FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Masters' Degree in Informatics Engineering
Dissertation

COSMO

Evolutionary Approach to Inverse Shortest Path in an Urban Mobility Context

September 4, 2013

Gustavo Fernandes
gnvf@student.dei.uc.pt

Advisor at DEI:
Penousal Machado

Abstract

Cities across the world are fighting a losing battle against the increasingly large tides of private and public transports that daily roam urban road networks. In an effort to tackle this problem, we present a thorough research of the current traffic management and routing algorithm paradigm, and compare the available traffic simulator options. We also elaborate a plan of action by establishing goals, tasks and milestones. We propose an evolutionary approach to generate a behavior model, based on the inverted shortest path, to later use in SUMO (Simulator of Urban Mobility).

Keywords: inverted shortest path, evolutionary computation, traffic microsimulation, collaborative route planning, intelligent transport systems

Acknowledgements

I would like to dedicate the first words of this thesis to thank the people who made this work possible and led me through this important stage in my life.
Thus firstly I would like to thank my advisor Professor Fernando Jorge Penousal Martins Machado for all the guidance, advice and insights he provided, which were fundamental to the completion of this research.
I also would like to thank my family and friends for their unconditional support, advice and encouragement throughout my academic life. Furthermore I would like to thank my Bachelor and Masters degree colleges for their shared wisdom and companionship all over these years.

Gustavo Fernandes

Contents

List of Figures

1 Introduction

In this first chapter we will present the reader with current issues that will benefit from this study. In section 1.1 we will analyze and discuss how this research will contribute to addressing current issues related to urban mobility, which will ultimately have an impact on day-to-day activities in real world scenarios. Then in section 1.2 we will contextualize the reader to problem at hand and, finally, explain our goals in section 1.3.

1.1 Motivation

The inhabitants of metropolitan areas follow a lifestyle that stands on the ability to travel within the city for both work and leisure. Moving has become more and more essential in the urban routine, either to: run a errand; commute; attend an event; etc. Because of this generalized need to circulate within the city road planning and traffic handling has become a major issue in the last decades [17]. Considering that there is a finite infrastructure of roads, if there is a growing, vastly large amount of vehicles used for transport, they will sooner or later overflow the road network's capacity and ultimately cause traffic jams or congestions and therefore lower mobility.
To cope with the rapidly decreasing ability to move from one place to the other within urban bounds society has attempted many different approaches and yet the problem remains. Many traditional attempts have been made to address the issue such as: rerouting traffic; increase/decrease certain area's motorization factor; promote the use of public transportation; etc. Although it temporarily eased the situation it did not manage to solve it. This problem even transcends the advent of ubiquitous system, geo-localization and other mobility enhancement technology, which like the other methods have worked as a ephemeral or minor solution, as mobility within cities limits still kept

decreasing [17].
In the recent past, with the advent of car navigation systems, instead of trying to work their way in the physical infrastructures researchers attempted to shape how people navigate, developing algorithms to get people from "A to B" following shortest (or fastest) path possible. Once we know where we are, where we want to go and the composition of the road network, we can calculate what is the shortest path is, but we cannot expect drivers to always follow the theoretical shortest path, as they tend follow the paths that have lesser perceived costs[7] which is mostly subjective. One way to approach this problem would be to reconsider the distances between nodes in the road network and adapt it to the perceived costs, or preferences, of the drivers. If we can reach the desired outcome, in our attempt to reconsider the distances between nodes of the road network according to driver usage, we can provide traffic modelers information that they could use to improve city's land-use planning [7]. In order to achieve this we consider the Inverted Shortest Path approach.

1.2 What is COSMO?

The COSMO project (COllaborative System for Mobility Optimization), undertaken by Faculdade de Ciências e Tecnologia of the University of Coimbra, is a research study focused in developing traffic prediction algorithms and software based on a non-invasive collaborative approach. The main target is to study the issues detailed in the previous section and develop a collaborative traffic control system in order to tackle them.
To achieve this COSMO takes advantage three areas of knowledge to do its three main tasks:

- **Ubiquitous Computing** to get the data from real vehicles traces in real maps, in order to achieve a realist setting build-up.
- **Intelligent Transport Systems** are taken in account when designing the algorithms, which are used alongside with Vehicle Routing; Evolutionary Computation; Map Matching; Geo-Referenced applications.
- **Complex Systems** are used to validate the data and tune the algorithm. A traffic micro-simulator platform, such as SUMO, which deals fairly efficiently with this kind of random disorganized networks, will be adopted for simulation and validation purposes.

COSMO already produced contributions towards the theoretical improvement of traffic flow, using a rerouting system based on a ant colony algorithm, which provided valuable results handling not only academic road networks but real-world maps, namely Coimbra's road network. This work will be focused on using evolutionary computation on the inverse shortest path problem to provide better guidelines to urban planners to cleverly construct and improve road networks.

1.3 Objectives

The main objective of this thesis is to analyse the current approaches to the Inverse Shortest Problem and provide a different alternative will be competitive with the state of the art. We will adopt an evolutionary approach, with the goal of promoting traffic flow and ease road congestion.
We will set a parallel with the Open Shortest Path First and then use the Inverted Shortest Path Problem to alter the network perception hopping to achieve better trip times and lengths for the road network users.

1.4 Outline

In this chapter it was presented the motivation for this work, alongside with the area of research in which it fits. It was also exposed this work's framing within the COSMO project, this thesis objectives, and a brief summary of the documents structure.
On Chapter 2 state-of-the-art research will be presented and briefly analyzed, providing some insight on work, by other authors, related to this dissertation.
Then in Chapter 3 we explain the tool used to produce our evolutionary algorithm, how it underwent a optimization process and setting side by side with Open Shortest Path First problem.
In Chapter 4 we will begin to set the problem in a urban mobility context. In this part, we explain how we used the chosen traffic simulator and how the designed algorithm fits in its structure.
Afterwards in Chapter 5, we set the stage for our testing phase, detailing process pursued while performing the test as well as the results they produced. We then proceed to analyze and discuss said results, drawing conclusions and comparing them to our expectations.
Finally in Chapter 6, we make a brief analyzes of the problem and the approach, suggesting ways to complement this study in the future and stating the author's final thought's on this subject.

2
State-of-the-Art

In order to take steps in an innovative fashion we must comprehend what was and is being done to tackle the problem at hand, thus helping the reader understand how this work differs from others and how other works help on the definition and execution of this one.
We will start by analyzing the current traffic simulation state-of-the-art, then we introduce the reader to the current paradigm on path-choosing algorithms. To do so we must start by presenting at which points do graph theory and road networks meet. Then we will expose how other studies have addressed path choosing, weight setting and related problems.

2.1 Traffic Simulation

Experimentation is an essential part of the scientific method as it is a mean of confirmation of the preceding hypotheses, however it's not always possible or convenient to perform it in a real-world scenario [14], which leads us to search for viable alternatives.
In our specific context, testing navigation algorithms using real drivers and vehicles simply is not feasible. It would take too much time, resources, volunteers and funding [14] to perform in such a experimentation, especially during the tunning part of the algorithm development. Because of that, tanks to current machine power, simulators have been developed to cope with necessities such as ours [23]. Due to the plentifulness of traffic simulator software, we are bound to make an analysis of the area in order to make an informed decision. Therefore, in this section, we will compare and discuss the main simulation software choices that are currently available within the scope of our work.

2.1.1 Simulator Type Selection

Traffic simulating software can be divide in three main types:

- **Macroscopic simulators** which model the traffic flow in accordance with high-level mathematical models most commonly based in fluid dynamic, allowing a continuous simulation. They are most commonly used in simulating wide areas with little detail and, therefore, not adequate to urban scenarios [28] [23].

- **Microscopic simulators** that comprehend individual entities separately with high-level detail in a discrete context. This traffic modeling type is the most realistic due to their individual approach to vehicles and drivers and is widely use to evaluated traffic control along with vehicular routing [28] [23] [14].

- **Mesoscopic simulators** are a in-between designation of an hybrid of the latter two. Although they consider groups of entities, they still describe their interaction and behavior with the same level of detail of macroscopic simulators [28].

Since our scope relates to individual traffic routing we found that the Macroscopic Simulation approach did not serve our interest as it does no deal with individual vehicles and drivers. Furthermore it is ill advised to use this simulation technique in an urban setting, which we set to use. Between the remainder types of simulation, we found best approach for this research to be Microsimulation, since it was already used at COSMO to keep the experiments as realistic as possible.

2.1.2 Selection Criteria and Individual Analysis

Due to the abundance of microsimulation software alternatives we were forced to cut down the list by reading third party reviews. Thus, we found the following to be the most well reviewed amongst the virtual simulation community:

- SUMO - *Simulation of Urban Mobility*

- QuadStone Paramics Modeller

- Aimsum

- Trafficwere SimTraffic

. Which we proceeded to deploy and experiment (when it was possible) and compared according to these criteria:

- Open Source and Free Usage
- Documentation and Interface
- Realism
- Performance

2.1.2.1 Open Source and Free Usage

Of the five simulators considered only SUMO, was gratis and Open-Source. Being open-source allows the user to use it at his will with no boundaries, shaping it to more adequately fit his intent. This degree of customization along with it being free is apparently a rare feature in this type of software and can provide a unique advantage.
Although none of the others were neither Open-Source and free, we experimented their trial versions, and drawn conclusion from them as well as from the available documentation.

2.1.2.2 Documentation and Interface

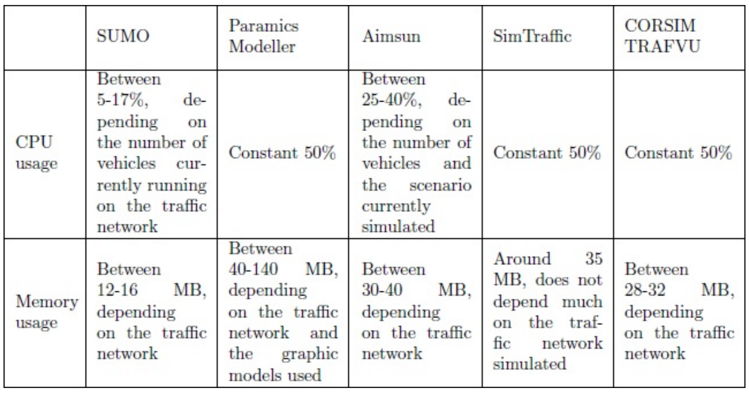
When dealing with other software and code it is important to have access to a solid documentation base. Because you need to know how things work and these are very complex pieces of computer code, if everything is well explained it minimizes programming difficulty.
Regarding the documentation of the previously selected programs, the only noteworthy finding is that Aimsum lacks both a user manual or documentation. On a more positive side, we discovered that SUMO has a considerably large community, which can be used to clarify doubts about its inner workings.

2.1.2.3 Realism

As previously stated it is of paramount importance to infuse this research with the maximum realism in our grasp. Therefore we must consider the similarities between the simulators and real life traffic.
Regrading this topic, our choice will go towards a simulator allowing different kinds of vehicles (e.g. bicycle, truck, bus, car), pedestrians, traffic lights, ability to integrate real maps, etc.

2.1.2.4 Performance

Has we have limited time to carry out this research, we must consider performance as a factor in this decision. There was no need to do performance testing as this analysis has been already performed by Kotushevski [23].

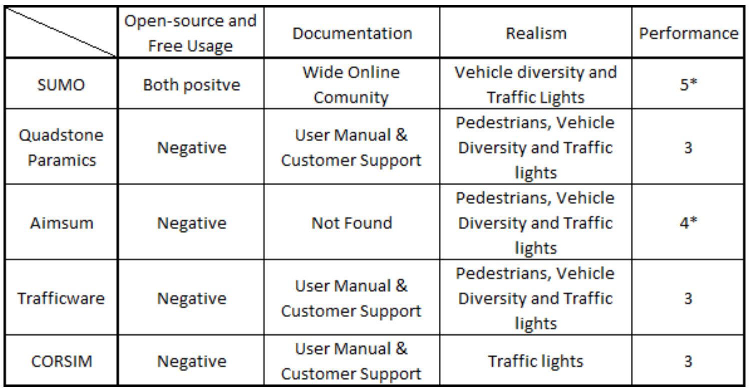


	SUMO	Paramics Modeller	Aimsun	SimTraffic	CORSIM TRAFVU
CPU usage	Between 5-17%, depending on the number of vehicles currently running on the traffic network	Constant 50%	Between 25-40%, depending on the number of vehicles and the scenario currently simulated	Constant 50%	Constant 50%
Memory usage	Between 12-16 MB, depending on the traffic network	Between 40-140 MB, depending on the traffic network and the graphic models used	Between 30-40 MB, depending on the traffic network	Around 35 MB, does not depend much on the traffic network simulated	Between 28-32 MB, depending on the traffic network

Figure 2.1: Performance analysis. Adapted from Kotushevski [23]

As Figure 2.1 displays, SUMO when used with a small number of vehicles uses less processing resources as well as less memory than the other simulators. However, in terms of scalability SUMO tends to perform worse than Paramics, SimTraffic and CORSIM TRAFVU.

2.1.3 Comparative Analysis



	Open-source and Free Usage	Documentation	Realism	Performance
SUMO	Both positve	Wide Online Comunity	Vehicle diversity and Traffic Lights	5*
Quadstone Paramics	Negative	User Manual & Customer Support	Pedestrians, Vehicle Diversity and Traffic lights	3
Aimsum	Negative	Not Found	Pedestrians, Vehicle Diversity and Traffic lights	4*
Trafficware	Negative	User Manual & Customer Support	Pedestrians, Vehicle Diversity and Traffic lights	3
CORSIM	Negative	User Manual & Customer Support	Traffic lights	3

Figure 2.2: A general comparison of all simulators according to the previously stated criteria. * in low vehicle number scenarios.

As you can see in Figure 2.2, Sumo has the best classification in every criteria except realism, where is only slightly behind. In light of these, it is our opinion that SUMO is the best microsimulation software choice for our purposes and thereby it will be used our in the simulation period.

2.2 Basic Notions

We will now expose basic concepts regarding the to main aspects of the thesis: Graph Theory and the Inverted Shortest Path Problem.

2.2.1 Road Networks in Graph Theory

The similarities between road networks and geometric graphs are the foundations that support current route predicting approaches. In order to know which formula's, assessment tools and proprieties used in graph theory can be used when dealing with road networks it is paramount to cogitate in which type of graph do road networks are more approximated to.
A geometric graph consists on a set of objects where some of are connected to each other, which are represented by mathematical abstractions called vertices or nodes and the connections between them are named edges or arcs. In a specific type of graph each edge has an attribute called weight which is commonly used to represent a cost once you used that edge to go from one node to another. [33] Although road networks consist basically in geometric graphs, has they can also be described as set of road intersections linked by a number of roads, each of the latter with a cost of travel(length), because a mathematical graph is an abstraction and road networks exist in the reel world, there are some differences worth depicting. The two most noticeable of the dissimilarities between them are related to the size of the set and weight of the vertices: a road network as a physical man-made structure is finite and a graph as a mathematical abstraction may be not; an edge may have a non-negative weight, but such concept does not make sense when applied to a road.
In graph theory graphs can be divided in two categories: planar and non-planar. From merely graphic point of view, a graph is planar if a graph can be drawn on a plane without having any of its edges crossed over each other, otherwise is a non-planar graph [33]. Kuratowski's theorem takes it to a more mathematical point of view stating that a graph is only planar if it does not contain as a sub-graph either a K_5 (a complete graph with five vertices) or a

$k_{3,3}$(complete bipartite graph on six vertices, three of which connect to each of the other three,) [24].
As the earth consists of an approximated sphere one can easily think of a road network as a graph drawn upon the flatten surface of a spread out earth's crust. Although road networks can be thought of as planar graphs, they are non-planar. Consider the roads as the edges and the road intersections as the nodes in a graph, if you would envision a bridge overpass or a tunnel under another road you be envisioning the graph equivalent of a edge intersection, making the road network a non-planar graph. Despite this, most of statistical and measurement tools used only in planar graphs are commonly used on road networks. For instance the gamma and alpha indexes which are a measure of connectivity in a graph, are used to evaluate the density of roads and though they suffice it must not be forgotten that we're dealing with non-planar graphs [13].
Therefore in this paper, in order to later find possible applications in the real world, it will only be considered finite planar graphs with non-negative edge weights.

2.2.2 Shortest Path Problem

The Shortest Path Problem (SPP) has found a considerably large number of different approaches and strategies each one with its own complexities and it's own applications. Dijkstra's algorithm design in 1956 is probably the most well know method used to find the shortest path between two vertices in a finite graph with non-negative edge weights. To summarize the method consists in navigating through every vertex labeling each one with the current distance traveled to get there and if it's already labeled relabel with the shortest distance, until it gets to the desired destination vertex, running in a $O(|V|^2)$ complexity [11]. Since it was conceived there have been some improvements to it, namely by Fredman and Tarjan which by the use of a min-priority queue implemented Fibonacci heap rendering it running in $O(|E| + |V|log|V|)$ [16].
More contextualized with this paper is the work of Benjamin Zhan. In his work he conducts a comparative study between graph searching methods to establish which would be more beneficial to be used in a large road network scenario. Zhan begins to compare the computation time of Dijkstra's and the A* algorithm and then introduces the concept of "radius" which is based in a heuristic and pre-processes the graph enabling to determine the relevance of certain paths (roads), thus discarding unneeded search paths, reducing the computation time. This concept brings a certain level of inaccuracy, which increases with the number of vertices, but was found an acceptable trade-off [35].

2.3 Evolutionary Routing

Ever since the dawn of mankind people looked at nature's intricate mechanisms mimicking them to improve quality of life and assure survival. We did so since we first started reproducing plants to sustain ourselves, calling it agriculture; building our own artificial caves, which now goes by the named of civil engineering and we still do it by imitating how burrs attach to fur and inventing Velcro. This trend as been science's uppermost, used and viable, source of inspiration and computer science has not been indifferent to it.
Evolutionary computation has been is a sub-field of artificial intelligence, since the sixties and is currently set to be used in combinatorial optimization problems which exhaustive search methods are not feasible nor fast enough [9]. Has weight setting is a combinatorial problem is np-hard [7], evolutionary computation is a evident choice to use as an approach.

2.3.1 Genetic

Genetic algorithms (GA) adapt the principles of natural selection to eugenically reach a desired outcome. The basic principle behind this algorithms is the same in an iterative method, a given a population and applied environmental pressure, only the fittest individuals will survive and pass their genes through the next generation. [12] In broad terms, randomly generated sets of input data are created and put through an algorithm. Then their outputs are compared to each other, with the use of a fitness function and the inputs with the best outcome will then "mate", having some of their data mixed, thus forming a new generation, repeating the whole process, until the maximized results are achieved [2] [12] [9].
For He, Li and Zhang found the need to used genetic algorithms as traffic routing is a non-linear np-complex problem, they used Linear Optimization alongside with and Hyper-graph context and optimized it with a genetic algorithm. In the latter a priority path based approach was used, with the each path's usage priority was a gene. Then they applied a random multi-position crossover and ran it through the fitness function bellow where let $L(j)$ be the path length of the chromosome j which is also calculated.

$$eval(k) = \frac{\frac{1}{L(k)}}{\sum_{j=1}^{size} \frac{1}{L(j)}}$$

Their ultimate goal was on improve public transportation trip times within a city and they found the new travel times to be rather satisfactory and the algorithm robust and convergent [19]. However their validation scenario was very limited, as they considered non-static parameters as static, such as traffic congestion; car-velocity; accident and other unpredictabilities and they did not consider real-world maps or realistic routes.
Another use for this approach is predicting traffic congestions as Tahilyani's research depicts. His work focuses on intelligent traffic control management exploring and comparing different approaches generated by genetic algorithms to ease congestion on intersections by controlling the traffic flow. He uses the traffic-light system status and timings as population for is algorithms and he measures the congestion in each of the individual to assess fitness [32]. Although it does not technically or directly chooses routes, this work can be applied to route making, if instead of controlling traffic flow is designed to avoid a given area or find an alternative route.

2.3.2 Memetic

Memetic algorithms in a broad sense are in most ways like their genetic counterparts, having they're memes (genes) can evolved during their life time. A memetic algorithms consists on the combinations of evolutionary and local-search methods and they are postulated to be faster and more accurate than genetic algorithms in a path finding context. [30] [5]. The search operator is the main focus on this approach, as it can swap a specific road segment for another if he finds it compensatory. This can improve the algorithm's processing time and making it more suitable to dynamic factors. In an attempt to solve the *Salesman Problem*, which objective is to find the cheapest way of visiting all the elements in a given set of cities, Botzheim employs what he calls a *Eugenic Bacterial Memetic algorithm* based on a GA first produced by Holland [20] [5] and then optimized by himself previously [4]. Bacterial Evolutionary Algorithm (BEA) are an adaptation of the standard GA with main differences residing in the mutation stage, where instead of adroitly swapping chromosomes specimen share chunks of their genes, and in the mutation which allows inter-generational changes in the specimen genome [5]. Focusing in the memetic portion of his experiment, he applies a 2-opt and 3-opt algorithm to his search method, which removes edges from the tour and reconnects them optimally. [5] The eugenic component, is also relevant as he states that previous work indicates that this is an effective way to solve the Salesman Problem [5]. In his work this component is located at the generation of the population where not only random itineraries are generated but some deterministic ones too, following the directive that the next node to visit is the closest or the second closest to their current position. Furthermore in the mutation phase randomly produces deterministic clones where some parts of the gene sequence are reversed.

2.3.3 Ant Colony

Ant Colony based algorithms imitate the biological process observable in ant scout and scavenging for food. Each ant leaves a trail of pheromone which attract other ants, leaving the path most traveled the most desirable. This works great for ants but in a urban context we want the exact opposite, therefore we consider a Inverted Ant Colony algorithm, in which the vehicles will avoid the less traveled roads leading to less congestion.
In their research Burrows, Reed, Templer and Walker [26] explore the benefits and disadvantages of this kind of approach by testing Ant Colony Optimization (ACO) under different situation both real and academic. They find that although the algorithm's remarkable adaptability to new situation, to prevent a large computation time they have to lay down a set of *soft* constraints. They attest that ACO is fully implementable in real world situations and that provides results in a fraction of the times other methods would .
José Capela's research he compares the ACO to other algorithms through real and academic maps (lattice and radial ring), shows that for a elevated number of car ACO can vastly improve circulation alongside with fuel consumption and CO_2 emissions. [10] He also gives the agents a since of personality which translates in the willingness to allow merges or the likeliness to attempt to over take another agent. In he most realistic scenario he uses a real map with actual route usage information, conferring a real world validity to his results.

2.4 Route Prediction Approaches

In this section we will explain the most commonly used route prediction approaches.

2.4.1 Adaptive Route Prediction

Considering that most urban vehicles within a city have a daily/weekly routine and their behavior has been found to be predictable with a good level of accuracy [34]. In his study Xue attempts to model vehicle patterns tracing the path of 4000 taxis cars in Shangai over the course of ten months. They tried to obtain what they call a Vehicle Mobility Pattern (VMP), using a

Variable-Order Markov (VOP) from the data they received from the traces. One of their initial findings was that although the human choice of route was an extremely complex problem that depended on a great number of variables, such as individual habits, number of lanes, number of ways, etcetera, they could still spot patterns on the travels. This algorithm chain has been vastly used in other location prediction problems [34] .The clear downside of this algorithm is that it generates a large number of patterns, because each variable depends on other random variables. To avoid this pruning can be used to reduced the processing cost.

2.4.2 Inverted Shortest Path Problem

Thus far we established that although graph theory has a great application in road networks, there are some situations that are not applicable in the real world. Graphs have an underlying assumption that we possess all the correct or accurate data concerning the real world structures which is not true and may induce errors. [8] Furthermore it may not actually interest us to know the shortest path between two vertex but to relate our comprehension of the road network to the driver's perception. In a practical sense, due to for instance morphology of the terrain, heavy traffic affluence, distance is not a singular factor when predicting vehicular routes among a road network. People, even the most experienced driver's and navigator familiar with the road network, tend to follow other paths.
The need to model route choices is paramount if we aim to explore the routes' characteristics perceptions, in order to forecast travels, anticipate future traffic conditions (congestions, accidents, etc.) and to understand how drivers react to change and adapt to new data [27]. The road network itself benefits from this pattern mining as obtained information though driver modeling can be used to traffic assignment and coordination systems [27].
Essentially the Inverse Shortest Path Problem in this case, consists in processing a weightless graph and finding its weights that are as close as possible to the previously obtain shortest path data and compliant with the observations of the paths more used in the network [8] [7]. More specifically in this thesis context, we can adjust the real road lengths to the driver perception allowing us to build a new "road map" and which results on a enhanced view of individual and possibly group preferences.
In his thesis Didier Burton starts to shorten the scope of graphs he works, choosing only finite, sparse, non-negatively weighted, loop-less graphs which is the closest approximation to the reality of road networks [7]. He proceeds to successfully use a specialized version of Goldfarb and Idani's method in

quadratic programming, introducing the concept of an island, which is created when a explicit constraint to the shortest path problem is reached. In addition they modify the algorithm further in order to test the correlation between observed data and expected data. [7]

2.4.3 Open Shortest Path First

This link-state routing protocol is most widely used in large enterprise computer networks. It uses shortest paths for routing the packets applying a equal cost multipath principle. The packets arriving at a given node are "shipped" to the next shortest path node, regardless of origin. Each node has a reference to packets' shortest path to destination.
There is an obvious parallel between computer and road networks. Therefore it is important to take in account on which methods are being used to cope with computer network traffic congestion.
In OSPF when a weight change occurs it has to be broadcast through the newtwork which may lead to flooding and should be avoided. [15] In our case, either to decrease computation resources or as a method of map stabilization, such changes should occur separated by medium intervals. Kennington addresses the weight setting problem by trying to divide equally the amount of flow on each arc. By limiting the number of prospective paths and selecting a pattern of flow he assigns weights, using linear programming, to the arcs subject to each arcs capacity. To cope with the fact that in this context linear programming proves to use too much computation time, he tries to restrict by a set of static rules the paths of the possible solution, trading off accuracy for fastness, within reasonable boundaries. [22] This optimization approach based on rule setting may be useful for the fitness function later on.
Using a cutting-edge Mixed Integer Optimization Solver (MILPS) called CPLEX accompanied and, due to the difficult nature of the wight setting problem, by developing a set of inequalities and incorporating a branch-and-cut algorithm, Parma was able to some problem in an as optimal as possible way. [1] In our work's scope, the key-part of his study is the branch-and-cut. this algorithm consists in on arrival to step where the MILPS is a few nodes away to the destination a search heuristic is utilized and if it is not the best match yest is cut off from search. This can be viewed has a kind of pruning used mainly in tree searches.
Very much alike the idea behind this thesis, in Mulyana's research a Heurisct/-Gentic Algorithm (HGA) is used. The population is composed of individuals which on their turn are comprised by chromosomes and each chromosome is

completed by net's complete set of wights and corresponds to a certain load distribution computed by Dijkstra's algorithm. At the begining of each generation some vectors of high quality are selected to produce better solutions. The heuristic search is also used to check the best chromosome of a population and to interfere in the mating process as it mutates chromosomes always guiding them towards the fitness functions. [15] This might shorten the number of generations needed, therefore improving computation time, but it may also lead to a local optimal state rather then a absolute one.

To avoid the local optimal solutions, Resende in his research on the OSPF weight setting problem, introduced a partitioned mating method. The population would be divided by 3 categories according to their fitness and in each iteration one individual form the fittest partition would mate either with the lower or the lowest fitness partition. Parent selection was also altered as the same specimen could reproduce multiple times in a single generation, leading to better offspring, given that the ones in the fittest partition always reproduce. Moreover when dealing with the crossover he lessened the probability of mutation of the genes which came from the elite parents. [29]

3 OSPF Genetic Approach

Although this is not the main focus of the thesis there is almost no data in an urban context to relate our future results to, so in order to measure how our method compares to others we chose to firstly use it in a more explored context, which has relevant data available.
Both problems refer to structures which can be represented by graphs and the main goal is to redistribute the weights of the graph in order to improve efficiency. The main difference is that the OSPF implies a turn based redistribution and the inverted shortest path does not. So it is safe to assume that this problem relates in many aspects to the one in hands.
In this chapter we begin by explaining how we are going to use a genetic approach on the OSPF, followed by setting the basic parameters in our genetic algorithm, then we explain the genetic algorithm (GA) and finally we present and discuss the tests results, comparing it to other author's previous work.

3.1 The ECJ Framework

In order to test evolutionary scenarios a JAVA Framework called Evolutionary Computation in Java (ECJ) was used. This framework was designed applying design patterns to ensure its re-usability and adaptability to different scenarios and with little programming effort.
The above mention adaptability along with the advisor's advice and its large documentation collection, were the main reasons supporting the usage of this framework [25].

3.2 How does it work

The top-level object is Evolve and has the sole purpose of initializing a subclass of EvolutionState. This subclass includes the number of generations to run and/or the ending condition that maybe set in case we need to stop it when we reach a ideal individual.

It also comprises other top-level objects which include:

- The Initializer class responsible for creating the initial population;
- The Population class which stores an array of sub-population classes that in its turn stores an array of individuals. We will only be using a population with a single sub-population;
- A Evaluator class where our fitness function will be defined;
- A Breeder class which is responsible for the breeding stage;
- An Exchanger, who trading individuals between sub-populations (which we wont be using);
- The Finisher which cleans up when the system is about to quit, which wont be used since we will be running it through a batch file and wont require cleaning up;
- And the Statistics component which gathers produced data and returns the statistics from tests.

Further information on each component may be found in the ECJ's documentation on its website [25].

3.3 Using the ECJ

This following set-up was used as a basis to produce the first results displayed in Figure 4.1 and every test after was a result of tempering with some of this parameters.
To start setting up our Genetic Algorithm we must configure the Population Object.

Listing 3.1: Population Parameter Configuration

```
pop.subpops = 1
pop.subpop.0.size = 300
pop.subpop.0.duplicate-retries = 0
pop.subpop.0.species = ec.vector.VectorSpecies
pop.subpop.0.species.genome-size = 100
pop.subpop.0.species.mutation-prob = 0.05
```

Listing 3.1 reflects in setting a single sub-population comprised by 300 unique individuals with a genome size of 100, with a probability of mutation of 0.05. The ECJ has a breeding mechanism is designated by “breeding pipeline”, which is not an accurate description as it is represented by a tree. The leaves of the tree are selection methods, which purpose is selecting individuals from the current generation. Non-leaf nodes then are breeding operators, which use

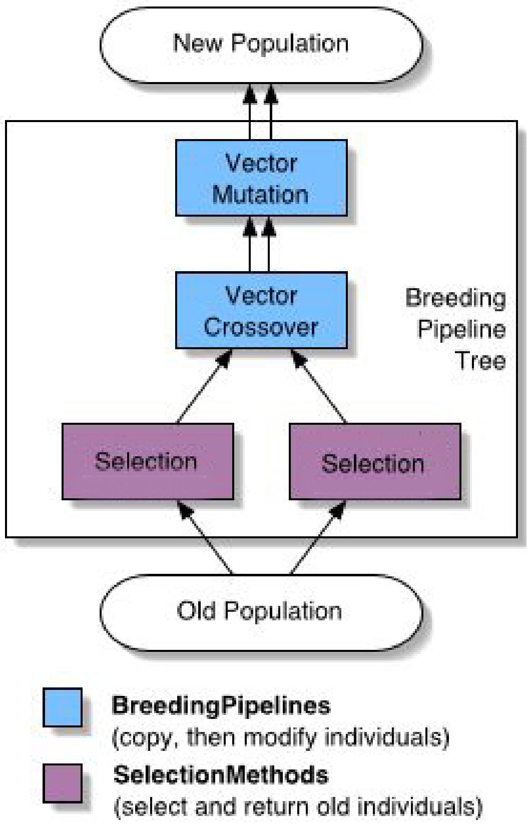


Figure 3.1: ECJ’s breeding pipeline-tree

individuals/specimens from the child-nodes (originated from the leaves) and breed them, sending them to a parent node. The root of the tree then returns the fully bred specimen to be inserted in the new population, as displayed in Figure 3.1. Breeding parameters may be establish withing the Breeding sub.class.

Listing 3.2: Breeding Parameter Configuration

```
pop.subpop.0.species.pipe =
ec.vector.breed.VectorMutationPipeline

pop.subpop.0.species.pipe.source.0 =
ec.vector.breed.VectorCrossoverPipeline

pop.subpop.0.species.pipe.source.0.source.0 =
ec.select.TournamentSelection

pop.subpop.0.species.pipe.source.0.source.1 =
ec.select.TournamentSelection
```

The parameter configuration displayed in Listing 3.2, stipulates that the mutation and cross-over will be handled by the mutation and cross-over operating pipelines accordingly. The last two lines of code stipulate that the selection will be handled by the tournament selection pipeline. We can establish the size of the tournament if we wish to do so by setting it in the breeding class also:

```
pop.subpop.0.species.pipe.source.0.source.0.size = 3
pop.subpop.0.species.pipe.source.0.source.1.size = 3
```

We can set two different tournament sizes for each selection operator, but in this study we will use the same
Although we can run the statistics class anywhere on the algorithm, we chose only to run it after the program has ended and have it produce the best individual's fitness and the execution time, so we can draw conclusions from it.

3.4 Methodology

Having explained how the parameters for the GA are in ECJ we will put them in the OSPF context, explain how the tests were executed and draw conclusions from their analysis.

3.4.1 Genetic Representation

In order to approach the OSPF problem within the scope of this study we must establish a connection between the genetic representation in the ECJ and the OSPF pardigm. In the genetic algorithm we consider a population comprised by specimens which are on its turn comprised by genes. So on the OSPF in a Gentic Algorithm scenario an individual is the network graph and its genome contains an array of weights which are the connection from an node to another.

3.4.2 Fitness Function

In order to pass the best individuals to the next generation a way of accessing if the individual's fitness is required. In order to achieve this we created a Target Graph with random connections between them. Our goal will be to achieve on a similar graph using a Genetic Algorithm. Between generations the specimen will be compared to the Target Graph and if they are similar enough they will pass to the next generation. This method is illustrated in the Listing 3.3.

Listing 3.3: Fitness Function Pseudo-code

```
for (i=0; i<numberOfGenes; i++)
  weight_difference = abs(specimen_weight[i] - target_weight[i])

  if(weight_difference < 2*target_weight)
    difference_factor_sum +=(weight_difference * 100)/
        target_weight
  else
    difference_factor_sum +=100

difference_factor_sum = weight_sum/numberOfGenes

similarity_factor = 100 - difference_factor_sum
```

The function begins by checking if the difference between the specimen and the target's weight is over two times the latter. If it is not it calculates a percent value of the disparity between the specimen and the target, through a simple cross-multiplication. If the difference is greater than two times the target's weight it considers that the disparity between the specimen's gene and the target is 100%. Then it calculates an avarage of the disparity factor among the genes of the current specimen and it subtracts it to 100 converting it into a parity factor which is used as a fitness value for that specimen.

3.4.3 Algorithm

To simulate a load balancing scenario, in an initial phase we started out by randomly generating a graph composed of 100 connections and its weights, which will be further addressed as the Target Graph, representing a network in its perfect load balancing state. We then proceeded to use the Initializer class to generate a population of graphs with the same number of weights as the Target Graph to be used as our first generation. This initial generation was devised randomly using the Initializer class and restrict it's random function, so that all genes would have a value between 0 and three times the value of the corresponding gene on the target graph. These restrictions were set in place because in this context, weights are positive and if the weight is over 2 times the target value it is considered to be of a 0 fitness by our fitness function, so there will be no interest in creating greater disparities.
Now that stage is set, we can initiate the cyclic part of the algorithm. Each iteration of this part consists in:

- Selection
- Mutation
- Recombination
- New generation

3.5 Retrieved Data

Firstly we need to determine what are the best parameters to be used in our genetic algorithm so we can achieve the best results possible.
Each of the following plots represent the data obtained from 30 different tests, all of which underwent the same parameters except the one in focus, but with different first generations and Target Graphs, in order to produce coherent results. The fitness value displayed on the following figures in a percent factor of similarity to the Target Graph, explained on the fitness function section above.

3.5.1 Population Size and Generation Number

As can be observed in the Figure 3.2 the fitness is proportional to the size of the population but after a size of 300 it increases slowly.

Through the analysis of this figure it can be ascertained that 300 specimen was the most adequate size of population considering that testing time greatly

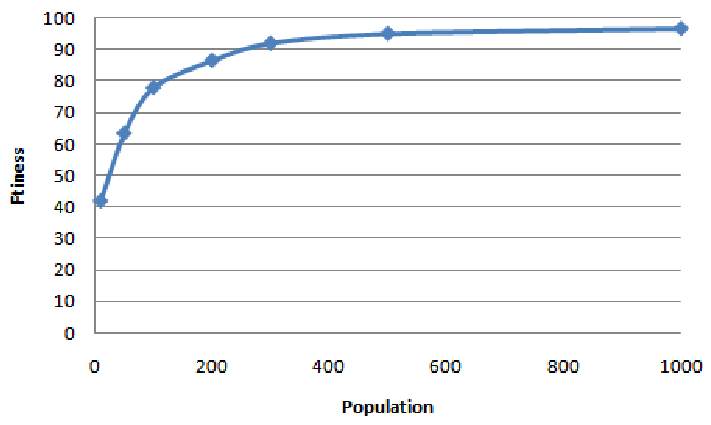


Figure 3.2: Average fitness of the best individual on the 1000th generation with the increase of the size of the population

increases after this number. It is also observable that with this population size we were able to have results fairly close to the desired balanced original graph.

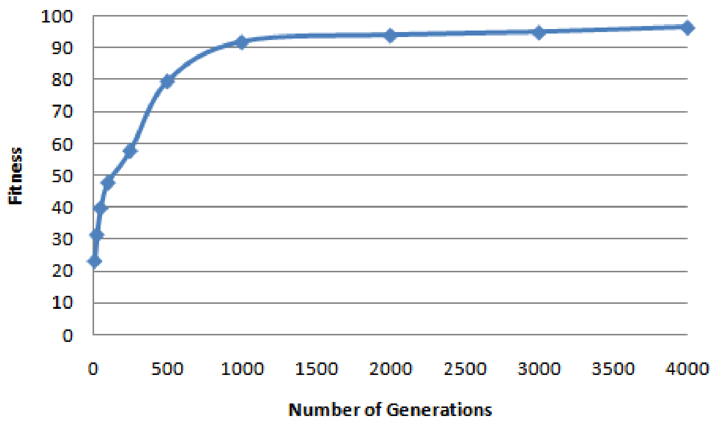


Figure 3.3: Average fitness of the best individual with the increase of the number of generations

In a similar way we determined that best number of generations considering fitness and test running time. As it can be observed in Figure 3.3, when we reach 1000 generations we have achieved a good fitness value and after that point the improvements are not significant enough to compensate for the extra testing tine.

3.5.2 Tournament Size

This parameter fits in the Selection phase of the algorithm. This type of selection, places a individual against an defined number of individuals and the fittest among them is chosen to proceed to the next generation.

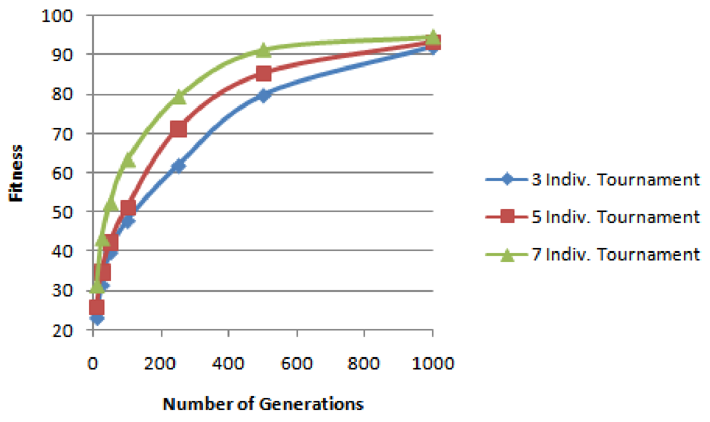


Figure 3.4: Average fitness of the best individuals with for different tournament sizes

Has we can see in Figure 3.4 the greater the tournament size the better fitness is seen since the first generations, which led us to choose a tournament size of 7 individuals.

3.5.3 Elitism

The Elitism factor is best describe by choosing the top fitness percentage of individuals and having them directly inserted in the next generation without recombine their genome with others.
We can predict that the higher the elitism less improvements we will observe from a generation to the next, as there are more repeating individual and thus less newer ones.
A high percentage of elitism will translate in less changes from on generation to the next, which will mean that from a certain percentage of elitism onward, for the same number of generations the fitness will decrease as the elitism increases. With a quick observation of Figure 3.5 we can obverse that, as we predicted when a high percentage of elitism is enforced the fitness value decreases. It can also be observe that it reaches it's best value when the elitism percentage is 25%

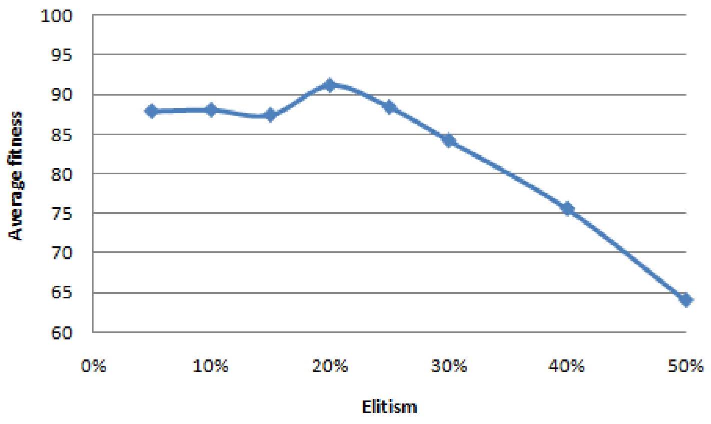


Figure 3.5: Average fitness of the best individuals with the variation with the percentage of Elitism

3.5.4 Mutation

In this section we consider how likely it is for a gene be randomly altered. In the ECJ this randomness factor us applied to each gene, meaning that different individuals will possibly have distinct amounts of genes mutated.

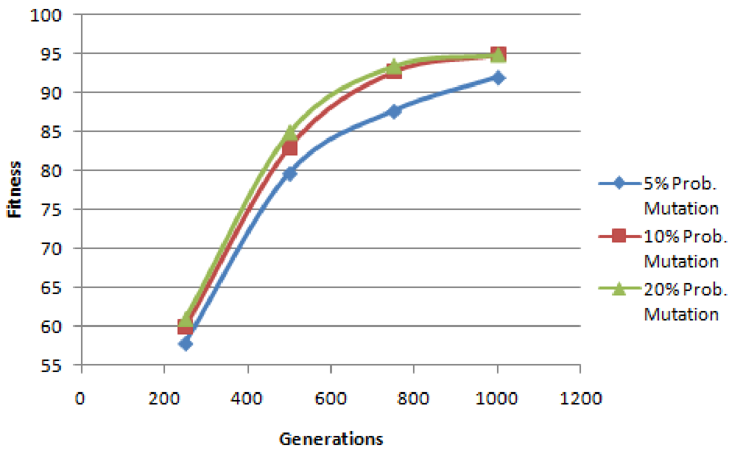


Figure 3.6: Average fitness of the best individuals with the variation of number of genes mutated and probability of mutation

As Figure 3.6 portrays we can achieve a better fitness than before, with a higher the probability of mutation. Mutation factors are commonly introduced to avoid the production of local extrema in a Genetic Algorithm [21].

3.5.5 Recombination

Recombination is the process where two selected individuals from a population have their genome combined generating a new individual which will continue on to the next generation.
We will study the advantages of using a 1-point and a 2-point crossover method. In a 1-point situation there is 1 cutting point on the genome of the parents resulting in two halves, then the genome is put back together producing the individual that will be inserted in the next generation.

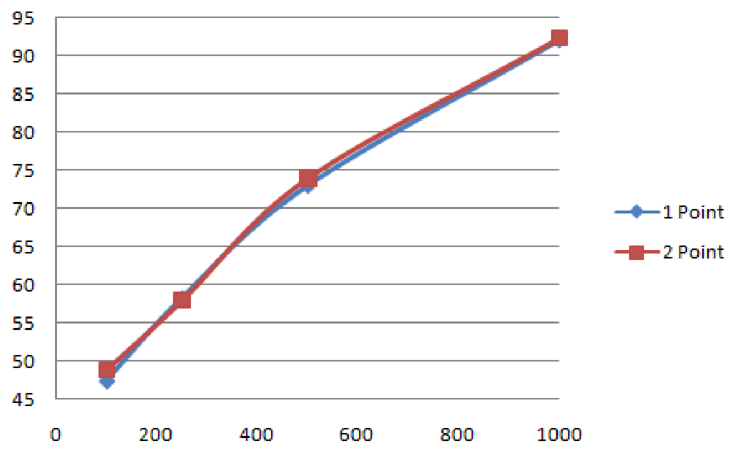


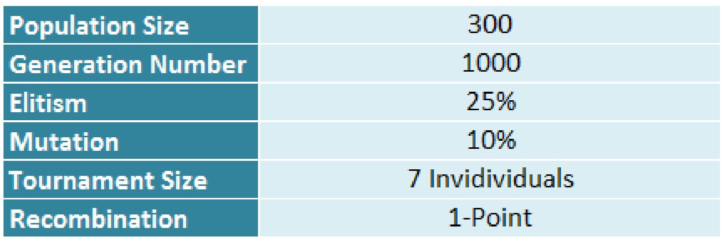
Figure 3.7: Average fitness of the best individual of each generation using a 1-point and a 2-point crossover

As we can observe in Figure 3.7 it seems there is little impact on the crossover amount of cutting points. We will use the 1-point default crossover, as it is the default recombination method in the ECJ.

3.5.6 Analysis

Every test was performed without changing any other variable than the one explicitly expressed on that subsection. There are endless other tests that could be performed and we could even cross-test parameters to see how each would influence the others, but such study wasn't performed has it is too extensive to be done in the required time of this thesis.
To better expose the of the best values obtained in every test we produced a table which holds all parameters which originated the best fitness. All of them surpassed the original fitness values.

As it is displayed in Figure 3.8 we managed to improve the fitness through



Population Size	300
Generation Number	1000
Elitism	25%
Mutation	10%
Tournament Size	7 Invididuals
Recombination	1-Point

Figure 3.8: The parameters that produced the best results achieved in every trial

the several by testing individual parameters of the genetic algorithm. We manage to improve our fitness from 92.01 to 94.37%. Though we only achieved a slight optimization it will translate to be significant when we extend the size of the network in an urban context later on.

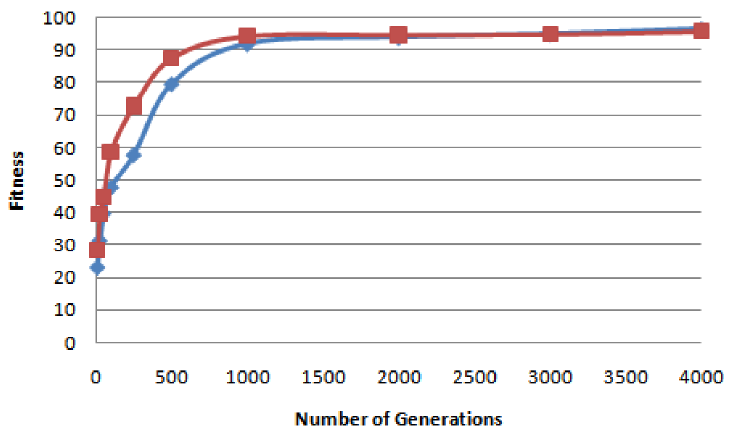


Figure 3.9: Average fitness of the best individual with the increase of the number of generations before and after optimizing parameter

To enhance the understanding of the extent of the optimization performed we display in Figure 3.9 a comparison between the fitness values achieve before and after the optimization.

3.6 Validation

So far we have explained and optimized our algorithm's approach, order to validate it we must compare its results to others produced by credited sources.

Most of the available data, in the OSPF context is produced solely under computer network scenarios which elude the focus of this dissertation. Thus instead of interfacing our algorithm with network simulator we proceeded to change it in order to mimic the conditions met in a computer network.
The OSPF is mainly used to balance the loads on each connection in order to improve traffic flow. This means that it is used in a turn-based paradigm, so that whenever the traffic flow changes or the algorithm sets new route priorities to accommodate the traffic variance.
To parallelize our algorithm with this paradigm we must change the way it works. Thus firstly we let the algorithm adjust himself to the evaluation function in the same amount of generations has before, at this stage it is as if our "network" has stable traffic flow. Then we periodically introduce slight changes the Target Graph in the evaluation function between generations, to simulate the modifications in the traffic flow. The rate that we change the graph it's similar to the way the OSPF as to adjust to an change in traffic flow in a computer network context. If there is a rapid variation of the traffic flow, then is little time to allow the algorithm to set new and better route priorities to the network, so if we shorten the number of generations between our changes to the Target Graph of the evaluation we will mimic this process. Having said that it is important to know that it is not the objective of this work to improve algorithms or data on this paradigm. We are only aiming to set a parallel so that we can establish that this algorithm behaves in the similarly to those applied to the OSPF.

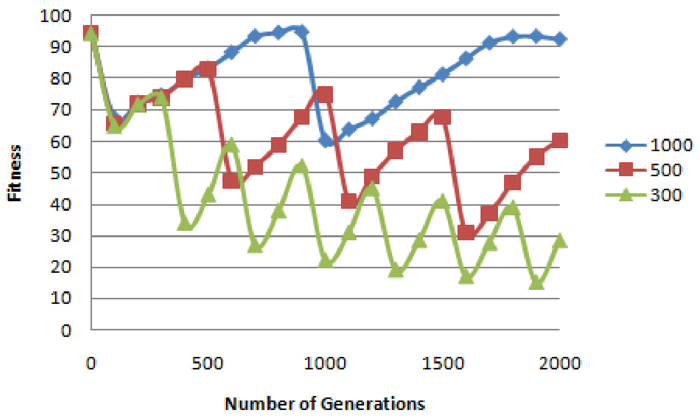


Figure 3.10: Evolution of the average fitness of the best individual with the passing of generations with different fitness goal changing intervals

In Figure 3.10 we can observe the different behaviors of the fitness depending on how many generations we give the algorithm to adjust. As it was predictable less generations you it to adapt to the new desired output (established within the evaluation function) progressively worst there will be the outcome. This observation agrees with the findingss of both Eueung Mulyana [15] and Luciana Buriol [6].
Mulyana's research which uses a Hybrid Gentic approach (as reference before) of the OPSF, portrays this exact behavior, demonstrating that when traffic congestion rapidly decreases the ability of their algorithm adjust to the new conditions as observable in Figure 3.10
In this way we were able to validate our algorithm, establishing similarities with the OSPF, so we can use it later on an attempt to use the Inverted Shortest Path in a urban context, which is the main scope of this thesis.

4
Traffic Simulation

In this chapter we will describe the tool used to simulate traffic and how we modeled the behavior pattern of the users to simulate their preferences. We will follow by describing the tests performed and finish off by making predictions of what results are expected to be observed.
In this way we will put to test our premise, that evolutionary computation may be used to provide more individually custom made routes, watch the impact on the user and the whole network system.

4.1 SUMO - Similation of Urban MObility

SUMO is a Open-Source traffic microscopic simulation package designed to reproduce vehicle movements withing a road network. It was implemented in C++ on Linux and Windows.
It is by a core module which collects the information about the routes, agents and network using XML files and then performs a discrete simulation. Other modules such as NETCONVERT, which plan and format the routes which can be inputed in the core module later and the TRaCi module which can be used to control a running simulation using a TCP connection to interface with it.
It comprehends multi-modal simulation modeling not only cars are modeled but also public transportations such as buses and trains. The most basic unit in the simulation is considered to be an individual rather than a vehicle. This individual is defined by its departure time and the route it takes which comprises several sub routes, which describe a modality.
Despite the atomic unit of the simulator is the human being it does not consider the pedestrian traffic, which might make it not the most realistic of its are, but it is safe to assume that it would greatly increase the already significant large average test time.

4.1.1 SUMO's Usage Example

To have a better and detailed understanding of how SUMO was used on this project we will present a description of its usage.

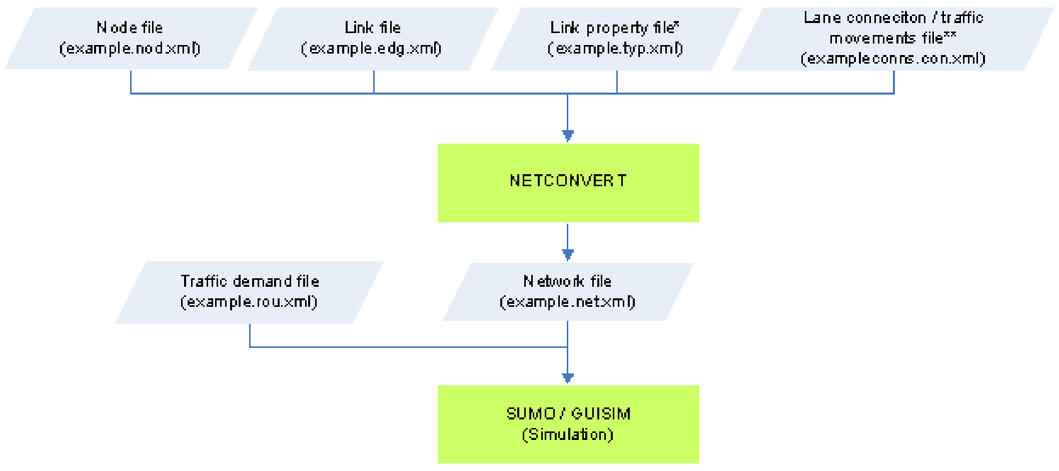


Figure 4.1: The required input files for running a simulation in SUMO[31]

Figure 4.1 portrays how SUMO handle it's input data. Before the data reaches the SUMO's core it must be processed by the NETCONVERT tool. This tools is what allows us to import road networks, from different types, in our case OpenStreetMap, formating them to fit the input system od the SUMO simulator.
Although there are four XML files on the input of the NETCONVERT the last two: Link Propriety File and Lane Connection File are optional, being used the alter default values, such as lane connections and traffic movements. The traffic demand file contains the agents that will participate in the simulation, their personal characteristics and routing information. Additionally configuration specifications can be provided in a file or otherwise in the command line which can comprehend the output style, starting and finishing times, among other parameters.
In this simulator the network is comprised by one way arcs with their own set of features such as number of lanes, speed limit and the nodes which they connect to and from. The nodes must have coordinates and may have different types which reflects on how traffic will behave on arriving at that node, if it will yield to the left; stop; wait for the green light or if it is unregulated, the latter being the default type. Moreover there can additionally be created restrictions to certain connections between lanes or routes.

Vehicles are defined on the traffic demand file and possess their own characteristics, such as departing time; maximum and minimum acceleration; driving skill and vehicle length.

4.1.2 How to run a simple SUMO simulation

To run a simulation we firstly need to generate three XML files containing the data which will be used in the simulation. To convert the network data file so it can be interpreted by the SUMO core we must pass it through the NETCONEVRT module, we do this in the following way:

```
netconvert --xml-node-files=lattice.nod.xml--xml-
edge-files=lattice.edg.xml--output-file=lattice.net.xml

netconvert --osm-files= lattice.nod.xml
--output-file= lattice.net.xml
```

While the first converts node and edge information the second one allows the conversion from OpenStreetMap files.
Next we need to generate the trips, which tell SUMO the start and end point of each journey made by the vehicle during the simulation. As in our work the we will be tempering with the routes and not the trips we will use a random trip generator provide by SUMO called randomTrips.py. To generate these random trips we simply type this command in the command line:

```
Python randomTrips2.py -n lattice/lattice.net.xml
-r lattice/lattice.rou.xml -e 100 -p 1 -B 1
```

In this particular case we are generating a set of trips which are comprised by 100 agents(-B), every simulation step (-p), until the 100th step (-e).
As mention before we can produce a configuration file for the simulation. That configure file must follow the XML syntax displayed and can have many customizable options. In Listings 4.1 is a simple example of a configuration file which expresses our input, output and the port where the TRaCi server will be ready for us to connect and follow the simulation as it unfolds.

Listing 4.1: XML Configuration File

```xml
<configuration>
  <input>
    <net-file value="lattice.net.xml"/>
    <route-files value="lattice.rou.xml"/>
  </input>
  <output>
    <tripinfo value="lattice.out.tripinfo.xml"/>
  </output>
  <traci_server>
    <remote-port value="8814"/>
  </traci_server>
</configuration>
```

After readying all the files the simulation can begin using a command such as this:

```
python cosmoController.py
../simulations/lattice/lattice.sumo.cfg
```

The “.sumo.cfg” extension defines the XML file which we have beforehand exemplified, which connects all the generated files and tell SUMO which they are.
For more information regarding simulation running and generating the necessary files to do so is explained with further detail on SUMO’s website and its wiki [31].

4.2 Connecting our algorithm with sumo

Now that we establish how we can run a simulation we will explain how we will link it with our routing algorithm. We will also clarify how will our algorithm behave and set up a basis of comparison to our findings. To actively affect a running simulation we must access it through the TRaCi module. To interface with the SUMO using the TRAcI took advantage of an implementation performed by José Capela’s[10], also in the COSMO project, which was programed in python and modified it to serve our purpose. Two routing algorithms were tested.
The first algorithm simply finds the shortest path between origin and destination, which we will use later on the establish a comparison.
The second one we implemented our GA, explained in the last chapter. The Initializer on our GA is prepared by setting the size of the genome corresponding to the number of arcs/edges in each city. Then we randomly generates

the values of each gene wich in the simulator correspond to distance. Each individual will then be outputted into a file which will then be parsed by the interface we altered which will process this data and then send it to sumo so that a next simulation that is executed produces the average trip time which will be used in the Evaluator class which will rate the individuals according to their trip time.

5
Tests and Data Analysis

In this chapter we will explained the reasons which led us to pursuit this test course, detailing its execution and data recovered. Then we will analyze the data produced by the test of best individual generated by our G.A. and discuss its impact of the algorithm in the vehicular mobility in an urban context.

5.1 Test Planning

As there is a randomness factor within the algorithm (in the mutation stage) all tests were performed only 10 times with the same parameters to ensured a relatively small standard deviation and coherency among the results and guarantee that testing time was within reasonable limits.

5.1.1 Non-real scenarios

It is important to see how the algorithm measures up in diverse conditions and so we experimented with standard road simulation scenarios and then real data.

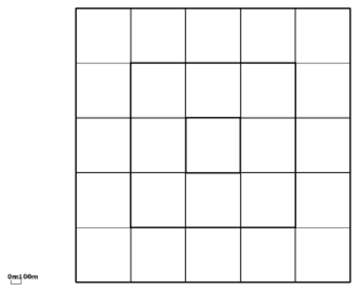


Figure 5.1: Lattice Network

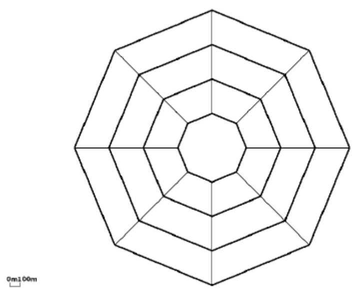


Figure 5.2: Radial and Ring Network

The Lattice network which can be observed in Figure 5.1 corresponds to a stereotypical typology of road networks called “grid”. This type of city plan can be observe since the Romans which used it extensively to plan their cities and can now be observed in almost every capital city in the United States [18]. The Radial and Ring road typology in portrayed in Figure 5.2 is more commonly seen in old European cities, as the city grew larger a broader road was built around it to improve traffic flow, which would be later encapsulated by the city it self [3].
We used both this stereotypical scenarios as a way of testing our hypothesis in less realistic scenario absent real ones.

5.1.2 Real-Scenarios

To try to make our results as close to those that would be obtained in real life testing we representation from OpenStreetMap of the city of Coimbra shown in Figure 5.3 and imported it as explained before to the simulator.

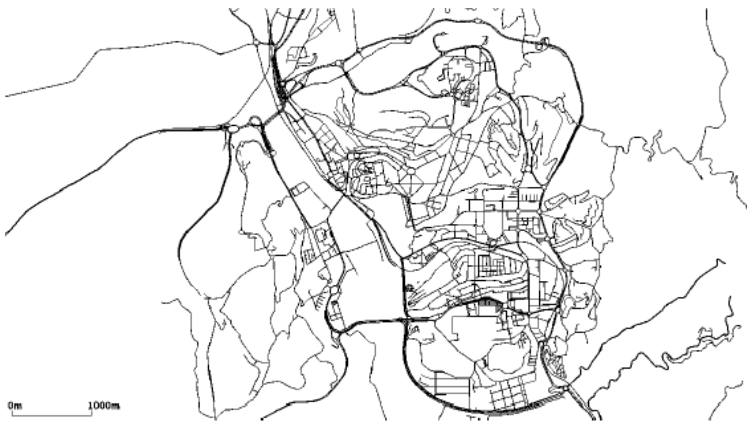


Figure 5.3: Radial and Ring Network

In figure 5.3 we can observe the generated network in SUMO’s GUI which is accessed by the TRaCi. The main challenge here was to work around the large testing times and bad network representations which the NETCONVERTER mis-formated, to do so we elminated them, in order to have a workable representation of the city of Coimbra without deadlocks or unreal junctions. We know that this infers less realism to the simulation but otherwise there would be no viable data. This was a very extenuating task but no other solution was available.

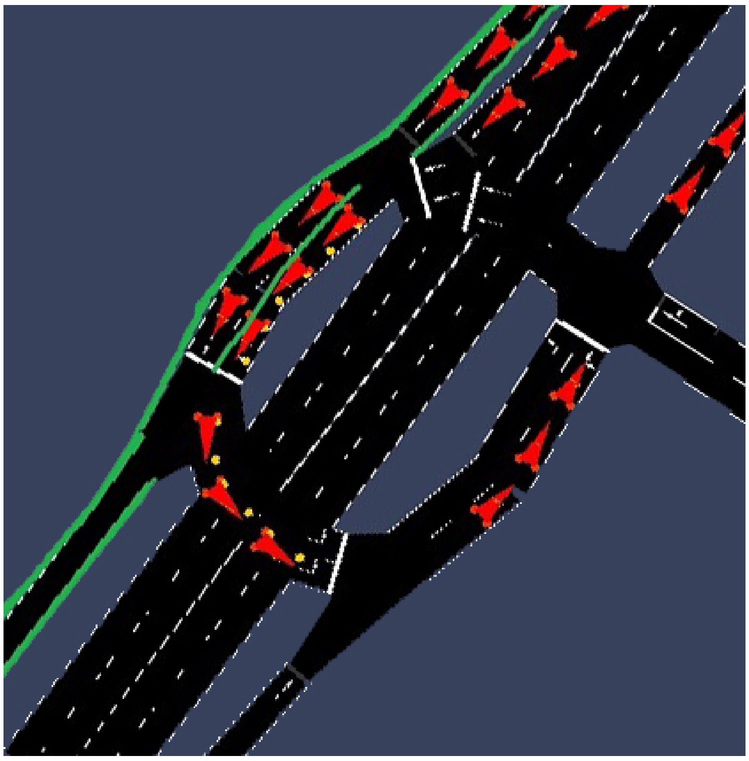
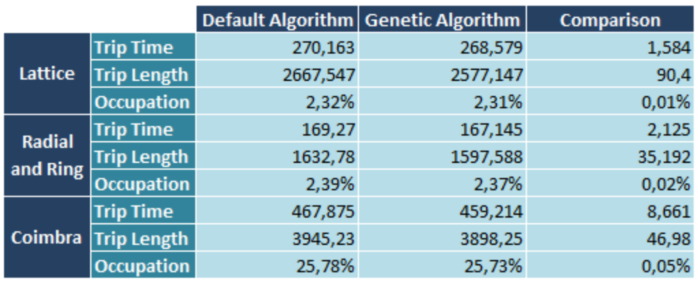


Figure 5.4: An example of a removed junction due to a deadlock.

In figure 5.4 we can see a example of these deadlocks which made the output data useless as it either did not the test finish as there are still trips to be concluded or sent trip times and road congestions values through the roof on the rare occasion the deadlock was inexplicably broken.

5.2 Analysis

In this section all the tests were only repeated 10 times, as the test's execution duration is overwhelming. Each test was set perform 1000 trips, which origin and destination were generated through the randomTrips.py.



		Default Algorithm	Genetic Algorithm	Comparison
Lattice	Trip Time	270,163	268,579	1,584
	Trip Length	2667,547	2577,147	90,4
	Occupation	2,32%	2,31%	0,01%
Radial and Ring	Trip Time	169,27	167,145	2,125
	Trip Length	1632,78	1597,588	35,192
	Occupation	2,39%	2,37%	0,02%
Coimbra	Trip Time	467,875	459,214	8,661
	Trip Length	3945,23	3898,25	46,98
	Occupation	25,78%	25,73%	0,05%

Figure 5.5: This figure illustrates the results obtained through the routing of the default algorithm and the routing produced by our own.

Figure 5.5 displays the results produced by the SUMO simulator, when taken into account the best individual of the last generation, produced by the G.A. we developed and explained earlier.

5.2.1 Lattice Scenario

As we can observe in Figure 5.5, there is a slight difference between both the average trip and length of the both the untempered algorithm and our GA. Although implementation of the GA produced a better occupancy average, but the maximum occupancy was of around 31% which is worse the default routing algorithm which only as 23%.

5.2.2 Radial and Ring Scenario

In the radial and ring scenario although improvement a little better than latter, concerning average trip length, trip time and occupancy, it still produces a worse maximum occupancy than the default algorithm.

5.2.3 Real Scenario

In this scenario we found the results somewhat better than in the previous ones. When we compare the values from the default algorithm and our own we are able that the G.A. was able to produce better results than the default algorithm, with a greater difference than in the previous scenarios. This might be due to the existence of more routing options. The maximum occupancy is 30% which is as bad as the default algorithm's but all studied variables: average trip time; average trip length and average road occupancy have in a small way, but further than on the other scenarios.

5.2.4 General Analysis

When set out to explore this approach we were expecting to make a significant impact on the average trip time and length and the road occupancy on the road network. Instead we were only able to improve it slightly. We consider that the results were not what we expected but we find that there is much that can be improved with enough time and tweaking our algorithm. Surprisingly the best improvements were observe in the real scenario which can also indicate that this approach works better on a more complex network than on a simple stereotypical academic one. We can also conclude that in the non-real scenarios although the algorithm produces slight general improvements it does so by creating a point during the simulation where traffic congestion is a necessary evil to later ease the traffic flow.

6 Conclusions

For the past decades attempts to cope with the increase of traffic flow within cities have not serve it right as road congestion is still a part of the dayly life in big cities. Route generator does not account for the human factor, as it only targets a driver to the theoretical shortest path, not having into to account the human factory of the equation.
Facing the road choosing process from the human perspective can be very thought as a multi-variable complex system of the Inverted Shorted Problem. Being NP-Hard, it's no viable to make a combinatorial exhaustive search, thereby to get around it we designed a evolutionary approach.
We aimed explore the hypotheses that and an evolutionary approach taken upon the urban mobility paradigm would offer solutions to improve traffic routing and thus decrease congestion rates within road networks.
Our main focus was to include in our Evolutionary approach the Inverted Shortest Path Problem, in a way that we would alter the perception of the road network. We did so by firstly addressing it in a computer network context and established similarities between our genetic algorithm and other approaches to the OSPF in a computer network context. Then using SUMO and an real-time interface designed in python we were able to put to test the Genetic Algorithm introducing changes in road perception and extracted data from it.
Although the algorithm did not measure up to the expectations, having not revealed an significant advantage in its usage, it is still important to mention that it can be later used for basis further research. Some examples of further work are:

- Implementation of a different fitness function that would rely on CO_2 emissions instead of average trip time or road congestion;
- Rerouting only a percentage of the population;

- Design a real-time implementation;
- Optimize the algorithm further by cross testing the tested values among themselves or adding new evolution parameters;
- Design a tag system to classify what roads would prioritized to decrease the traffic threshold and take it in to account on the fitness function.

As a final thought, this approach needs a large collection of vehicle information to be enforced in real life. This might be a problem as it raises many privacy, because of the sensitivy of the general public to release personal data. Although we know that privacy has been decreasingly being a subject of importance considering the amount of information people are willing to share on social networks and similar systems, there are always some people that wont give up their right to be private. This algorithm will only work if we can obtain the most accurate vehicular position and route. So we hope that one day privacy will still be preserved and data can be shared without causing any concerns.

Bibliography

[1] S. A. Amandeep Parmar and J. Sokol. An integer programming approach to the ospf weight setting problem. 2006.

[2] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming : An Introduction : On the Automatic Evolution of Computer Programs and Its Applications (The Morgan Kaufmann Series in Artificial Intelligence).* Morgan Kaufmann Publishers, Nov. 1997.

[3] R. Bartlett. *The Making of Europe. Conquest, Colonization and Cultural Change 950-1350.*

[4] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Fuzzy rule extraction by bacterial memetic algorithms. *Int. J. Intell. Syst.*, 24(3):312–339, Mar. 2009.

[5] J. Botzheim, P. Földesi, and L. T. Kóczy. Solution for fuzzy road transport traveling salesman problem using eugenic bacterial memetic algorithm. In J. P. Carvalho, D. Dubois, U. Kaymak, and J. M. da Costa Sousa, editors, *IFSA/EUSFLAT Conf.*, pages 1667–1672, 2009.

[6] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, Mikkel, and L. S. B. U. Campinas. A memetic algorithm for ospf routing, 2002.

[7] D. Burton. On the inverse shortest path problem, 1993.

[8] T. Cui and D. S. Hochbaum. Complexity of some inverse shortest path lengths problems. *Netw.*, 56(1):20–29, Aug. 2010.

[9] K. De Jong. Evolutionary computation: a unified approach. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2245–2258, New York, NY, USA, 2008. ACM.

[10] J. A. C. Dias. Using error to optimize the city, 2012.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[12] A. Eiben and J. Smith. *Introduction to Evolutionary Computing.* 2003.

[13] D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. *CoRR*, abs/0808.3694, 2008.

[14] J. Esser and M. Schreckenberg. Microscopic simulation of urban traffic based on cellular automata, 1997.

[15] U. K. Eueung Mulyana. A hybrid genetic algorithm approach for ospfweight setting problem.

[16] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[17] R. Gakenheimer. Urban mobility in the developing world. *Transportation Research Part A: Policy and Practice*, 33(7-8):671 – 689, 1999.

[18] M. Gelernter. *A history of American architecture: buildings in their cultural and technological context.*

[19] R. He, Y. Li, and X. Feng. Models and genetic algorithms for the optimal stochastic riding routes in urban public transportation and its applications. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 5, pages 431 –435, aug. 2007.

[20] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[21] M. A. . M. C. K. Tesch. Genetic algorithm search for stent design improvements.

[22] J. Kennington. An arc-path model for ospf weight setting problem, 2009.

[23] G. Kotushevski and K. A. Hawick. A review of traffic simulation software. Technical Report CSTN-095, Computer Science, Massey University, Albany, North Shore 102-904, Auckland, New Zealand, 2009.

[24] K. Kuratowski. Sur le Probleme des Courbes Gauches en Topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

[25] T. E. O. Manual. The ecj owner's manual, 2010.

[26] K. T. J. W. Philip Burrows, Kate Reed. Efficient traffic routing using acom, 2009.

[27] C. G. Prato. Route choice modeling: Past, present and future research directions. *Journal of Choice Modelling*, 2(1):65–100, 2009.

[28] N. T. Ratrout and S. M. Rahman. A comparative analysis of currently used microscopic and macroscopic traffic simulation software, 2008.

[29] M. G. C. Resende. A memetic algorithm for ospf routing. 2002.

[30] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, and C. Lucas. Memetic algorithm based path planning for a mobile robot. In A. Okatan, editor, *International Conference on Computational Intelligence, ICCI 2004, December 17-19, 2004, Istanbul, Turkey, Proceedings*, pages 56–59. International Computational Intelligence Society, 2004.

[31] SUMO. Import tutorials. `http://sumo-sim.org/wiki/Main_Page`, 2013. Last visited in 12 June 2013.

[32] S. Tahilyani. A new genetic algorithm based lane-by-pass approach for smooth traffic flow on road networks, 2012.

[33] R. J. Trudeau. *Introduction to Graph Theory.* Dover Publications, New York, 1993.

[34] G. Xue, Y. Luo, J. Yu, and M. Li. A novel vehicular location prediction based on mobility patterns for routing in urban vanet. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):222, 2012.

[35] F. B. Zhan. Three fastest shortest path algorithms on real road networks: Data structures and procedures, 1997.