives of this project are defined. On this section, a wider vision of the project's aim and each step that will lead to the goal is defined, trying to follow a train of thought on how each step connects to each other to accomplish the final goal of having a genetic algorithm to deal with the routing of garbage trucks in a city, using the parish of Campolide at the Lisbon municipality as a model to validate the solution.

The problem definition and motivation, besides having some theory to be based, was explained in a broader aspect because was not the main proposal of this project. In chapter 2, a literature review of the theory used to accomplish the research project was analyzed in a deeper way, presenting each topic that is important to the development of this project with more details. First, on section 2.1, the concept of open data was addressed. Since this project relies on data from governments departments, this concept was analyzed, and a state of the art was built on how the world deals with open data. This is important to make viable that the approach applied in this research can be replicated in other areas. In the subchapter 2.2, a literature review was made on the CARP algorithms and the current solutions available. The definition of the CARP is formally presented, as a variety of works about different algorithms to solve the CARP and its extensions. Then, in the subchapter 2.3, the genetic algorithm is described, with explanations of its core concepts. The subchapter 2.3 has subsections that explain in detail each step that most genetic algorithms have in common, like how to generate the offspring using crossover and mutation operations.

Following the chosen research methodology, chapter 4 describes in detail every aspect of the development of the project. The subchapters 4.1 and 4.2 define the tools utilized to accomplish the final goal, and how these tools were applied in the context of the project. Then, in the subchapter 4.3 the genetic algorithm built to deal with the CARP problem is described, specifically the garbage trucks routing problem in Campolide. The construction of the genetic algorithm follows every aspect presented in the literature review, adapting some steps that in order to solve the routing problem.

In chapter 5, some tests are performed, and the results are described. The subchapters 5.1 and 5.2 discusses multiple tests made in order to define the parameters and conditions that best fit the GA developed, and the definition of waste weight that will be considered in each served street for the algorithm. Later, on the subchapter 5.3, the algorithm is executed with the best-found parameters and the results are assessed. Some discussion and analysis on the best result found can be found at the end of the subchapter 5.3.

The conclusion, limitations and future developments can be found in the chapters 6 and 7. The chapters summarize the comprehension on the results found and its feasibility on real case scenarios including entire cities, taking into consideration the current limitations and future works to be done in the project to solve these limitations.

4. DEVELOPMENT

This project aims to develop a genetic algorithm to deal with the problem of planning efficient routes to garbage trucks in a city. This problem, as stated before, is closely related with the CARP problem. But in real case scenarios, and in the Campolide's case, the available trucks don't have the same capacity. To deal with this difference some adjustments in the algorithms found in the literature must be made. This section gives an overview of the entire development of the project, the problems faced, the solution applied to solve these problems and why the reason on the way the problems were approached.

The next subchapters will briefly contextualize the tools that were used to assist in the GA construction. This is important to understand the structure of data that are gathered and explain the choices on how to deal with this data.

4.1. A BRIEF WORD ON OPENSTREETMAP

OpenStreetMap (OSM) is a collaborative mapping project where volunteers are free to create and edit geographic map information over the world to an open database. This database is also available for free under the Open Database License.

The street data from OSM, that are relevant to this project, are organized in nodes and ways. The node is the smallest point of data in the map, it represents a single point by defining its latitude and longitude, figure 4.1 shows an example of a node in OSM. The nodes can also contain more information inside it, called tags. The tags are used to better qualify the node when it makes sense, for example, if the node represents a pedestrian crossing or bus stops, this information will be referred in tags. Some nodes have no tags, which is not a problem as they are used to represent a path.

```
<node id="21433116" visible="true" version="10" changeset="31019058"
timestamp="2015-05-11T21:05:16Z" lat="38.7244842" lon="-9.1772710"/>
```

Figure 4.1 – OSM's node example

Streets in the OSM are represented using one or more ways. Ways are an ordered list of nodes, the way direction is defined by the order of the nodes in the way, having that the way starts at the first node and end at the last. The way representation can be seen at figure 4.2 Tags are also presented in ways to define things like the street name, the possible directions of the road, among other relevant information. Nodes shared between two or more ways define intersections between streets. Not all nodes are shared, there are nodes that are solely included in one way, for example, a node representing a pedestrian crossing may not also represent a street connection, among other cases.

OSM is a powerful collaborative project that allows a variety of applications and studies on its data. There are much more about nodes, ways and how they relate to each other than discussed above, but those are not relevant to this project and therefore will not be addressed here.

```xml
<way id="233939235" visible="true" version="1" changeset="17392454" timestamp="2013-08-18T08:21:23Z">
        <nd ref="2422521543"/>
        <nd ref="2422521485"/>
        <nd ref="2422521542"/>
        <tag k="highway" v="residential"/>
        <tag k="lanes" v="1"/>
        <tag k="name" v="Rua de Campolide"/>
        <tag k="oneway" v="yes"/>
</way>
```

Figure 4.2 – OSM’s way example

In order to develop this project, data from Campolide streets are required, such as streets directions, length, and connections, this information can be obtained using OSM. Data from OSM can be exported directly from their main website, but in this project, a package called OSMnx was used. OSMnx is a Python package created by Boeing, G. (2017) that facilitates the download of administrative boundaries streets and perform useful calculations using data from OSM.

Using OSMnx to download the data, a graph structure is retrieved with the nodes and ways representing the vertexes and edges of a graph respectively. OSMnx performs data cleansing process automatically on downloads, the nodes that are not used in an intersection and are presented in OSM’s ways are removed by an algorithm. This result in a simplified graph with just the relevant edges and nodes of a certain location.

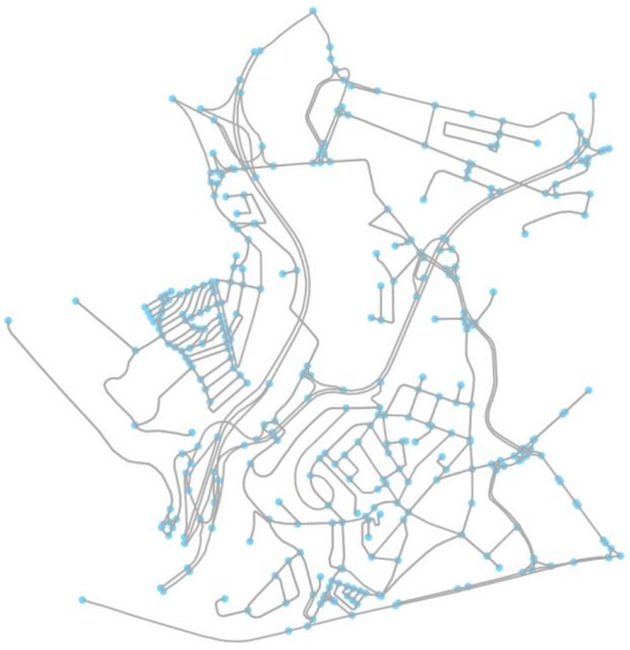


Figure 4.3 – Campolide representation with OSMnx

After the download of data from Campolide and Lisbon, both can be stored in the local machine using the library function *save_graphml* to reduce the amount of time required to the upcoming runs of the algorithm, the graph is saved as a GraphML file into the disk.

4.2. Distance Matrix

The method to calculate the distances between edges in this project's GA will follow the distance matrix proposed by Arakaki & Usberti (2018). The first step is to build a distance matrix between edges of the generated graph with Campolide's ways. To build this matrix, each edge from the graph must have its shortest path value to the other edges, in the case of the edge to itself, the distance is defined as the length of the edge.

In order to accomplish the creation of this Matrix, Lisbon graph data is needed above the Campolide's one. In a city, from any point that a person is, he or she need to be able to get to any other point, if this is not possible, would have streets in the city that could not be reached by any means. In this case, Campolide is not an entire city and is contained inside Lisbon, so it does not provide a path from one edge to all other edges, as can be observed in the figure 4.4 where there are some isolated vertexes. For that reason, a larger graph that contains the studied graph needs to be used to be able to provide valid paths through its edges.

To calculate the distances, the method *shortest_path_length* from *network* package is used. Internally this method uses the Dijkstra's algorithm to calculate the shortest path. The parameters required by the method are the graph, both nodes and the weight that must be taken into consideration, that in this case is the length of the way. The graph provided must be the wider one, the Lisbon graph, because of reasons already defined above.

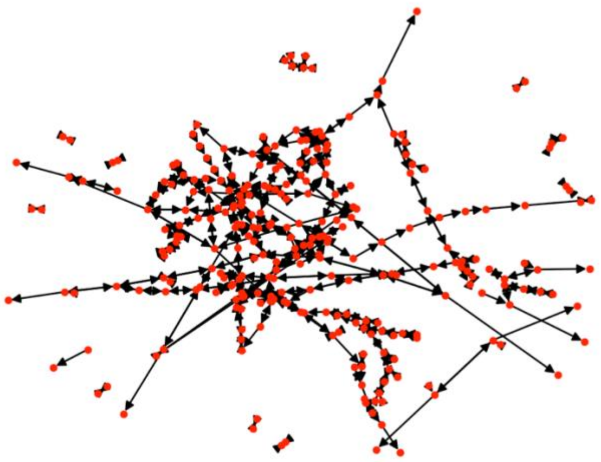


Figure 4.4 – Graph with Campolide's data after removing highways

Although using Lisbon graph as a parameter to the shortest path method, the edges used to iterate the process are from Campolide. Also, only edges that have tags with keys "highways" and values "residential" and "secondary" are taken into consideration as they are the ones where garbage trucks do regular collections. After this filter, 473 edges and 274 nodes are left to be calculated, this graph is represented in figure 4.4. The narrow two-way streets, the one on that a single edge can have both directions, are represented as two directed edges. Both of these edges do not need to be served, because it would be like serving the street twice, first on the right side and then on the left side, and as these streets are narrow enough that they need only be traversed only once. To achieve the objective of serve the edge only once if the truck has one of the ways in its route, a list of corresponding edges needs to be constructed. This list stores every edge that are narrow two ways and its corresponding edge, this mean, the contrary edge that represents the opposite direction on that street. For example, if there is a two way street represented by the edge (a, b), the edges build by the algorithm for it will be (a, b) and (b, a), the list will keep track of this street and correspond the edge (a, b) with the edge (b, a) and the edge (b, a) with the edge (a, b). By doing this, once a truck already served the edge (a, b), it will not be required to serve the edge (b, a), and this can be easily checked by consulting the previously corresponding list.

Every distance is measured in meters and the distances calculated are not distances between edges but nodes, and nodes distances do not make sense in the CARP problem that serves the edges. To solve this problem, the edges distances are calculated according to two different cases.

The equations below exemplify the distance calculations. With the edges $A = (a_i, a_f)$ and $B = (b_i, b_f)$, and having that $D(x_1, x_2)$ is the Dijkstra's algorithm that calculates the shortest distance between two nodes. The equation (1) represents the distance within the same edge, while the equation (2) is the distance between two different edges.

$$d(a, a) = D\left(a_i, a_f\right) \tag{1}$$
$$d(a, b) = D\left(a_f, b_i\right) + d(a, a) + \ d(b, b) \tag{2}$$

When the first and last edges are the same, as shown in (1), the distance will be the total length of the edge, basically, the distance between the first and last node. With different edges (2), will be considered the last node from the first edge, and the first node of the last edge. Added to that are the previously calculated distances of the first and second edges alone. This sum gives the total distance between the two edges.

This process is repeated until every edge in the graph has a distance calculated for each other edge presented. It is a time-consuming task but done only once, with the results stored on the disk, since cities do not change the streets and its orientations often.

After this calculation, the distances between edges can be retrieved in O(1), as it just needs to access the exact point in the matrix. These distances will be heavily used during the GA, so having the information with low complexity worth the time spent at first.

4.3. GENETIC ALGORITHM

Towards the implementation of a genetic algorithm, the representation of the problem, how to measure found solutions and the operations needed in order to generate the upcoming populations must be defined. In the literature review of genetic algorithms was seen the most common representations and operators of a genetic algorithm. This section will discuss how there these apply in this project.

A candidate solution in the population need to the comprehensive, it must carry organized data that allows it to be measured. This is important to the next subchapter that will discuss chromosome representation created for this project. In the subsequent subchapters, the initialization process and genetic operators will be addressed.

4.3.1. Chromosome Representation

This project will make use of a similar approach to the new GVR (Genetic Vehicle Representation) proposed by Pereira, Tavares, Machado & Costa (2002). Besides the fact that their approach deals with the VRP problem, small changes can be made in order to fit the project's problem.

The representation must contain the number of vehicles used, the served ways by each truck, as the order of the service. In this representation, the individuals are composed by a list of garbage trucks, each one containing an ordered subset of ways, each way listed in the set is a served edge by the corresponding truck. One solution must contain every edge that needs to be served in the problem, independent of when a truck will serve it. Each chromosome is a valid solution to the problem.

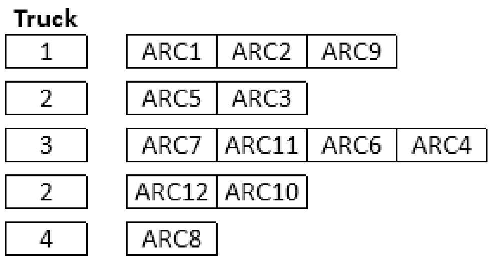
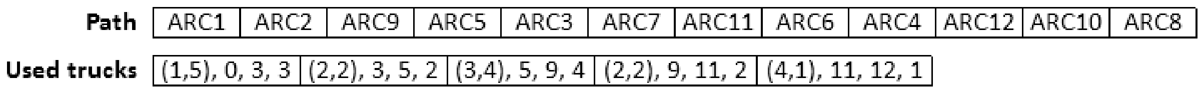


Figure 4.5 – Chromosome representation

In the example above in figure 4.5, we have at least four trucks available to allocate in the routes. The number of ways that each truck is serving is related to its capacity. For simplification, every way demand is 1, the truck 1 can serve at least 3 ways, while the truck four at least 1. In the case of the capacity of the truck exceed, a new truck is chosen, and the remaining edges are allocated, the new truck must have enough capacity for at least the first edge that was not able to fit the previous truck, this process is repeated until every edge is allocated in a truck.

To represent this in a data structure in the program, a class representing the chromosome holds two arrays, one is the array with the path followed by the trucks, this array just keeps a sequence of ordered ways disregarding garbage trucks and capacities. The second array keeps track of the trucks used in the solution, the first and last ways each one serves, their capacity and their load. A truck item in the second array always points to two subsequent indexes in the path array, this is how it keeps track on the first and last way that it serves. It's important to note that the index range referenced by the truck item is closed in the left side and open in the right $[x_1, x_2)$.



Path	ARC1	ARC2	ARC9	ARC5	ARC3	ARC7	ARC11	ARC6	ARC4	ARC12	ARC10	ARC8
Used trucks	(1,5), 0, 3, 3	(2,2), 3, 5, 2	(3,4), 5, 9, 4	(2,2), 9, 11, 2	(4,1), 11, 12, 1							

Figure 4.6 – Chromosome data structure

(truck_id, truck_capacity), start_index, end_index, load

Figure 4.7 – Truck data explanation

This combination of arrays provides the information where each truck serves one or more ways in the solution. This representation also allows the genetic operator to act simpler in the paths array, just needing the trucks array to be updated accordingly.

4.3.2. Fitness function

The fitness of a chromosome is defined as the length of all the routes each garbage truck must accomplish. The route of each truck is given by the edges served by them, adding the distance from the deposit to the first edge and the last edge to the deposit. The deposit is a variable containing an edge previously defined in the algorithm, this edge represents the deposit where the trucks start and end their routes.

$$d(a,b) = d(a,b) + d(a,a) + d(b,b) \qquad (1)$$

$$\textstyle\sum_{i=2}^{n} d(e_{i-1}, e_i) - d(e_{i-1}, e_{i-1}) \qquad (2)$$

The equations above are used to calculate the fitness of a chromosome in the GA. Having that n is the number of edges in a trip, considering that $d(x_1, x_2)$ is the distance between two edges x_1 and x_2, and the edges a and b. The equation (1) calculate the distance of two edges a and b. The equation (2) is the iteration performed by the fitness function in order to calculate the total distance that a truck travels inside a solution.

To calculate the distance between edges the distance matrix previously built is used to retrieve data for the distance function. The distance from the two firsts edges $d(e_0, e_1)$ is retrieved and added into the chromosome fitness. Then, for each edge in the solution skipping the first three edges, the distance between the current edge against the previous edge is calculated using the equation (2). In the equation (2), the distance of the previous edge e_{i-1} is retrieved and subtracted from the sum, this subtraction exists to correct double counts of the distances from the first edge as shown in figure 4.8, the distance matrix stores the distance from the beginning of the first edge to the end of the last edge and this can be counted twice without a subtraction of one of those. In the end, the distances from the deposit to the first edge and the last edge to the deposit are considered using (1), and as before, the correction needs to be made to remove the double count of the edge's distance. The lower the fitness the best is the solution.

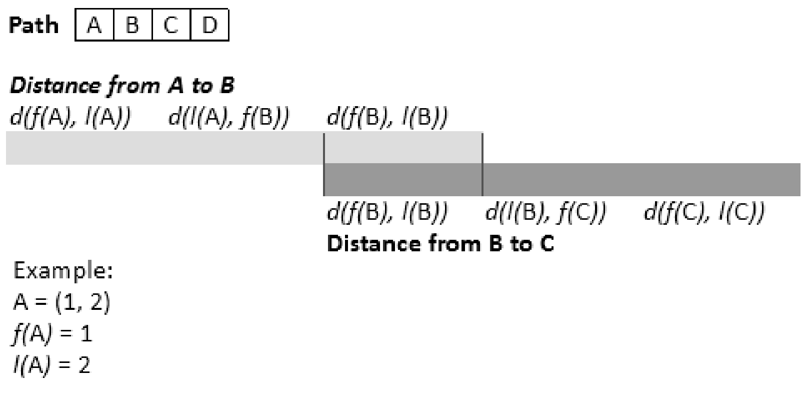


Figure 4.8 – Path distance calculation example

Once the fitness is calculated, it is stored in the chromosome until it suffers some operation that can let the fitness outdated, in this case, the chromosome forget its fitness and once requested needs to be calculated again. This is done to avoid consecutive calculations of the same routes even if these calculations have a low cost.

4.3.3. Initialization

For the initialization of the population, N random chromosomes are generated, N is a variable previously defined in the GA implementation. Each chromosome must represent a feasible solution containing every way that needs to be served by the garbage trucks. To generate a random chromosome, every edge that needs to be served is randomly shuffled, the result of this shuffle will be the exact order that these edges will be served.

The trucks used in the solution are set to carry 90% of their capacity, this is to avoid errors on the difference between the predicted weight and the actual weight of the waste found in the streets. This project performance will be analyzed based on the plastic and metal collection of Campolide, to convert the truck capacity that is in cubic meters to this type of waste kilograms, a conversion table found in the Our Seas Our Future organization website was used.

To know which truck will serve the edges, an iteration over the available trucks is done. In each step, a truck in randomly selected from the poll of trucks. With the truck selected, a new ordered iteration over the unserved edges is done, the edges are assigned to the truck until the capacity of the truck is reached, or no more edges need to be served. In the case of the truck wasn't assigned with an edge, it will not be inserted in the solution and no edge will be marked as served, a new truck will be selected, and the process will continue. On the other hand, if the truck actually served one or more edges and its capacity was reached, the truck will be composing the solution with the edges that it serves, and if there are still more edges to be served, the process will continue selecting a new truck.

The iteration method keeps track of the selected garbage trucks and remove it from the selection poll until every truck had the chance to participate in the process, this gives at least one chance for every truck to enter in the line to serve an edge. Is not guaranteed by the method that every truck will actually serve an edge, but the aim here is to try to utilize every asset that is available to the task.

4.3.4. Operations

The representation, fitness function and the initialization of the population were defined. To use these definitions to evolve the population to the next generations in a genetic algorithm, the operators and how they will act also must be defined, the upcoming subchapters will discuss how the population generate their offspring and evolve over time.

4.3.4.1. Selection

The selection step is when the chromosomes who will act as parents to the new population are selected. These chromosomes will be used for breeding in the crossover step. To select solutions that are diverse among them also relying on how good they are compared to the population, the tournament selection was chosen in this project. The tournament size, that is the number of chromosomes that will participate in the tournament, is a previously chosen parameter. The best fit chromosome of the group wins the tournament and is selected to be a parent in the crossover. The process repeats to generate the next parent. This tournament selection does not remove already selected parents from the population, so the same chromosome that just won a tournament can be selected again in the next tournament. In this project the lower distance is the best fitness, them even if very low fitness solution appears in the tournament group, they are very likely to lose against better fit solutions with lower distance, but this also gives a chance to not so good solutions, as it can happen to select a group where none of the best fit chromosomes are presented.

Additionally, to always carry over the chromosome from the current generation with the best fitness to the next generation of individuals, a method known as elitism or elitist selection will be applied. This method guarantees a spot in the next population to the best individual of the current population, the solution will be carried unchanged, but still can suffer with a mutation in the construction of the offspring. The best solution will be inserted once more in the end of the evolution, to guarantee that it is not changed by the mutation step. This is done to make sure that there is a chance on evolving by mutation the best solution, and to prevent the best solution to be lost in case of the genetic operations change it into a higher fitness solution.

4.3.4.2. Crossover

The crossover operation, as one of the most important operations in a GA, must work with the representation chosen on the chromosomes. In this project it works differently of the common crossover from most of the literature, the genetic material is not exchanged between parents, but a piece of the route is given by one of the parents, while the remaining genetic material is totally got from the other parent. This generates only one child in the crossover operations, because of that, more crossovers must be done to complete the next population. This crossover methods follows the one proposed by Pereira, Tavares, Machado & Costa (2002) but applying small changes on how to generate new routes on truck's capacity overflow.

The crossover randomly selects a sub-route of one of the parents and insert it on the other parent. The insertion point is defined to be just after the way which has the minimum distance between itself and the first way in the sub-route. This operation must guarantee that every solution generated from it is a valid solution.

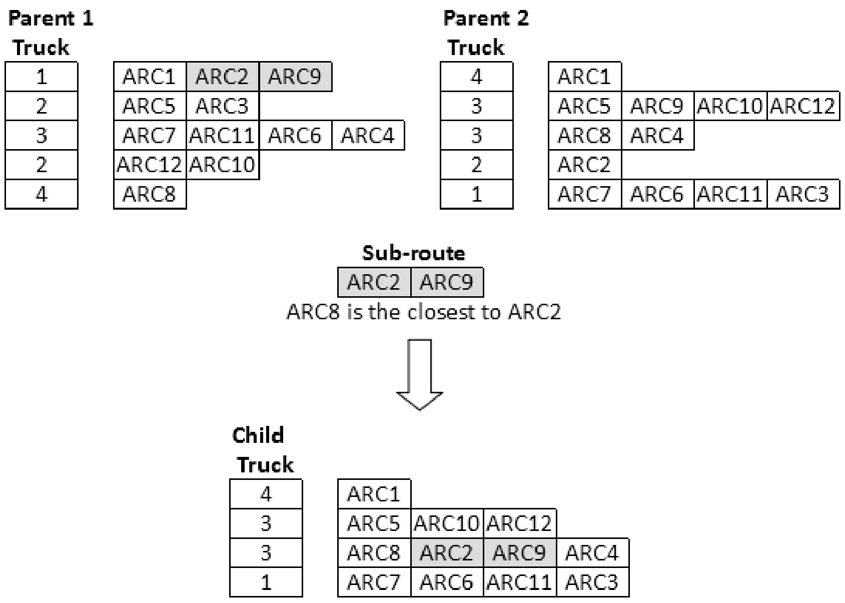


Figure 4.9 – Crossover operation

Summarizing the crossover process, two parents are previously selected in the selection step of the algorithm and a new chromosome is created to store the data generated by the crossover. A sub-route is randomly selected from one of the trucks of the second parent, this sub-route must have at least one edge. The selected sub-route weight is calculated and stored in a variable. One time only, the algorithms verify which edge has the lower distance between itself and the first edge of the sub-route, this verification is done accessing the distance matrix. From now one, the method will go through every truck from the parent 1, and consequently every edge each truck servers. This iteration will append new edges in the child path, and new trucks on the child used trucks array. It is important to notice that if the current edge of the iteration is presented inside the sub-route selected in the first parent, the edge will be not considered, because it would generate duplicated edges when the sub-route is inserted in the new solution. When the iteration finds the edge that is closest to the first edge of the sub-route, the edge will be normally appended in the solution and the sub-route will be inserted just after it. After the insertion of the sub-route in the child chromosome, the process continues until every edge is served by the child.

An example can be observed in figure 4.9. In this example, the sub-route [ARC2, ARC9] is selected from the parent 1. With the distances' matrix was discovered that the ARC8 was the closest edge before the ARC2. While copying the parent 2 edges to the child, the ARCs presented in the sub-route must be skipped, the first one was the ARC9 from the second truck. After that, the ACR8 was reached in the third truck, inserted as normal, followed by the sub-route selected. Having the sub-route inserted in the child, the process continues normally still skipping the ARCs present in the sub-route. Notice that fourth truck was removed from the solution because it became with no edges to serve.

Capacities overflow can happen multiple times in this algorithm, to manage that, there is a list that stores every edge that could not be served for some reason. Once the previous process is complete, an iteration over the edges not served will take place assigning each edge to a truck as it is done in the initialization process of the GA. This guarantee that all edges are served by a truck.

4.3.4.3. Mutations

As the crossover, the mutation step must be specific for this problem being able to deal with the chromosome representation. The mutation is applied to every member of a new population with a low probability. This project implements three kinds of mutation, called swap mutation, inverse mutation, and insertion mutation, each of them have their own probability of being applied. Mutation operators, as the crossover ones, can only generate valid solutions.

In the swap mutation, a truck is randomly select from the chromosome, then a random edge served by this truck in the solution is selected. The same occurs once more in the chromosome. Even if the edges are inside an array without the truck's array influence, is crucial to first select the truck so the capacity and the load of the truck are previously known for the mutation, this information is then utilized to only generate a valid solution. Having that two edges from the chromosome were randomly selected, a swap operation is done, the first edge takes the place of the second edge and the second edge is placed on the first's spot. The swap is done with validation to stop weight overflows from occurring. In the case of the validation fails, the swap process is repeated until the validation is successful, with a maximum of ten attempts. After that, if the validation fails more than ten times, the mutation swap is not done.

The inverse mutation acts just in a sub-route of a truck in the solution. Like the sub-route selection in the crossover step, a sub-route is selected in this mutation. Then the order of the sub-route in inverted and inserted again in the path. This type of mutation differently of the swap mutation can not generate invalid solutions, as the number and weight of the ways are not changed and only the distance can change, if this GA had other limitations like the total length a truck can travel because of its fuel, this could generate an invalid solution and would require a validation. New limitations can be developed and will be discussed in the future works section.

The last type of mutation implemented in this GA is the insertion mutation. This mutation acts on the chromosome changing the position of one way. First, it selects randomly a single way server by a truck, then this way is removed from that truck and inserted in a new truck. This mutation is the only of all three that has a chance of creating a new route in the solution by adding a new truck. The probability of adding a new route is defined as $\frac{1}{2t}$ where t is the number of trucks used in the solution. Like the other types of mutation, this one also guarantees that the solutions are valid in order to be executed.

4.3.5. Termination conditions

The termination condition can be defined using several techniques, as the number of iterations, some criteria, and the fitness evolution. In this project, after the definition of the parameters the termination conditions were defined based on tests done using the algorithm execution.

Was defined that this project will have two different termination conditions methods, one with the maximum number of iterations, and another one that analyzes the evolution of the fitness of the best solution in the population. The maximum number of iterations was set to 4000. In the fitness evolution, if the best fitness does not improve within 800 iterations, the GA will stop and take the best chromosome as the solution found. The first of the two conditions that are reached stop the algorithms, and the best solution will be the chromosome with the best fitness in the end.

5. RESULTS AND DISCUSSION

In order to validate the proposed project, this chapter discusses the results obtained by applying it in the real case scenario of Campolide. The implementation and tests of the proposed GA were done using the Python 2.7 programming language on a MacBook Pro from late 2013 with the following specifications: 2.4Ghz Core i5-4258U processor with 3MB L3 cache, 4GB of 1600MHz DDR3 RAM and macOS 10.13 High Sierra. To decide on the genetic algorithm required parameters addressed in the previous chapter, multiple runs were made, and their results were analyzed. The next paragraphs will present the chosen test cases over the possible parameters combinations and the result of the algorithm execution with these combinations of parameters.

5.1. PARAMETERS DEFINITION

The parameters to be defined are the population size, the tournament size, the crossover rate, and the mutation rate. In the literature a variety of combinations of these parameters can be found, mostly of them set a high probability of crossover, and low probabilities for mutation. Following the parameters adjustment in Pereira et al. (2002) and Karadimas et al. (2007) a set of values for each parameter were chosen based on the value range found in the literature review. There where proposed 2 different values for the population size, two values for the tournament size, three distinct crossover rates, and three values for the mutation rate. This generates a total of 36 different combinations, presented in table 5.1.

#	Population size	Tournament size	Crossover rate	Mutation rate	Initial fitness (km)	Iteration 200 (km)	Iteration 600 (km)
1	76	10%	75%	0%	544	273	180
2	76	10%	75%	1%	540	277	182
3	76	10%	75%	5%	542	266	177
4	76	10%	85%	0%	544	266	178
5	76	10%	85%	1%	545	271	177
6	76	10%	85%	5%	541	271	177
7	76	10%	100%	0%	541	244	163
8	76	10%	100%	1%	542	248	154
9	76	10%	100%	5%	545	242	165
10	76	20%	75%	0%	539	254	173
11	76	20%	75%	1%	540	274	185
12	76	20%	75%	5%	539	258	172
13	76	20%	85%	0%	540	241	159
14	76	20%	85%	1%	538	244	163
15	76	20%	85%	5%	542	250	162
16	76	20%	100%	0%	536	231	163

17	76	20%	100%	1%	540	236	169
18	76	20%	100%	5%	538	233	158
19	126	10%	75%	0%	537	240	158
20	126	10%	75%	1%	537	248	166
21	126	10%	75%	5%	540	238	154
22	126	10%	85%	0%	537	238	159
23	126	10%	85%	1%	539	229	159
24	126	10%	85%	5%	537	231	157
25	126	10%	100%	0%	537	223	156
26	126	10%	100%	1%	539	214	148
27	126	10%	100%	5%	540	227	149
28	126	20%	75%	0%	536	224	154
29	126	20%	75%	1%	539	235	160
30	126	20%	75%	5%	539	228	159
31	126	20%	85%	0%	534	214	151
32	126	20%	85%	1%	540	225	161
33	126	20%	85%	5%	536	238	164
34	126	20%	100%	0%	537	222	153
35	126	20%	100%	1%	538	222	152
36	126	20%	100%	5%	537	214	152

Table 5.1 – GA parameters combinations

For each one of the 36 combinations, the genetic algorithm was executed 10 times performing 10,000 iterations in each run. In total, the algorithm was executed 360 times and performed 3,600,000 iterations. The last three columns from table 5.1 show the average of the ten executions of each combination for the initial population, that was randomly generated, the iteration 200 and 600. In the initial population, the values oscillated 11km between the best (minimum) and worst fitness. In the iterations 200 and 600, the range goes from 11km to 64km and 38km respectively, showing that the parameters indeed influence in the convergence of the solutions.

From the data can be seen that the population size and the crossover rate are the parameters that most influence the solutions' convergence of the algorithm. In figure 5.1 a chart with the iterations of each previously present combination is shown, in this chart can be seen that the combination number 26, represented with a thicker pink line, is the one that has a higher performance on the tests, and its parameters will be used to run the GA. Giving that, the default parameters are set as 126 chromosomes in the population, the tournament size will have 10% of the total size of the population with 13 chromosomes. Every child from the evolved populations will be generated using the crossover, except the two children that have a spot in the next population due to the elitism approach, being that one of them is prone to mutation to give an opportunity to improve its fitness due to mutation.

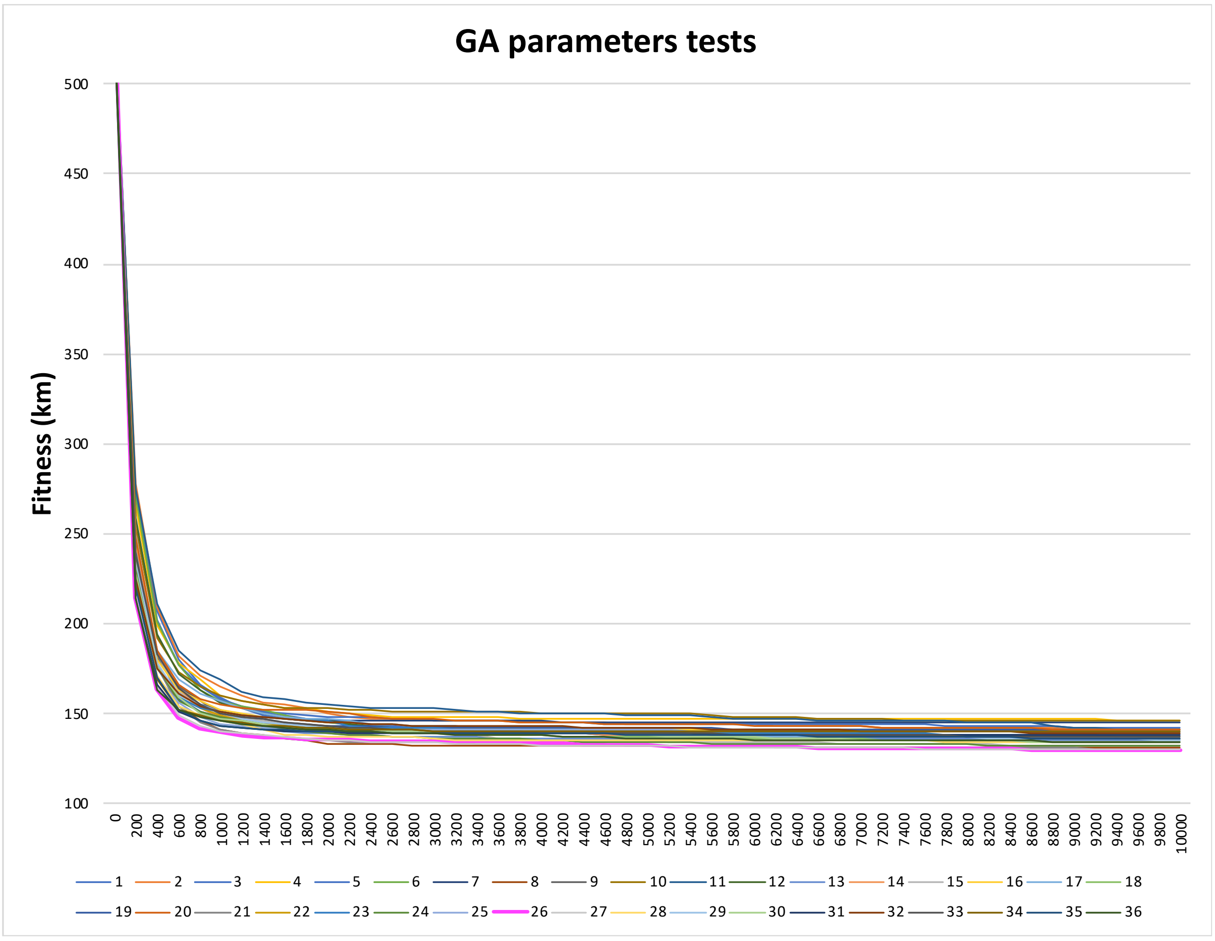


Figure 5.1 – GA parameters choice

Besides the GA's parameters also have to be defined the termination condition of the algorithm. As can be seen in figure 5.1, the convergence of the algorithm is concentrated basically on the first 1000 runs. From 1000 iterations onward the best fitness suffers little changes in its values. Using the chosen combination 26, from the iteration number 1000 to the iteration number 10,000, the fitness dropped from 139 kilometers to 130, a drop of 10 kilometers only, while from the iteration 600 to 1000, the drop was 9km from 148km to 139km. The same approximate amount decreased in the fitness that was observed during 9000 iterations happened in the previous 400 iterations. This leads that the first 1000 iterations must be within the range of the termination condition, but still, that the search is not stagnated and given enough time to the algorithm it still can explore the search space looking for best solutions. Based on the execution results data, the number of 4000 iterations was chosen to be the maximum number of iterations in the termination condition allowed in the GA.

The second termination condition defined in this project is the number of iterations without improvement on the best fitness. This number was defined based on the runs of the GA with the combination 26. The executions show that if 800 iterations have been performed since the last best fitness, it will be hard or take a longer time to find the next best fitness that in all cases do not improve the solution as much to justify the time it takes to compute. Figure 5.2 shows the average evolution of the combination number 26, in the graph can be seen the stated before that the bigger improvement of the GA is located on the first iterations, and as soon it reaches a certain number of iterations, the improvement drops until become irrelevant to the amount of time it needs to be computed.

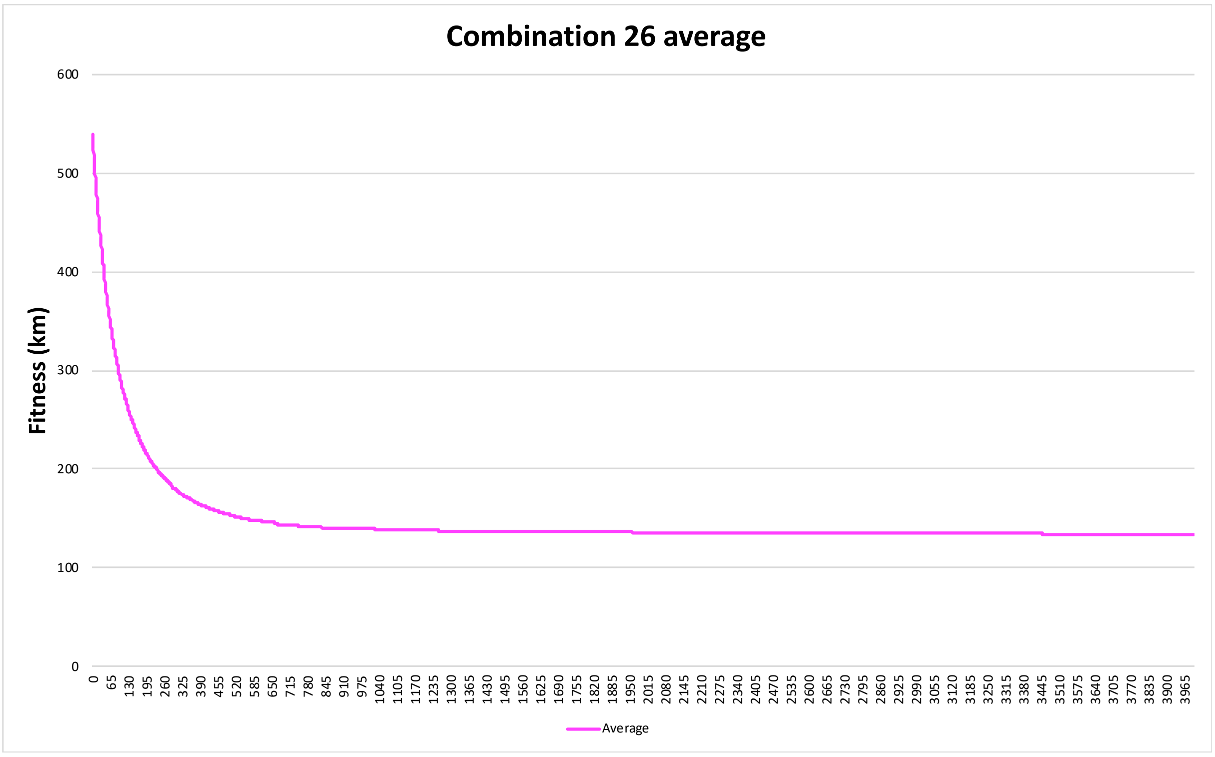


Figure 5.2 – Combination 26 fitness evolution

5.2. Defining the garbage weight to be collected

Besides having the termination condition and parameters defined, the actual weight of each edge must be defined in order to relate the roads collections with the truck's capacities. Fadzli et al. (2015) state that managers usually find it is hard to send the ideal number of vehicles to perform the waste collecting due to the fact that guessing the waste weight to be collected is a difficult process. In the residential door-to-door waste collection, the total garbage weight of each street for collection is unknown, methods to determine the amount of garbage can be found in the literature as the case of CARP with stochastic demands (CARPSD) (Fadzli et al., 2015).

To solve this issue and create a solution that actually meets the reality of the city, this project makes use of open data from the government. Having the collection data from the preceding months, a good approximation can the done on the amount of garbage the city generates, and how it is distributed along the roads. The collected data of the past months provided by the Câmara Municipal de Lisboa is organized by collection route, which means that each current route traveled in a day have the collected weight amount measured. A route is defined as a set of streets traveled by the garbage truck. The routes that contain streets of Campolide can also pass through other parishes. A percentage of the weight collected by the route was measured considering only the streets within Campolide. With the measure of weight collected only in Campolide, a mean of the amount of weight per meter is done for each route in the parish. With that mean calculated, the distance of a way can be used to set the garbage weight of each served street in the problem.

In this real case scenario, the plastic and metal household collection in a period of 4 months was analyzed in order to assign each road the corresponding amount of weight that it will have. Then, the solutions found in the tests are aiming the recyclable collection of plastic and metal only. For other types of waste, different information needs to be loaded into the weight of the edges according to the objective. Using the real amount of garbage collected, the solution implicitly considers the streets population density and its residents' habits of garbage disposal.

5.3. Results evaluation

With the GA operations and parameters defined, the algorithm was executed 10 times in order to get the best result of these runs. Using the best chromosome, the routes can be calculated and plotted on a map allowing to access a visual representation of the solution. The best solution found has a fitness of 113,378.20 meters, using 4 trucks to collect the garbage from the city. The distance traveled going back and forward to the deposit in the route is 57,639.47 meters, and the actual distance used in the collection step is 56,738.73 meters. The route is represented in the map of Lisbon, as can be seen in figure 5.3.

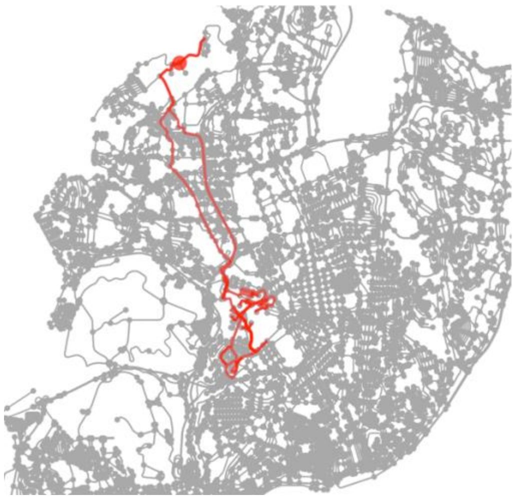


Figure 5.3 – Example of route

The routes, as stated before, starts and end a deposit, the deposit can be observed in figure 5.3 as a big red dot in the top of the figure, can be clearly observed a path going and return from the deposit to the parish of Campolide. To better visualization of the routes found by the GA execution, the map was zoomed in to show only Campolide and the four routes were plotted, figures 5.4, 5.5, 5.6 and 5.7 shows the path that each garbage truck needs to travel in order to perform the collection from every edge defined in the GA.

The routes built by the algorithm clearly shows that the served edges create groups in specifics areas of the Campolide region. Even if it is possible to improve even more the solution, this is a good approximation on which path a set of trucks must travel to serve the city. The actual path can be seen just looking at the edges in the route and listing it in the order it appears in the route, so the driver can follow the directions using both the map and the list of streets that need to be traveled.

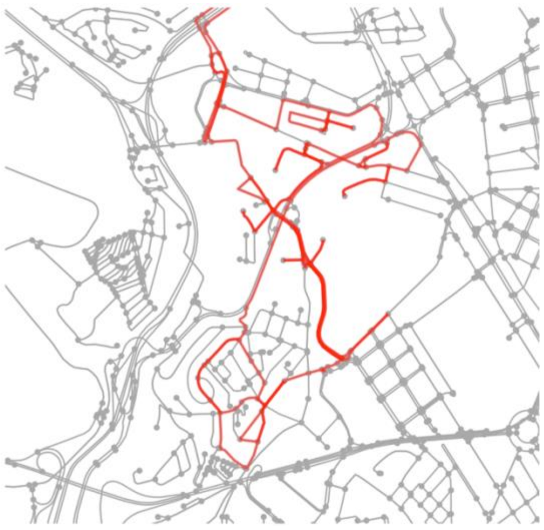


Figure 5.4 – Route 1

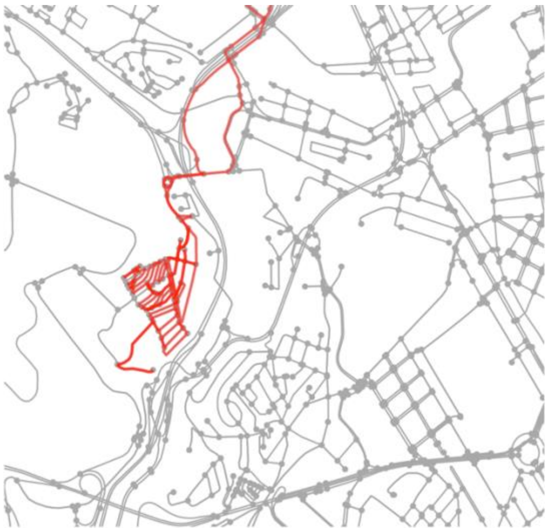


Figure 5.5 – Route 2

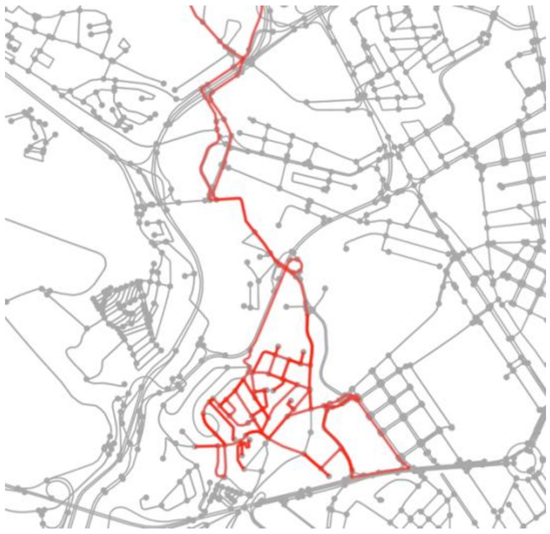


Figure 5.6 – Route 3

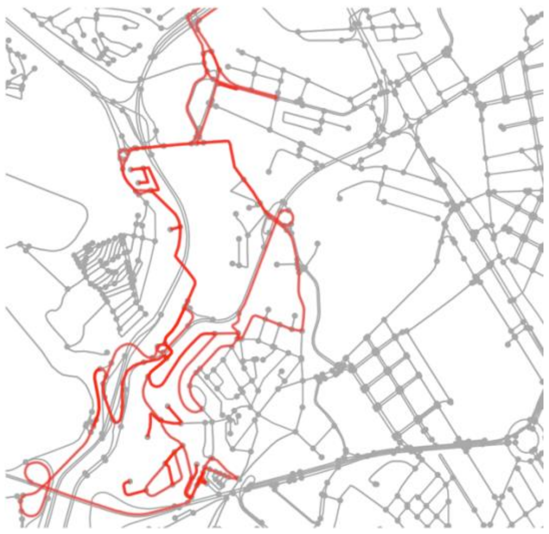


Figure 5.7 – Route 4

Table 5.2 shows the final results for the routes constructed. As can be observed, only one class of truck was chosen to be a part of the solution, which make sense, as the classes on the current stage are not limited by streets constraints, which is one of the limitations and future works that will be discussed further. But the table also shows that the algorithm takes advantage of trucks capacities to allocate the required edges, the percentage of use is greater than 97% in three cases, and 74% in one, probably because it run out of edges that required to be served. The project maximized the use of each truck in the solution, also searching for the shortest distance that each truck needs to travel to complete the collection on all the required streets.

Route	Number of edges served	Truck capacity	Truck load	% load
1	75	1456	1421	97,63
2	70	1456	1439	98,87
3	78	1456	1082	74,32
4	86	1456	1442	99,08

Table 5.2 – Routes trucks information

As this project deals with a simplification of the Lisbon waste collection, addressing just a central region of the entire city of Lisbon, the comparison with the actual routes can not give the exact result on which method is better to deal with the waste collection. The routes within Lisbon are not exclusively in some region and can pass through several regions instead on focus only inside one. This leads that routes serving only Campolide do not exist in the current context. Even more, the actual data do not address the streets covered by the route 2 of the GA's best solution, represented in figure 5.5.

As an addendum, one can realize that the edge highlighted in figure 5.8 was not served by any routes displayed in the figures 5.4 to 5.7. This happens because to plot the graph with the path, the python function receives a list of nodes and draws a red line through the shortest path between each pair of nodes. That edge that is never highlighted, share its two nodes with another road, that tends to always be the shortest between these nodes. So, even if the edge that represents that street is within a route, and it indeed is, it will not show highlighted with red in the map using the current plot function.

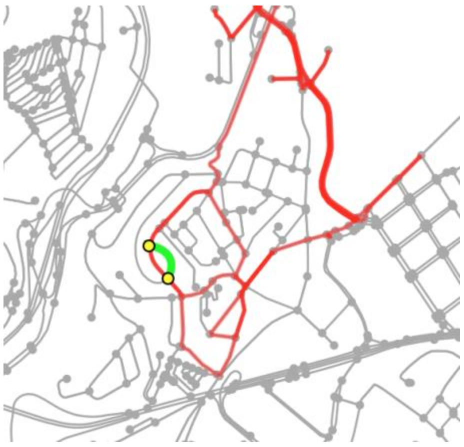


Figure 5.8 – Piece of route 1 highlighting road without red highlight

Even that on this project only the data from Campolide, which is a region inside the city of Lisbon, was used to evaluate the algorithm, the project showed itself feasible for approaching large instances as a city. Because only within Campolide the GA had to deal with 473 edges, and there is still space for optimization on performance.

6. CONCLUSIONS

Build routes for waste management collection within a city is a hard computational problem. Some cities solve this problem by using ICT components as sensors to assist the reduction of distance traveled by the garbage trucks in order to perform the waste collection of the city. But this method can only be applied where the collection is done mainly in pre-defined spots where waste bins are located, and can use algorithms that solve VRP to address it. For the door-to-door collection, which includes trucks going through each household to perform the collection, this method is harder to be applied. The door-to-door collection issue is usually modeled as a CARP problem. Even though the CARP attracts less attention than other routing problems like VRP, several authors in the literature address the CARP in order to solve issues related with routes that must serve edges instead of nodes in a graph.

This project seeks environmental and economic improvement to the city stated by the concept of smart cities. Making use of information and technologies that already exist in the majority of big urban sites around the world, it mixes GA, open data and geographic information to solve the CARP problem using vehicles with different capacities. The differences from this approach to others in the literature are that it uses actual past collection data to determine the garbage weight of the served streets, this generates a much more accurate information to feed the GA in the search for a solution to the problem. And the GA uses a new genetic representation, following the representation built by Pereira et al. (2002). Additionally, trucks with different capacities can be represented using this representation. The size of the vehicle can be used to limit it to certain roads only, a usual situation found in the city of Lisbon that has many narrow streets due to its historical nature, this means that bigger vehicles can't serve narrow streets that are prohibited for larger vehicles. Even though the roads and vehicles constrain are possible in this representation, more specific data from the cities' streets must be collected in order to apply these limitations.

The results proved that the GA can be good enough to build the routes traveled by the garbage trucks, reducing the amount of distance needed to perform the street garbage collection. Parameters and termination conditions were set based in multiple tests in order to minimize the computing time required to the GA converge and assess the results. An example of a result showing the routes in a map and a table with the trucks loads can be found in the session 5.2, this solution addresses the principal objective of the project. Another advantage of this project if that the solutions generated by the project adapts itself to the current state of the city. As stated in the introduction, the urbanization process will continue in the next years, and due to how the cities manage the allocation of people within its borders or even due to how smart cities create centers that attract more people over time, the distribution of garbage will continually change over time. Since this project uses current data to predict the waste that streets generate in the near future, the adaptation process occurs naturally while executing the project.

7. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

This project enables the use of different types of collection vehicles, that is more realistic than the usual CARP representation where a homogeneous vehicle's fleet is available to serve the entire city. A limitation that can be found in this project, as discussed before, is the lack of information on the street width, that prevents to set the width of the edges in the GA. This reduces the reality of the result as exists streets where a larger truck can not traverse. Also, without these constraints set in the streets, the trucks with greater capacities will always generate better results since it can carry more waste until return to the deposit to unload. Once the street widths start feeding the GA, it will make sense to perform a multi-objective optimization instead of a single objective as it is currently done. The Pareto frontier can be used to minimize both the total length of the routes as the number of trucks used in the solution.

Other limitation found and future work to be done in the project is the ability to prevent prohibited turns, best known as U-turns. This leads to trucks performing prohibited turns when it is more convenient to follow the shortest path between the current location and the next edge to be served. This can be solved by changing the graph to keep a list of possible successors for each street piece represented in the graph, with this technique turn prohibitions become transparent and a vehicle will be only able to follow the designed paths from its current location (Lacomme et al., 2001).

The major limitation of this project was to find directed CARP instances with distinct trucks to compare the results. Since the CARP study is not as popular as other types of routing, few materials are available to enable a comparison on the built model and the existing algorithms built to solve this problem. Even the actual collection data do not address all the streets that this algorithm does, making hard and unfair the comparison between both of them.

The prediction of garbage weight to be collected from the streets can take more into consideration in future works. Fadzil et al. (2015) presented some studies done that apply external factors like hot weather or rainy days to the garbage collection to minimize the cost and environmental impact. For example, in rainy days household waste can act as a sponge that absorbs rains drops and increases its weight, consequently increasing its impact on the capacity of the collection vehicle.

Another constraint that will be implemented in a future version of the algorithm is the total amount of fuel that a truck can carry and its consumption trying to consider the load. The consideration of the capacity already used is important because as the truck traverses the required streets and increase its load, its weights get higher requiring more energy to move. With this constraint the GA will enable the truck to travel until it has fuel.

There are still many areas that can be explored since that the CARP receive little attention compared with other routing methods. The algorithm developed in this project can still be improved to address much more limitations to improve its performance. The future developments of this project will take into consideration the discussed limitations and future works, as try to compare it with other solutions.

8. BIBLIOGRAPHY

Arakaki, R. K., & Usberti, F. L. (2018). Hybrid genetic algorithm for the open capacitated arc routing problem. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2017.09.020

Bătăgan, L. (2011). Smart Cities and Sustainability Models. *Revista de Informatică Economică*, *15*(3), 80–87. Retrieved from http://revistaie.ase.ro/content/59/07 - Batagan.pdf

Beliën, J., De Boeck, L., & Van Ackere, J. (2011). Municipal Solid Waste Collection and Management Problems: A Literature Review. *Transportation Science*. https://doi.org/10.1287/trsc.1120.0448

Buenrostro-Delgado, O., Ortega-Rodriguez, J. M., Clemitshaw, K. C., González-Razo, C., & Hernández-Paniagua, I. Y. (2015). Use of genetic algorithms to improve the solid waste collection service in an urban area. *Waste Management (New York, N.Y.)*. https://doi.org/10.1016/j.wasman.2015.03.026

Caragliu, A., del Bo, C., & Nijkamp, P. (2011). Smart cities in Europe. *Journal of Urban Technology*, *18*(2), 65–82. https://doi.org/10.1080/10630732.2011.601117

Carr, J. (2014). An Introduction to Genetic Algorithms. *Whitman College Mathematics Department*. https://doi.org/10.1016/S0898-1221(96)90227-8

Catania, V., & Ventura, D. (2014). An Approach for Monitoring and Smart Planning of Urban Solid Waste Management Using Smart-M3 Platform. *15th Conference of Open Innovations Association FRUCT*. https://doi.org/10.1109/FRUCT.2014.6872422

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., … Scholl, H. J. (2012). Understanding smart cities: An integrative framework. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (pp. 2289–2297). https://doi.org/10.1109/HICSS.2012.615

History of OpenStreetMap. (2018, July 28). OpenStreetMap Wiki, . Retrieved 19:04, July 31, 2018 from https://wiki.openstreetmap.org/wiki/History_of_OpenStreetMap.

de Oliveira Simonetto, E., & Borenstein, D. (2007). A decision support system for the operational planning of solid waste collection. *Waste Management (New York, N.Y.)*. https://doi.org/10.1016/j.wasman.2006.06.012

Deng, X., Zhu, Z., Yang, Y., Li, X., Tian, Y., & Xia, M. (2007). A Genetic Algorithm for the Capacitated Arc Routing Problem. In *Proceedings of the IEEE: International Conference on Automation and Logistics*. https://doi.org/10.1145/1569901.1569947

Fadzli, M., Najwa, N., & Luis, M. (2015). Capacitated arc routing problem and its extensions in waste collection. In *AIP Conference Proceedings*. https://doi.org/10.1063/1.4915634

Fujdiak, R., Masek, P., Mlynek, P., Misurec, J., & Olshannikova, E. (2016). Using genetic algorithm for advanced municipal waste collection in Smart City. In *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*. https://doi.org/10.1109/CSNDSP.2016.7574016

Giffinger, R. (2007). Smart cities Ranking of European medium-sized cities. *October*, *16*(October), 13–18. https://doi.org/10.1016/S0264-2751(98)00050-X

Grimm, N. B., Faeth, S. H., Golubiewski, N. E., Redman, C. L., Wu, J., Bai, X., & Briggs, J. M. (2008). Global Change and the Ecology of Cities. *Science*, *319*(5864), 756–760. https://doi.org/10.1126/science.1150195

Guerrero, L. A., Maas, G., & Hogland, W. (2013). Solid waste management challenges for cities in developing countries. *Waste Management*. https://doi.org/10.1016/j.wasman.2012.09.008

Gurstein, M. B. (2011). Open data: Empowering the empowered or effective data use for everyone? *First Monday*. https://doi.org/10.5210/fm.v16i2.3316

HAN, H., & Cueto, E. P. (2015). Waste Collection Vehicle Routing Problem: A Literature Review. *PROMET - Traffic&Transportation*. https://doi.org/10.7307/ptt.v27i4.1616

Holland, J. H. (1992). Genetic Algorithms. *Scientific American*. https://doi.org/10.1038/scientificamerican0792-66

Hollands, R. G. (2008). Will the real smart city please stand up? *City*, *12*(3), 303–320. https://doi.org/10.1080/13604810802479126

Hoornweg, D., & Bhada-Tata, P. (2012). *What a waste: a global review of solid waste management*. *World Bank, Washington DC*.

Huijboom, N., & Broek, T. Van Den. (2011). Open data : an international comparison of strategies. *European Journal of EPractice*. https://doi.org/1988-625X

Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, Adoption Barriers and Myths of Open Data and Open Government. *Information Systems Management*. https://doi.org/10.1080/10580530.2012.716740

Jebari, K., & Madiafi, M. (2013). Selection Methods for Genetic Algorithms. *International Journal of Emerging Sciences*.

Karadimas, N. V., Papatzelou, K., & Loumos, V. G. (2007). *Genetic algorithms for municipal solid waste collection and routing optimization*. *IFIP International Federation for Information Processing*. https://doi.org/10.1007/978-0-387-74161-1_24

Kumar, M., Husian, M., Upreti, N., & Gupta, D. (2010). Genetic Algorithm: Review and Application. *International Journal of Information Technology and Knowledge Management*.

Kumar, V., & Panneerselvam, R. (2017). A Study of Crossover Operators for Genetic Algorithms to Solve VRP and its Variants and New Sinusoidal Motion Crossover Operator. *International Journal of Computational Intelligence Research ISSN*.

Lacomme, P., Prins, C., & Ramdane-Cherif, W. (2001). A genetic algorithm for the capacitated arc routing problem and its extensions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-45365-2_49

Mendes, A. C. (2017). Almeida Henriques: “Smart cities” precisam de apoios. Retrieved June 1, 2018, from https://www.jornaldenegocios.pt/negocios-iniciativas/cidades-inteligentes/detalhe/almeida-henriques-smart-cities-precisam-de-apoios

Mitchell, M. (1995). Genetic algorithms: An overview. *Complexity*, *1*(1), 31–39. https://doi.org/10.1002/cplx.6130010108

Mohammed, M. A., Abd Ghani, M. K., Hamed, R. I., Mostafa, S. A., Ahmad, M. S., & Ibrahim, D. A. (2017). Solving vehicle routing problem by using improved genetic algorithm for optimal solution. *Journal of Computational Science*. https://doi.org/10.1016/j.jocs.2017.04.003

Molloy, J. C. (2011). The open knowledge foundation: Open data means better science. *PLoS Biology*. https://doi.org/10.1371/journal.pbio.1001195

Mourão, M. C., Nunes, A. C., & Prins, C. (2009). Heuristic methods for the sectoring arc routing problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2008.04.025

Nam, T., & Pardo, T. A. (2011). Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference on Digital Government Innovation in Challenging Times - dg.o '11* (p. 282). https://doi.org/10.1145/2037556.2037602

OpenStreetMap contributors. (2015) Planet dump [Data file from June, 2018]. Retrieved from https://planet.openstreetmap.org.

Pereira, F. B., Tavares, J., Machado, P., & Costa, E. (2002). GVR: A New Genetic Representation for the Vehicle Routing Problem. *Artificial Intelligence and Cognitive Science: 13th Irish International Conference, AICS 2002, Limerick, Ireland, September 12-13, 2002 - Proceedings*. https://doi.org/10.1007/3-540-45750-X_12

Ramdane-Cherif, W. (2006). Evolutionary algorithms for capacitated arc routing problems with time windows. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*.

Saini, N. (2017). Review of Selection Methods in Genetic Algorithms. *International Journal Of Engineering And Computer Science*. https://doi.org/10.18535/ijecs/v6i12.04

Science for Environment Policy. (2015). Indicators for sustainable cities. *European Commission*, (12), 1–189. https://doi.org/10.2779/61700

United Nations. (2014). *World Urbanization Prospects 2014*. *Demographic Research*. https://doi.org/(ST/ESA/SER.A/366)

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*. https://doi.org/10.1007/BF00175354

Willemse, E. J., & Joubert, J. W. (2016). Constructive heuristics for the Mixed Capacity Arc Routing Problem under Time Restrictions with Intermediate Facilities. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2015.10.010

Wøhlk, S. (2008). A decade of Capacitated Arc Routing. *Operations Research/ Computer Science Interfaces Series*. https://doi.org/10.1007/978-0-387-77778-8_2