

Bio-inspired Computing and Smart Mobility



Daniel H. Stolfi

Supervisor: Dr. Enrique Alba

Departamento Lenguajes y Ciencias de la Computación
University of Malaga

PhD Thesis Dissertation in Computer Science


E.T.S. Ingeniería Informática

September 2018



UNIVERSIDAD
DE MÁLAGA

AUTOR: Daniel Héctor Stolfi Rosso

 <http://orcid.org/0000-0002-1138-8130>

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga



Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:

<http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode>

Cualquier parte de esta obra se puede reproducir sin autorización
pero con el reconocimiento y atribución de los autores.

No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es



Carta de Aval



Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

El Dr. **Enrique Alba**, Catedrático de Universidad perteneciente al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,

certifica

que D. **Daniel Héctor Stolfi Rosso**, Ingeniero en Informática por la Universidad de Málaga, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada:

Bio-inspired Computing and Smart Mobility

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo. Y para que conste a efectos de lo establecido en la legislación vigente, autorizo la presentación de la Tesis Doctoral en la Universidad de Málaga.

En Málaga, Septiembre de 2018

Firmado: Dr. Enrique Alba Torres



UNIVERSIDAD
DE MÁLAGA

Acknowledgements

It has been six years since I finished my master's degree study and started this long, sometimes hard, sometimes joyful, path that has ended with this PhD thesis. The path is not over yet, wonderful things are coming, but I wouldn't be able to be writing these lines without the support I have received from my parents, Elena and Héctor, and from my whole family to be honest, since my childhood. Dealing with me was not easy, thank you for not saying no to all my crazy dreams, despite the Christmas fireflies.

During my secondary school my mind started to wander through the unknown roads of science. Thanks to my teachers for encouraging me to continue, lending me technical and scientific books and always believing in me. Also thanks to those who said no, because they provided me with an extra challenge to fight off.

Writing a PhD thesis is not an easy work. It requires lots of time to conduct many research studies, present them in conferences, writing journals articles and get them successfully published. That time has to be borrowed from somewhere and somebodies. I must then give a big thank to Isabel, patient lovely wife, and Leila and Alejandra, brave daughters, for understanding and taking care of me.

Since every PhD candidate needs an advisor without whom it is impossible to sail the seas of science, I would like to thank to Professor Enrique Alba, for guiding me during this journey, giving me hints, helping me to gain new knowledge, and enriching my research work. Consequently, I am grateful with the help and support provided by the NEO Research Group. All its members were there for answering my questions, so I cannot forget to say them: thank you guys.

Finally, I met lots of valuable people during my stays in the University of Shinshu (Japan) and in the University of Birmingham in (U.K). I am grateful to Professors Kiyoshi Tanaka, Hernán Aguirre, and Xin Yao for having me.

Financial Support:

This PhD thesis has been partially funded by the Spanish Ministry of Economy and Competitiveness (MINECO) and European Regional Development Fund FEDER, under contract TIN-2011-28194 (roadME <http://roadme.lcc.uma.es>), TIN2014-57341-R (moveON <http://moveon.lcc.uma.es>), TIN2016-81766-REDT (CI-RTI <http://cirti.es>), and TIN2017-88213-R (6city <http://6city.lcc.uma.es>). Daniel H. Stolfi was supported by an FPU grant (FPU13/00954) from the Spanish Ministry of Education, Culture and Sports.



UNIVERSIDAD
DE MÁLAGA

Table of Contents

| | |
|---|-------------|
| Acknowledgements | v |
| List of Abbreviations | xiii |
| 1 Introduction | 1 |
| 1.1 Motivation | 1 |
| 1.2 Objectives and Phases | 3 |
| 1.3 Thesis Contributions | 4 |
| 1.4 Thesis Organization | 5 |
| I Scientific and Technological Bases | 7 |
| 2 The Main Scientific Challenge: Smart Mobility Problems | 9 |
| 2.1 Introduction | 9 |
| 2.2 Long Travel Times | 11 |
| 2.3 Polluted Cities | 13 |
| 2.4 Finding an Available Car Park Spot | 16 |
| 3 Our Scientific Base: Bio-inspired Computing | 19 |
| 3.1 Introduction | 19 |
| 3.2 Metaheuristics | 19 |
| 3.2.1 Evolutionary Algorithms (EA) | 21 |
| 3.2.2 Simulated Annealing (SA) | 22 |
| 3.2.3 Ant Colony Optimization (ACO) | 23 |
| 3.3 Statistical Validation | 24 |
| 4 The Main Technological Base: Microsimulation | 27 |
| 4.1 Introduction | 27 |
| 4.2 Traffic Microsimulators | 28 |
| 4.2.1 TRANSIMS | 29 |
| 4.2.2 VISSIM | 30 |
| 4.2.3 MATSim | 30 |
| 4.2.4 SUMO | 31 |

| | | |
|-----------|---|-----------|
| 4.3 | SUMO: Simulation of Urban MObility | 32 |
| 4.3.1 | Main Characteristics of SUMO | 32 |
| 4.3.2 | Building Mobility Scenarios with SUMO | 35 |
| 5 | Facing Technology Gaps: Incomplete Maps and Data | 37 |
| 5.1 | Introduction | 37 |
| 5.2 | Flow Generator Algorithm (FGA) | 39 |
| 5.2.1 | Route Generator (RG) | 42 |
| 5.2.2 | Evolutionary Algorithm (EA) | 43 |
| 5.3 | Case Studies | 48 |
| 5.4 | Parameterization | 48 |
| 5.5 | Results | 51 |
| 5.5.1 | Setup Stage | 52 |
| 5.5.2 | Optimization Stage | 52 |
| 5.6 | Discussion | 54 |
| II | Modeling and Solving Problems | 55 |
| 6 | Red Swarm: Reducing Travel Times | 57 |
| 6.1 | Introduction | 57 |
| 6.2 | The Red Swarm Architecture (RS) | 58 |
| 6.2.1 | Evolutionary Algorithm (EA) | 59 |
| 6.2.2 | Rerouting Algorithm (RA) | 64 |
| 6.3 | Case Study | 65 |
| 6.4 | Competitor Techniques for our EA | 67 |
| 6.4.1 | Dijkstra (DJK) | 67 |
| 6.4.2 | Distance Vector (DV) | 68 |
| 6.4.3 | Ant Colony Optimization (ACO) | 68 |
| 6.5 | Parameterization | 69 |
| 6.5.1 | Parameterization of the ACO Algorithm | 70 |
| 6.5.2 | Parameterization of the EA | 70 |
| 6.6 | Experimental Analysis | 71 |
| 6.6.1 | Optimization | 71 |
| 6.6.2 | Parallel EA | 74 |
| 6.7 | Discussion | 77 |
| 7 | Green Swarm: Reducing Carbon Footprint | 79 |
| 7.1 | Introduction | 79 |
| 7.2 | The Green Swarm Architecture (GS) | 80 |
| 7.2.1 | Eco-friendly Route Algorithm (EfRA) | 82 |
| 7.2.2 | Green Algorithm (GrA) | 85 |
| 7.3 | Case Studies | 85 |
| 7.3.1 | Alameda (ALA) | 88 |

| | | |
|----------|--|------------|
| 7.3.2 | Malaga (MGA) | 88 |
| 7.3.3 | Stockholm (STO) | 89 |
| 7.3.4 | Berlin (BER) | 89 |
| 7.3.5 | Paris (PAR) | 89 |
| 7.4 | Competitor Techniques | 89 |
| 7.4.1 | Minus 20% (-20%) | 90 |
| 7.4.2 | Maximum 30 km/h (30km/h) | 90 |
| 7.4.3 | HDV-LDV | 90 |
| 7.5 | Experimentation | 90 |
| 7.5.1 | Metric Study | 91 |
| 7.5.2 | Optimization | 92 |
| 7.5.3 | Green Swarm Combined with Other Strategies | 94 |
| 7.5.4 | Study on Unseen Scenarios | 94 |
| 7.5.5 | Study of User Acceptance Rates | 97 |
| 7.5.6 | A Better Context for Understanding the Contributions of GS | 99 |
| 7.6 | Discussion | 100 |
| 8 | Yellow Swarm: Low-Cost Infrastructure for the City | 101 |
| 8.1 | Introduction | 101 |
| 8.2 | The Yellow Swarm Architecture | 102 |
| 8.2.1 | Evolutionary Algorithm (EA) | 102 |
| 8.2.2 | Panel Manager | 106 |
| 8.3 | Case Studies | 106 |
| 8.3.1 | Malaga and Madrid | 106 |
| 8.3.2 | Quito | 108 |
| 8.4 | Experiments and Results | 110 |
| 8.4.1 | Malaga and Madrid | 110 |
| 8.4.2 | Quito | 115 |
| 8.4.3 | Optimization Interval | 115 |
| 8.4.4 | Validation on 30 Unseen Scenarios | 118 |
| 8.5 | Discussion | 119 |
| 9 | Smarter Routes for GPS Navigators | 121 |
| 9.1 | Introduction | 121 |
| 9.2 | Dynamic User Equilibrium (DUE) | 122 |
| 9.2.1 | DUE.r & DUE.rp | 123 |
| 9.2.2 | DUE.ea | 124 |
| 9.3 | Case Study | 126 |
| 9.4 | Results | 127 |
| 9.4.1 | Optimization | 127 |
| 9.4.2 | Penetration Rate | 130 |
| 9.5 | Discussion | 130 |

| | |
|--|----------------|
| 10 Know Your City: Car Park Spots | 131 |
| 10.1 Introduction | 131 |
| 10.2 Car Park Occupancy Prediction | 132 |
| 10.2.1 Polynomial Fitting (P) | 133 |
| 10.2.2 Fourier Series (F) | 133 |
| 10.2.3 K-Means (KM) | 133 |
| 10.2.4 KM-Polynomial (KP) | 133 |
| 10.2.5 Shift & Phase (SP) | 133 |
| 10.2.6 Time Series (TS) | 134 |
| 10.3 Case Studies | 134 |
| 10.3.1 Birmingham | 134 |
| 10.3.2 Glasgow | 135 |
| 10.3.3 Norfolk | 136 |
| 10.3.4 Nottingham | 136 |
| 10.4 Training | 136 |
| 10.5 Testing | 138 |
| 10.6 Web Prototype | 140 |
| 10.7 Discussion | 142 |
| III New Intelligent Algorithms | 143 |
| 11 New Bio-inspired Algorithms | 145 |
| 11.1 Introduction | 145 |
| 11.2 Background | 146 |
| 11.3 Epigenetics From an EA Representation | 148 |
| 11.4 Epigenetic operators | 149 |
| 11.4.1 Genomic Imprinting | 150 |
| 11.4.2 Reprogramming | 151 |
| 11.4.3 Paramutation | 151 |
| 11.4.4 Position Effect | 151 |
| 11.4.5 X-Inactivation | 151 |
| 11.4.6 Bookmarking | 151 |
| 11.4.7 Gene Silencing | 152 |
| 11.5 The epiGenetic Algorithm (epiGA) | 152 |
| 11.5.1 Population Initialization | 153 |
| 11.5.2 Selection | 155 |
| 11.5.3 Nucleosome Generation (NG) | 155 |
| 11.5.4 Nucleosome Based Reproduction (NBR) | 155 |
| 11.5.5 Epigenetic Mechanisms (EM) | 156 |
| 11.5.6 Replacement | 159 |
| 11.6 Discussion | 159 |

| | |
|---|------------|
| 12 Solving Problems with epiGA | 161 |
| 12.1 Multidimensional Knapsack Problem (MKP) | 161 |
| 12.2 Competitors | 163 |
| 12.2.1 IBM ILOG CPLEX | 163 |
| 12.2.2 SACRO-PSO Algorithms | 164 |
| 12.2.3 Resolution Search + Branch & Bound (RS + B&B) | 164 |
| 12.2.4 Genetic Algorithm (GA) | 165 |
| 12.2.5 Simulated Annealing (SA) | 165 |
| 12.3 Parameterization | 166 |
| 12.3.1 epiGA | 167 |
| 12.3.2 GA and SA | 168 |
| 12.4 Evaluating epiGA | 170 |
| 12.5 Discussion | 175 |
| 13 Bio-inspired Computing and Smart Mobility | 177 |
| 13.1 Introduction | 177 |
| 13.2 Yellow Swarm Revisited | 177 |
| 13.2.1 Evaluation Function | 178 |
| 13.2.2 Problem Representation | 178 |
| 13.2.3 The epiGenetic Algorithm (epiGA) | 178 |
| 13.2.4 Evolutionary Algorithm (EA) | 178 |
| 13.3 Case Study | 179 |
| 13.4 epiGA vs. EA | 179 |
| 13.5 Discussion | 180 |
| IV Conclusions and Future Lines of Research | 181 |
| 14 Conclusions and Future Work | 183 |
| 14.1 Global Conclusions | 183 |
| 14.2 Future Lines of Research | 185 |
| V Appendices | 187 |
| Appendix A List of Publications Supporting this PhD Thesis | 189 |
| Appendix B Resumen en Español | 193 |
| B.1 Introducción | 193 |
| B.2 Bases Tecnológicas y Científicas | 194 |
| B.2.1 Problemas de Movilidad Inteligente | 194 |
| B.2.2 Computación Bioinspirada | 194 |
| B.2.3 Microsimulación | 195 |
| B.2.4 Mapas y Datos de Tráfico Incompletos | 196 |

| | | |
|---------------------------|--|------------|
| B.3 | Modelado y Resolución de Problemas | 198 |
| B.3.1 | Red Swarm: Reducción de Tiempos de Viaje | 198 |
| B.3.2 | Green Swarm: Menos Emisiones de Gases | 199 |
| B.3.3 | Yellow Swarm: Infraestructura de Bajo Coste Para la Ciudad | 200 |
| B.3.4 | Rutas más Inteligentes Para Navegadores GPS | 201 |
| B.3.5 | Conoce tu Ciudad: Plazas de Aparcamiento | 203 |
| B.4 | Nuevos Algoritmos Bioinspirados | 204 |
| B.5 | Conclusiones y Trabajo Futuro | 207 |
| List of Figures | | 209 |
| List of Tables | | 211 |
| List of Algorithms | | 213 |
| Index | | 215 |
| References | | 219 |

List of Abbreviations

| | |
|-----------------|--|
| -20% | Minus 20% |
| 30km/h | Maximum 30 km/h |
| ABC | Artificial Bee Colony |
| ACO | Ant Colony Optimization |
| ACR | Ant Colony Routing |
| AIS | Artificial Immune System |
| ALA | Alameda |
| ANN | Artificial Neural Network |
| ATLC | Adaptive Traffic Light Control |
| AVCAS | Ant-based Vehicle Congestion Avoidance System |
| BA | Bat-inspired Algorithm |
| BCO | Bee Colony Optimization |
| BER | Berlin |
| BM | Blind Mutation |
| CA | Cellular Automaton |
| CAVE | Congestion Avoidance in Vehicular Environments |
| CH ₄ | Methane |
| CO | Carbon Monoxide |
| CO ₂ | Carbon Dioxide |
| CRO | Check and Repair Operator |
| CS | Cuckoo Search |
| DE | Differential Evolution |
| DESX | Destination Crossover |
| DJK | Dijkstra Algorithm |
| DNA | Deoxyribonucleic Acid |
| DUA | Dynamic User Assignment |
| DUE | Dynamic User Equilibrium |
| DV | Distance Vector |
| EA | Evolutionary Algorithm |
| EC | Evolutionary Computing |
| EDA | Estimation of Distribution Algorithm |
| EfRA | Eco-friendly Route Algorithm |
| EM | Epigenetic Mechanisms |
| epiGA | epiGenetic Algorithm |

| | |
|-----------------|--|
| F | Fourier Series |
| FA | Firefly Algorithm |
| FC-VSL | Fuel-Consumption-aware Variable-Speed Limit |
| FFM | Flow Focused Mutation |
| FGA | Flow Generator Algorithm |
| GA | Genetic Algorithm |
| genGA | generational GA |
| GeS | Gene Silencing |
| GPS | Global Positioning System |
| GrA | Green Algorithm |
| GRASP | GRASP |
| GS | Green Swarm |
| GUI | Graphical User Interface |
| HBEFA | Handbook Emission Factors for Road Transport |
| HC | Hydrocarbons |
| HDV | Heavy Duty Vehicles |
| HGA | Hybrid Genetic Algorithm |
| HNN | Hopfield Neural Network |
| ILS | Iterated Local Search |
| IPR | Intelligent Parking Reservation |
| IT | Information Technologies |
| ITS | Intelligent Transportation Systems |
| IWD | Intelligent Water Drops |
| JOSM | Java OpenStreetMap |
| KH | Krill Herd |
| KM | <i>K</i> -Means |
| KP | KM-Polynomial |
| LDV | Light Duty Vehicles |
| LED | Light-Emitting Diode |
| MADM | Multi-Attribute Decision Making |
| MATSIM | Multi-agent Transport Simulation |
| MGA | Malaga |
| MKP | Multidimensional Knapsack Problem |
| MPC | Model Predictive Control |
| MS | Monkey Search |
| MSE | Mean Squared Error |
| MTS | Multiple Trajectory Search |
| NBR | Nucleosome Based Reproduction |
| NG | Nucleosome Generation |
| NN | Neural Networks |
| NO _x | Nitrogen Oxides |
| O ₃ | Ground Level Ozone |
| OBU | On Board Unit |

| | |
|----------|---|
| OD | Origin-Destination |
| OGL | Open Government Licence |
| OSM | OpenStreetMap |
| P | Polynomial Fitting |
| PAR | Paris |
| pEA | Parallel Evolutionary Algorithm |
| PHEM | Passenger car and Heavy duty vehicle Emission Model |
| PM | Particulate Matter |
| PSO | Particle Swarm Optimization |
| RA | Rerouting Algorithm |
| RG | Route Generator |
| RNA | Ribonucleic acid |
| RS | Red Swarm |
| RT | Regression Tree |
| SA | Simulated Annealing |
| SACRO | Self-Adaptive Check and Repair Operator |
| SFM | Sensor Focused Mutation |
| SI | Swarm Intelligence |
| SIDRA | Signalized Intersection Design and Research Aid |
| SP | Shift & Phase |
| SS | Scatter Search |
| ssGA | steady state GA |
| STO | Stockholm |
| STPX | Street Two Point Crossover |
| SUMO | Simulation for Urban MObility |
| SVR | Support Vector Regression |
| TCO | Termite Colony Optimization |
| TraCI | Traffic Control Interface |
| TRANSIMS | TRansportation ANalysis and SIMulation System |
| TS | Time Series |
| UTU | User Terminal Unit |
| V2I | Vehicle to Infrastructure |
| V2V | Vehicle to Vehicle |
| VANET | Vehicular Ad-Hoc Networks |
| VMO | Variable Mutation Operator |
| VNS | Variable Neighborhood Search |
| VTRS | Vehicle Traffic Routing System |
| XML | Extensible Markup Language |



UNIVERSIDAD
DE MÁLAGA

Chapter 1

Introduction

This volume presents a summary of the research work done with the aim of addressing and solving Smart Mobility problems in a smart city context. Several big cities are modeled to be optimized using new evolutionary techniques and the traffic simulator SUMO. Three new architectures, Red Swarm, Green Swarm and Yellow Swarm are proposed, analyzed and used to reduce travel times, greenhouse gas emissions, and fuel consumption of vehicles. A new method for calculating alternative routes for GPS navigators and the prediction of car park occupancy rates are also included in this PhD thesis. Moreover, a novel algorithm for generating realistic traffic flows is developed and tested in different scenarios: working days, Saturdays, and Sundays. Finally, a new family of bio-inspired algorithms based on epigenesis was designed and tested on the Multidimensional Knapsack Problem and used in the Yellow Swarm architecture.

1.1 Motivation

Nowadays, cities are growing in number of inhabitants, many of whom are arriving at the city for the first time [230]. As a consequence, the number of vehicles in streets is continuously increasing [72] while the infrastructure is not scaling at the same pace to support the demand, which in turn produces congestion, affecting all aspects of daily life.

Whether road traffic sources are public, private, fleets, deliveries or services, there is a notable increase in the number of trips citizens have to take and their duration [224]. These journeys are often to commute or take children to school, which usually occurs at the same time of day. Other sources of traffic in big cities are people visiting hospitals, going shopping, or making short trips to meet each other [215].

Evidence of the problems described above can be seen in the number of traffic jams [215] which have increased in frequency over the last decade and have become a serious issue for residents living in cities. As a result, traveling by car is becoming slower than it used to be and it is a common source of delays, economic loss, and stress because of the effect traffic congestion has on peoples' leisure time and work.

Another consequence is the amount of greenhouse gases emitted to the atmosphere since the more people driving at low speeds or even stuck in traffic jams, the greater the emissions from the vehicles' motor [101]. Some of the main emissions are:

1. Carbon Dioxide (CO_2), the main factor behind the significant climate change the planet is undergoing, especially because of anthropogenic emissions [89], unused gas flaring during oil production, fossil-fuel combustion, natural gas consumption, road transport, etc. [167].
2. Carbon Monoxide (CO), which is emitted by the incomplete combustion of fossil fuels and biofuels. Exposure to CO can reduce the oxygen-carrying capacity of the blood, thereby reducing oxygen delivery to the body's organs and tissues. Furthermore, CO slowly oxidise when is in atmosphere contributing to the formation of ozone which has associated effects on human health and ecosystems. Additionally, CO can turn into CO_2 through chemical processes in the atmosphere [240]
3. Particulate Matter (PM). They are microscopic solid or liquid matter suspended in the atmosphere which can penetrate deep into the lungs and blood streams unfiltered, posing a great risk to human health.
4. Nitrogen Oxides (NO_x), emitted during fuel combustion in domestic heating and industrial facilities. In high concentrations, they cause inflammation of the airways and reduced lung function.
5. Hydrocarbons (HC), which usually correspond to partially burned fuel produced by motor vehicles. They are the main contributor to smog and a prolonged exposure to these gases may cause asthma, lung disease, and cancer.
6. Methane (CH_4). It is produced by organic matter decomposition in oxygen-poor environments. It is also a greenhouse gas so that it might contribute to global warming.
7. Ground Level Ozone (O_3), which also has a marked effect on human health such as breathing problems, reduced lung function, and asthma. It contributes also to global warming. Ozone is formed in the troposphere, from complex chemical reactions involving NO_x , CH_4 , and CO.

Air quality is an important issue for the economy, the environment, and of course, human health. Greenhouse gas emissions not only contribute to global warming, but also jeopardize people's health via different respiratory and cardiovascular diseases as well as lung cancer [101, 137]. They also have an economic impact, shortening lives, increasing medical costs, and reducing productivity through the loss of working days. Additionally, air pollution can also damage buildings and has a clear impact on the climate, since some air pollutants act as greenhouse gases [90].

One of the European Union's objectives for the year 2020 is the reduction of greenhouse gas emissions [64], although emissions from fuel combustion have been rising worldwide since 1971, and this growing tendency seems to be hard to reverse in the near future [163].

Several strategies have been proposed to prevent traffic jams and reduce the amount of gases emitted to the atmosphere [70, 133]. Some of them are based on traffic microsimulation

where each vehicle is modeled as an agent, subject to a car following model, moving by a realistic city map made of streets with multiple lanes, roundabouts, traffic lights, left and right turn restrictions, etc. This very accurate studies have the drawback of long computation times as it is necessary to simulate many vehicles in a big scenario. As an example of this, one hour of simulation time could be equivalent to several minutes in real time using the latest hardware available. This is a matter to be taken into account in optimization studies where it is necessary to evaluate (simulate) each configuration to obtain its fitness value. Moreover, the high complexity found in mobility problems, especially those where there are several routes to choose based on an optimization criterion, make them to be very hard to solve using a deterministic, exact heuristic.

The use of Information Technologies (IT) to solve the problems found in the city of the 21st century, facilitates the combination of several techniques for collecting data and intelligent algorithms based on metaheuristics. Metaheuristics for combinatorial optimization problems [27] are frequently inspired by natural processes such as Darwins' theory of evolution: Evolutionary Algorithms (EA) are today a classic example [14].

Concretely, bio-inspired algorithms such as Genetic Algorithms (GA) [86, 102], Simulated Annealing (SA) [118], Particle Swarm Optimization (PSO) [116], Ant Colony Optimization (ACO) [58], among others are able to find good solutions, usually the optimum, of highly complex real-world problems in reasonable computing times. Usually, they start with a set of initial candidate solutions and iteratively generate new ones in a chain of increasingly fitted populations towards the optimum of the problem. Their non-deterministic guided and intelligent search balances the exploration of the search space and exploits its more promising regions, to hopefully find the optimal solution to the problem being solved.

This PhD thesis is focused on the design of new algorithms inspired in epigenesis and the applicability of their results to improve road traffic in cities. It involves modeling new simulation scenarios, solving problems related to road traffic generation, and generating new paradigms to optimize them. The research work done has been developed in connection with several research projects aiming at real world applications, holistic intelligence, and Smart Mobility: roadME [183], MAXCT [145], moveON [153], CI-RTI [184], and 6city [1].

1.2 Objectives and Phases

Among the objectives of this PhD thesis are the design, implementation, and evaluation of solutions for the Smart Mobility problems found in modern cities, using metaheuristics and bio-inspired algorithms. Concretely:

- O1 Study the existing techniques which are part of the state of the art in Smart Mobility, increasing existing knowledge about the problem to innovate and improve the current techniques by using hybridization, parallelism, etc.
- O2 Design, develop, and analyze a new type of bio-inspired algorithm, based on epigenesis, with the aim of being used to solve problems of combinatorial and continuous optimization.

- O3 Generate realistic scenarios to be optimized by the algorithms and architectures developed keeping in mind the possibly application in real world models besides working with simulators.
- O4 Solve real problems generating new scientific knowledge while collaborating with the industry.
- O5 Dissemination of research results among the society by using new technologies such as YouTube, Twitter, and a personal web site.

The phases of this PhD thesis are based on the scientific method [54, 77] in order to ensure a rigorous and well-defined working methodology. Specifically:

1. *Observation*: We identify interesting, real problems in typical cities and analyze the existing bio-inspired algorithms to build new models and simulations to better understand the current problems, discover new ones, and solve them. Additionally, we study the solutions already proposed by academic researchers and those that are being used by industry.
2. *Hypothesis*: We propose new bio-inspired algorithms using sequential and parallel techniques to better solve hard problems. Our hypothesis is that parallelism and machine learning techniques can be ways of solving open problems closer to reality than the existing in the current literature.
3. *Experimentation*: This is a very important phase in which each experiment has to be thoroughly designed and conducted. We carry out several experiments in order to test our algorithms and Smart Mobility proposals in very realistic scenarios. We analyze the results achieved using statistical tests, validating or refuting our previous hypotheses.
4. *Analysis*: In this phase we analyze existing algorithms and literature, different microsimulations, several maps of European cities, and the experiments conducted in those cities.
5. *Conclusions*: Finally, after the research process, we draw the corresponding conclusions. We confirm our main hypothesis that our bio-inspired proposals can be used to improve the road traffic in the city, not only reducing greenhouse gas emissions and fuel consumption but also shortening travel times. Our results perform well, not only against other data-based solutions, but also against those that have been obtained by human experts according to their own intuition.

1.3 Thesis Contributions

The main contribution of this PhD thesis are the design, study and implementation of new bio-inspired techniques to address road traffic problems in big cities, such as long travel times, high greenhouse gas emissions and fuel consumption. Our objective is to make scientific

contributions not only to solve Smart Mobility problems but also in the techniques used for it. The contributions can be summarized as follows:

- A review of the state-of-the-art proposals for solving Smart Mobility problems.
- A deep study of the road traffic problems found in big cities, including traffic jams, long travel times, and high gas emissions.
- A set of tools to build realistic simulation scenarios based on public data available (open data) such as street distribution, traffic light locations, number of vehicles at measurement points, flows based on people's needs, vehicle types, and car park occupancy rates.
- Three new architectures for optimizing road traffic, shortening travel times and reducing greenhouse gas emissions and fuel consumption.
- A prediction system, based on machine learning techniques, to forecast car park occupancy rates.
- A new family of bio-inspired algorithms based on epigenesis for solving combinatorial and continuous problems, which have been applied to a road traffic problem.

1.4 Thesis Organization

This PhD thesis volume is organized in three parts and two appendices. The Part I introduces the scientific and technological bases of this PhD thesis. There, Smart Mobility problems are described in Chapter 2, metaheuristics as a method for solving hard combinatorial problems are discussed in Chapter 3, microsimulation techniques as a tool for modeling road traffic in a city are analyzed in Chapter 4, and finally, the design and test of the Flow Generator Algorithm are addressed in Chapter 5.

In Part II several problems are modeled and solved by using three new architectures, Red Swarm (Chapter 6), Green Swarm (Chapter 7), and Yellow Swarm (Chapter 8). Additionally, alternative routes for GPS navigators and the prediction of car park occupancy rates are also addresses in Chapter 9 and Chapter 10, respectively.

In Part III, the design of a new bio-inspired algorithm based on epigenesis is presented (Chapter 11), tested in the Multidimensional Knapsack Problem (Chapter 12), and used to optimize the configuration of the Yellow Swarm architecture (Chapter 13).

Finally, our conclusions and future lines of research are given in Chapter 14.



UNIVERSIDAD
DE MÁLAGA

Part I

Scientific and Technological Bases



UNIVERSIDAD
DE MÁLAGA

Chapter 2

The Main Scientific Challenge: Smart Mobility Problems

This chapter presents a review of the Smart Mobility problems that are present in almost all the big cities of the world. Starting with long travel times which make citizens to waste several hours in their car every day, following with air pollution which represent one the most important health issues nowadays. Finally, another consequence of crowded cities is addressed: finding a free parking spot in the city.

2.1 Introduction

It was reported that 50% of Europeans use a car every day [224], while 38% of them encounter problems as they travel around cities. Furthermore, an important number of Europeans believe that the truly serious problems within cities are caused by air pollution (81%), road congestion (76%), traveling cost (74%), accidents (73%), and noise (72%).

Human health, economic development, energy, traffic jams, environmental pollution, and waste management are some of the problems that strongly affect different aspects of our society. These problems represent a challenge for city governments to manage such growing issues in smarter ways. Research to Smart Cities and Intelligent Transportation Systems (ITS) [207] is therefore a must and so is reported by major agencies worldwide [55, 159].

The concept of a smart city has not yet been totally defined [103]. In fact, the term *smart* is usually confused with *digital* or *intelligent* [41], and normally it focuses on the relationship between infrastructures, services, government, and citizens, in a sort of holistic vision. As a result, several definition of a smart city have been given:

- i) “A city connecting the physical infrastructure, the IT infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city” [92]
- ii) “Those cities that utilize information and communication technologies with the aim to increase the life quality or their inhabitants while providing sustainable development” [17]

- iii) “The use of Smart Computing technologies to make the critical infrastructure components and services of a city – which include city administration, education, health care, public safety, real estate, transportation, and utilities – more intelligent, interconnected, and efficient” [235]
- iv) “A city that monitors and integrates conditions of all of its critical infrastructures, including roads, bridges, tunnels, rail/subways, airports, seaports, communications, water, power, even major buildings, can better optimize its resources, plan its preventive maintenance activities, and monitor security aspects while maximizing services to its citizens” [31]

Another way of defining a smart city and identifying its strengths and weaknesses is by measuring its performance and level of development across a broad range of characteristics and factors, as proposed in [80]. The authors identify six characteristics with their corresponding factors (Figure 2.1):

- i) *Smart Economy*: innovative spirit, entrepreneurship, productivity, flexibility of labor market, etc.
- ii) *Smart People*: level of qualification, affinity with life long learning, social and ethnic plurality, flexibility, creativity, etc.
- iii) *Smart Governance*: participation in decision-making, public and social services, transparent governance, etc.
- iv) *Smart Mobility*: local accessibility, international accessibility, availability of infrastructure, sustainable, innovative and safe transport systems
- v) *Smart Environment*: attractiveness of natural conditions, pollution, environmental protection, and sustainable resource management
- vi) *Smart Living*: cultural and education facilities, health conditions, individual safety, etc.



Figure 2.1: The six main axes of smart cities. We are focused on Smart Mobility and Smart Environment.

There are many ways of approaching a modern smart city. They should be based on a holistic model in order to deploy new services according to the city’s priorities. Interesting initiatives are green buildings, electrical cars and buses, optimized water distribution systems, waste reduction, processing and recycling, fair share of goods and services, ITS, digitalization,

robotics, waste bins equipped with capacity sensors, dynamic lightning, smart car parks, smart cards, shopping and street activity monitoring, and systems of geolocation of public transport. Furthermore, global alert systems about pollution, pollen, ultraviolet radiation, ozone, water quality, flooding, fire, storms, hurricanes, etc., could be implemented by using Wireless Sensor Networks (WSN) [69]. Finally, the reduction of travel times, greenhouse gas emissions and fuel consumption ought to be present in a smart city project as well.

The implementation of these systems have to be done keeping in mind not only the local administration but also citizens, who should be the main beneficiaries of them. These technologies must be robust, reliable, intelligent, and easy to use, for instance by delivering services through smartphones or tablets which allow a bidirectional communication and become also a valuable source of data [165].

In the following section several solutions to different problems related to the intersection of two main topics in smart cities are presented. Concretely, road traffic (Smart Mobility) and greenhouse gas emissions (Smart Environment) which are the main topics addressed in this PhD thesis. The main idea is to make greener cities by using evolutionary techniques to optimize road traffic.

2.2 Long Travel Times

There are several proposals for shortening travel times which have been published in the last five years. Some of them deal with urban traffic congestion problems, others with traffic light control, route planning, etc.

The study in [60] presents a recent review of some techniques used for detecting traffic jams and for avoiding congestion on roads. The authors conclude that a GPS based system can be a better alternative technique for traffic jam detection as it can monitor the whole road network and can be incorporated with the strategies for congestion avoidance which will help to improve the traffic flow.

A distributed and cooperative system dedicated to road self-organization is presented in [223] with the aim of detecting traffic jams and transmitting traffic alerts. The authors present a theoretical model based on the FORESEE cooperation model [79] composed of a set of agents which are physically installed in each vehicle. Each agent evaluates the traffic conditions and exchanges information with other agents over wireless media. The working scenario is quite big and the results are achieved by using a traffic simulator. In our studies there is not a minimum number of vehicles to sense and communicate the traffic state. When we use radio communications, they rely only upon a fixed number of spots or nodes.

In [45], the authors present an optimization method that determines routes for drivers and then increases the performance of the traffic network via dynamic traffic routing. A novel algorithm, called Ant Colony Routing (ACR), based on Ant Colony Optimization (ACO) with stench pheromone and colored ants, is proposed for the optimization. The different vehicles routes are modeled by using the colored ants so that they are only sensitive to their own color. Moreover, the stench pheromone is used to disperse ants throughout the network thereby preventing traffic jams. We work with an evolutionary algorithm, use scenarios made

of real streets and we test them by simulating the dynamics of the whole city and players (vehicles, driving rules, traffic lights, etc.).

In [134] a proposal based on an integrated macroscopic traffic model (S model) which includes a macroscopic urban traffic flow model and a microscopic traffic emission model (VT-Micro) is presented. While the former provides macroscopic traffic states for each link, the latter evaluates the emissions based not only on the speed of all vehicles but also on the acceleration or the deceleration of each of them. Moreover, a Model Predictive Control (MPC) [176] is applied to urban traffic networks with the aim of reducing both travel delays and gas emissions based on the aforementioned models, by regulating the stop-and-go behavior and distributions of traffic flows within the network with the aid of traffic signals. Although we also reduce travel times, we use a different approach consisting in rerouting vehicles to avoid congested streets in real geographical areas in a customized manner for every driver.

Another method to reduce travel times is presented in [147]. It consists of an algorithm capable of controlling traffic signals that relies on traffic observations made by available sensor devices and local communication between traffic lights. To evaluate the system developed, a realistic traffic model was made using information supplied by the city of Ottawa, Canada. The advantages presented by the authors are failure tolerance, dynamic response, and the fact that the simulations used to validate this approach are based on historical data. The model of traffic used is composed of just a small number of single intersection snapshots, while we address bigger geographical areas.

In [127] the authors propose an Adaptive Traffic Light Control (ATLC) using Vehicular Ad-Hoc Networks (VANET), which takes into account the vehicle density as well as the relative position of vehicles with respect to junctions. They present a case study based on a specific intersection in the city of Moncton, Canada. The proposed system is validated with real traffic data by dynamically adjusting the periods of green lights. The results of the simulation show that the algorithm proposed improves traffic flows and the current configuration of the city just as well as some other algorithms in the literature. Although the scenario chosen is intended to be a realist one by using historical traffic data and the results are promising, the study is limited to just one intersection.

In [185] a system to prevent traffic jams and reduce congestion by assigning new routes to vehicles is proposed. The route assignation, which excludes heavy load streets, is done during each vehicle's journey of each vehicle so that the driver is able to react to unexpected situations such as accidents, etc. To do this, the authors also use on-board systems such as tablets on which the position of each vehicle as well as the new route calculated are shown. They have experimented with the road network of the island of Manhattan in New York imported from OpenStreetMap [169], and reduced travel times by to 33%. This article is different from our proposal in that each driver needs a wireless device with GPS, Internet connectivity, and a screen to be able to use it.

In [75] the authors introduce a vehicle-to-vehicle (V2V) congestion avoidance mechanism to minimize travel times by detecting congestion levels and rerouting vehicles in real time, based on VANETs. They create a distributed congestion avoidance scheme and consider a reactive mechanism instead of periodic broadcasts. Additionally, a dynamic route planning

technique helps cars to avoid jams by choosing the route with the minimum travel time which is calculated using the congestion information available, previously collected by each car. Our studies are based on a different approach as we do not use vehicular networks but rather previously calculated routes and their probability of being chosen to improve the road traffic in the city.

In [148] the authors propose an architecture to control and manage the utilization of road transport networks to prevent traffic congestion. Their architecture divides an urban area into smaller regions while the capacity of each road segment within these regions is reserved by users on demand, spatially and temporally. Additionally, a real-time scheduling algorithm to solve the route reservation problem is analyzed using a realistic road transport scenario in a large area in Nicosia, Cyprus, extracted from OpenStreetMap and imported into SUMO [123]. Their results indicate that congestion can be avoided and travel times improved after the application of a route reservation algorithm over a specific region. In our architectures we use different approaches and algorithms to optimize other, different cities.

A bi-level optimization framework to settle the optimal traffic signal setting problem is presented in [188]. By using a Hybrid Genetic Algorithm (HGA), the authors decouple the original bi-level problem into two single-level problems employing SUMO and then solve them sequentially. The upper-level problem sets the traffic signal to minimize the drivers' average travel time, and the lower-level problem achieves network equilibrium using the settings calculated in the upper level. The experiments were conducted in an urban area of Chicago obtained from OpenStreetMap with a number of vehicles obtained from the average daily traffic counts. In our proposal, the focus is on the rerouting of vehicles to prevent traffic jams, without changing traffic light cycles.

All in all, long travel times in a city are studied and reduced in this PhD thesis by suggesting new alternative routes which are customized to each driver. Evolutionary techniques and traffic simulation are used to achieve that objective in several big cities whose street layout are quite realistic as they are imported from OpenStreetMap into the SUMO traffic simulator.

2.3 Polluted Cities

Reducing pollution from road traffic is another Smart Mobility problem which is related to Smart Environment initiatives. The following papers are focused on the reduction of gas emissions from vehicles not only by using greener, alternative routes, but also by taking into account traffic light cycles and the design of the city.

A green Vehicle Traffic Routing System (VTRS) that reduces fuel consumption and consequently CO₂ emissions via a bio-inspired algorithm, combined with a fuel consumption model, is introduced in [110]. It consists of an Ant-based Vehicle Congestion Avoidance System (AVCAS) that uses the Signalized Intersection Design and Research Aid (SIDRA) fuel consumption and emission model in its vehicle routing procedure. By using various criteria such as average travel time, speed, and distance, this system is able to reduce fuel consumption by finding the least congested shortest paths and reducing the vehicle traffic congestion and emissions. This approach is evaluated by using simulation environments on a map of Kuala Lumpur imported from OpenStreetMap into the SUMO traffic simulator [123].

In contrast, the approaches presented in this PhD thesis center on reducing gases as a way of improving the rest of the metrics by preventing jams, in several case studies.

An approach for dynamic calculation of optimal traffic routes is presented in [7]. It comprises a multi-objective optimization algorithm, which combines Simulated Annealing (SA) with cost function based on both, Multi-Attribute Decision Making (MADM) and TOPSIS [107] to provide the driver with optimal paths. They use real-time data using Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communications to reroute the vehicles and reduce the congestion on the roads. The results of the proposed algorithm have been compared to the shortest path Dijkstra algorithm [50] and other strategies in two real cities (Sheffield and Birmingham) imported into the SUMO traffic simulator from OpenStreetMap. In our study we use an evolutionary algorithm to optimize our case study, focusing on the reduction of travel times and greenhouse gas emissions by suggesting alternative routes to vehicles driving through the area under analysis.

In [227] the authors address the optimization of vehicular traffic flows by using road-side units (V2I) to gather information with which to redirect vehicles to less congested roads and reduce CO₂ emissions. The proposed algorithm, called Congestion Avoidance in Vehicular Environments (CAVE), uses the rerouting strategy for vehicles in order to spread them over several available road segments, reducing vehicular congestion. They modeled the vehicular network with an oriented and dynamically weighted graph updated according to the number of vehicles in the streets. To manage vehicle mobility they use the OMNet++ simulator with the Veins framework connected to SUMO. The results presented show that the CAVE algorithm reduces travel times, gas emissions, fuel consumption, and road congestion by showing less congested routes to the drivers. Although our proposal also reduces travel times and gas emissions, our aim is to implement a lightweight infrastructure and a high reutilization of urban devices such as traffic lights and existing networks to reduce costs.

Several studies have focused on reducing the gas emissions from vehicles in urban areas. In [251] the authors discuss the conflict between a reduction in travel times and gas emissions. They implement a Model Predictive Control (MPC) and propose an objective function especially built to weigh the different emission parameters as well as the traffic flow, and test them in a traffic simulator. They solve the MPC optimization problem by using a multi-start sequential quadratic programming optimization method and conclude that an improvement in the traffic flow does not necessarily guarantee reduced emission levels. This is the reason why we explicitly include in our studies both traffic flow and emissions.

In [124] two different scenarios which model the roads within the city of Bologna and its surroundings are optimized by using different emission metrics as edge weights for the Gawron algorithm [78]. The authors observe that optimization seems to depend on the type of roads available in the area analyzed and that there are a series of inter-dependencies between pollutant gas emissions and road networks. We have tested our proposals on different case studies to check how they behave and observe several metrics in detail using our micro-simulations.

In [6] a new protocol is proposed. Called the environmentally friendly geocast protocol, it focuses on minimizing CO₂ emissions from vehicles approaching a traffic light signal. By using the information delivered by the signal, vehicles calculate their recommended

environmentally friendly speed in order to avoid some actions such as stop-and-go conditions, high speeds, and high accelerations. The paper deals with just one intersection (we use extensive areas or the whole city itself) and does not reroute vehicles, as we do in our studies.

In [132] a real time traffic light control scheme for reducing vehicle CO₂ emissions is proposed. The road conditions are obtained by the wireless communication between an electronic toll collection transponder installed in vehicles and traffic lights. Vehicles send the passing requests to the traffic lights, so that the traffic control center knows the road conditions in real time and dynamically adjusts the traffic lights cycle length based on a decision tree algorithm. Despite the important reduction in CO₂ emissions achieved in this work, the authors only present the results of a simple intersection instead of a larger geographical area.

In modern civil engineering, a few cities have been designed from scratch taking CO₂ emissions into account [149], but the overwhelming number of existing cities have to find some other ways of reducing greenhouse gas emissions. The authors of the aforementioned article evaluate the city of Yokohama in Japan as a case study with different urban forms and traffic, and analyze the relationship between them and CO₂ emissions. The proposed method is a useful tool for urban planners to test some land use and transportation policies [212] for designing sustainable cities. However, what we are interested in is optimizing already existing cities, taking advantage of their current infrastructure.

In [142] the authors implement three strategies in order to reduce local traffic emissions: i) reducing traffic demand by 20%, ii) replacing heavy duty vehicles by 1.5 light duty vehicles, and iii) introducing a speed limit of 30 km/h, in a single intersection located at Bentinckplein in the city of Rotterdam, the Netherlands. The authors analyze only one intersection instead of large districts of a given city as we do.

In [16] the authors develop a methodology to estimate the effectiveness of ramp metering in reducing CO₂ emissions. Based on their findings, they suggest that ramp metering could be used to decrease CO₂ emissions regardless of the number of vehicles taking the detours at ramps. The implementation of this system requires a traffic detection system and pre-timed ramp controls to be installed, which does not take advantage of the existing infrastructure. In our case, we aim to implement a lightweight infrastructure and a high reutilization of urban devices such as traffic lights.

In [56] a novel eco-friendly algorithm called EcoTrec is introduced. It considers road characteristics as well as traffic conditions in order to improve the fuel savings of vehicles and reduce gas emissions, using VANETs for collecting and disseminating information to each other. It was tested in the simulated scenario of 6 km² of the city of Cologne, Germany. Its results show that EcoTrec achieves a reduction of 20% in CO₂ emissions in that scenario. In our studies we want not only to reduce emissions, but also to test our proposal in different scenarios and cities to prove its robustness.

A carbon-footprint/fuel-consumption-aware variable-speed limit (FC-VSL) traffic control scheme is presented in [135]. The authors minimize fuel consumption for a single vehicle under certain traffic conditions, and obtain the optimal vehicular trajectory. To do that, they designed the FC-VSL scheme based on the optimal trajectory and applied it to all vehicles on the road, and evaluated its performance through simulation. Their results show that the

FC-VSL can reduce average fuel consumption and outperform another VSL scheme which was designed for smoothing vehicular traffic. Our proposal differs from this one, especially in the strategy used to achieved a reduction of fuel consumption and emissions.

In [11] the reductions of travel times and gas emissions are achieved through the traffic signal settings using a single (travel time) and bi-objective (travel time and fuel consumption) evolutionary algorithm. The study shows that allowing different cycle lengths between signals and coordinating them by correctly setting their offsets can significantly reduce both travel times and fuel consumption. Despite the fact we have also considered Quito's mobility behavior as a new case study for one of our solutions, we have used a completely different strategy based on rerouting vehicles instead of adjusting traffic lights.

To sum up, we address in this PhD thesis the reduction of greenhouse gas emissions and fuel consumption by suggesting alternative routes to drivers, preventing traffic jams, and fostering eco-driving through the city's streets. The proposals discussed are based on evolutionary techniques, microsimulation, and real maps and traffic data.

2.4 Finding an Available Car Park Spot

Finding an available parking space is hard in most big cities, especially in the city center. Off-street car parks are a viable alternative since on-street parking spaces are quite limited and usually it is cheaper to find an off-street car park or pay and display bays rather than wasting time (and fuel) in finding a free space. Nevertheless, whichever is the parking modality chosen, it is a big advantage to know whether a free space would be available for us, in advance. The prediction of car park availability has been studied in a context of smart cities for many years, especially now when most parking facilities have installed sensors as part of their infrastructure.

In [119] the authors fit a continuous-time Markov model to predict future occupancies in several parking locations to propose different alternatives to drivers. They consider not only the car park occupancy rate but also the estimated time of arrival obtained from the vehicle's navigation system in which the calculations are done. They provide two *ad hoc* examples to test their proposal, showing promising results. The approach taken in this PhD thesis is based on open data published by local authorities instead of using users' personal devices.

In [253] two smart car park scenarios based on real-time information are presented. The authors use historical data made available by the authorities of the cities of San Francisco, USA and Melbourne, Australia. They employ Regression Tree (RT), Neural Networks (NN) and Support Vector Regression (SVR) as prediction mechanisms for the parking occupancy rate. Their experiments reveal that the regression tree using the historical data in combination with time and day of the week, performs best for predicting parking availability on both data sets. We have analyzed different predictors which present alternative results depending on the number parameters used.

In [33] the authors propose a methodology for predicting parking space availability in Intelligent Parking Reservation (IPR) architectures. It consists of a real-time availability forecast algorithm which evaluates each parking request and uses an aggregated approach to iteratively allocate parking requests according to drivers' preferences, and parking avail-

ability. They employ historical information of entering and leaving to update and predict the availability for each parking alternative. The results provided, obtained from contrasting predictions with real data, show that the forecast is adequate for potential distribution in real-time. Our approach studies different predictors without interacting with the current demand, relying just on the historical data.

In short, the car park prediction strategies analyzed in this PhD thesis consist in analyzing the historical occupancy rates of car parks and forecast the future availability, presenting this information to the users in a web page or a mobile application. The data source used comes from open data published by local councils which are periodically collected by our system, stored for historical purposes, and processed in order to predict available parking spaces.



UNIVERSIDAD
DE MÁLAGA

Chapter 3

Our Scientific Base: Bio-inspired Computing

In this chapter metaheuristics are described, especially those used in this PhD thesis. As they are well-known techniques for solving hard combinatorial problems we have used them either to compare their results with our algorithms, or as the starting point for building new ones. Finally, the statistical methods used for validating our results are presented.

3.1 Introduction

Metaheuristics for combinatorial optimization problems [27] are frequently inspired by natural processes such as Darwins' theory of evolution: evolutionary algorithms are today a classic example [14]. They are used to solve highly complex real-world problems. Usually, they start with a set of initial candidate solutions and iteratively generate new ones in a chain of increasingly fitted populations towards the optimum of the problem. Their non-deterministic guided and intelligent search balances the exploration of the search space and exploits its more promising regions, to hopefully find the optimal solution to the problem being solved.

3.2 Metaheuristics

Formally, an optimization problem is defined as a pair (S, f) , where $S \neq \emptyset$ is the search space, and f is the objective (fitness) function defined as: $f : S \rightarrow \mathbb{R}$. Solving an optimization (minimization) problem consists in finding a solution $i^* \in S / f(i^*) \leq f(i), \forall i \in S$. The solution of maximization problems is equivalent as proposed in [14, 86]: $\max \{f(i), i \in S\} \equiv \min \{-f(i), i \in S\}$.

Metaheuristics [84, 178] are approximate algorithms capable of finding good solutions (usually the best) to hard problems which cannot be solved by using traditional exact techniques, because they would need extremely long computation times and/or their high

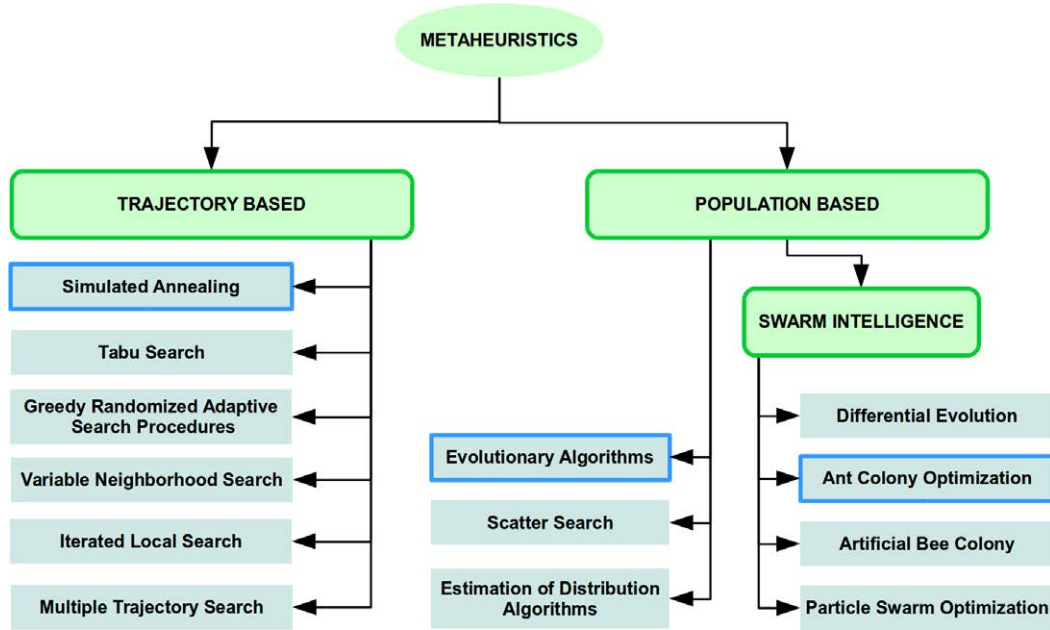


Figure 3.1: Classification of metaheuristics. The algorithms used in this PhD thesis are outlined.

memory requirements. There are two different search strategies in metaheuristics: trajectory based and population based.

On the one hand, trajectory based algorithms (Figure 3.1) handle a single element of the search space at time, i.e. one solution. These algorithms use some mechanism to escape from a local optimum in their basic local search methods. Examples of trajectory based algorithms are Simulated Annealing (SA) [118], Tabu Search (TS) [84], Greedy Randomized Adaptive Search Procedures (GRASP) [67], Variable Neighborhood Search (VNS) [150], Iterated Local Search (ILS) [85], and Multiple Trajectory Search (MTS) [228].

On the other hand, population based algorithms (Figure 3.1) work with a set of elements or solutions, i.e. population, colony or swarm. There exists a learning factor in these algorithms as they try to identify regions of the search space which present high quality solutions by using the individuals in the population. We could say that these methods perform a biased sampling of the search space. Evolutionary Algorithms (EA) [85], Scatter Search (SS) [83], Estimation of Distribution Algorithms (EDA) [139], Differential Evolution (DE) [213], Ant Colony Optimization (ACO) [57], Artificial Bee Colony (ABC) [115], and Particle Swarm Optimization (PSO) [116] are all examples of population based algorithms.

This PhD thesis focuses on bio-inspired computing [143, 171] which takes the inspiration from nature to design algorithms capable of solving high complex problems. From the mathematical model of a neuron [146] to ants using pheromones to better foraging food as an example of emerging intelligent behavior [114], bio-inspired algorithms have been grouped in three main types depending on the source of inspiration:

1. Evolutionary Computing (EC): Evolutionary biology ideas used for designing Evolutionary Algorithms (EA).

2. Swarm Intelligence (SI): Algorithms in which a set of simple agents mimic the behavior of social organisms.
3. Artificial Immune Systems (AIS): Models followed by immune systems are used to develop computational tools.

Among the nature inspired metaheuristics we can name, Genetic Algorithms (GA) [86, 102], Simulated Annealing (SA) [118], Particle Swarm Optimization [116], Artificial Neural Network (ANN) [146], Ant Colony Optimization (ACO) [57], Harmony Search [254], Bat-inspired Algorithm (BA) [245], Artificial Immune Systems (AIS) [47], Bee Colony Optimization (BCO) [220], Cuckoo Search (CS) [246], Termite Colony Optimization (TCO) [99], Firefly Algorithm (FA) [247], Krill Herd (KH) [74], Monkey Search (MS) [154], and Intelligent Water Drops (IWD) [106]. The bio-inspired algorithms used in this PhD thesis are described in the following sections.

3.2.1 Evolutionary Algorithms (EA)

As mentioned, evolutionary algorithms (EA) [85] are stochastic search methods which solve a wide range of combinatorial (and continuous) problems very efficiently and effectively, especially those that cannot be solved, using classic optimization tools: guided or efficient exhaustive enumeration like in A^* , mathematical programming, branch-and-bound and dynamic programming, to name a few. The multiple search at the same time, the representation of the solutions in any convenient manner for the search, the absence of requirements of continuity/derivability of the function being optimized, and many other advantages (like dealing with any type of constraints, mixed variable domains, large dimensionality, etc.) make EAs a great tool in modern research.

Evolutionary Algorithms are a population-based method and at each iteration several operators are applied to the μ individuals of the population. After each iteration, the λ new individuals are obtained in order to be incorporated in the next generation. Recombination, mutation, selection, and replacement operators are commonly found in EAs as a way of producing new individuals which can experiment self-adaptation and be naturally selected based on their fitness value which is provided by the objective function. One of the most commonly used EA is the Genetic Algorithm.

Genetic Algorithm (GA)

Genetic Algorithm (GA) [86, 102] is a very popular subclass of EA with proven efficacy in solving combinatorial optimization problems, either static or dynamic versions [4]. Genetic Algorithms simulate processes present in evolution such as natural selection, gene recombination after reproduction, gene mutation, and the dominance of the fittest individuals over the weaker ones. A typical GA consists of a population of μ individuals from which a subset is selected using a selection operator. Then these individuals are recombined using a crossover operator to obtain a new set of λ individuals based on the original ones. After that, a probabilistic mutation is applied (mutation operator) introducing little modifications

Algorithm 3.1 Pseudocode of Genetic Algorithm (GA).

```

procedure GA( $N_i, P_c, P_m$ )
   $t \leftarrow 0$ 
   $P(0) \leftarrow \text{PopulationInitialization}(N_i)$ 
   $Q(0) \leftarrow \emptyset$ 
  while not TerminationCondition() do
     $Q(t) \leftarrow \text{Selection}(P(t))$ 
     $Q(t) \leftarrow \text{Crossover}(Q(t), P_c)$ 
     $Q(t) \leftarrow \text{Mutation}(Q(t), P_m)$ 
    Evaluation( $Q(t)$ )
     $P(t+1) \leftarrow \text{Replacement}(Q(t), P(t))$ 
     $t \leftarrow t + 1$ 
  end while
end procedure

```

▷ P = population
 ▷ Q = auxiliary population
 ▷ Selection operator
 ▷ Crossover operator
 ▷ Mutation operator
 ▷ Evaluation function
 ▷ Replacement operator

to the individuals chromosome. Finally, after evaluating the new offspring, the replacement operator selects the fittest to replace the former population.

Algorithm 3.1 shows the pseudocode of a canonical GA. There are two variants of GA: steady state GA (ssGA) where the new individuals are directly generated and inserted into the population using the replacement operator, and generational GA (genGA) where a new auxiliary population is created which will replace the original one after each generation.

First, the number of steps t is set to 0 and the population $P(0)$ is initialized. Then, after initializing the auxiliary population $Q(0)$, the main loop is executed while the termination condition is not fulfilled. Inside the main loop, the *Selection* operator is applied to fill the working population $Q(t)$. Next, the *Crossover* operator is applied and after that, the *Mutation* operator modifies the new offspring. Finally, after the *Evaluation* of $Q(t)$, the new population $P(t+1)$ is obtained by applying the *Replacement* operator.

3.2.2 Simulated Annealing (SA)

Simulated Annealing (SA) [35, 118] is a well-known metaheuristic applicable to a wide range of problems. This is a probabilistic method used to find an approximate global optimum in a large search space. It is inspired in annealing in metallurgy where a previously heated material is gradually cooled in order to increase its ductility and reduce its hardness. This temperature decrement is interpreted in SA as a reduction of the probability of accepting worse solutions while the search space is being explored.

At the beginning of the SA (Algorithm 3.2) an initial solution is randomly generated and the initial temperature is set. Then, while the termination condition is not met, a new solution is generated from the previous one. After that, the solution acceptance is checked. If the evaluation of the new solution (X') turns out to be better than the previous one (X), this new solution replaces the older. If not, there is a probability of accept this new worse solution, depending on the current temperature (T_k). Finally, the temperature is updated (decreased) and a new iteration begins.

Algorithm 3.2 Pseudocode of Simulated Annealing (SA).

```

procedure SA
   $X \leftarrow \text{GenerateInitialSolution}()$ 
   $t \leftarrow 0$ 
   $T_k \leftarrow \text{InitialTemperature}()$ 
  while not  $\text{TerminationCondition}()$  do
     $X' \leftarrow \text{PickRandomNeighbour}$  ▷ Generates a new solution
    if  $(c(X') < c(X))$  then ▷ New solution acceptance
       $X \leftarrow X'$ 
    else
       $\text{Accept}(X, X', T_k)$  ▷ Acceptance
    end if
     $T_{k+1} \leftarrow \text{Update}(T_k)$  ▷ Temperature decrement
     $k \leftarrow k + 1$ 
  end while
end procedure

```

The acceptance probability in SA often adopts one of the two forms [248] shown in equation 3.1 and 3.2 where $c(X)$ is the cost function which provides the fitness value of the solution X .

$$\text{Accept}(X, X', T_k) = \min\{1, e^{-\frac{c(X') - c(X)}{T_k}}\} \quad (3.1)$$

$$\text{Accept}(X, X', T_k) = \frac{1}{1 + e^{\frac{c(X') - c(X)}{T_k}}} \quad (3.2)$$

3.2.3 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) [57] is an optimization technique inspired by the natural behavior of ants. It is a general-purpose heuristic method for identifying efficient paths through a graph and has been successfully applied to solve different combinatorial optimization problems with discrete representations.

The ACO algorithm simulates the foraging behavior of ants in search of food and their collaborative effort model by pheromone trails. Algorithm 3.3 presents its pseudocode.

First, an initial solution is generated and the pheromones are initialized. Then, while the termination condition is not met a new solution X_i is built and if it represents an improvement on the current best solution X , it becomes the best one. At the end of each iteration, the pheromones τ are updated to dissuade following solutions already visited.

Algorithm 3.3 Pseudocode of Ant Colony Optimization (ACO).

```

procedure ACO
   $X \leftarrow \text{GenerateInitialSolution}()$  ▷ Initialization
   $\tau \leftarrow \text{InitializePheromone}()$ 
  while not  $\text{TerminationCondition}()$  do
    for  $i = 1 \rightarrow m$  do
       $X_i \leftarrow \text{ConstructSolution}(\tau)$  ▷ Generates a new solution
      if  $(c(X_i) \leq c(X))$  then
         $X \leftarrow X_i$ 
      end if
    end for
     $\tau \leftarrow \text{EvaporatePheromone}(\tau)$  ▷ Pheromone update
     $\tau \leftarrow \text{UpdatePheromone}(\tau)$ 
  end while
end procedure

```

3.3 Statistical Validation

As metaheuristics are stochastic algorithms which include several random operators, one single run is not enough to compare their results with another competitor algorithm. On the contrary, a series of runs for each algorithm's configuration is required in order to calculate a global indicator such as median, mean, and standard deviation. However, using a single global indicator can produce biased conclusions on any empirical analysis.

In this PhD thesis we will use a standard procedure recommended in [189] for statistical comparison of metaheuristics, especially, parametric and non-parametric tests [49]. Parametric tests are meant to detect differences in distributions when they are obtained from independent executions, they follow a Gaussian distribution, and they have sub-populations presenting different variabilities from others. Non-parametric tests, are less restrictive and can be applied regardless of the three previous conditions.

Figure 3.2 shows the general framework to perform a statistical analysis proposed in [3]. We follow it by performing first, a normality test using Kolmogorov-Smirnov to check if the variables' values follow a normal distribution. Then, if they are normally distributed, ANOVA and Student's t -test will be used to analyze the variance and ensure the statistical significance. If not, Friedman and Wilcoxon are used instead. As the resulting distribution could easily be non-normal in metaheuristics, we have used in this PhD thesis non-parametric statistical validation (Friedman and Wilcoxon) and set our null hypothesis with a confidence level greater than 99%, i.e. the statistical differences in our test are with a p -value < 0.01 .

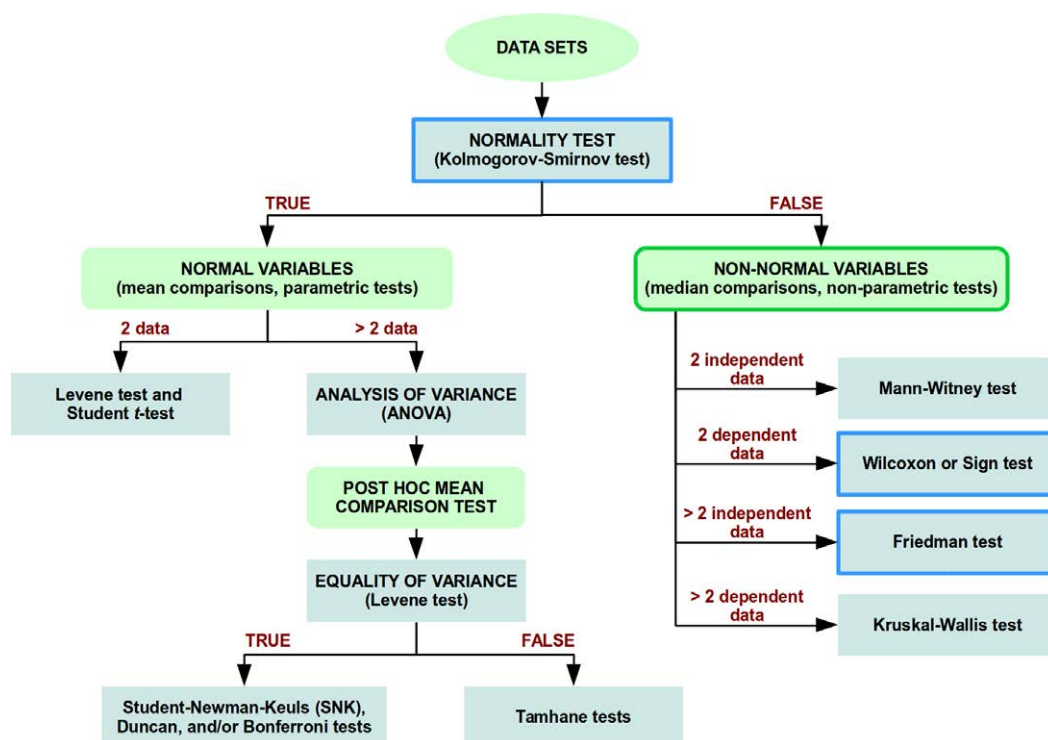


Figure 3.2: Statistical validation of results in metaheuristics. The tests used in this PhD thesis are outlined.



UNIVERSIDAD
DE MÁLAGA

Chapter 4

The Main Technological Base: Microsimulation

In this chapter we discuss what is a traffic simulator and why it is present in our experiments. We describe some well-known traffic simulators which are widely used nowadays as well as OpenStreetMap as a source of geographical maps. Finally, we describe the SUMO traffic simulator in detail as it is the one we are used in the studies conducted in this PhD thesis.

4.1 Introduction

The experimentation in a real city with real vehicles and people is very complicated and almost impossible if we want to study an area bigger than just an intersection or two, not to say that it has to be done in real time which would require experimenting for weeks or even months. As in several disciplines such as robotics and mechanics, Smart Mobility requires simulation to analyze and optimize a city case study. Of course, nothing prevents anyone from going to the city later and checking the validity of the results obtained in the laboratory. In fact, a third option is to experiment in a controlled environment with real data, which represents an almost perfect option for carrying out scientific studies like the ones in this PhD thesis.

Computer simulations [241] have been used in several disciplines since the very beginning of the computer age. Whether it is about simulating physical objects, chemical processes, weather phenomena, economic markets, astrophysics, etc., a model representing the behavior of the real system is required, subject to simplifications and generalizations, especially in those systems so complex that a describing equation is not available.

Traffic simulators [18, 25] are a very valuable tool for representing all the factors involved in an actual scenario where city streets, traffic flows, vehicles, and even pedestrians can be analyzed *in vitro* and in a short, affordable time, usually with a high degree of realism. Then, a whole set of output metrics could be retrieved to analyze the vehicles' performance and suggest improvements for the real city (*in silico*), which would not be possible otherwise.

Traffic simulators implement different flow models [32, 95, 140] to specify rules for car movement, lane changing, etc. According to the level of granularity, they can be categorized as macroscopic, mesoscopic, and microscopic simulators [18].

Macroscopic models represent traffic as continuous flows of vehicles inspired by the fluid theory, hence, they do not consider each car individually. These models are used when detailed information about the behavior of individual vehicles is not required but only a general evaluation of traffic flows in a network. As this is the highest level of abstraction, it is difficult to apply to an actual urban scenario composed of intersections, traffic lights, left-turn restrictions, etc. However, they are often used for regional transportation planning, instead [140].

Mesoscopic models use an intermediate level of detail. They describe some interactions between cars at an individual level, although they move in groups. The position of each car depends on a probability value, which also makes this model unsuitable for the details we need in this PhD thesis.

Microscopic models describe the mobility parameters of each vehicle with respect to others in detail, while macroscopic and mesoscopic models work at a higher level of abstraction [140]. They deliver estimated, but reliable and detailed information about the behavior of each single vehicle in the simulation. Additionally, microscopic simulators allow us to better know what is happening at each intersection, modify traffic light cycles, define individual routes, etc., increasing the reality of our studies.

In the following section, microscopic traffic simulators are discussed as they will be used to build and analyze realistic scenarios in Chapter 5 and in our Smart Mobility architectures presented later in Part II.

4.2 Traffic Microsimulators

Microscopic traffic simulators implement the highest level of detail in the simulation involving not only the vehicles moving through streets, but also traffic lights, pedestrians, buses, bicycles, etc. They need more computational resources than mesoscopic and macroscopic simulator as each single vehicle is modeled and updated at a defined time step. In spite of that, the outputs obtained such as travel times, emissions, queue lengths, and distance traveled, are very accurate because they are calculated for each vehicle while traveling throughout the road network.

Car following models are used to update the vehicles' position during the microsimulation. This makes it possible to model drivers' behavior by using parameters such as acceleration, deceleration, driver imperfection, eagerness for performing lane changing, driver impatience, red light violation probability, among others. Furthermore, it is more likely to detect traffic jams produced by saturated lanes, wrong traffic light cycles, or specific turn restrictions than in the other simulation models.

As we are going to work with traffic microsimulators, we will analyze some of the most commonly used in the following sections, before choosing the most suitable for the objectives of this PhD thesis.

4.2.1 TRANSIMS

TRANSIMS (TRansportation ANalysis and SIMulation System) [226] is an integrated system of travel forecasting tools for modeling regional transport systems. This software package has been made available through open-source licensing and is divided into modules, being the traffic microsimulator the one which implements the microscopic simulation. The transportation modeling and simulation is done from population synthesis, which is iteratively obtained according to the first Wardrop's principle [234].

The simulation is based on a Cellular Automaton (CA) to implement different car following models, lane changes, etc. Consequently, each link is segmented into small cells of equal length which can be either occupied by a vehicle or empty. TRANSIMS's simulation creates detailed snapshot data which can be visualized (Figure 4.1) and also processed to obtain several metrics. The emissions reported by TRANSIMS are estimated from the aggregate data due to the use of the aforementioned CA based simulation.

Other modules available in TRANSIMS are the activity generator to generate household activities, priorities, locations, travel preferences, etc., and the route planner, which reads the activities previously generated and calculates the fastest routes.

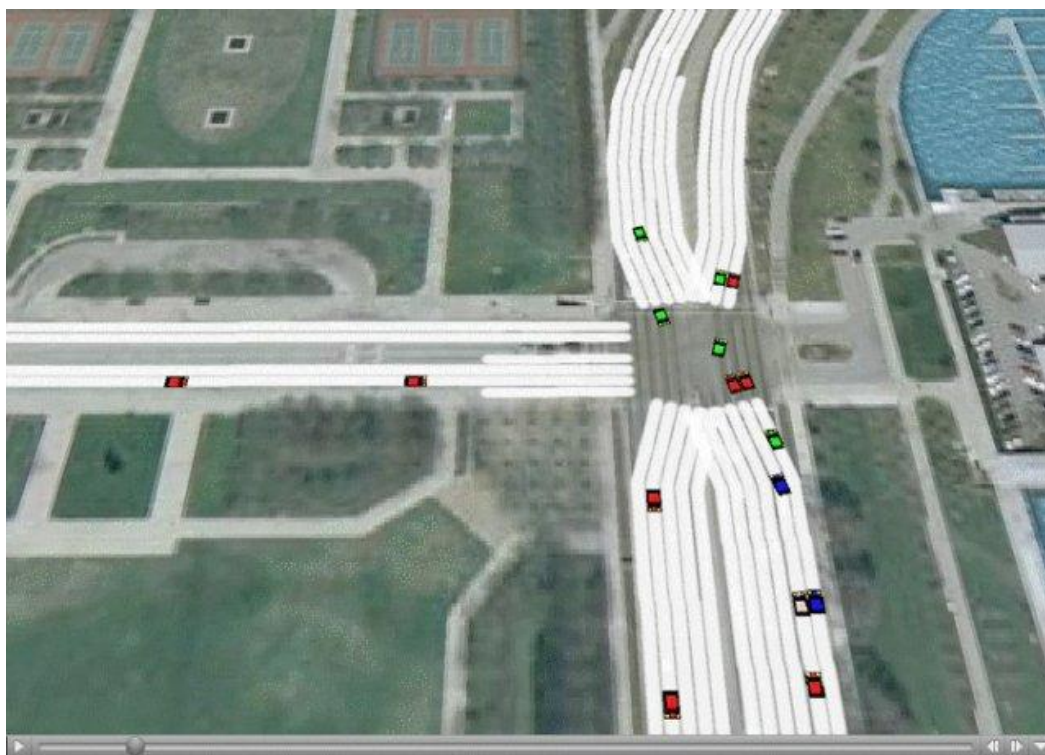


Figure 4.1: Snapshot of the TRANSIMS's visualization component (TRANSIMS Studio Wiki¹).

¹<https://sourceforge.net/p/transimsstudio/wiki/Home/>

4.2.2 VISSIM

VISSIM [232] is a microscopic traffic simulator developed by PTV [174]. It uses the psycho-physical driver behavior model proposed by Wiedemann [238, 239], which considers the psychological and physical aspect of the drivers, and the Helbing's social force model for the pedestrian dynamic [100].

This commercial software uses one-way links connected with connectors for representing a road network instead of a graph of nodes, which allows modeling geometries with any level of complexity. VISSIM can simulate several types of vehicles such as trams, cars, motorbikes and also pedestrians, enabling individual parameterization of the drivers.

The graphical capabilities of VISSIM allow the creation of high-detailed 2D and 3D animations (Figure 4.2) while collecting data from the simulation such as vehicles' speed, acceleration, emissions, trajectories, etc.



Figure 4.2: Unity Interface to PTV Vissim (PTV Vissim 10 Highlights²).

4.2.3 MATSim

MATSim (Multi-agent Transport Simulation) [144] is an open source software for microscopic modeling of traffic. It is based on a multi-agent simulation framework designed for large-scale scenarios which uses a queue-based model instead of a computationally expensive car-following behavior [105].

The MATSim traffic flow model is based on the storage capacity and flow capacity of links. The former is the maximum number of vehicles fitting on a network link, while the latter represents how many vehicles can leave a link per time step. Additionally, a

²<http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/release-highlights/>

co-evolutionary algorithm is used to optimize each agent's plan to reach an equilibrium where agents cannot further improve their plans.

MATSim's working scenarios can be manually defined or imported from OpenStreetMap [169]. It is possible to define traffic lights, number of street lanes, maximum speeds, etc. Moreover, several vehicle types can be defined such as car, bike, bicycle, bus, taxi, and train.

After performing a MATSim simulation, several output metrics can be collected and analyzed such as waiting times, travel times, emissions, among others. Furthermore, there are two visualizers available for MATSim: the original one, OTFVis (Figure 4.3), which is an open source software implemented as a MATSim extension, and Via, a commercial visualizer developed by Senozon which presents a better user interface and stability.

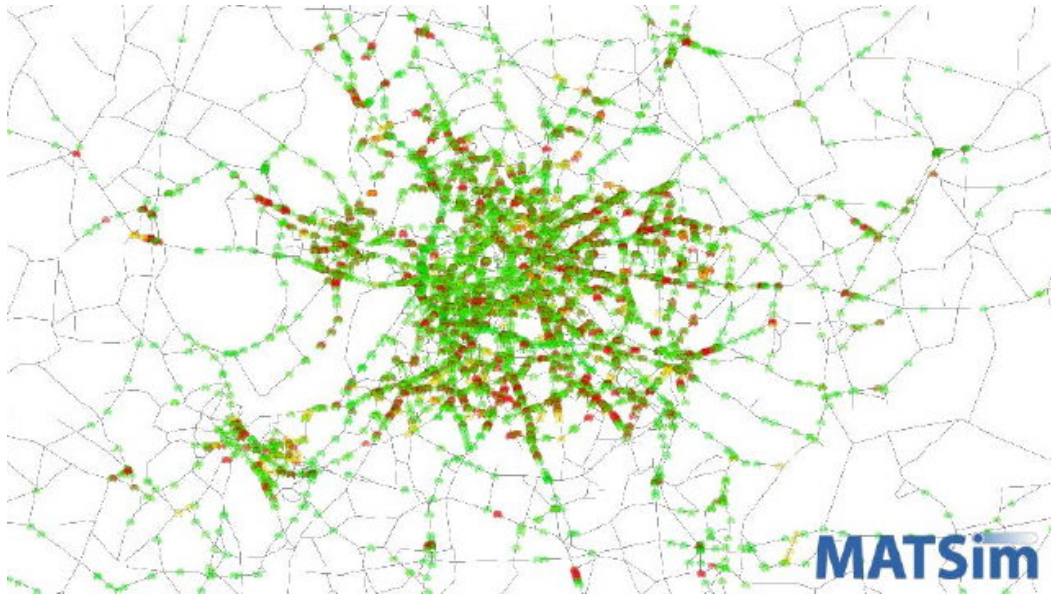


Figure 4.3: Traffic and Public Transit in Berlin (BVG and PTV, MATSim scenario gallery³).

4.2.4 SUMO

SUMO (Simulation for Urban MObility) [123, 216] is a free and open microscopic traffic simulator developed by the German Aerospace Center (DLR) [53]. It is actually a software package which includes not only the traffic simulator but also visualization tools, a network generator, a route generator, etc. SUMO is also capable of performing mesoscopic simulations, although this characteristic was not used in our studies.

It implements several car-following models and several vehicle's characteristics can be easily defined. Moreover, simulation scenarios are manually defined or imported from several sources, including OpenStreetMap [169]. Furthermore, SUMO can be externally controlled via a socket-based interface to add more versatility to simulations.

SUMO includes a 2D graphical visualizer (Figure 4.4) of the simulation where the user can interact and modify several vehicle and simulation parameters. Regarding the available

³<https://www.matsim.org/gallery/berlin/>

outputs, the simulation generates individual vehicle metrics, simulation summary, trip data, emissions and noise generated, routes used, queue lengths, etc. SUMO is explained in detail in the following section.

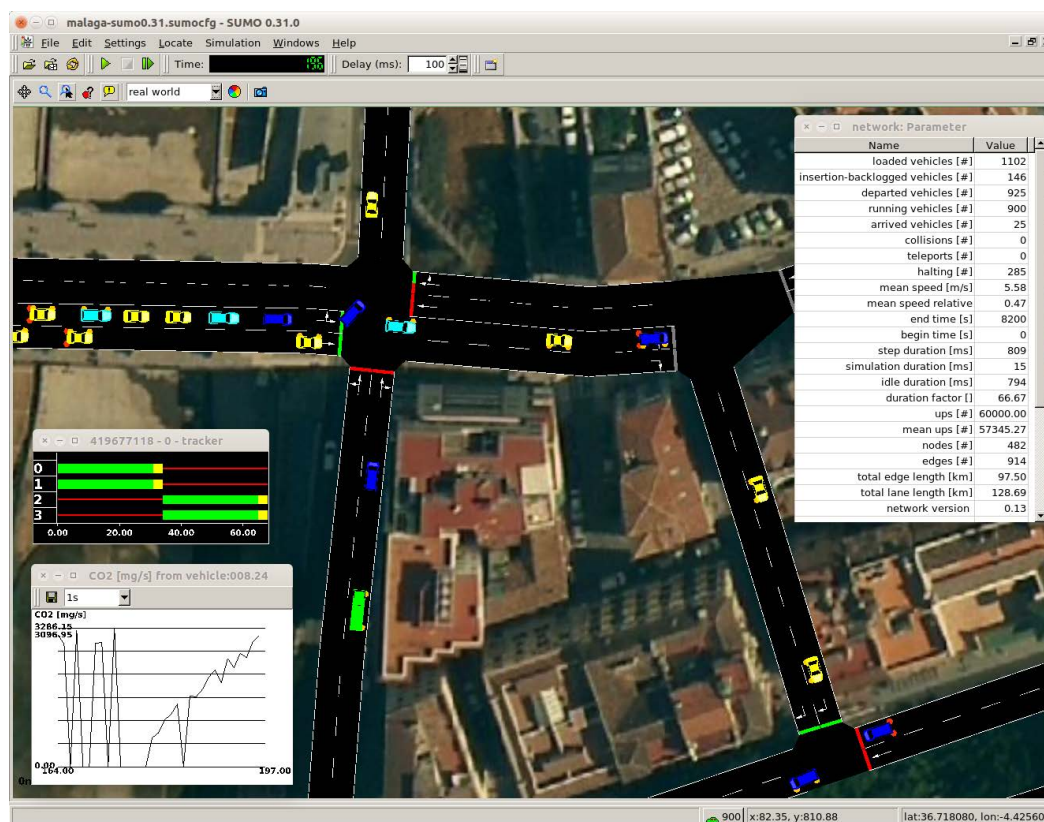


Figure 4.4: SUMO's GUI (Graphical User Interface).

4.3 SUMO: Simulation of Urban MObility

In this PhD thesis SUMO (Simulation for Urban MObility) [123, 216] is used for simulating the mobility scenarios to be optimized in order to evaluate them using thousands of vehicles and realistic maps.

4.3.1 Main Characteristics of SUMO

SUMO was chosen to be used as a simulation tool in our studies due to the following main reasons:

- SUMO is an open source project which allowed us to compile and customize the simulator to ours experiments.
- As a cross-platform software, the source code of SUMO is freely available.

- There are several sources of documentation, tutorials, and forums to learn how to use SUMO and solve possible doubts.
- It implements several car following models which can be highly customized.
- Maps can be imported from OpenStreetMap increasing the realism of the city layout.
- There are lots of data available after the simulation to collect and report as metrics.
- Simulation can be externally controlled and its entities modified to implement complex behaviors that do not exist in SUMO.
- Despite the fact the SUMO's GUI (Graphical User Interface) has not 3D capabilities, it provided everything we needed to visually identify road traffic issues and make videos to disseminate our results.
- SUMO's development team is constantly evolving this software, adding new features such as pedestrians, trains, electric vehicles, energy consumption models, and wireless onboard devices.

SUMO includes several utilities related to traffic simulation:

- SUMO: The microscopic, space-continuous, and time-discrete traffic flow simulator.
- SUMO-GUI: The simulator plus a graphical user interface to visualize and interactively modify the simulation.
- NETCONVERT: An utility to import and generate traffic roads networks.
- POLYCONVERT: Imports geometrical shapes and converts them into a graphical representation to be used by NETCONVERT while building the network and then visualized using SUMO-GUI.
- NETEDIT: A graphical network editor for SUMO.
- NETGENERATE: It is used to generate abstract road networks.
- OD2TRIPS: Imports Origin-Destination (OD) matrices and splits them into single vehicle trips.
- DUAROUTER: Computes vehicle routes using a shortest path algorithm (Dijkstra [50] or A* [94]). It can also be iteratively called to perform Dynamic User Assignment (DUA).
- JTRROUTER: Computes routes based on traffic volumes and junction turning ratios.
- DFROUTER: Computes routes by using data from induction loop sensors.
- MAROUTER: It is used to compute a macroscopic user assignment from OD matrices, trip files or route files.
- ACTIVITYGEN: Reads the definition of a population and computes mobility wishes for its members.

SUMO implements several car-following models, including the one developed by Krauß in [125], an extension of the Gipps model [81], and the lane change model proposed by Krajzewicz in [121].

Vehicle type definition in SUMO includes acceleration, deceleration, vehicle length, empty space between vehicles, maximum speed, vehicle class, color, emission class, shape, driver's impatience, person capacity, etc. It allows the simulation of multimodal traffic, including sedans, vans, trucks, bicycles, motorbikes, public transport, trains, pedestrians, and recently electric vehicles. Dynamic User Assignment (DUA) can be done by iteratively computing the approximate Dynamic User Equilibrium (DUE) as proposed by Gawron in [78]. Some of these features are shown in Figure 4.5.

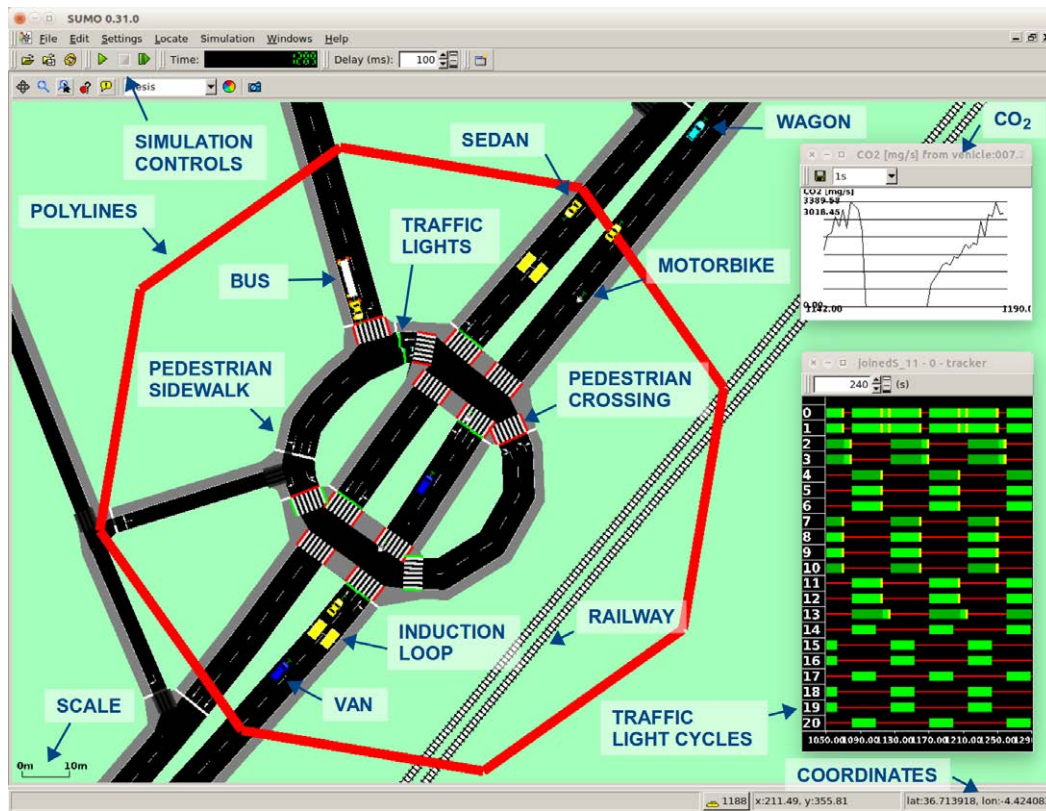


Figure 4.5: Some of the features of SUMO.

In SUMO, road networks can be manually defined by positioning junctions (nodes) connected by streets (edges) or be imported from OpenStreetMap. Additionally, NETCONVERT is able to import road networks from VISSIM, MATSim, VISUM, OpenDRIVE, etc. Street definition in SUMO includes number of lanes, street width, vehicle classes permitted/forbidden, priority, maximum speed, street name, sidewalk width, etc. Moreover, time schedules of traffic lights are automatically generated and can also be manually defined and adjusted during the simulation using programs.

SUMO allows external control of vehicle interactions and the simulation itself by using the Traffic Control Interface (TraCI) [236]. Retrieving simulation values online, as well as modifying traffic lights cycles, closing streets, etc., can be done by an external controlling

program by using TraCI. As TraCI uses a client/server architecture and there are several interfaces available written in Python, Java, C++, etc. We have used the TraCI Python interface included into the SUMO package.

Finally, SUMO can generate several output files on demand, normally written in eXtensible Markup Language (XML) format. The most used are:

- **Summary:** Contains the number of vehicles that are loaded, inserted, running, waiting to be inserted, have reached their destination, etc., in each simulation step.
- **Trip Info:** This output file contains the information about each vehicle's departure time, travel time, arrival time, distance traveled, departure and arrival lanes, gas emissions and fuel consumption, etc.
- **Emissions:** A really big file containing the gases emitted, noise generated, and fuel and electricity consumed, for every vehicle and simulation time step.
- **Traffic Light States:** It contains the state of each traffic light (green, yellow, or red) in each simulation step for the current program.
- **Induction Loop Detectors:** This output file contains the number of vehicles that have completely passed the detectors placed in the streets within each desired interval.
- **Edge and lane emissions and noise:** Contains edge/lane-based vehicular pollutant emission and noise.
- **Raw vehicle positions dump:** Another huge file containing all vehicle positions, speeds, etc., over time.

SUMO calculates the emission model according to the Handbook Emission Factors for Road Transport (HBEFA) [98] or Passenger car and Heavy duty vehicle Emission Model (PHEM) [96] standards. Formerly, it used the HBEFA v2.1 emission data base, however, the last version supported (3.1) is recommended, especially due to the correspondence between its classes and the European emission standards [65].

The use of an emission class in a vehicle's definition, e.g. HDV_D_EU3 for diesel driven heavy duty vehicle Euro norm 3, provides a set of constants internally used by SUMO to calculate the vehicle's emissions taking into account, not only its acceleration/deceleration in each simulation step but also the terrain slope among others.

The calculated edge/lane noise output is based on Harmonoise [186] and the electricity-consumption model is described in [126].

4.3.2 Building Mobility Scenarios with SUMO

In this PhD thesis we work with scenarios based on realistic maps imported from OpenStreetMap (OSM). To build each case study we first select the desired area and then, using the Java OpenStreetMap (JOSM) utility, extra objects are removed, including buildings, POIs, car parks, pedestrian crossings, etc., which are not useful for our experiments and may be misinterpreted by SUMO. Finally, the map is imported into SUMO by using the

NETCONVERT utility and the vehicles' flows between origin and destination are added (Figure 4.6).

Usually, each flow consists of several routes between the same origin and destinations, so that each vehicle has different alternatives for its trips. However, there are scenarios in which it is interesting to study what happens if all vehicles are taking the shortest (fastest) routes.

These flows, generated by the DUAROUTER utility, are called the experts' solution from SUMO in this PhD thesis. We use different cost functions such as travel time but also different gas emissions, fuel consumption and noise emission in order to obtain several routes even between the same origin and destination points as DUAROUTER uses the Dijkstra [50] algorithm to calculate routes.

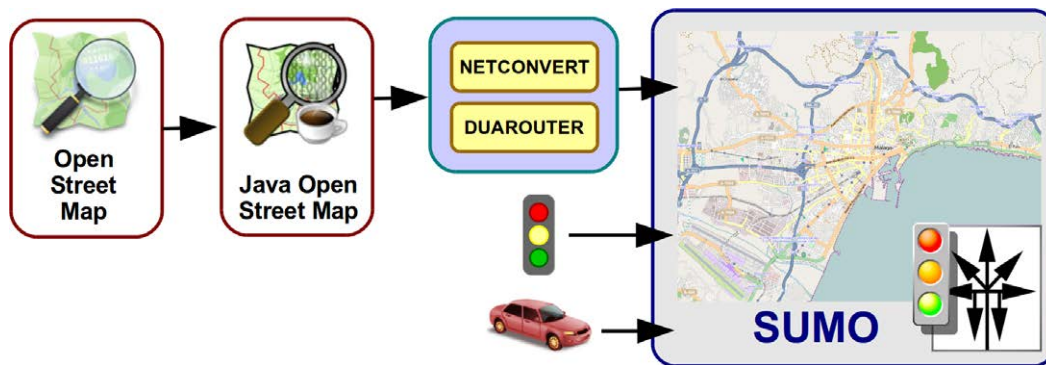


Figure 4.6: Scenario building schema.

Chapter 5

Facing Technology Gaps: Incomplete Maps and Data

In this chapter we present the Flow Generator Algorithm for calculating realistic traffic flows for traffic simulators. We start with an original map from OpenStreetMap and traffic data collected at different measurement points, published by the city's authorities, to produce a model consisting of the simulation map and a series of traffic flows (routes + vehicles) which match the real number of vehicles at those streets. This is extremely useful in practice, since no city has all flows for cars (just some sensors that measure them), while Smart Mobility services often need such flows. This is possible thanks to the use of evolutionary algorithms for such a complex task and the SUMO traffic simulator to evaluate the generated scenarios. We have tested our proposal on two geographical areas of the city of Malaga, comprising different map sizes, number of sensors and vehicles. Our algorithm, as well as the realistic scenarios generated by using it, can be used as the basis for other research approaches, especially those focused on road traffic optimization.

5.1 Introduction

There are several ways of addressing a real world problem. Some of them are based on mathematical models used to generate possible candidate solutions or to evaluate possible sets of solutions [217]. Among generative models we can find mathematical models such as linear, integer, dynamic, nonlinear programming, differential equations, network flow models, decision analysis, number theory, tabu search, genetic algorithms, fluid dynamics, and game theory [196]. While evaluative models include queueing models, queueing network theory, Petri nets, decision models, data envelope analysis, simulation, and perturbation analysis [196]. Due to the complexity of the problems we are solving in this PhD thesis, we will use evolutionary techniques and a simulation environment to conduct our experiments. The former is to address problems made of hundreds of variables, and the latter to allow us to use a virtual world where the impossible has become possible.

Traffic simulators have been frequently used in the last decade to validate different research approaches involving not only mobility issues, but also other different disciplines.

Some examples of these are traffic light optimization [10], intermodal traffic systems in disaster management [61], reducing the required total vehicle fleet size [28], alternative routes for preventing traffic jams [203], evacuation planning [73], optimization of transmission of live on-road videos [48], taxi dispatching [141], vehicle platooning [113], among others.

A mobility scenario is mainly composed of the map of a city (including streets, roundabouts, turn restrictions, traffic lights, etc.) and its traffic flows. These traffic flows are obtained from a origin-destination matrix (OD-matrix) where the travel demands between vehicles' origin and destination are specified. Since it is almost impossible to obtain data for an entire large city so as to estimate the OD-matrix, flows must be generated based on measurements from sensors.

In this chapter we study a new methodology to build realistic traffic flows, based on evolutionary techniques, to be used in mobility scenarios supported by maps from OpenStreetMap and vehicular data obtained from sensors placed around the city's streets. We feed these inputs into our Flow Generator Algorithm (FGA) and using an evolutionary algorithm (EA) and a traffic simulator, SUMO in our case, we obtain a realistic simulation model. This model contains traffic flows calculated according to an estimated OD-matrix, so that the number of vehicles at each measurement point matches the real one. The resulting map can be directly used by researchers to test their Smart Mobility proposals and other research work involving road traffic simulations.

There are many studies (see survey in [22]) which focus on the estimation of origin-destination matrix based on traffic counting locations. They can be static [131, 136] or dynamic [97, 160]. However, these algorithms assume that all link costs are available, which may not be true in practical situations such as our case study. Unfortunately, authors do not usually detail the scalability of their algorithms for larger networks, which is a key issue for the interest in their solutions.

In [252] a Hopfield Neural Network (HNN) model is used to estimate the urban origin-destination distribution matrix. The author claims that due to the ability of quick computation, parallel distributed processing and hardware realization of neural networks, it is possible to overcome the difficulties of mathematical optimization models. He finds the global optimal solution to the problem and experiments on a graph made of just five nodes representing the same number of zones. To the contrary, our method focuses on individual streets rather than zones (finer grain, higher realism) and we need to route vehicles via individual streets.

An open-source software, called TrafficModeler, is presented in [170]. This program implements a traffic definition model consisting of a set of layers placed over a road network. By using those layers it is possible to represent specific traffic patterns associated with different attributes. Additionally, traffic flows can be obtained from virtual populations based on demographic data (i.e. transportation between home, school and work). This tool for modeling traffic flows differs from our proposal in that it cannot be applied when the only source of data is the number of vehicles measured by sensors.

Finally, there are two utilities included in the SUMO [123] software package called ACTIVITYGEN and DFROUTER. The former computes the mobility wishes for a group of citizens matching a map, while the latter uses values from induction loops (sensors) to compute vehicle routes. ACTIVITYGEN is quite similar in some aspects to the aforemen-

tioned article, analyzed in this section, although it does not provide a graphical user interface. DFROUTER is a tool that may be used in the same way as FGA, however, it assumes that the map is completely covered by sensors, especially on its borders, and it requires the exact timestamp in which vehicles were detected and their speed in all the measurement points. None of these options are suitable for the problem we are solving as they cannot be applied to calculate the traffic flows based on just the number of vehicles counted by each sensor.

There exist several methodologies for obtaining a valid city map, many of them are based on importing it from OpenStreetMap [91, 169, 180]. While data regarding the number of vehicles on city streets is being collected by different methods [51]. Some of them use Wireless Sensor Networks (WSN) [229], magnetic sensors [38], or even Wi-Fi and Bluetooth technology [68]. Furthermore, there are several studies (see [128] for a survey) about the allocation of sensors on city streets, taking into account aspects such as maximum flow coverage, and route coverage.

FGA is different from those discussed in this section as we can address real, large maps, calculate the traffic flows by using an evolutionary technique, especially useful when there are just a few sensors, and provide a simulation model which matches not only the urban layout of the city but also the real number of vehicles at the measurement points.

5.2 Flow Generator Algorithm (FGA)

The Flow Generator Algorithm (FGA) [197, 201] is a new strategy to generate traffic distributions in a city by using the data previously collected from sensors which count vehicles in a few streets. In most of the cities the available data is scarce and many of them cannot offer more than a few points of sensing. As this limitation comes from the very nature of the problem (and belongs to the realm of activities that city managers do, not us as scientists) we just tried to deal with this restriction and proposed a strategy to overcome it.

FGA is based on an evolutionary algorithm (EA) especially adapted to work with the difficulties that are present in this problem, such as high complexity due to the high number of vehicles and large scenarios, long evaluation times, and the high probability of traffic jams occurring in a city scenario when the number of vehicles moving through its streets increases.

Formally, let $\vec{v}^* = (v_1^*, \dots, v_N^*)$ be a vector containing the values collected from N sensors in the real city, and $\vec{v} = (v_1, \dots, v_N)$ a vector containing the values obtained from the evaluation of the city map. Our objective is to minimize the error $\vec{e}_i = |\vec{v}_i^* - \vec{v}_i|, i \in \{1, \dots, N\}$ by modifying the vehicle flows $f = (f_1, \dots, f_M)$ in the city.

In short, by looking for appropriate flows (decision variables) we compute estimated flows on a simulator with the goal that they match real measured ones in the city where they are available. The set of flows also contain a subset of proposed ones for the streets where no measurements are available at all, thus allowing the researcher to further study the city by using existing and approximated flows for almost all the streets.

The FGA architecture is presented in Figure 5.1 where the inputs of the algorithm are on the left, the processing carried out by the FGA in the center, and the output achieved, i.e. SUMO's real map imported from OpenStreetMap plus the vehicle flows, on the right.

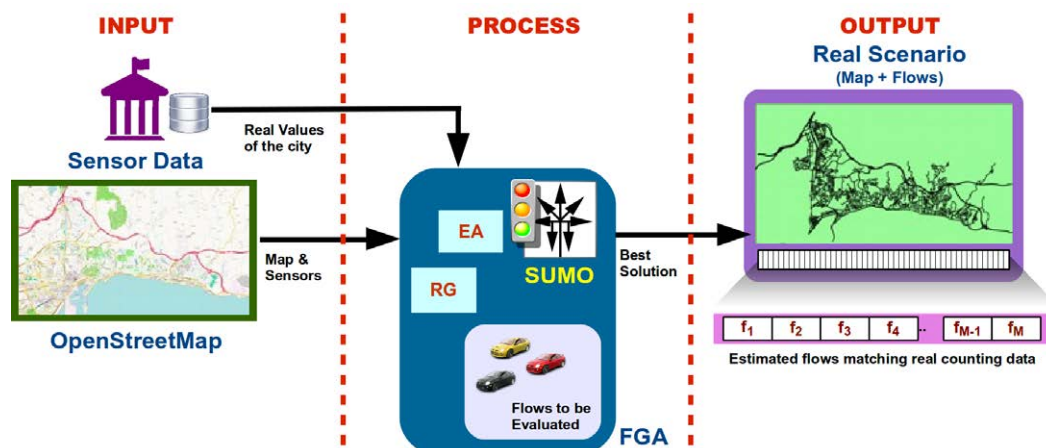


Figure 5.1: Architecture of FGA. The inputs are the map from OpenStreetMap and data from sensors. The output is a realistic scenario consisting of the simulation map plus the calculated routes and vehicles.

We have chosen SUMO [123] because it is a widely used open source, microscopic, multimodal traffic simulator. However, other traffic simulators could be used as our proposal is not uniquely targeted at SUMO. This makes FGA generally useful for the domain, suitable for many approaches, different technologies, and researchers' preferences.

Regarding the measurement point problem, it has already been addressed by several authors (see [190, 233, 244]). In this study we focus on using the already existing data which are influenced by a number of factors, such as weather conditions, holidays, and special events. Despite the difficulty of knowing the true demand, every single hour, throughout the year, we wish to offer here a valid, average scenario which is closer to reality than just adding random traffic, as it can be seen in many published studies. Moreover, to make our scenarios still more realistic, we have included a warm up period before counting vehicles so that the city already has vehicles on its streets when the study begins. Note that the number of vehicles in each flow includes those vehicles filling the city during the warm up period.

A diagram of FGA is depicted in Figure 5.2. It is divided into two stages: The *Setup Stage* which calculates the base scenario where each sensor is fed (covered) by at least one route carrying vehicles, to be measured; and the *Optimization Stage*, which optimizes the number of vehicles in each route in order to minimize the difference (error) between the real number of vehicles measured in the city and the number in the simulated scenario.

At the very beginning of the execution of FGA, in the *Setup Stage*, the different routes are generated from the origins placed at the borders of the area analyzed, to the destinations, also placed at the borders. As a result, we obtain different routes throughout the entire area (Figure 5.3a) which are generated by our Route Generator (RG). Note that RG does not provide all the possible routes between two points as it relies on different weight values for streets and the Dijkstra algorithm [50], and some streets might not be on an optimum route.

Then, if each sensor is covered (has counted at least one vehicle), the *Setup Stage* ends because the calculated flows are covering all the streets with sensors. If not, a new origin is placed in a street before reaching one of the sensors (randomly picked) have not yet registered any vehicle (Figure 5.3b) and the routes are again calculated by RG.

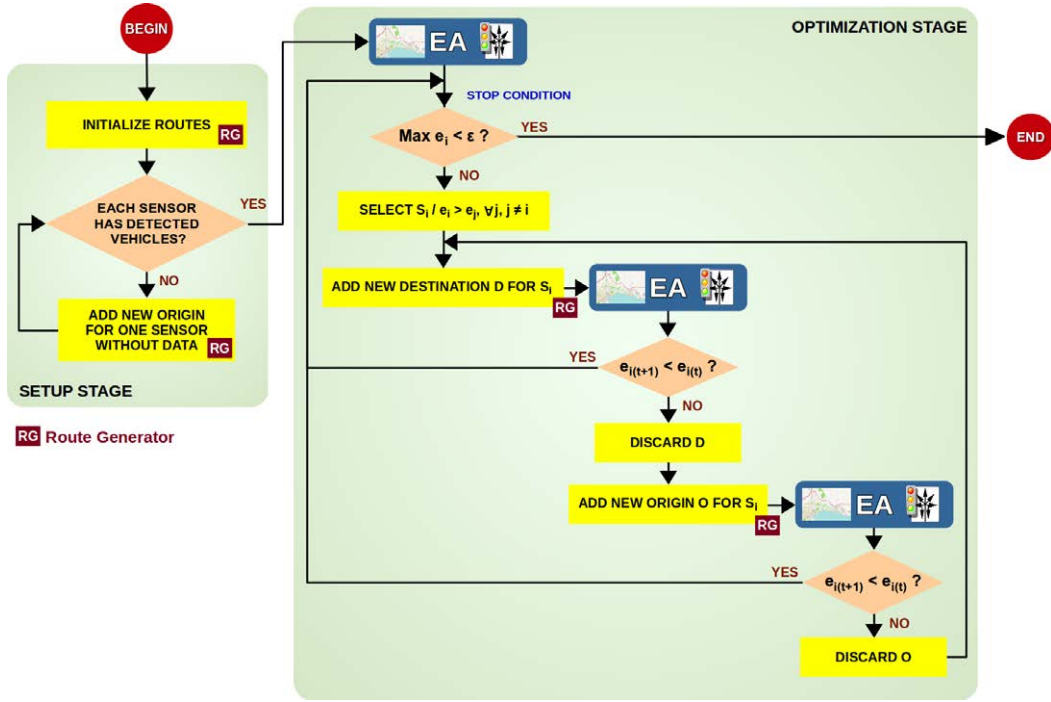


Figure 5.2: Flow Generator Algorithm (FGA). In the *Setup Stage* the initial routes are calculated by the RG in order to cover all the available sensors. Then, in the *Optimization Stage* the number of vehicles for each flow is calculated by the EA, using RG to generate new routes if needed.

The *Optimization Stage* begins when all sensors are measuring vehicles from at least one route. The first step consists in optimizing the current set of routes by using our EA to calculate the optimal number of vehicles in each route so that the measurements done by each sensor are closer to the real values. After that, the *Stop Condition* is checked, i.e. if the maximum error $E = \max\{e_i\}$ is smaller than the maximum desired error ϵ . Should it be fulfilled, the FGA ends and the current set of flows is returned as the solution.

If there is at least one sensor S_i whose error e_i is greater than or equal to ϵ , the next step will consist in selecting the sensor whose error is the biggest, adding a new destination D to the map for it, and running RG again. By doing so, we force the generation of new routes which contributes to incrementing the number of vehicles detected by the sensor S_i (Figure 5.3c).

After that we need to optimize the traffic flows again, using our EA, and then check if the error for S_i has been reduced after the optimization process ($e_{i(t+1)} < e_{i(t)}$). If it is true, the last optimization process was successful, and the *Stop Condition* will be checked again, closing the loop. If not, the recently added destination D is discarded and a new origin O is tried instead.

Adding a new origin O makes RG add new routes from O to the available destinations which will increase the number of vehicles measured by S_i (Figure 5.3d). If this fails, origin O is discarded and a new, different destination D is selected and the process is repeated. Otherwise, if e_i has been reduced, the *Stop Condition* is checked again, and depending on the

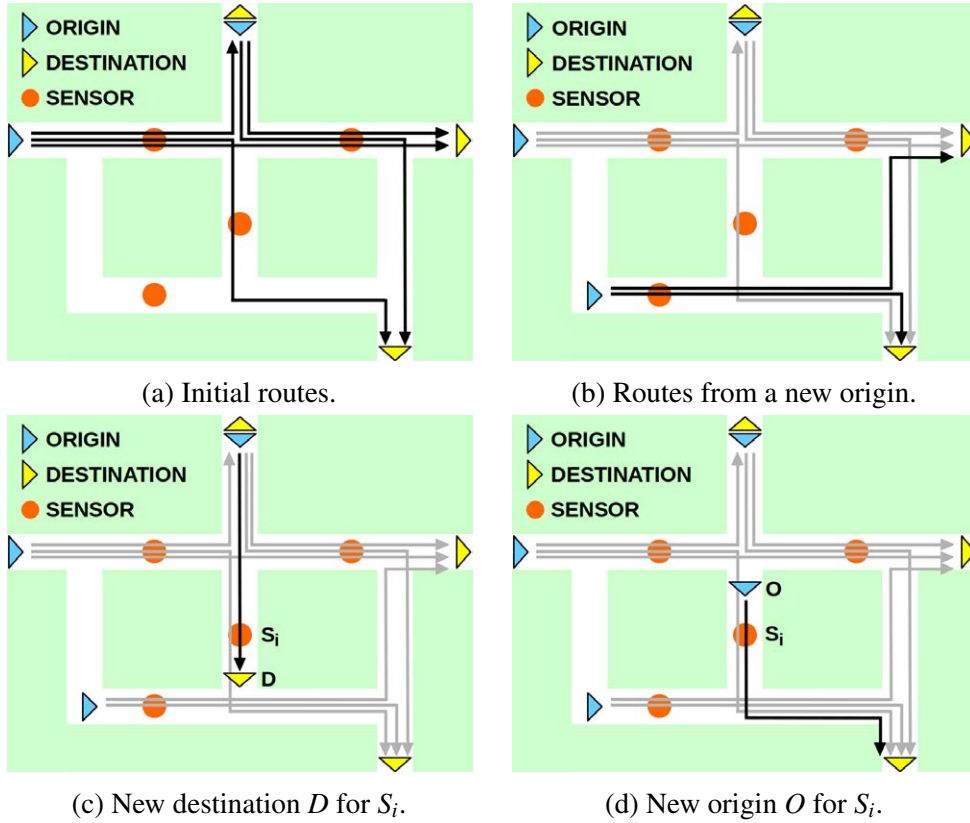


Figure 5.3: Different phases when adding routes. In Figure 5.3a, there is one uncovered sensor. In Figure 5.3b, a new origin has been added. In Figure 5.3c, a new destination D is tested for S_i . And in Figure 5.3d, an alternative origin O is used for increasing the number of vehicles in S_i .

new maximum error value, FGA could end at that point or start a new loop, optimizing the scenario by reducing the error of a new selected sensor.

Having explained how FGA works we move on to describe the internals of RG and EA.

5.2.1 Route Generator (RG)

The RG is invoked several times during the execution of FGA. It takes a road traffic scenario, imported from OpenStreetMap, consisting of streets, roundabouts, traffic lights, junctions, etc., and generates routes from the given origins and destinations. In our experiments we have used the DUAROUTER utility provided with SUMO which generates routes using the Dijkstra algorithm [50], although other traffic simulators and utilities can be used. Additionally, RG discards routes not involving sensors as they do not affect the result of the optimization. Finally, RG also ignores routes that have a high impact on results, i.e. those which affect many sensors, as according to our preliminary tests, they make the scenarios harder to optimize.

The RG, presented in Algorithm 5.1, has five parameters: O and D which are the set of origins and destinations of the routes to be generated; $BanList$ which is a list of destinations unreachable from each origin (urban layout) or those which make unrealistic loops; $Attributes$

Algorithm 5.1 Route Generator (RG).

```

function RG( $O, D, BanList, Attributes, MaxS$ )
     $R = \emptyset$  ▷ Initialize the set of routes
    for all  $o \in O$  do
        for all  $d \in D$  do
            if  $d \notin BanList(o)$  then ▷ Not in the ban list
                for all  $a \in Attributes$  do
                     $r = dijkstra(o, d, a)$  ▷ New route
                     $n = sensors(r)$  ▷ Number of sensors in the route
                    if  $0 < n \leq MaxS$  then
                         $R = R \cup route(o, d)$  ▷ Adds the new route
                    end if
                end for
            end if
        end for
    end for
    return  $R$  ▷ Returns the new set of routes
end function

```

which is a set of the different attributes (weights) to be used in the consecutive runs of the Dijkstra algorithm, so that we get several routes for the same pair origin and destination (diversity); and $MaxS$ which is a positive integer number indicating the maximum number of sensors allowed in a route (simplicity).

First, the algorithm initializes the empty set of routes R . Second, for all origins o and destinations d whereas d is not in the forbidden destinations for the current origin o , the algorithm calculates a new route using the Dijkstra algorithm (through DUAROUTER in our case). Each new route from o to d is calculated using a different attribute in the given attribute list $Attributes$ (travel time and noise in our experiments).

Then, the number of sensors n included in the route is calculated and if n is into the desired range $(0 - MaxS]$, the new route is added to the set R . We only add routes that affects the sensors' readings as we do not wish to raise the complexity of the problem to be addressed by the EA later. Additionally, for the same reason, we wish to work with routes that make small changes to the sensors' measurements. As a consequence, we have limited the maximum number of sensors in each route.

After visiting all the combinations of origins and destinations the algorithm ends returning the new set of routes R . Now it is time to assign vehicles to each route that will be addressed by our EA every time is needed (Figure 5.2).

5.2.2 Evolutionary Algorithm (EA)

An EA is proposed to calculate the number of vehicles in each route so that the differences between the real values measured by the sensors and the ones obtained from the simulated scenario can be minimized.

In Algorithm 5.2 the pseudocode of our (10+2)-EA is presented. It has a population of ten individuals ($\mu = 10$) and generates a two-individual offspring per generation ($\lambda = 2$). This configuration was chosen because our evaluation function (described later) is a costly one as it requires a simulation to calculate an individual's fitness value. Furthermore, each simulation takes up to about 60 seconds to finish when evaluating our most complex scenarios.

The EA has the following parameters: crossover probability (P_C), initial mutation probability (π_1), final mutation probability (π_2), threshold (θ) for commuting from π_1 to π_2 , α and β used by the mutation operator, the minimum and maximum number of vehicles per route (V_{MIN} and V_{MAX}), maximum admitted error (ϵ_M), initial fill (I_F), and the initial value I_V .

First of all, an initial population is created by generating ten new individuals consisting of a vector of integers (vehicles in each flow). We randomly assign vehicles when creating the initial population, under the following conditions: i) the vehicles are only assigned to a reduced number of routes given by the *Initial Fill* parameter (I_F); and ii) the initial maximum number of vehicles in each route is given by the *Initial Value* parameter ($V_{MIN} \leq I_V \leq V_{MAX}$). After generating an individual, the configuration represented by its flows is simulated in order to check if there are any sensors exceeding the maximum deviation ϵ_M . If so, one of those sensors is randomly selected and also one of the routes sending vehicles to it. Then the number of vehicles in that route is set to V_{MIN} , and then the individual is evaluated once more. This process is repeated until all the sensor values are under the maximum error ϵ_M .

Having generated the initial population, the main loop begins and it will continue until the *Termination Condition* holds. In the main loop, the parents are selected by using binary tournament [87]. Then, the recombination operator is applied to obtain the offspring. We have used a Uniform Crossover [87] here with a crossover probability P_C . Now, the offspring is changed by the Mutation Operator. We have tested different mutation strategies which are analyzed later in this chapter. Before applying the replacement operator, the new individuals are evaluated. We have chosen an elitist replacement [86] because we are working with a small population and long evaluation times.

Algorithm 5.2 Evolutionary Algorithm (EA).

```

procedure EA( $P_C, \pi_1, \pi_2, \theta, \alpha, \beta, V_{MIN}, V_{MAX}, \epsilon_M, I_F, I_V$ )
   $t \leftarrow 0$ 
   $P(0) \leftarrow \text{createPopulation}(I_F, I_V)$  ▷ P = population
  while not terminationCondition() do
     $parents \leftarrow \text{selection}(P(t))$  ▷ Binary tournament
     $offsp \leftarrow \text{UniformCrossover}(P_C, parents)$ 
     $offsp \leftarrow \text{MutationOperator}(\theta, \pi_1, \pi_2, \alpha, \beta, V_{MIN}, V_{MAX}, offsp)$ 
     $\text{evaluateFitness}(\epsilon_M, offsp)$ 
     $P(t+1) \leftarrow \text{replace}(offsp, P(t))$  ▷ Elitist replacement
     $t \leftarrow t+1$ 
  end while
end procedure

```

Representation

Our representation consists of a vector of integers (Figure 5.4). Each value corresponds to the number of vehicles arriving by origins O_i to destinations D_j of each route R_m in the simulated scenario which are represented by the traffic flows f_m .

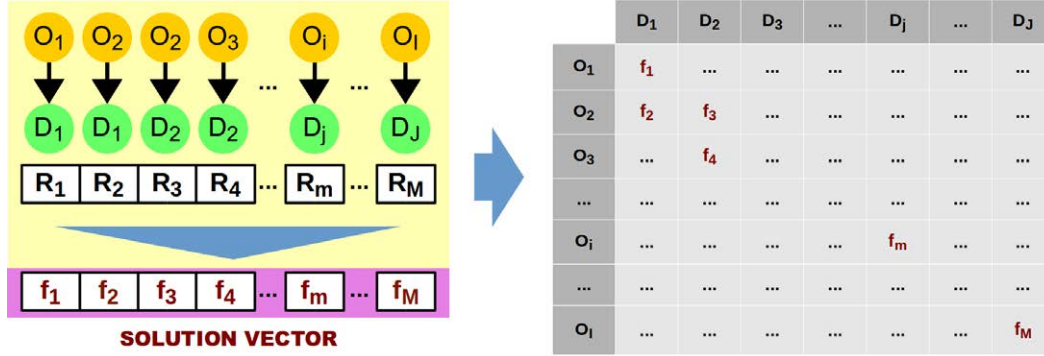


Figure 5.4: Problem Representation: Solution vector of M integer values corresponding to the number of vehicles injected into each flow of the scenario. Note that we are using the EA to fill an OD-Matrix whose rows and columns were calculated by RG.

We have restricted the range of vehicles between V_{MIN} and V_{MAX} especially to reduce the complexity of the problem by limiting the possible values for each position of the solution vector. Additionally, we have observed that if there are many vehicles in a route, they end up in traffic jams, so this is another reason for keeping this number as low as possible. The total number of flows, M (integers in the solution vector), depends directly on the number of routes generated by the RG (Dijkstra algorithm and urban layout).

Evaluation Function

The evaluation function assigns a numeric value to an individual according to the configuration of the vehicle flows (Equation 5.1). It calculates the square value of the difference between the real values (\vec{v}^*) measured in the city and the ones (\vec{v}) collected during the simulation using the flows represented by the individual under evaluation.

$$F(\vec{v}) = \begin{cases} \sum_{i=1}^N \frac{(\vec{v}_i - \vec{v}_i^*)^2}{\vec{v}_i^*} & \text{if } C(\vec{v}) \leq \epsilon_M, \\ \infty & \text{if } C(\vec{v}) > \epsilon_M. \end{cases} \quad (5.1)$$

The fitness value of an individual is calculated by applying the evaluation function so that the numeric value of $F(\vec{v})$ is the summation of the squared values previously calculated divided into the corresponding real value for each sensor. However, if at least one sensor i has exceeded the real value \vec{v}_i^* by a percentage greater than ϵ_M ($C(\vec{v})$ in Equation 5.2) we apply a penalization of a large constant value in the algorithm because we are minimizing, so the lower, the better.

$$C(\vec{v}) = \max \left(\frac{\vec{v}_i - \vec{v}_i^*}{\vec{v}_i^*} \right), i \in \{1, \dots, N\} \quad (5.2)$$

It does not seem appropriate to fix each individual during the evaluation as we did during the generation of the population. It is a very costly process that considerably increases the running time of the EA. Consequently, we decided to penalize inviable solutions which ended up in traffic jams.

Operators

On the one hand, we have used the well-known standard Uniform Crossover [87] as the recombination operator, whose crossover probability P_C has been calculated according to the parameterization shown in Section 5.4. On the other hand, we have evaluated three different mutation operators in order to improve the results achieved by the EA: Blind Mutation (BM), Flow Focused Mutation (FFM), and Sensor Focused Mutation (SFM).

First, we implemented the Blind Mutation (BM) operator where the number of vehicles in each flow f_i might be modified depending on the mutation probability P_M (Algorithm 5.3).

Algorithm 5.3 Blind Mutation (BM).

```

function BM( $\vec{f}, \delta, P_M, V_{MIN}, V_{MAX}$ )
  for all  $f_i \in \vec{f}$  do
    if  $rnd() < P_M$  then
      if  $rnd() < 0.5$  then                                ▷ Equiprobable
         $f_i \leftarrow \max(f_i - \delta, V_{MIN})$                 ▷ Decrement
      else
         $f_i \leftarrow \min(f_i + \delta, V_{MAX})$                 ▷ Increment
      end if
    end if
  end for
  return  $\vec{f}$ 
end function

```

In BM, the decision on whether a flow is to be incremented or decremented is equiprobable and the number of vehicles to be added to or subtracted from each flow is given by δ as described in Equation 5.3.

$$\delta = \alpha - \sqrt{\beta} + \sqrt{\beta \cdot \min_i \{fitness_i\}}, \forall i \in P \quad (5.3)$$

We can see that δ depends on the best individual (minimum fitness) in the population and two parameters: α and β . The former is used to define the minimum value of δ and the latter is to control the incremental rate of it. The final value of f_i , after being modified, is kept in the valid range $[V_{MIN}, V_{MAX}]$.

Second, we wished to evaluate a more complicated operator which takes into account the number of vehicles that are needed in each flow f_i , to reach the desired value in one of the sensors s visited by the corresponding route. Consequently, we have designed the Flow Focused Mutation (FFM) operator as described in Algorithm 5.4.

Algorithm 5.4 Flow Focused Mutation (FFM).

```

function FFM( $\vec{f}, \vec{v}, \vec{v}^*, \delta, P_M, V_{MIN}, V_{MAX}$ )
  for all  $f_i \in \vec{f}$  do
    if  $rnd() < P_M$  then
       $s \in S(f_i)$  ▷ Randomly picks a sensor  $s$  from  $i$ 
      if  $v_s - v_s^* < 0$  then ▷ More vehicles for sensor  $s$ 
         $f_i = \min\{f_i + \delta, V_{MAX}\}$ 
      end if
      if  $v_s - v_s^* > 0$  then ▷ Less vehicles for sensor  $s$ 
         $f_i = \max\{f_i - \delta, V_{MIN}\}$ 
      end if
    end if
  end for
  return  $\vec{f}$ 
end function

```

As in the previous operator, the flows to be modified are selected according to probability P_M . Then, one sensor s is randomly selected from the sensors affected by the current flow. If the number of vehicles v_s measured by s is under the desired value v_s^* , the number of vehicles in f_i is increased, and vice versa. Note that the value of δ for this operator is calculated in the same way as for the BM operator.

Finally, a different approach was followed to design the third mutation operator under evaluation, as shown in Algorithm 5.5). With the Sensor Focused Mutation (SFM) operator we wished to fine tune the number of vehicles we were adding to or subtracting from each flow f_i . We then used a calculated value δ' , based on the error ε_i corresponding to the difference between v_s and v_s^* .

First, we randomly select a sensor s and then a flow f_i which modifies the number of vehicles counted by s . Then, we calculate δ' as stated and modify the number of vehicles in f_i accordingly, always taking into account the valid range of values. We have divided δ' into twice the number of flows to take small steps towards the EA's convergence depending on the problem size. Note that δ' could be also a negative number.

Algorithm 5.5 Sensor Focused Mutation (SFM).

```

function SFM( $\vec{f}, \vec{v}, \vec{v}^*, S, V_{MIN}, V_{MAX}$ )
   $s \in S$  ▷ Randomly picks one sensor from  $S$ 
   $f_i \in \vec{f} : s \in S(f_i)$  ▷ Randomly picks one flow for  $s$ 
   $\delta' \leftarrow \frac{v_s - v_s^*}{2 \cdot \text{length}(\vec{f})}$  ▷  $\delta'$  depends on the error  $\varepsilon_s$ 
   $f_i = \min(\max(f_i + \delta', V_{MIN}), V_{MAX})$ 
  return  $\vec{f}$ 
end function

```

Having defined the components of the EA we move on to describe the case studies and the parameterization done. This is a much needed step prior to using any non-deterministic algorithm on a given problem.

5.3 Case Studies

We address the flow calculation for two different geographical areas of Malaga, Spain. We have imported them from OpenStreetMap into the SUMO traffic simulator by using the utility provided by SUMO (Figure 5.5).

We worked first with a small area corresponding to the city center (*Alameda Principal*) which encompasses an area of about 3 km². In this area the local council has placed 12 sensors on the city's streets for measuring the road traffic and their data are publicly available on its web page [13]. Using this valuable resource we were able to calculate the average number of vehicles for the fourth quarter of 2014 and the first of 2015, for Working Days (WD), Saturdays (SAT), and Sundays (SUN). After those dates, works to build the subway began which reduced the number of sensors available and severely altered the road traffic in the area, which would have made our study incomplete.

For our second case study, we dealt with a bigger area, the whole of eastern Malaga, which comprises an area of about 32 km². In this more complex study, there are 23 sensors (11 new plus the former 12 of the small area). We studied the traffic during the same quarters as in the small area for the same reasons.

We have named the three scenarios of 2014 as *12.2014.WD*, *12.2014.SAT*, and *12.2014.SUN*, of the first case study (12 sensors), and *12.2015.WD*, *12.2015.SAT*, and *12.2015.SUN* the three of 2015. Additionally, we have labeled *23.2014.WD*, *23.2014.SAT*, *23.2014.SUN*, *23.2015.WD*, *23.2015.SAT*, and *23.2015.SUN* the scenarios of the second case study (23 sensors).

5.4 Parameterization

We have conducted the parameterization study by optimizing the scenario *12.2014.WD* in a computer cluster comprising four processors Intel Xeon E5-2670v3 @ 2.30 GHz (96 cores).

First, we have analyzed the Uniform Crossover operator in our EA. Concretely, we tested different crossover probabilities on the same problem instance by performing 300 independent runs of the EA (30 per probability value under test) as shown in Table 5.1.

We can see that 0.3 is the best ranked value (4.87) according to the Friedman test, followed by 0.9 (5.13), 0.4 and 1.0 (both 5.27). Although it is not the best ranked value, we have chosen 0.4 as probability value because it presents the lowest minimum and maximum values (better stability). A crossover probability value of 0.4 has also improved the robustness of the algorithm achieving by far the lowest standard deviation value. Note that the calculated Wilcoxon *p*-value between results of 0.3 and 0.4 is 0.813 which shows us that the distribution of their values are quite similar.

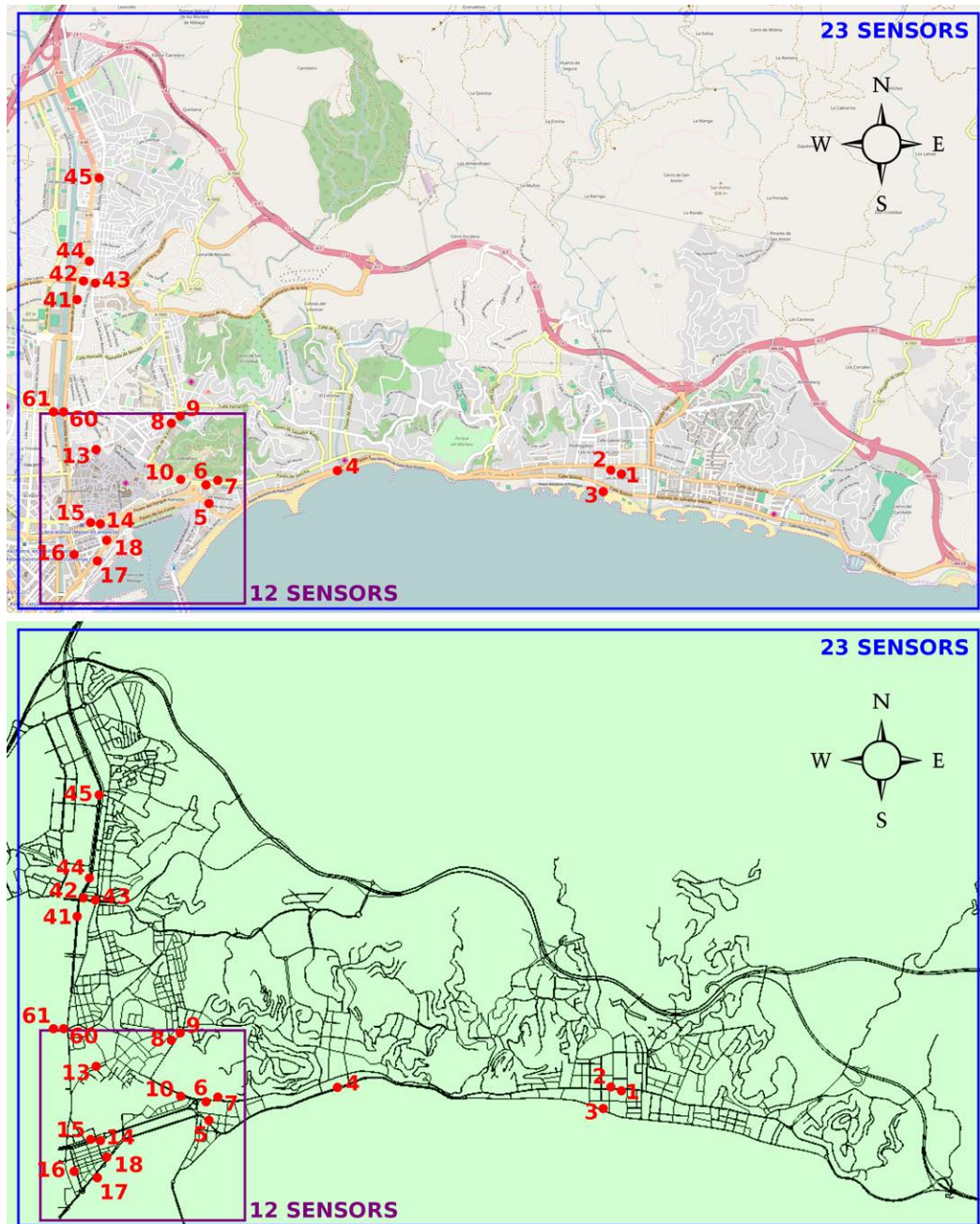


Figure 5.5: Two case studies in Malaga, imported from OpenStreetMap (upper picture) into SUMO (lower picture).

Regarding the mutation operator, we have conducted 30 independent runs of the EA using each mutation operator (90 in total) to obtain the results presented in Table 5.2. We can see there that BM presents the lowest fitness values and Friedman Rank. On the other hand, FFM and SFM despite being more complex and using more information about the problem, have not obtained good results. Consequently, we have chosen Blind Mutation (BM) as the

Table 5.1: Fine tuning of the Uniform Crossover Operator.

| P_C | Fitness | | | | Friedman Rank | Wilcoxon p -value |
|-------|--------------|--------------|--------------|--------------|---------------|---------------------|
| | Mean | StdDev | Min | Max | | |
| 0.1 | 3.558 | 1.529 | 1.659 | 8.435 | 5.70 | 0.600 |
| 0.2 | 3.731 | 1.592 | 1.717 | 8.554 | 5.47 | 0.688 |
| 0.3 | 3.618 | 2.038 | 1.319 | 11.276 | 4.87 | — |
| 0.4 | 3.482 | 1.135 | 0.989 | 6.147 | 5.27 | 0.813 |
| 0.5 | 3.969 | 1.883 | 2.011 | 9.064 | 5.60 | 0.349 |
| 0.6 | 3.810 | 1.859 | 1.420 | 8.964 | 5.33 | 0.734 |
| 0.7 | 3.966 | 1.722 | 1.119 | 8.737 | 6.03 | 0.102 |
| 0.8 | 4.277 | 2.236 | 2.025 | 9.956 | 6.33 | 0.229 |
| 0.9 | 3.401 | 1.400 | 1.734 | 7.929 | 5.13 | 0.673 |
| 1.0 | 3.630 | 1.814 | 1.304 | 8.014 | 5.27 | 0.719 |

Table 5.2: Fine tuning of the Mutation Operator.

| Strategy | Fitness | | | | Friedman Rank | Wilcoxon p -value |
|----------|-------------|-------------|-------------|-------------|---------------|---------------------|
| | Mean | StdDev | Min | Max | | |
| BM | 2.83 | 1.43 | 0.83 | 7.02 | 1.23 | — |
| FFM | 6.42 | 4.09 | 1.11 | 17.98 | 1.80 | 0.00 |
| SFM | 14.70 | 4.13 | 7.43 | 21.98 | 2.97 | 0.00 |

mutation operator. Note that we have also calculated the Wilcoxon p -value to confirm that the results are statistically significant.

With respect to the rest of parameters, we have experimentally set the V_{MIN} and V_{MAX} values so that the former was set to 10 and the latter to half the maximum number of vehicles in a sensor. This V_{MIN} value guarantees that each flow visited by the algorithm has some vehicles running through it (take into account that the simulation has a warm up stage before beginning to count vehicles). The value of V_{MAX} is shown to be appropriate as it was never reached by the EA in our tests as we did not wish to restrict the possible solutions by limiting the number of vehicles in each flow (although we also wanted a reduced search space).

V_{MIN} and V_{MAX} are used with α and β (Equation 5.3) to calculate the value of δ according to the minimum (best) fitness value in the population. We show an example of curve in Figure 5.6 where $\alpha = V_{MIN}$ and $\beta = \frac{1}{V_{MAX}}(\frac{V_{MAX}}{5} - \alpha)^2$ so that when the best fitness of the population is equal to half the maximum number of vehicles in a sensor (V_{MAX}), the value of δ is equal to $\frac{V_{MAX}}{5}$.

We can see that initially, when the best fitness of the population is high, the number of vehicles (δ) to be added to/subtracted from the changing flows is also high. Then, when the population evolves to better (lower) fitness values, δ is lower as well, so that the algorithm better exploits the solutions found by taking small steps towards an optimal.

The rest of the parameters have been experimentally set to better explore the search space ($\pi_1 = \frac{5}{L}$, $\pi_2 = \frac{1}{L}$, and $\tau = 2.0$), to avoid overcharged scenarios full of vehicles stuck in traffic jams ($I_F = 10\%$ and $I_V = 10\%$), and to keep the sensor dependencies simple ($MaxS = 2$).

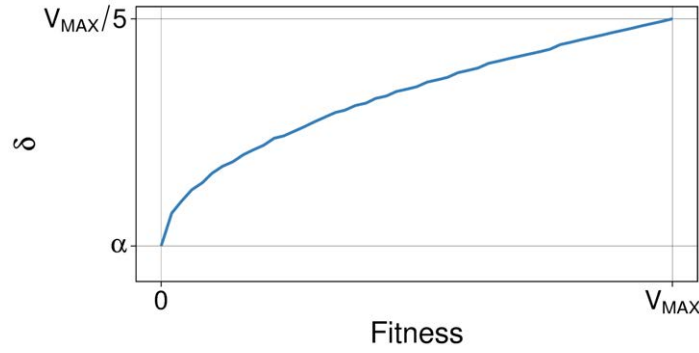


Figure 5.6: Evolution of δ vs. the fitness value of the best individual in the population.

Table 5.3 shows the chosen parameters of FGA including those used in the simulation and in EA. Note that the warm up time is different in the first case study (12 sensors) from the second (23 sensors) as we needed more time to populate a larger scenario with vehicles.

Table 5.3: Parameters of FGA. Brief description and values.

| | | |
|-----------------|--|---|
| | # Generations | 5000 |
| | Termination condition | $\varepsilon_s \leq 10\%, \forall s \in S$ |
| | Warm up time (seconds) | 600 (12 sensors) & 1200 (23 sensors) |
| | Simulation time | Warm up time + 1 hour |
| μ | # of individuals in the population | 10 |
| λ | Offspring size | 2 |
| P_C | Crossover probability | 0.4 |
| P_M | Mutation probability | π_1 if $\min(fitness_i) > \tau$, π_2 otherwise |
| π_1 | Initial mutation probability | $\frac{5}{L}$ |
| π_2 | Final mutation probability | $\frac{1}{L}$ |
| τ | Fitness threshold | 2.0 |
| V_{MAX} | Maximum number of vehicles in each route | $\frac{\max(\bar{v}_s^*)}{2}, \forall s \in S$ |
| V_{MIN} | Minimum number of vehicles in each route | 10 |
| α | First parameter of the δ formula | V_{MIN} |
| β | Second parameter of the δ formula | $\frac{1}{V_{MAX}} \left(\frac{V_{MAX}}{5} - \alpha \right)^2$ |
| ε_M | Maximum error rate in valid configurations | 10% |
| I_F | Initial percentage of routes with vehicles | 10% |
| I_V | Initial percentage of vehicles in each route | 10% |
| $MaxS$ | Maximum number of sensors in a route | 2 |

5.5 Results

Our experiments consisted in testing the FGA in different scenarios to validate it as a method to achieve traffic flows which match the real ones in the city. In the following sections, we first describe the results of the *Setup Stage* of FGA when generating the initial routes. After that, once all the initial routes have been set, we will proceed with the *Optimization Stage* where the number of vehicles in each route are calculated.

5.5.1 Setup Stage

As the street distribution is the same in each case study (only the number of vehicles differs) during the *Setup Stage* of FGA we used one scenario of 12 sensors and another of 23. The objective of this stage is to generate the initial routes to work with in the next stage, taking into account that each sensor needs at least one route with vehicles. The results obtained after calculating the initial routes are presented in Table 5.4. We can see that only two steps were enough to cover all the sensors in our small case study (12 sensors) (columns on the left). The first step showed that there were only 11 sensors covered by the initial 55 routes generated by RG. In the second step a new origin was added for the uncovered sensor (8) in order to generate an extra route during a new execution of RG.

A bigger map with more sensors as in our second case study requires more steps of the FGA during the *Setup Stage*. We can see in the columns on the right side of Table 5.4 that nine steps were performed by executing RG to generate routes to cover all 23 sensors. Initially, there were 100 routes, however, RG almost doubled this number to complete this *Setup Stage*.

Table 5.4: Optimization of both case studies using FGA (*Setup Stage*). We report the new origins added, the number of sensors measuring vehicles, and the number of routes in the scenario.

| Step | 12 sensors | | | 23 sensors | | |
|------|------------|-----------|----------|------------|-----------|----------|
| | O | # Sensors | # Routes | O | # Sensors | # Routes |
| 1 | - | 11 | 55 | - | 15 | 100 |
| 2 | 8 | 12 | 56 | 1 | 16 | 115 |
| 3 | | | | 2 | 17 | 127 |
| 4 | | | | 3 | 18 | 139 |
| 5 | | | | 6 | 19 | 157 |
| 6 | | | | 8 | 20 | 163 |
| 7 | | | | 9 | 21 | 179 |
| 8 | | | | 13 | 22 | 185 |
| 9 | | | | 14 | 23 | 195 |

With all the sensors covered by at least one route, the *Setup Stage* of FGA ended and the case studies were ready to be optimized in the *Optimization Stage*.

5.5.2 Optimization Stage

After the *Setup Stage*, it was time to assign vehicles to each route. The FGA uses our fine tuned EA to do that and also RG if more routes are needed (Figure 5.2). Table 5.5 shows the results achieved after running the *Optimization Stage* of FGA on our 12 scenarios.

We can see that only one or two steps were necessary to match the desired values (maximum error under 10%) for the first case study (12 sensors, 2014 and 2015). Both scenarios corresponding to working days needed two steps and just one step for Sundays. Note that each step in this stage involved running the EA. It seems that the higher the number of vehicles in a scenario, the harder the optimization is, as expected. Additionally, the same

Table 5.5: Optimization of 12 scenarios using FGA (*Optimization Stage*). We report the number of steps, the origin or destination added in each step (O/D), the total number of routes (# R), the total number of vehicles (# V), maximum error (Max ϵ), and average number of hours taken by each step. Note that the final Max ϵ values achieved (below 10%) for each scenario are in bold.

| 12.2014.WD | | | | | | 12.2014.SAT | | | | | 12.2014.SUN | | | | |
|------------|------|-----|-------|----------------|-----|-------------|-----|------|----------------|-----|-------------|-----|------|----------------|-----|
| Step | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs |
| 1 | — | 56 | 4949 | 20% | 62 | — | 56 | 4371 | 10% | 49 | — | 56 | 3955 | 9% | 36 |
| 2 | D.7 | 57 | 5281 | 8% | 46 | D.7 | 57 | 4383 | 6% | 51 | — | — | — | — | — |
| 12.2015.WD | | | | | | 12.2015.SAT | | | | | 12.2015.SUN | | | | |
| Step | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs |
| 1 | — | 56 | 5024 | 14% | 57 | — | 56 | 4253 | 9% | 43 | — | 56 | 3501 | 7% | 28 |
| 2 | D.7 | 57 | 5331 | 4% | 53 | — | — | — | — | — | — | — | — | — | — |
| 23.2014.WD | | | | | | 23.2014.SAT | | | | | 23.2014.SUN | | | | |
| Step | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs |
| 1 | — | 195 | 10678 | 33% | 163 | — | 195 | 9096 | 45% | 78 | — | 195 | 8047 | 41% | 58 |
| 2 | D.5 | 203 | 11610 | 30% | 110 | D.5 | 203 | 9365 | 34% | 90 | D.5 | 203 | 8278 | 38% | 82 |
| 3 | D.44 | 214 | 11434 | 32% | 90 | D.44 | 214 | 9497 | 32% | 90 | D.44 | 214 | 8089 | 36% | 65 |
| 4 | O.44 | 205 | 11373 | 16% | 117 | O.44 | 205 | 9801 | 9% | 94 | O.44 | 205 | 8292 | 9% | 89 |
| 5 | D.4 | 220 | 11472 | 9% | 105 | — | — | — | — | — | — | — | — | — | — |
| 23.2015.WD | | | | | | 23.2015.SAT | | | | | 23.2015.SUN | | | | |
| Step | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs | O/D | # R | # V | Max ϵ | Hrs |
| 1* | — | 220 | 11420 | 9% | 102 | — | 205 | 9500 | 8% | 90 | — | 205 | 7685 | 4% | 68 |

sensor (7) always presented the biggest error in the first step making the FGA to add a new destination (D.7) when it was needed (Max $\epsilon \geq 10\%$). The time spent in the optimization process was between 28 hours (*12.2015.SUN*) and 110 hours (*12.2015.WD*).

The optimization of the second case study was harder. A bigger map containing more sensors required more routes and vehicles increasing the complexity. We addressed the optimization of the three scenarios of 2014, whose results are also shown in Table 5.5. We can see that again the scenario for Working Days required an extra step (and more routes) to reduce the maximum error under 10%.

Nevertheless, the steps one to four for the scenarios of 23.2014 used the same routes, as observed in the previous case study. We had at that point some evidence that routes seemed to depend only on the street distribution. Consequently, we tried to do something different with the 23.2015's scenarios: instead of going through all the previous steps of the *Optimization Stage* as we did until that moment, we took the routes from the last step of each 23.2014 scenarios and optimized them to match the sensor values of 23.2015.

What we did was to assume that we would reach the same configuration in the last step of the optimization of each scenario, so we directly jumped to it, prescinding of the previous steps, changing only the desired number of vehicles in 2015 as they are different from 2014. By taking this shortcut, not only have we significantly reduced the number of steps and shortened the optimization time (102 hours vs. 585 hours for Working Days), but in addition we achieved good (even better) solutions in just one step as can be seen in the last row of

Table 5.5. This will allow us to calculate flows for different traffic demands in just one step when all the base routes are set, saving a lot of time and computing resources. Note that the optimization process is performed only when building the models of the city (offline) and then they can be used without running FGA again (no computation time needed whatsoever).

5.6 Discussion

In this chapter we have presented the Flow Generator Algorithm (FGA), based on the designed RG and a new specialized EA, for generating routes according to real data measured on the city's streets. FGA is not a genetic algorithm, nor an optimization algorithm. FGA is a novel method proposed in this PhD thesis, which is able to generate realistic scenarios based on data published by local councils, using an EA, the SUMO microsimulator and its tools.

We have optimized two case studies (12 scenarios) imported from OpenStreetMap into SUMO and compared the number of vehicles at measurement points against the real ones published by the mobility department of Malaga council. This is a very complex problem not only because the number of decision variables but also due to the few data available plus some limitations such as the unknown traffic light cycles (which could reduce the flow capacity of the streets), the wrong U-turns in available maps. We overcame many of these limitations by filtering and correcting the information on OpenStreetMap.

FGA has been developed due to the necessity of validating the Smart Mobility proposals of this PhD thesis. Being suitable for many approaches and different technologies, FGA is a valuable tool that can be used by researchers to test their Smart Mobility proposals and other research work involving road traffic simulations. The resulting flows can be useful for performing different types of studies with the confidence of being able to work with a traffic distribution close to reality.

Despite the fact that an OpenStreetMap model is needed to import the city layout into SUMO, fortunately almost all cities are present in that platform nowadays. On the other hand, the need for sensor measurements may be a problem, although the open data initiative encourages the public administration to publish such data, and this is becoming very common.

After the experiments conducted and the results obtained showing an accuracy greater than 90% in all sensors and scenarios, we can conclude that FGA actually computes realistic traffic flows based on incomplete measurements obtained from a few sensors in the city. FGA fills the current research gap with incomplete data on cities, a very common situation for most cities in the world which have at most a few sensors installed. FGA also computes a distribution of flows compatible to the measured ones in an efficient way, so that any tool for Smart Mobility can use them to perform a wide range of applications.

Part II

Modeling and Solving Problems



UNIVERSIDAD
DE MÁLAGA

Chapter 6

Red Swarm: Reducing Travel Times

This chapter presents an innovative approach to solve one of the most relevant problems related to Smart Mobility: the reduction of travel times. This original approach, called Red Swarm, suggests a potentially customized route to each vehicle by using several spots located at traffic lights in order to avoid traffic jams by using V2I communications. That is quite different from other existing proposals, as it deals with real maps and actual streets, as well as several road traffic distributions. An Evolutionary Algorithm is proposed to optimize our case studies which have been imported from OpenStreetMap into SUMO as they belong to a real city. Additionally, a Rerouting Algorithm is developed which accesses the configuration of the Red Swarm and communicates the route chosen to vehicles, using the spots (via Wi-Fi link). Moreover, three competing algorithms have been developed in order to compare their results to those of Red Swarm and have observed that Red Swarm not only achieved the best results, but also outperformed the experts' solutions in a total of 60 scenarios tested.

6.1 Introduction

One of the aforementioned problems that can be found in big cities when we are traveling through their streets is experiencing a delay in our trip produced by an unexpected traffic jam [215]. This is becoming more common nowadays, especially in city center, where the number of vehicles in streets is continuously increasing [72]. As a consequence, citizen's quality of life is decreasing, not only because they take longer to reach their destination, but also because these situations can become very stressful.

We propose Red Swarm as a solution for this matter. Red Swarm is a data information system spread throughout the city at a low cost and able to redirect vehicles in movement in the city to finally achieve shorter travel times and fewer traffic jams in urban areas. The deployment of Red Swarm, a real time system for suggesting distributed personalized routes in a modern city, not only provides customized routes to every single vehicle, but it is also able to collect information from the road traffic (anonymously) enabling local authorities to better know the online and historical data of the city.

6.2 The Red Swarm Architecture (RS)

We propose a new system called Red Swarm (RS) [203, 204, 205] to optimize the traffic in the whole city with the aim of reducing travel times. It implies a continuous distributed exchange of data between vehicles and spots that will allow us to run intelligent algorithms that compute optimized route segments, customized to each driver in the city.

The Red Swarm architecture consists of:

1. Several spots distributed throughout the city, installed at traffic lights, which use a Wi-Fi connection to suggest new routes to vehicles.
2. The *Rerouting Algorithm* (RA), which selects the route to be suggested based on the configuration of the system and the vehicles' destination.
3. The Evolutionary Algorithm (EA), which computes the configuration of the system.
4. User Terminal Units (UTU), usually mobile phones or tablets which are able to communicate with the spots, send their data, and receive the new routes suggested. This function could also be fulfilled by On Board Units (OBU) installed in vehicles.

The Red Swarm architecture is divided into two stages: i) the configuration stage, and ii) the deployment and use stage (Figure 6.1). In the *Configuration Stage* the EA calculates the configuration for the spots by using the traffic simulator SUMO [122] in order to evaluate each solution. In the *Deployment and Use Stage*, the calculated optimal configuration for the Red Swarm spots is used by the RA (explained later) to suggest new routes to the vehicles that are approaching a junction controlled by a Red Swarm spot by using a Wi-Fi link.

Although the configuration is not recalculated in the deployment and use stage, the routes suggested to each vehicle are personalized, so as to split traffic into separate routes that will benefit both the individual vehicles and the overall traffic flow. Regarding the Wi-Fi communication, based on the results observed in [225], it presents a wide coverage area of 77 meters on average which supports our proposal.

As it was discussed in Chapter 4 we use in this PhD thesis realistic simulations, i.e. SUMO. While SUMO simulates the traffic flow of each vehicle, TraCI [237] makes it

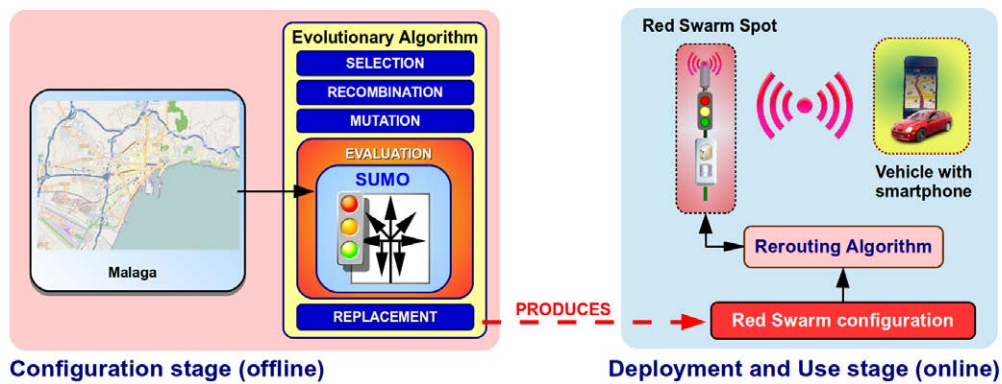


Figure 6.1: The Red Swarm architecture.

possible to control SUMO externally (RA) to reroute the vehicles according to each spot's configuration. When the simulation ends, data from the itinerary of each vehicle such as departure times, travel times, etc., can be obtained by parsing several XML files.

6.2.1 Evolutionary Algorithm (EA)

In the *Configuration Stage* of the Red Swarm architecture we use an EA to find the optimal arrangement of routes in the city to then spread the traffic out in a way that is efficient for drivers. Evolutionary algorithms are inspired by the evolution of individuals which are well adapted to their environment (see Section 3.2.1).

We have designed an steady state, (10+2)-EA to optimize the probabilities for each route (Section 3.2.1). We have chosen to work with a small population ($\mu = 10$) only creating two new individuals in each generation ($\lambda = 2$) to be more efficient, because our fitness function takes, on average, about 20 seconds to be computed, due to the complexity involved in analyzing the traffic distribution in the city.

Representation

Each Red Swarm spot should be placed at a traffic light situated at a street junction of the city. Consequently, it can be thought of a set of input and output streets as depicted in Figure 6.2a. The input streets (S_1 and S_2) are the streets by which the vehicles arrive at the junction and the output streets are the streets by which the vehicles leave that junction.

When a vehicle is approaching the junction by an input street, the Wi-Fi link is established which triggers the rerouting process. Then, this process suggests a new route to the vehicle depending on the configuration of the Red Swarm (probabilities P_1 , P_2 and P_3 in Figure 6.2a) and the vehicle's final destination in the city.

In Figure 6.2b an example of a possible rerouting is presented. When the vehicle is detected in the input street $S_{1,1}$ belonging to the Red Swarm spot RS_1 , a new route is suggested to the driver. The next step in the route of the vehicle might be the input street $S_{2,3}$ (spot

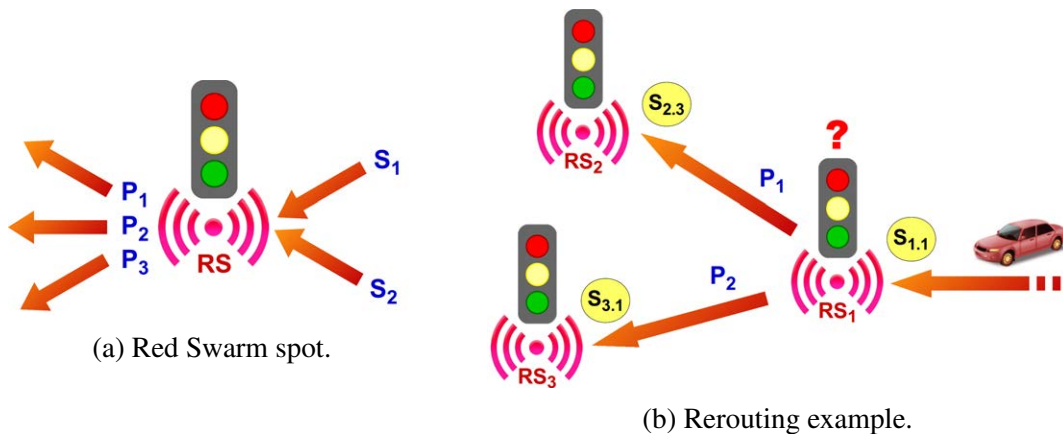


Figure 6.2: Red Swarm spots rerouting vehicles.

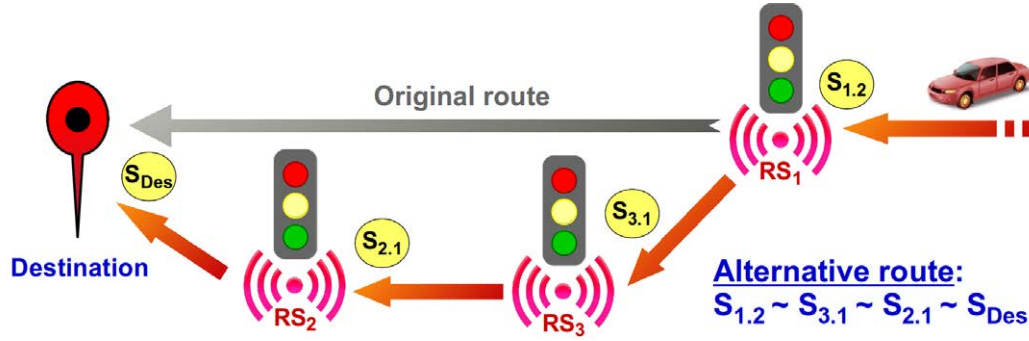


Figure 6.3: Rerouting of a vehicle through Red Swarm spots toward its final destination.

RS_2) or the input street $S_{3.1}$ (spot RS_3), depending on the values of the probabilities P_1 and P_2 which have been previously optimized for the traffic in this area in connection to the rest of the areas by the EA in the *Configuration Stage*.

This process is repeated in each Red Swarm spot until the vehicle arrives at its final destination. In this way, the new route of the vehicle is made up of several paths between input streets of Red Swarm spots, from the first spot which has detected and rerouted it, to the last spot which is placed in proximity to the vehicle's final destination (Figure 6.3).

In the case study analyzed (see Section 6.3) there are ten Red Swarm spots, each with several input streets which amounts to a total of 28 input streets in the area under analysis. Furthermore, there are also nine possible destinations, so that the different routes from one spot's input street to its reachable ones are arranged in nine chunks. Therefore, each reachable input street has a probability value associated with it which defines its chances of being suggested to a vehicle as the next step in its personalized route to destination.

Figure 6.4 shows the schematic representation of the problem, i.e. the probabilities for each reachable street of being selected, all of them mapped into a solution vector. In our study the solution vector is made of 1098 floating-point numbers (the probability values) which reveals the complexity of this problem. The probabilities included in each destination

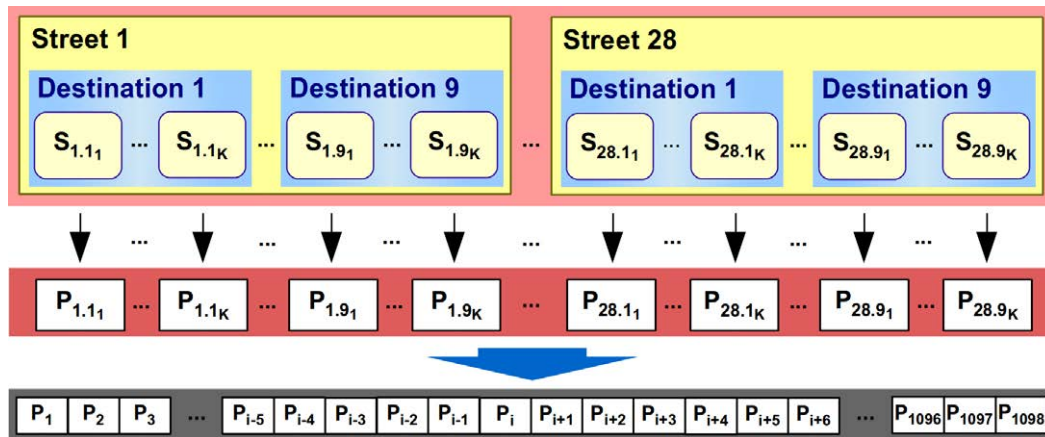


Figure 6.4: Schematic representation of the configuration of the Red Swarm mapped into a probability solution vector of floats.

chunk are normalized by following Equation 6.1, so that each value is in the range of $[0, 1]$. So, the sum of all the probabilities for the K_N reachable streets in the destination chunk M , which is part of the configuration of the street N , is equal to 1.

$$P_{S_N D_M} = P_{(N,M)_1} + \dots + P_{(N,M)_{K_N}} = \sum_{i=1}^{K_N} P_{(N,M)_i} = 1, \quad P_{(N,M)_i} \in [0, 1] \quad (6.1)$$

Evaluation Function

The evaluation of scenarios takes into account the average travel time of vehicles and the number of vehicles inside the area analyzed, both of which are collected after simulating several scenarios in the city. In Equation 6.2 we propose an evaluation function which computes a real number indicating the fitness of the configuration being evaluated. As we are minimizing this value, the lower it is the better.

$$F = \omega_1(N - n) + \omega_2 \frac{1}{n} \sum_{i=1}^n (delay + travel\ time)_i \quad (6.2)$$

In the first term, N is the total number of vehicles and n is the number of vehicles that have arrived at their destination when the simulation ends. This term guarantees that all vehicles arrive at their destination which is especially important as SUMO writes trip data in output files only for those vehicles which complete their itinerary. Additionally, if there are vehicles *en route* after the end of the simulation it is due to traffic jams, which is another important reason to penalize those poor configurations.

In the second term, the travel time of a vehicle is calculated by adding the time that it has waited before entering the area due to a congested input street (*delay*) and the time that it has spent in arriving at its destination (*travel time*). Both terms are weighted by two constants (ω_1 and ω_2). We assume that the vehicles which are in the city at the end of the analysis would have spent (on average) half of the analysis time in completing their itinerary ($\omega_1 = \frac{1}{2} \text{analysis time}$). The second term is weighted by one ($\omega_2 = 1$), so that both terms are in the same scale of time.

Selection Operator

The Selection Operator implemented chooses two individuals for reproduction by using a uniform probability distribution without taking into account their fitness value.

Recombination Operator

We have designed and tested two different recombination operators: The Street Two Point Crossover (STPX) and the Destination Crossover (DESX). The former keeps the configuration of each input street intact, while the latter does the same with each destination chunk.

STPX consists of a standard two point crossover in which two individuals are crossed by swapping their contents between two randomly selected points, to produce two descendants. In our case we exchange all the probabilities between the input street blocks selected (the

complete blocks including the destination chunks of probabilities). An example of STPX can be seen in Figure 6.5a where the probability values of the street blocks from six to twelve (crossover points) are exchanged.

The other operator, DESX, exchanges the configuration of destination chunks throughout all the street blocks of the parents instead of exchanging the configuration of streets, as STPX does. An example of DESX can be seen in Figure 6.5b where the probability values of destination chunks three and four are exchanged in all the street blocks of the solution vector.

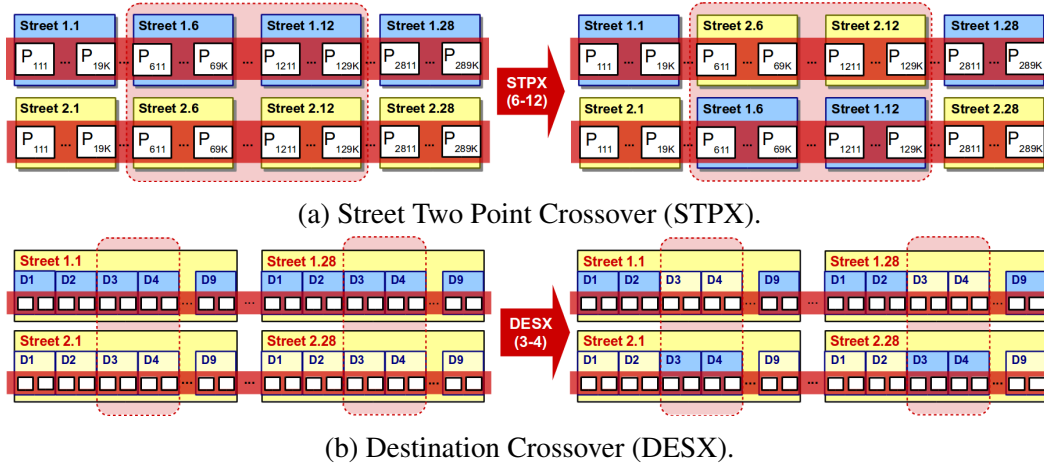


Figure 6.5: Red Swarm's recombination operators.

Mutation Operator

In [204] several mutation operators were proposed and tested. There, we decided to use two different operators for the mutation of the individuals in the EA. The former was meant to explore the search space and the latter, to exploit the accumulated search experience.

Then, we decided to unify these two operators in just one, called the Variable Mutation Operator (VMO). In order to preserve the variability presented by the two former mutation operators in VMO, we have used a variable mutation probability value. When the fitness value of the best individual of the population is greater than a threshold θ , π_1 will be used as the probability of changing the values in a destination chunk. Otherwise, π_2 will be the probability used. VMO changes only the probabilities in destination chunks of just one street block whether it is π_1 or π_2 used for the computations.

In Algorithm 6.1 the pseudocode of the VMO is presented. First, for each individual in the offspring the input street s is randomly chosen. Then, the list of destinations is obtained from the individual and some of the probabilities in each destination chunk are changed depending on the value returned by the *random* function, and the mutation probability P_m (which is equal to π_1 or π_2). Finally, when all the destinations in the street block have been processed, the individual mutated is returned.

For example, Figure 6.6 represents the VMO applied to an individual when *Street 4* has been randomly selected as the target street of the mutation. Then, destination chunks

Algorithm 6.1 Variable Mutation Operator (VMO).

```

procedure VMO( $P_m, offspring$ )
  for all  $individual \in offspring$  do
     $s \leftarrow getRandomStreet(individual)$  ▷ Input street
     $destinations \leftarrow getDestinations(individual)$  ▷ All destinations
    for all  $d \in destinations$  do
      if  $random() < P_m$  then ▷  $P_m \in \{\pi_1, \pi_2\}$ 
         $AssignNewProbabilities(d)$ 
      end if
    end for
  end for
  return  $offspring$ 
end procedure

```

Destination 1, *Destination 4*, and *Destination 5* have been randomly selected to be mutated. Consequently, in this example VMO changes the probability values of the ranges $P_{4.1.1}$ to $P_{4.1.K}$, $P_{4.4.1}$ to $P_{4.4.K}$, and $P_{4.5.1}$ to $P_{4.5.K}$ corresponding to the K input streets which are reachable from *Street 4*, when the final destination of the vehicle is either *Destination 1*, *Destination 4* or *Destination 5*.



Figure 6.6: Variable Mutation Operator (VMO).

Replacement Operator

We have used an elitist replacement [86] in which λ individuals of the population are replaced only if they have a fitness value worse than an offspring individual.

Parallel EA (pEA)

We have addressed the optimization of several scenarios in the same run of the EA by doing multiple evaluations (one per scenario) of the same individual in order to calculate its fitness. In doing so, we expect to achieve a more general configuration for the spots. However, as this implies an increment in the run time, we have developed a new parallel EA (pEA) to tackle it. The block diagram of the designed pEA is shown in Figure 6.7. It calculates the fitness function of an individual over n scenarios as the average fitness of them all.

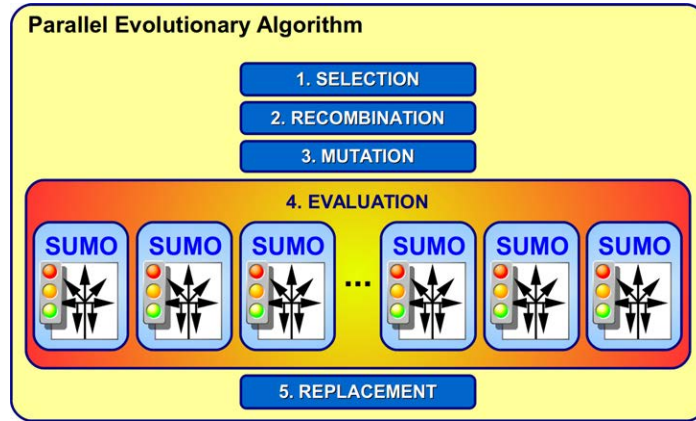


Figure 6.7: Parallel Evolutionary Algorithm (pEA).

6.2.2 Rerouting Algorithm (RA)

The Rerouting Algorithm (RA) is used to suggest a new route to vehicles which are approaching a Red Swarm spot. The pseudocode of the RA is presented in Algorithm 6.2 and the flow chart can be seen in Figure 6.8.

First, the current street is obtained from the data sent by the vehicle. Second, the final destination is checked to avoid rerouting a vehicle already on the last street of its itinerary, thus avoiding meaningless reroutings. To the contrary, if the vehicle has not yet reached its destination, all the routes from the current street to the destination of the vehicle are obtained from the system's configuration previously calculated by the EA. If the final destination is not directly reachable from the current street, the algorithm obtains the next input street (belonging to another spot) which is directly reachable from the vehicle's current street so that it is sent to another Red Swarm spot. This next input street is selected based on the probabilities stored in the configuration of the system. Finally, the next destination street

Algorithm 6.2 Rerouting Algorithm (RA).

```

procedure REROUTING(vehicle)
  current  $\leftarrow$  getStreet(vehicle)
  if isDestination(current, vehicle) then                                ▷ Last journey's street
    nextDestination  $\leftarrow$  current
  else
    nextDestination  $\leftarrow$  getDestination(current)
    if nextDestination =  $\emptyset$  then                                       ▷ Destination is unreachable
      nextStreets  $\leftarrow$  getReachableStreets(current)
      nextDestination  $\leftarrow$  getStreetByProbability(nextStreets)        ▷ Next RS spot
    end if
  end if
  setNextDestinationStreet(nextDestination, vehicle)
end procedure

```

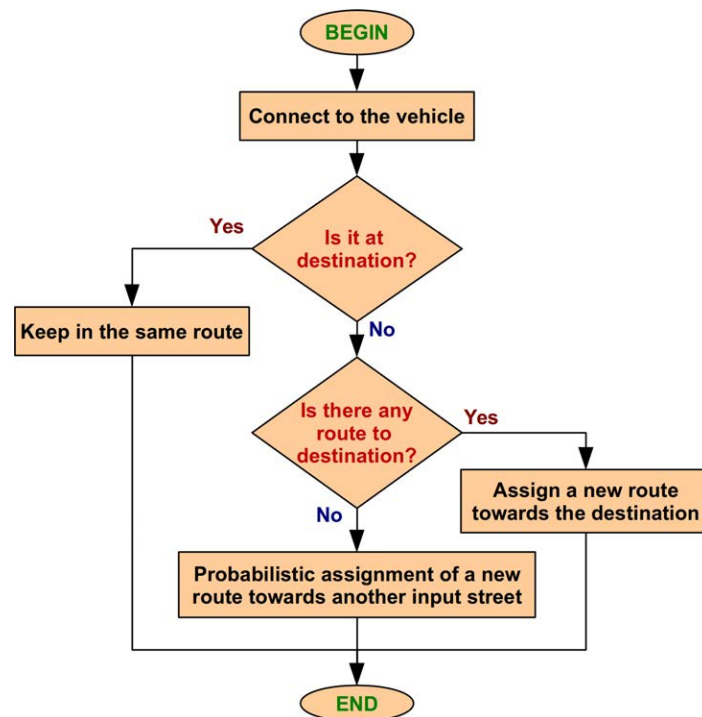


Figure 6.8: Flow chart describing the Rerouting Algorithm.

(thus the route) of the vehicle is set to the chosen one (here we have supposed that all drivers accept the change suggested) and the process ends. Note that if there is more than one route to the destination suggested, one of these routes will be randomly selected (driver's decision).

6.3 Case Study

We have applied the Red Swarm solution with the aim of reducing the average travel times of vehicles in an area of the city of Malaga (Malaga Park) which is well-known for suffering from traffic jams at peak times.

The geographical area analyzed which contains the park of Malaga is delimited to the north by Carretería Street, to the south by the Mediterranean Sea, to the east by Gutemberg Street, and to the west by the Guadalmedina River, and encompasses an area of about 2.5 km². Figure 6.9a shows a snapshot of the area analyzed taken from OpenStreetMap while Figure 6.9b the same area imported into SUMO is depicted. Finally, Figure 6.9c presents a snapshot exported from SUMO to Google Earth™ where our ten Red Swarm spots, placed at strategic junctions of the city, are represented by red circles.

We have used the method described in Section 4.3.2 to build our scenarios, all of which have been imported from OpenStreetMap [169]. Each traffic flow generated by DUAROUTER (experts' solution) consists of several routes between the same origin and the different destinations (including the different routes between them), so that we avoid



Figure 6.9: Area of Malaga Park imported from OpenStreetMap into SUMO, and exported to Google Earth™.

all vehicles driving along the same streets, which would be much too easy to optimize (just suggesting a pair of alternative routes).

In this PhD thesis we have worked with two different case studies called $z8$ and $z12$. These two share the same characteristics (listed in Table 6.1) such as 262 traffic lights, ten Red Swarm spots, four vehicle types (listed in Table 6.2), and nine input and output streets where vehicles arrive at and exit from the analyzed area, respectively. We have defined four different types of vehicles in order to create a more realistic approach. The differences between them are the arrival probability which defines the distribution of vehicles in each scenario, the maximum speed of vehicles which is also limited by the streets' speed limit, the acceleration and deceleration rates and the vehicles' length.

Table 6.1: Characteristics of the two case studies.

| Case study | z8 | z12 |
|-------------------|------|------|
| # vehicles | 800 | 1200 |
| Analysis time (s) | 2400 | 3000 |
| # Traffic lights | 262 | |
| # Red swarm spots | 10 | |
| # Vehicle types | 4 | |
| # Input streets | 9 | |
| # Output streets | 9 | |

Table 6.2: Type and characteristics of vehicles.

| Type | Arrival Prob. | MaxSpd. (km/h) | Accel. (m/s ²) | Decel. (m/s ²) | Length (m) |
|-------|---------------|----------------|----------------------------|----------------------------|------------|
| sedan | 0.50 | 160 | 0.9 | 5.0 | 3.8 |
| van | 0.25 | 100 | 0.8 | 4.5 | 4.2 |
| wagon | 0.15 | 50 | 0.7 | 4.0 | 4.3 |
| truck | 0.10 | 40 | 0.6 | 3.5 | 4.5 |

6.4 Competitor Techniques for our EA

We propose in this section three other competing algorithms that could reduce travel times by configuring the system in the configuration stage: i) The DJK algorithm, based on the Dijkstra shortest path algorithm [50]; ii) the DV algorithm, based on the Bellman-Ford algorithm [24]; and iii) the ACO algorithm, which is based on the Ant System algorithm presented in [59].

6.4.1 Dijkstra (DJK)

The Dijkstra algorithm (DJK) proposed here is based on the implementation of the well-known Dijkstra shortest path algorithm [50] included in the DUAROUTER utility, which is part of the SUMO suite. The solution vector to be used by the RA will be calculated by the DJK algorithm as an alternative to the EA.

DJK conceives the scenario as a graph in which inputs, the spots' input streets, and destinations are all the nodes, while the edges are the different paths between them. The weight values for each edge are calculated by counting the number of preplanned routes that include each street of that edge. Then, the solution vector is obtained by converting these weights to probability values so that they can be managed by the RA.

Equation 6.3 calculates the probability for the route between the input street S_N and the destination D_M . There, $\omega_{S_N D_M}$ is the weight of the edge between S_N and D_M , and $\sum_i \frac{1}{\omega_{S_N D_i}}$ is the summation of all the destinations reachable from S_N . As we want to distribute the road traffic through different streets, the more congested is a street (more routes), the less likelihood of being chosen the street has.

$$P_{S_N D_M} = \frac{1}{\omega_{S_N D_M} \sum_i \frac{1}{\omega_{S_N D_i}}} \quad (6.3)$$

6.4.2 Distance Vector (DV)

We propose in this section another competitor of our EA. The Distance Vector (DV) algorithm is based on the implementation included in the RIP Version 2 protocol (RFC 2453) of the Distance Vector algorithm, which in turn is based on the Bellman-Ford algorithm [24].

Most of the current Internet routers implements this algorithm to know the length of the shortest path from themselves to all the other destination routers. In this case, we calculate routes between the input streets of the Red Swarm spots of the city and the final destinations of vehicles. This algorithm and its implementation are found in most of the current Internet routers and its main characteristic is that each node of the network knows the length of the shortest path from itself to all the other destination routers.

The routing tables of each input street contain an entry for each destination in the city, which are updated with the ID of the next input street in the route. This input street is selected depending on the number of spots which will be in this journey to the vehicle destination when it takes this route, the fewer the better.

In this case, instead of the RA we run a simplified version of the Rerouting Algorithm so that the next street suggested to vehicles is taken directly from the routing tables calculated by the DV algorithm, thus we are not calculating a solution vector of probabilities but actual rerouting tables. When the rerouting takes place in each spot, the next street suggested to vehicles will be obtained directly from the routing tables calculated for each input street, instead of using the RA based on probabilities.

6.4.3 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) [58] is an optimization technique inspired by the natural behavior of ants, which has been described in Section 3.2.3. Inspired by the Ant System model [59], we here provide a new algorithm to suggest routes in our system in a new way.

Our case study is represented by nine graphs (one per destination), whose nodes are the 28 input streets and whose edges are the routes between them. In ACO, a set of ants construct a solution by traveling through a graph which represents the environment. So, we have created 56 ants per graph (twice the number of input streets) and placed them randomly in the nodes which are different from the destination ones.

Then, each ant makes a sequence of probabilistic decisions when arriving at each Red Swarm spot in order to choose the next input street to visit. This series of decisions represents the path that the ant has followed to reach its target destination. Thus, we build the ants' tours across the nodes of the graph (input streets) until the destination is reached.

When all the ants have ended their individual journey, a new iteration begins after marking each edge of the graph with artificial pheromones. These pheromones influence the ants' decisions in following iterations in such a way as to increase the likelihood of the paths which have been transited most, being chosen in the current iteration. There also exists an evaporation coefficient which prevents the pheromone values from having an influence for too long on one iteration propagation, as happens in the natural ant system.

The pheromone trail update is done as explained in Equation 6.4, where $\tau_{ij}(t+1)$ is the intensity of the trail in the next iteration, $\tau_{ij}(t)$ is the current intensity, $(1 - \rho)$ represents the

evaporation of the trail in the current iteration, and $\Delta\tau_{ij}$ is the sum of the quantity per unit of length of pheromones laid on the edge (i, j) by the k -th ant (Equation 6.5). Finally, the $\Delta\tau_{ij}^k$ value is calculated by following Equation 6.6, where Q is the relative importance of the trail and L_k is the tour length of the k -th ant.

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij} \quad (6.4)$$

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k \quad (6.5)$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if the } k\text{-th ant travels from street } i \text{ to street } j \\ 0 & \text{otherwise} \end{cases} \quad (6.6)$$

Moreover, the probability of transit from street i to street j is formalized in Equation 6.7, where η_{ij} is the heuristic value which defines the visibility of the next street, and $allowed_k$ is the set of the direct reachable streets. We have used the same heuristic as in DJK and DV, i.e., the number of routes that contain the streets. Furthermore, both α and β are the parameters that control the relative importance of trail versus visibility in the transition equation.

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{k \in allowed_k} [\tau_{ik}(t)]^\alpha \cdot [\eta_{ik}]^\beta} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (6.7)$$

As we need probability values to configure the system, when all the ants have ended their tours we evaluate the solution by mapping the paths (tours) into probabilities in order to get the status vector to configure the Red Swarm spots. Since we have a set of routes from the ACO system, at this point we count the number of paths transited by the ants, for each route, from the nine graphs that include this route. Then, we calculate the normalized solution vector of probabilities, so that the more transited a route is, the less likely it is to be selected.

6.5 Parameterization

Now, we describe the experiments that have been conducted in order to parameterize the ACO algorithm (Section 6.5.1) and the EA algorithm (Section 6.5.2).

As we are dealing with a very complex problem and consequently with long execution times, we cannot follow an exhaustive method of parameterization. Instead, we have tested several parameter values for the operators and made decisions based on the data collected.

In all the experiments we have applied a Friedman test to determine the best parameter and operator, and then we have studied the statistical significance of these data by using the Wilcoxon test to compare the best ranked distribution to the rest (pairwise comparisons) [189]. All the experiments were conducted by performing 30 independent runs on the same scenario.

6.5.1 Parameterization of the ACO Algorithm

Based on the values proposed in [59], we have conducted several experiments in order to perform the parameterization that better suits our problem. The results of 30 runs of the algorithm using different values for the quantity of trail laid by the ants (Q), the relative importance of the trail (α) and the visibility (β) are shown in Table 6.3. The number of ants (m) was set to 504 (twice the number of streets multiplied by the nine destinations, thus, the nine graphs), the trail evaporation in each iteration ($1 - \rho$) was set to 0.25, and the initial value for trails ($\tau_{ij}(0)$) was set to 0.50.

Table 6.3: Parameter tuning of the ACO algorithm.

| Parameters | | | Best Fitness | | Friedman Rank | Wilcoxon p -value |
|------------|------------|------------|---------------|---------------|---------------|---------------------|
| Q | α | β | Avg. | StdDev | | |
| 25 | 0.5 | 1.0 | 5215.6 | 141.6% | 3.73 | 0.19 |
| | 1.0 | 0.5 | 7613.4 | 216.1% | 2.00 | — |
| | 1.0 | 1.0 | 4317.3 | 183.7% | 2.33 | 0.99 |
| 50 | 0.5 | 1.0 | 7552.4 | 121.1% | 4.97 | 0.05 |
| | 1.0 | 0.5 | 8840.0 | 225.8% | 2.70 | 0.07 |
| | 1.0 | 1.0 | 13853.8 | 147.7% | 5.27 | 0.00 |

We have explored a large set of potential sets of parameters for ACO adding up to a final large number of 180 experiments. As we can see, the best ranked algorithm, $ACO_{25,1.0,0.5}$ ($Q = 25$, $\alpha = 1.0$, $\beta = 0.5$) and the second best, $ACO_{25,1.0,1.0}$ ($Q = 25$, $\alpha = 1.0$, $\beta = 1.0$), both share the same statistical benefits (p -value = 0.99), so we have chosen the latter because its average fitness and standard deviation are lower than the former (4317.3 vs. 7613.4 and 183.7% vs. 216.1%, respectively).

6.5.2 Parameterization of the EA

We have tested the STPX and DESX operators for recombination probabilities (P_c) of 0.2, 0.4, 0.6, 0.8, and 1.0, by doing 30 runs for each operator and probability values (300 runs). Table 6.4 shows the results of the experiments conducted as well as the statistical analysis performed. As we can see, STPX has outperformed DESX for all the recombination probability values tested.

Then, we have conducted several experiments in order to parametrize the mutation operator VMO, which are also listed in Table 6.5. We have tested π_1 and π_2 for probability combinations of $\frac{1}{9}$ (one over the number of destinations), $\frac{1}{3}$, $\frac{1}{6}$ and 1.

We have chosen $\pi_1 = 0.33$ and $\pi_2 = 0.11$ based on their average fitness value in spite of the fact that it is the second best ranked case (5.30 vs. 5.27). Moreover, the Wilcoxon p -value denotes that this configuration (0.33,0.11) and the best ranked one (0.66,0.33) are not so different (p -value = 0.91), which allows us to make this decision. All in all, the mutation probability of a route will depend on the number of input streets, π_1 , and π_2 , so that it will be $\frac{1}{28} \frac{1}{3} = \frac{1}{84}$ for π_1 and $\frac{1}{28} \frac{1}{9} = \frac{1}{252}$ for π_2 . All the parameters of the EA are shown in Table 6.6.

Table 6.4: Tuning of EA's recombination operator.

| P_c | Street Two Point Crossover (STPX) | | | | Destination Crossover (DESX) | | | |
|------------|-----------------------------------|-------------|---------------|---------------------|------------------------------|--------|---------------|---------------------|
| | Fitness | | Friedman Rank | Wilcoxon p -value | Fitness | | Friedman Rank | Wilcoxon p -value |
| | Average | StdDev | | | Average | StdDev | | |
| 0.2 | 917.3 | 8.3% | 4.07 | 0.27 | 964.0 | 14.6% | 6.00 | 0.00 |
| 0.4 | 3930.3 | 398.0% | 3.53 | 0.00 | 1301.2 | 151.3% | 5.17 | 0.00 |
| 0.6 | 887.5 | 5.7% | 2.37 | 0.50 | 14711.5 | 213.1% | 8.00 | 0.00 |
| 0.8 | 941.8 | 12.0% | 4.80 | 0.00 | 20301.9 | 179.1% | 8.93 | 0.00 |
| 1.0 | 1068.5 | 84.7% | 2.17 | — | 36507.2 | 140.0% | 9.97 | 0.00 |

Table 6.5: Tuning of EA's VMO ($P_c = 0.6$).

| π_1, π_2 | Fitness | | Friedman Rank | Wilcoxon p -value |
|-------------------|--------------|--------------|---------------|---------------------|
| | Average | StdDev | | |
| 0.11, 0.66 | 12707.1 | 484.9% | 6.77 | 0.34 |
| 0.11, 1.00 | 9827.1 | 275.1% | 8.90 | 0.05 |
| 0.33, 0.11 | 987.0 | 12.4% | 5.30 | 0.91 |
| 0.33, 0.66 | 34553.9 | 371.2% | 6.67 | 0.25 |
| 0.33, 1.00 | 2527.9 | 315.5% | 7.40 | 0.23 |
| 0.66, 0.33 | 4664.1 | 318.0% | 5.27 | — |
| 0.66, 1.00 | 4999.7 | 427.9% | 7.83 | 0.15 |
| 1.00, 0.11 | 13455.8 | 397.6% | 5.33 | 0.50 |
| 1.00, 0.33 | 10833.4 | 230.4% | 6.67 | 0.27 |
| 1.00, 0.66 | 11510.1 | 321.3% | 6.93 | 0.08 |

Table 6.6: Parameters of EA.

| Parameter | Value |
|------------------|-----------------|
| P_c | 0.6 |
| π_1 | $\frac{1}{84}$ |
| π_2 | $\frac{1}{252}$ |
| θ | 1500 |
| Max. generations | 5000 |

6.6 Experimental Analysis

In the following sections we address the optimization of the case studies $z8$ and $z12$. We have analyzed two versions of EA, EA10 where just one of the available routes between spots is used, and EA05 where up to two routes can be probabilistically used.

First, we have run the EA and its competitors algorithms to compare their performance as well as the solutions achieved. Second, we have set the best solution of each algorithm as the configuration of Red Swarm and tested it in 30 different scenarios in order to discover how scalable the solutions are. Finally, we have identified the best performing algorithms and reported the improvement achieved in the average travel time of vehicles.

6.6.1 Optimization

In this section we have performed the optimization of $z8$ and $z12$ in order to achieve an optimum for the configuration of the Red Swarm spots. This case study consists of 800 vehicles that arrive in the analyzed zone of the city via nine input streets and that drive through the city until reaching their destination.

We do not include DJK and DV in this first experiment because they are deterministic and only one execution is needed to get the configuration vector from those algorithms.

In the case of the EA10, EA05 and ACO algorithms, we have carried out 30 runs as they are nondeterministic in order to analyze their performance as well as achieve the best configuration for the Red Swarm spots. In Table 6.7 we present the results of the optimization of one scenario of $z8$ and another of $z12$ performed by our algorithms. We have included the average of the best fitness achieved in the 30 runs as well as the standard deviation. Furthermore, the average number of iterations and the standard deviation are also provided together with the Friedman Rank and the Wilcoxon p -value.

Table 6.7: Fitness, number of iterations and statistical tests for the optimization of one scenario of $z8$ and $z12$.

| Alg. | Fitness | | # iterations | | Friedman Rank | Wilcoxon p -value |
|-------|--------------|-------------|--------------|--------------|---------------|---------------------|
| | Avg. | StdDev | Avg. | StdDev | | |
| $z8$ | | | | | | |
| EA10 | 658.0 | 7.1% | 2531.6 | 36.4% | 1.00 | — |
| EA05 | 760.0 | 8.1% | 3485.0 | 32.5% | 2.67 | 0.00 |
| ACO | 1610.4 | 172.7% | 573.7 | 12.2% | 2.33 | 0.00 |
| $z12$ | | | | | | |
| EA10 | 855.7 | 8.5% | 3699.0 | 30.5% | 1.00 | — |
| EA05 | 1037.8 | 9.7% | 3411.7 | 30.5% | 2.67 | 0.00 |
| ACO | 7613.4 | 216.1% | 579.8 | 15.3% | 2.33 | 0.00 |

The first conclusion is that EA10 has achieved the lowest average fitness value for $z8$ and $z12$ (658.0 and 855.7 respectively). It is also notable the small number of iterations (573.7 and 579.8 on average) performed by the ACO algorithm (thus a faster convergence) despite its high average fitness. Based on the fitness values and the statistical analysis we have selected the best configurations of each algorithm and tested them on 30 unseen instances of both case studies, where we have also included the deterministic algorithms (DJK and DV). We have configured the Red Swarm spots with the solution obtained from the five algorithms in order to analyze how they behave in the 30 different scenarios of $z8$ and other 30 of $z12$.

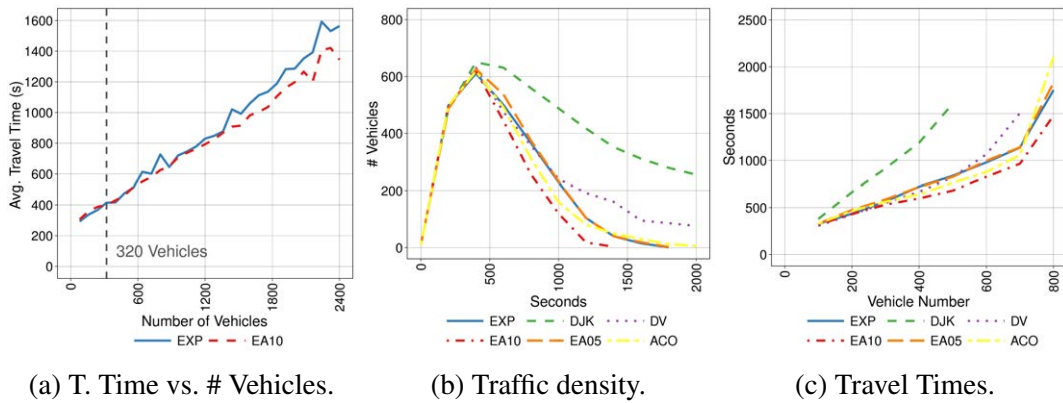
Table 6.8 present the results where the high fitness values of DJK and DV confirm they are not capable of rerouting vehicles to their destinations, while avoiding traffic jams in the period of time analyzed. EA05 and ACO are also unable to obtain competitive results while EA10 is the best performing algorithm although it only has improved the experts' solution (EXP) in some scenarios (53% of $z8$ and 83% of $z12$).

Therefore, in order to analyze how Red Swarm behaves when compared with the experts' solution (EXP) when it is configured by EA10, we have illustrated the travel time vs. the number of vehicles in the city in figures 6.10a and 6.11a. Although we have optimized case studies of 800 and 1200 vehicles, we have tested the configuration obtained by EA10 with up to 2400 vehicles.

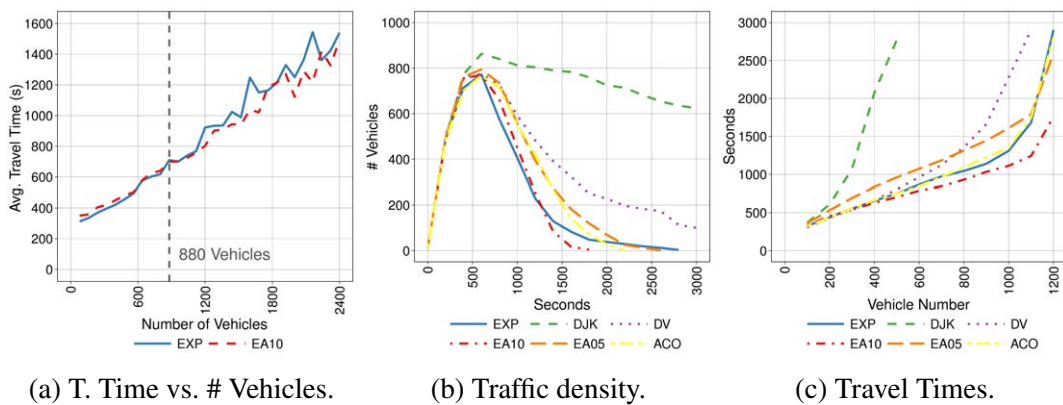
As can be seen, EA10 becomes effective when there are more than 320 vehicles in $z8$ and 880 vehicles in $z12$. This is an interesting and high impact conclusion, since it means that, as long as we have around 1000 vehicles in the geographical area (very likely) our solution is more efficient for drivers and modern urban policies.

Table 6.8: Fitness comparative and statistical test of 30 scenarios of $z8$ and $z12$. The best values are in bold.

| Alg. | $z8$ | | | | | $z12$ | | | | |
|------|--------------|-------------|--------------------|---------------|---------------------|--------------|-------------|--------------------|---------------|---------------------|
| | Fitness | | Scenarios Improved | Friedman Rank | Wilcoxon p -value | Fitness | | Scenarios Improved | Friedman Rank | Wilcoxon p -value |
| | Avg. | StdDev | | | | Avg. | StdDev | | | |
| EXP | 623.5 | 4.3% | — | 1.50 | — | 821.1 | 3.5% | — | 1.83 | 0.00 |
| DJK | 222588.6 | 8.3% | 0.0% | 6.00 | 0.00 | 1048164.2 | 12.8% | 0.0% | 6.00 | 0.00 |
| DV | 50229.1 | 15.2% | 0.0% | 5.00 | 0.00 | 146557.1 | 60.6% | 0.0% | 4.97 | 0.00 |
| EA10 | 658.1 | 34.9% | 53.3% | 1.53 | 0.75 | 788.0 | 3.3% | 83.3% | 1.17 | — |
| EA05 | 761.8 | 23.6% | 0.0% | 3.90 | 0.00 | 14494.9 | 510.0% | 0.0% | 3.83 | 0.00 |
| ACO | 837.6 | 71.3% | 3.3% | 3.07 | 0.00 | 923.2 | 6.8% | 0.0% | 3.20 | 0.00 |

Figure 6.10: Traffic density and travel times comparison for the best scenario of $z8$.

Moreover, in figures 6.10b and 6.10c we present the traffic density and the travel times of vehicles in $z8$. We can see in both graphs the effects of the best solution of each algorithm on the road traffic, which confirm that neither DJK nor DV are capable of managing such a number of vehicles and that EA10 routes vehicles out of the city faster than the experts' solution. The same behavior is observed in figures 6.11b and 6.11c for vehicles in $z12$.

Figure 6.11: Traffic density and travel times comparison for the best scenario of $z12$.

6.6.2 Parallel EA

Parallelism is central to carry out our experiments. We need to simulate the whole city, with thousands of cars following driving directions, interacting with each other, communicating, considering statistics and a large amount of data. Parallelism has been essential in reducing the study time from several years of computation to five weeks. We used a parallel version of our EA to optimize more than one scenario in the same run. Thus we have decided to calculate the average fitness value of two, four and even eight scenarios of $z8$ and $z12$ as the fitness value of an individual by using different versions of our pEA: pEA10.2, pEA10.4, and pEA10.8, respectively. So, we have named as pEA10.x, the parallel version of the EA10 algorithm which evaluates x scenarios in parallel and obtains the individual's fitness value by averaging them.

We have tested our parallel algorithms by optimizing one scenario of $z8$ and one of $z12$ in 30 independent runs. As can be clearly deduced from the values in Table 6.9, the more scenarios are optimized, the better the solution achieved. Consequently, pEA10.8 presents the best results in the optimization scenarios of $z8$ and also in the $z12$ ones.

Table 6.9: Average fitness, average number of iterations and statistical tests for the parallel optimization of two, four, and eight scenarios of the case studies $z8$ and $z12$. Note that the best values of each case study are in bold.

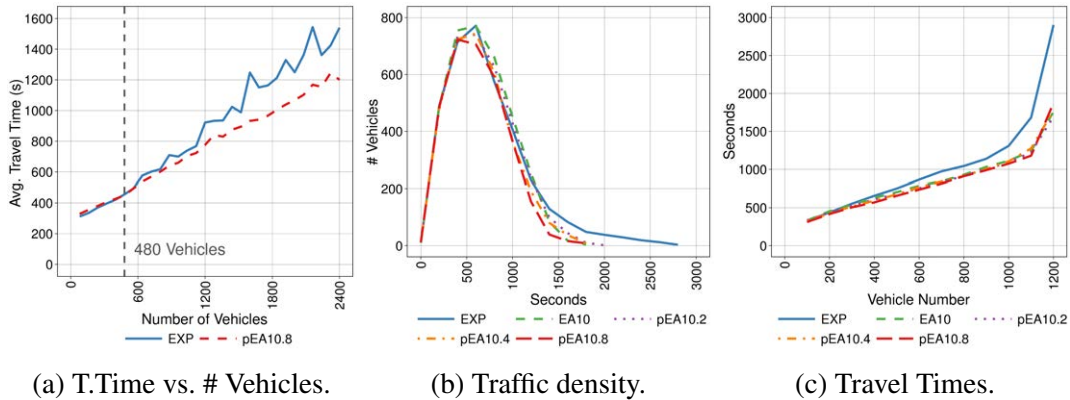
| Alg. | Fitness | | # iterations | | Friedman Rank | Wilcoxon p -value |
|---------|--------------|--------------|---------------|--------------|---------------|---------------------|
| | Avg. | StdDev | Avg. | StdDev | | |
| $z8$ | | | | | | |
| pEA10.2 | 746.3 | 95.6% | 2647.5 | 34.8% | 2.73 | 0.00 |
| pEA10.4 | 608.6 | 2.8% | 2940.3 | 32.8% | 2.07 | 0.00 |
| pEA10.8 | 603.3 | 2.9% | 3140.9 | 31.9% | 1.20 | — |
| $z12$ | | | | | | |
| pEA10.2 | 6428.3 | 332.2% | 2866.2 | 38.0% | 2.67 | 0.02 |
| pEA10.4 | 881.4 | 22.8% | 2717.2 | 35.4% | 2.57 | 0.00 |
| pEA10.8 | 874.0 | 41.8% | 3496.5 | 32.4% | 1.43 | — |

Next, we tested our solutions in 30 different scenarios for each case study. The results show that all the parallel algorithms have achieved better fitness values than EXP in all the scenarios of $z12$ and in mostly all of them in $z8$ as shown in Table 6.10. Our results seem to depend not only on the number of scenarios but also which scenarios are selected for optimization. Consequently, the more scenarios that are optimized, the more robust the solution achieved.

Figure 6.12a shows that the minimum number of vehicles from which Red Swarm is effective has been reduced to 480 vehicles (it was 880 for EA10) as the behavior of the configuration computed by pEA10.8 is more linear with respect to the number of vehicles than the EA10 one. In addition, figures 6.12b and 6.12c show the improvement in traffic density and travel times of the vehicles achieved in $z12$.

Table 6.10: Fitness comparative and statistical test of 30 scenarios of $z8$ and $z12$. The best values are in bold.

| Alg. | Fitness | | Friedman Rank | Wilcoxon p -value |
|---------|--------------|-------------|---------------|---------------------|
| | Avg. | StdDev | | |
| $z8$ | | | | |
| EXP | 623.5 | 4.3% | 2.30 | 0.01 |
| EA10 | 658.1 | 34.9% | 3.93 | 0.00 |
| pEA10.2 | 595.7 | 2.1% | 3.33 | 0.00 |
| pEA10.4 | 599.5 | 2.4% | 3.73 | 0.00 |
| pEA10.8 | 584.3 | 1.8% | 1.70 | — |
| $z12$ | | | | |
| EXP | 821.1 | 3.5% | 4.83 | 0.00 |
| EA10 | 788.0 | 3.3% | 3.53 | 0.00 |
| pEA10.2 | 779.3 | 2.1% | 3.13 | 0.00 |
| pEA10.4 | 768.5 | 2.0% | 2.50 | 0.00 |
| pEA10.8 | 744.8 | 1.7% | 1.00 | — |

Figure 6.12: Traffic density and travel times comparison for the best scenario of $z12$ using pEA.

Finally, the average travel times in the 30 scenarios tested as well as the average route length of vehicles are listed in Table 6.11. Additionally, the metrics of the most improved scenario are also listed.

We can see that pEA10.8 achieved an average improvement of 6.3% (19.2% maximum) in $z8$ and 9.3% (18.8% maximum) in $z12$ on the average travel time, i.e., travel times which are 173 seconds shorter (76 seconds on average). The average distance traveled by vehicles when they are being rerouted by Red Swarm is always longer than in the experts' solution. This was to be expected because we are rerouting vehicles via alternative streets which are not part of the shortest path, with the aim of reducing traffic jams. This extra length is however minimal (10% percent on average) and has a huge advantage in reducing times for drivers and the city.

In the last study carried out we have calculated the *weak orthodox speedup* [2] of our parallel algorithms by carrying out 10 runs of each algorithm with a different random seed.

Table 6.11: Results of the optimization of the vehicles' average travel time, divided in the average of 30 scenarios and the best of them. Note that the best values of each case study are in bold.

| Alg. | Average 30 scenarios | | | | | | Best scenario | | | | | |
|------------|----------------------|-------------|-------------|---------------|-------------|-------|---------------|--------------|--------------|---------------|--------------|-------|
| | Travel Time | | | Route Length | | | Travel Time | | | Route Length | | |
| | Avg. | StdDev | Impr. | Avg. | StdDev | Rate | Avg. | StdDev | Impr. | Avg. | StdDev | Rate |
| <i>z8</i> | | | | | | | | | | | | |
| EXP | 623.5 | 4.3% | — | 1770.2 | 1.2% | — | 726.6 | 48.2% | — | 1771.3 | 42.0% | — |
| EA10 | 658.1 | 34.9% | -5.5% | 1960.6 | 1.6% | 10.8% | 624.8 | 42.0% | 14.0% | 1945.9 | 45.9% | 9.9% |
| EA05 | 761.8 | 23.6% | -22.2% | 2076.9 | 1.4% | 17.3% | 739.5 | 74.5% | -1.8% | 2097.9 | 49.1% | 18.4% |
| ACO | 837.6 | 71.3% | -34.3% | 2066.4 | 1.8% | 16.7% | 699.0 | 51.5% | 3.8% | 2066.7 | 48.4% | 16.7% |
| pEA10.2 | 595.7 | 2.1% | 4.5% | 1895.8 | 1.4% | 7.1% | 599.0 | 41.1% | 17.6% | 1933.6 | 43.0% | 9.2% |
| pEA10.4 | 599.5 | 2.4% | 3.9% | 1948.3 | 1.4% | 10.1% | 598.2 | 40.8% | 17.7% | 1972.5 | 44.2% | 11.4% |
| pEA10.8 | 584.3 | 1.8% | 6.3% | 1901.4 | 1.6% | 7.4% | 587.1 | 40.7% | 19.2% | 1899.8 | 43.3% | 7.3% |
| <i>z12</i> | | | | | | | | | | | | |
| EXP | 821.1 | 3.5% | — | 1746.4 | 1.2% | — | 922.0 | 55.0% | — | 1716.5 | 43.5% | — |
| EA10 | 788.0 | 3.3% | 4.0% | 2029.0 | 3.5% | 16.2% | 794.6 | 42.4% | 13.8% | 2017.3 | 47.6% | 17.5% |
| EA05 | 14494.9 | 510.0% | -1665.3% | 2183.6 | 2.3% | 25.0% | 1085.7 | 46.7% | -17.7% | 2180.4 | 50.2% | 27.0% |
| ACO | 923.2 | 6.8% | -12.4% | 1927.0 | 1.9% | 10.3% | 935.1 | 55.1% | -1.4% | 1919.1 | 50.8% | 11.8% |
| pEA10.2 | 779.3 | 2.1% | 5.1% | 2047.7 | 1.8% | 17.3% | 770.2 | 41.8% | 16.5% | 1903.9 | 45.1% | 10.9% |
| pEA10.4 | 768.5 | 2.0% | 6.4% | 1973.5 | 1.0% | 13.0% | 776.4 | 44.5% | 15.8% | 1978.4 | 45.7% | 15.3% |
| pEA10.8 | 744.8 | 1.7% | 9.3% | 1921.1 | 1.2% | 10.0% | 749.0 | 43.4% | 18.8% | 1931.7 | 45.3% | 12.5% |

Because of the huge demand on resources that each parallel execution requires we have not done 30 runs in this case.

Table 6.12 lists the different experiments and results of the execution of the algorithm optimizing one scenario in a one core machine (sEA10.1), two scenarios in a one core machine (sEA10.2) and also in two core machines (pEA10.2), and so on. We have named the sequential calculation of n fitness functions sEA10. n and the parallel calculation pEA10. n .

The results show that both pEA10.2 and pEA10.4 have nearly reached a linear speedup by using parallelism (1.9 and 3.9 respectively). Moreover, when using 8 cores (pEA10.8) we

Table 6.12: Average execution times and speedup of ten independent runs of the sequential algorithm (sEA10. n) and parallel (pEA10. n). The best speedup reached is in bold.

| # Scenarios | Algorithm | # Cores | Avg. Time (h) | Speedup |
|-------------|-----------|---------|---------------|------------|
| 1 | sEA10.1 | 1 | 19.1 | 1.0 |
| 2 | sEA10.2 | 1 | 35.8 | 1.0 |
| | pEA10.2 | 2 | 18.8 | 1.9 |
| 4 | sEA10.4 | 1 | 40.0 | 1.0 |
| | pEA10.4 | 4 | 10.2 | 3.9 |
| 8 | sEA10.8 | 1 | 93.0 | 1.0 |
| | pEA10.8 | 8 | 13.8 | 6.8 |

got a very good speedup of 6.8 (an efficiency of 85%), not really the perfect 8 value because of the overload that the execution of eight parallel threads in the same computer represents.

6.7 Discussion

In this chapter we have presented a solution to one of the more important Smart Mobility problems, long travel times due to traffic jams.

The Red Swarm architecture configured by our EA suggests alternative routes customized to drivers in order to avoid traffic jams and find a quicker way to reach destination.

The solutions achieved when we tested the configuration obtained by the EA in 30 different scenarios show that we have outperformed all of them when we use pEA10.8 as the optimization algorithm. This indicates that our solution is robust enough in such a complex problem like the one analyzed here.

We have also tested Red Swarm in different traffic conditions (number of vehicles) and we found a threshold representing the minimum number of vehicles from which Red Swarm becomes effective as a system for preventing traffic jams. In each case study, the threshold is quite a bit lower than the usual average number of vehicles in the real geographical area. However, with a small number of vehicles Red Swarm still works fine, but the enhancements to the city are less noticeable.

Finally, we observed lower traffic densities and shorter travel times when we applied Red Swarm to the most likely traffic situations in a modern city. This represents valuable aid to the citizens of a smart city providing they become users of the Red Swarm.

In our work we have assumed that all drivers have a terminal (e.g. smartphone) and they follow the routes suggested by the system. We expect a reduction in the average improvement on travel times if a significant percentage of drivers do not follow the indications of Red Swarm; the gradual penetration of the system will be the next step in our research. Moreover, unexpected changes in the state of the city, such as accidents or suddenly closed streets, have to be addressed in future work so as to be able to change the active configuration of the spots according to the new situation. A city is a very complex organism, and so we need to move step by step, gradually incorporating layers of human behavior and technologies.

Altogether, we think that our Red Swarm is a solution which could be used in current modern cities in order to reduce traffic jams and improve the citizens' quality of life. Moreover, it could be utilized to collect anonymous information from cars allowing local authorities to better understand the online and historical state of the city.



UNIVERSIDAD
DE MÁLAGA

Chapter 7

Green Swarm: Reducing Carbon Footprint

This chapter proposes a mobility architecture, called Green Swarm, to reduce greenhouse gas emissions from road traffic in smart cities. The traffic flow optimization of four European cities: Malaga, Stockholm, Berlin, and Paris, is addressed with new case studies importing each city's actual roads and traffic lights from OpenStreetMap into the SUMO traffic simulator, so as to find the best ways to redirect the traffic flow, and advise drivers. Additionally, the proposal is compared with three other strategies, which are also combined with Green Swarm in order to improve metrics such as travel times, gas emissions, and fuel consumption. This results in reductions in gas emissions as well as in travel times and fuel consumption in more than 500 city scenarios. The proposal has also been tested in scenarios where not all drivers are using it, to observe the change in traffic conditions when it is only in partial use, successfully paving the way for future sustainable cities.

7.1 Introduction

Another source of problems in big cities is air pollution and road traffic is a well-known source of greenhouse gas emissions in urban areas [101]. Poor air quality contributes to respiratory and cardiovascular diseases as well as to lung cancer [137] Air pollution is not only an important issue for the economy, the environment, and human health, but it also damages cities' buildings and has a clear impact on our climate, since some air pollutants behave as greenhouse gases [90].

Having observed the vehicles' behavior when developing the Red Swarm architecture for preventing traffic jams, we proposed in this chapter Green Swarm, an evolution of our preliminary work [199] redesigned and adapted in this case to reduce not only travel times, but also greenhouse gas emissions, and fuel consumption.

7.2 The Green Swarm Architecture (GS)

The Green Swarm architecture (GS) [202], based on Red Swarm [203, 204, 205], is a significantly different line of research where a large number of weak points have been addressed, to finally design and implement a new system based on the following, new contributions:

1. GS uses a new mathematical function to measure the quality of the solutions. This generates a new search landscape where new algorithms and unseen performances are analyzed.
2. The algorithms (now EfRA and GrA) have been revised and their performance improved to get better results in shorter times.
3. There is a study of the relationships between metrics (travel time, CO₂, fuel, etc.).
4. Four different cities (Malaga, Stockholm, Berlin and Paris) have been optimized, plus one extra scenario consisting of real traffic flows (Alameda) which amounts to more than 500 scenarios in five case studies. In previous work just one city or parts of it has been tested. The conclusions drawn from working with four cities give this study a robust endorsement as a comparison analysis for future work in this area.
5. We have pushed the boundaries of existing algorithms by significantly increasing the number of vehicles in each scenario (up to 5800 vs. 1200). This means considerably larger computational times, and improved realism.
6. There were no competitors whatsoever in past articles of the literature. Thus three competitors have been introduced, not only to test our strategy, but also to complement it. In this sense, our new proposal has been strongly tested compared to related systems.
7. As not everyone is keen on using new technologies until they are firmly established, in our experimentation a user acceptance study has been conducted so as to address not only the scientific aspect of the proposal but also the social one.

GS can be installed in modern cities with a minimum investment as it is able to use already existing infrastructure such as traffic lights controlled by a computer, Wi-Fi connectivity, mobile phones, and tablets. GS comprises the following components: i) Nodes installed at traffic lights which communicate with vehicles to know their destination and send them a new route around the city; ii) The Eco-friendly Route Algorithm (EfRA) which calculates the configuration of the system; iii) The Green Algorithm (GrA) which is executed in the nodes to suggest eco-friendly routes to vehicles; and iv) Mobile devices such as smartphones and tablets for the user terminals, or even On Board Units (OBU) installed in vehicles.

In Figure 7.1 the schema of the GS architecture is depicted. It is divided into two stages: an offline stage called *Setup Stage* and an online stage called *Green Stage*. In the *Setup Stage*, the configuration of the nodes is calculated by EfRA so that each node will be able to suggest an eco-friendly route to a vehicle depending on its final destination, based on a probability value. These probabilities are calculated before deploying the system (training phase) by optimizing four different traffic distributions of the city, as this improves the robustness of

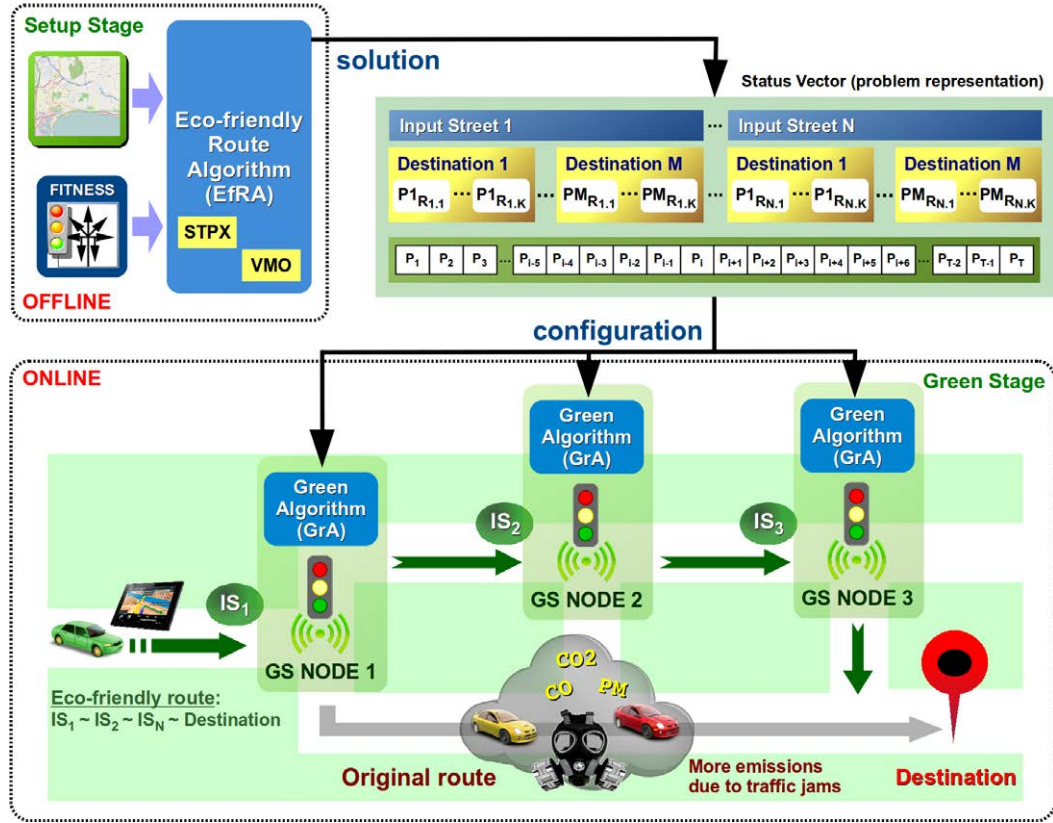


Figure 7.1: Green Swarm Architecture. In the *Setup Stage* the configuration of the GS nodes is calculated using the EfRA and then, in the *Green Stage*, vehicles are rerouted by the GrA to prevent traffic jams.

GS [203]. In doing so, EfRA is provided with more diverse real situations to help make better decisions when evolving the population toward configurations that are adjusted to more general solutions instead of a specific one.

In the *Green Stage*, vehicles connect to the GS nodes as they pass by, which triggers the execution of the GrA. Then, GrA suggests a new route for the vehicles according to the configuration calculated by EfRA in the previous stage. These new routes are customized as they are determined by the final destination of each vehicle.

Each node is implemented using a Wi-Fi spot connected to a processing unit capable of running the GrA. Additionally, they can be remotely updated (via the mobile network or the already existing connectivity found in traffic lights) to change the GS configuration in the case of possibly closed streets, events, etc. The software running in the mobile devices consists of a navigator-like screen with a graphical user interface for entering the driver's destination. Finally, the communication between a device and a node implies the former sending the desired destination and the latter answering with the route to the next GS node or to the driver's destination. According to [225] we estimate an operational radius for each node of 77 meters.

The placement of the nodes has been manually set for this study as it represents a challenge in itself which needs and justifies a future, separate, scientific article. The main

goal is: given a set of the more congested junctions controlled by a traffic light, identify those which better improve the rerouting of vehicles, preventing traffic jams, and use them as Green Swarm nodes.

An example of the rerouting performed by GS during the *Green Stage* is shown in Figure 7.1. When the vehicle connects to *Node 1* via a Wi-Fi link, the GrA suggests a new route toward *Node 2*, potentially different from the original one. It is assumed here that the driver accepts the new route, so that when he/she approaches *Node 2* by Input Street IS_2 , the vehicle will be routed to *Node 3*. Finally, in *Node 3*, which is near the vehicle's destination, the driver will be sent directly (no intermediate node) to the end of his/her journey.

By using GS the vehicle has probably traveled a longer distance than when following the shortest path (which is usually the default choice made by drivers), but it has avoided possible traffic jams while driving in an eco-friendly way. As a result, the amount of gas emitted into the atmosphere and travel times have both been reduced. Even if this seems not to be an intuitive result, it is demonstrated that taking into consideration the global flow and driving events, leads to a greener trip in the end. In order to evaluate each case study, the traffic simulator SUMO [123] has been used. SUMO implements realistic car following models and it can be externally controlled by TraCI [237] to perform the reroutings suggested by the GrA.

7.2.1 Eco-friendly Route Algorithm (EfRA)

What is being attempted in this study is finding a solution to a very difficult real problem requiring high evaluation times and managing large vectors of numbers which encompass a huge search space, very hard to explore by exhaustive methods. Furthermore, there is no analytic equation, so traditional methods are not viable. In addition, low complexity operations as used in metaheuristics are needed. All these reasons make this problem suitable for solving with a bio-inspired algorithm [29]. Concretely, we have designed a new evolutionary algorithm, based on a (10+2)-EA [15] and called Eco-friendly Route Algorithm.

EfRA is an elitist steady state EA, with a population of ten individuals, generating two new individuals at each step, mainly because the evaluation of each individual requires a simulation which takes more than 30 seconds to complete. EfRA is a light-weight algorithm (compared to other metaheuristics like common EAs, PSOs, etc.), it performs well without the need for an analytic equation which is impossible in this domain.

First, in EfRA (Algorithm 7.1), the number of steps t is set to zero and the population $P(0)$ (10 individuals) is initialized with random values. Then, while the termination condition is not fulfilled the main loop is executed. In our experiments EfRA ends when the maximum number of steps (5000) or the convergence criterion (500 generations without improvements) are reached. Inside the main loop, after initializing the auxiliary population $Q(0)$, two parents are selected from the population by using binary tournament [87]. Next, the offspring (two individuals) are obtained after applying the recombination operator (STPX) and after that, the offspring are mutated by applying our Variable Mutation Operator (VMO), both described later. Then, the new individuals are evaluated and inserted in the auxiliary population $Q(t)$.

Algorithm 7.1 Eco-friendly Route Algorithm (EfRA).**procedure** EfRA $t \leftarrow 0$ $P(0) \leftarrow \text{createPopulation}()$

▷ P = population

while not *terminationCondition*() **do** $Q(0) \leftarrow \emptyset$

▷ Q = auxiliary population

 $\text{parents} \leftarrow \text{selection}(P(t))$

▷ Binary tournament

 $\text{offspring} \leftarrow \text{STPX}(P_c, \text{parents})$

▷ Street Two Point Crossover

 $\text{offspring} \leftarrow \text{VMO}(\pi_1, \pi_2, \theta, \text{offspring})$

▷ Variable Mutation Operator

 $\text{evaluateFitness}(\text{offspring})$ $\text{insert}(\text{offspring}, Q(t))$ $P(t+1) \leftarrow \text{replace}(Q(t), P(t))$

▷ Elitist replacement

 $t \leftarrow t + 1$ **end while****end procedure**

Finally, the new population $P(t+1)$ is generated by replacing the current one ($P(t)$) with the individuals of the auxiliary one ($Q(t)$) in an elitist way, that is, the worst individuals in $P(t)$ (highest fitness values) are replaced by the individuals in $Q(t)$ if and only if the new ones have better (lower) fitness values and they are not yet in the population.

Representation

The goal is to suggest routes to vehicles as they are approaching a junction controlled by a GS node, so the different probabilities for each route need to be stored in a configuration vector. These probabilities are computed by an intelligent automatic technique according to the layout and dynamic features of the traffic in the city: our EfRA. Additionally, the suggested routes have to be personalized for each driver depending on his/her destination. For this reason, the route probabilities have to be separated into groups (chunks) assigned to each destination.

The problem representation chosen is shown in Figure 7.1 where it can be seen blocks of routes starting in the same street (Input Street 1, Input Street 2, etc.) which are inputs to a junction controlled by a GS node. These input streets are the points where the rerouting takes place (providing that the driver takes into account the suggestion given). Then, the available routes are replicated in several destination chunks in the same street block so as to personalize the trip based on drivers' destinations.

Finally, each route has a probability value associated with it, to define how likely it is to be suggested to drivers. Note that the summation of probability values in the same chunk must be equal to 1.

Evaluation Function

According to our experimentation, explained in Section 7.5.1, several relationships have been observed between the metrics, which has led to only CO₂ being included in the evaluation function to calculate the fitness value of the individuals.

The fitness function for EfRA is presented in Equation 7.1 where two terms can be seen. The first is meant to penalize the individuals representing configurations for the GS nodes which are unable to route all vehicles to their destination within the analysis time. Therefore, N is the total number of vehicles and n is the number of vehicles which have completed their itineraries, so we penalize the resulting fitness value with the number of vehicles which are inside the area under analysis when the analysis time ends.

$$F = (N - n) + \alpha^{-1} \frac{1}{n} \sum_{i=1}^n CO_{2_i} \quad (7.1)$$

The second term of Equation 7.1 represents the average CO₂ emissions from vehicles. It is normalized by the α coefficient calculated as shown in Equation 7.2. There, λ represents the number of training scenarios (four in this study), and n_i is the number of vehicles in the training scenario i . By using α in Equation 7.1 the fitness function is normalized, so that the experts' solution we are improving has a fitness value equal to 1. As the idea is to minimize, values below 1 represent an improvement over the experts' solution, i.e. the lower, the better.

$$\alpha = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \frac{1}{n_i} \sum_{j=1}^{n_i} CO_{2_{ij}} \quad (7.2)$$

Evolutionary Operators

Some of the operators tested in Chapter 6 have again been used in this new problem. Binary Tournament is used as the selection operator; Street Two Point Crossover (STPX) (Section 6.2.1) as the recombination operator, where the cross points are blocks of input streets' configurations; Variable Mutation Operator (VMO) (Section 6.2.1) where two different mutation probability values are combined using a threshold value to switch between them (the first value is meant to explore the search space whilst the second is to refine the solution by exploitation); and Elitism in the replacement operator. The recombination probability value used is 0.6, the threshold $\theta = 1.0$, and the mutation probabilities are $\pi_1 = 0.04$ and $\pi_2 = 0.01$. Table 7.1 shows a summary of the parameters of EfRA.

Table 7.1: Parameters of the EfRA.

| Parameter | Value |
|---|------------|
| Maximum iterations | 5000 |
| Crossover probability (P_C) | 0.6 |
| Mutation probabilities (π_1, π_2) | 0.04, 0.01 |
| Threshold (θ) | 1.0 |

7.2.2 Green Algorithm (GrA)

Our GrA runs in each GS node. When a vehicle connects with a node via Wi-Fi, GrA reads the configuration previously calculated by EfRA for this node and suggests an alternative route based on the probability values and the vehicle's destination (*Green Stage*). Even though the GrA cannot guarantee that each vehicle will reach its destination (as each spot is only responsible for a section of the whole route), the evolution of the configurations in EfRA toward an optimum makes it highly likely that each vehicle will reach its final destination.

The pseudocode of GrA is presented in Algorithm 7.2. First, the current street and the vehicle's destination zone are obtained from the approaching vehicle itself. Second, the destination zone is checked to avoid rerouting a vehicle already in it. If the vehicle has not yet reached its destination zone, all the routes from the current street to the vehicle's destination zone are considered in the GS configuration. If the destination zone is not directly reachable from the current street ($route = \emptyset$), the algorithm obtains the next Input Street (belonging to another node) which is directly reachable from the vehicle's current street so that it is rerouted to another GS node. This Input Street is selected based on the probabilities stored in the GS configuration. Finally, the route from the current to the next Input Street is suggested to the vehicle in the last step.

Algorithm 7.2 Green Algorithm (GrA).

```

procedure GRA(vehicle)
  current_street  $\leftarrow$  getStreet(vehicle)
  destination  $\leftarrow$  getDestinationZone(vehicle)
  if current_street  $\in$  destination then                                 $\triangleright$  At destination?
    route  $\leftarrow$  getCurrentRoute(vehicle)
  else
    route  $\leftarrow$  getRouteToDestination(current_street, destination)
    if route =  $\emptyset$  then                                               $\triangleright$  Rerouting to the next GS node
      nextInputStreet  $\leftarrow$  getStreetByProbability(current_street, destination)
      route  $\leftarrow$  getRouteToInputStreet(nextInputStreet)
    end if
  end if
  suggestNewRoute(route, vehicle)
end procedure

```

7.3 Case Studies

For this approach four large European cities: Malaga (Spain), Stockholm (Sweden), Berlin (Germany), and Paris (France) were chosen. This enabled the study of specific zones which are prone to traffic jams, with the aim of improving traffic flow and reducing gas emissions. Furthermore, a reduced area of Malaga (Alameda Principal) was also studied, where real traffic conditions could be faithfully recreated and the accuracy of the study, improved.

First, the GS system was applied to a small case study (0.4 km^2) comprising the *Alameda Principal* area of Malaga (Spain). In this case study data published by the local council for peak time traffic at 2 p.m. on working days [13] was used and its real traffic flows were generated by using our Flow Generator Algorithm [197, 201] described in Chapter 5. Second, four new and larger geographical areas were used, representing zones not only in the city of Malaga, but also in three major European cities: Stockholm, Berlin, and Paris which are all shown in Figure 7.2.

In spite of the fact that the real number of vehicles in the larger areas was unavailable, they were included to test our proposal against different cities, urban maps, and traffic distributions. By doing so, a real case study to validate GS was addressed, and then a variety of new case studies were analyzed (generalization and robustness).

To build each new case, the geographical areas in OpenStreetMap [169] were first selected and then exported to individual map files (.osm files). The maps were then modified by using the application JOSM (Java Open Street Map) thereby removing unhelpful, irrelevant data such as parks, housing blocks, and pedestrian walkways. Based on these, the working maps for SUMO were generated by NETCONVERT following the steps described in Section 4.3.2.

Finally, the traffic flows between the streets were defined using the DUAROUTER utility and used as the inputs to the areas being analyzed (source streets) and the streets which are destinations. Each flow contained several routes representing different, alternative paths between the same source and destination pairs. These were obtained by using the different weight metrics available in DUAROUTER such as travel times, emissions, and fuel consumption. By using these flows the difficulty of the problem being addressed increased, as vehicles do not always take the same routes toward their destination.

Other different case studies where vehicles are actually taking the fastest routes in the cities under consideration have also been included. This was done so as to also address a more realistic problem (people usually drive along avenues). The traffic light cycles were assigned by NETCONVERT while generating the map, using the algorithms included in SUMO. However, some corrections, especially in the lights' synchronization, were made to avoid problems of misconfigured cycles.

Wishing to provide a more realistic study, four different vehicle types were used (Table 7.2) having different emission classes from the HBEFA [96] model, as it would not make sense to have sedans and trucks emitting the same amount of gas nor consuming the same liters per kilometer. The arrival probability states that half of the vehicles are sedans while the rest are less common as they are heavier, which is to be expected in a city center.

Table 7.2: Characteristics of the four types of vehicles.

| Type | Arrival Prob. | MaxSpd. (km/h) | Accel. (m/s^2) | Decel. (m/s^2) | Length (m) | Emission class |
|-------|---------------|----------------|---------------------------|---------------------------|------------|----------------|
| sedan | 0.50 | 160 | 0.9 | 5.0 | 3.8 | P_7_7 |
| van | 0.25 | 100 | 0.8 | 4.5 | 4.2 | P_7_5 |
| wagon | 0.15 | 50 | 0.7 | 4.0 | 4.3 | P_7_6 |
| truck | 0.10 | 40 | 0.6 | 3.5 | 4.5 | HDV_3_1 |



Figure 7.2: Case studies: *Alameda* (ALA), *Malaga* (MGA), *Stockholm* (STO), *Berlin* (BER), and *Paris* (PAR), imported from OpenStreetMap into the SUMO traffic simulator.

In each working scenario, vehicles arrive at different times, through different streets and taking different routes, which generates a variety of situations to train and test the proposal. Since the assigned vehicles' type and route depend on the random number generator included

Table 7.3: Characteristics of the case studies: *Alameda* (ALA), *Malaga* (MGA), *Stockholm* (STO), *Berlin* (BER), and *Paris* (PAR). The number of probability values in the solution denotes the complexity of each city.

| Case study | ALA | MGA | STO | BER | PAR |
|---------------------------------|------|------|------|------|------|
| Analysis time (s) | 3600 | | | | |
| # Vehicles | 4104 | 4700 | 4600 | 5800 | 5700 |
| # Traffic lights | 28 | 89 | 75 | 76 | 58 |
| # GS nodes | 3 | 7 | 6 | 6 | 4 |
| # Input streets | 9 | 17 | 14 | 24 | 16 |
| # Vehicle types | 4 | 4 | 4 | 4 | 4 |
| # Source streets | 4 | 26 | 21 | 16 | 15 |
| # Vehicle flows | 4 | 25 | 14 | 16 | 15 |
| # Vehicle routes | 15 | 430 | 196 | 229 | 210 |
| Studied area (km ²) | 0.4 | 10.0 | 2.9 | 7.0 | 5.6 |
| # Probability values | 168 | 840 | 1314 | 450 | 732 |

in SUMO, by changing the simulation seed the different scenarios were defined for each case study. These mobility solutions based on traffic distributions are called the experts' solution as they were generated by the SUMO tools. The characteristics of the case studies are presented in Table 7.3. All of them were analyzed for one hour, while the rest of the characteristics were dependent on the road distribution obtained from OpenStreetMap as well as the size of the geographical area. Note that we give the name *Input Street* to the different streets which are, in fact, inputs to a junction controlled by a GS node, and *Source Street* to each street by which the vehicles enter the case study being analyzed. Each case study is described as follows.

7.3.1 Alameda (ALA)

The geographical area selected from the city of Malaga to build the case study called *Alameda*, is delimited to the north by *Calle Carretería*, to the south and east by *Avenida de Manuel Agustín Heredia*, and to the west by *Avenida del Comandante Benítez*, which encompasses an area of about 0.4 km². We defined 15 routes arranged in 4 different vehicle flows which start at a Source Street by using DUAROUTER. Then, when each one of the 4104 vehicles of the case study enter the area under analysis, one of these routes is assigned depending on the flows calculated by the FGA (Chapter 5).

7.3.2 Malaga (MGA)

The area of *Malaga* analyzed is about 10 km². It is delimited to the north by the *Autopista del Mediterráneo*, to the south by the Mediterranean Sea, to the east by *Paseo de Cerrado de Calderón*, and to the west by the *Guadamedina* River. There are 430 routes arranged in 25 different vehicle flows in *Malaga* which are assigned to 4700 vehicles as they enter this geographical area.

7.3.3 Stockholm (STO)

Our *Stockholm* case study consists of 196 routes arranged in 14 different vehicle flows to analyze the behavior of 4600 vehicles in a geographical area delimited to the north by *Odengatan*, to the south by *Riddarfjärden*, to the east by *Birger Jarlsgatan*, and to the west by *Klarastrandsleden*. The total area included is about 2.9 km².

7.3.4 Berlin (BER)

The geographical area of *Berlin* is delimited to the north by *Oranienstraße*, to the south by *Columbiadamm*, to the east by *Kottbusser Damm*, and to the west by *Potsdamer Straße*, which encompasses an area of about 7 km². In this case study we define 229 different vehicle routes arranged in 16 flows and experiment with 5800 vehicles.

7.3.5 Paris (PAR)

The area chosen from the city of *Paris* is delimited to the north by *Avenue Villiers*, to the south by *Cours Albert 1er*, to the east by *Avenue Franklin Delano Roosevelt*, and to the west by *Boulevard Pershing* and *Boulevard Lannes*. There are 15 vehicle flows which contain 210 routes defined in our *Paris* case study as well as 5700 vehicles. The total area of *Paris* analyzed is about 5.6 km².

7.4 Competitor Techniques

Although comparing a contribution to existing competitors is a must in science, research papers in this area frequently do not consider competitor systems. The reason is not only the difficulty of finding closely similar work, but also that it is very difficult to find and manage studies reporting so many technological tools, open data and algorithms. Notwithstanding, an effort has been made to compare our proposal with others by including several competitors

Consequently, three different strategies presented in [142] in order to reduce local traffic emissions have been chosen: i) reducing traffic demand by 20% (-20%), ii) introducing a speed limit of 30 km/h (30km/h), and iii) replacing Heavy Duty Vehicles (HDV) with 1.5 Light Duty Vehicles (LDV). These strategies may seem at first glance to be trivial as they are not based on optimization. However, they are widely applied by local councils, particularly when the pollution levels are so high that people's health is put at risk [221].

The authors of the aforementioned article tested these strategies in a single intersection located in Bentinckplein in the city of Rotterdam, the Netherlands. Although they achieved reductions in emissions of between 13% and 30% depending on the strategy and metrics used, they analyzed only one intersection instead of large districts. This encouraged us to test those strategies in our case studies as an additional contribution. In the following paragraphs, we describe the modifications applied to the traffic demand implemented by each strategy. The routes taken by vehicles are the same as those used in the experts' solution to ensure a fair comparison.

7.4.1 Minus 20 % (-20 %)

A reduction in the number of vehicles of 20% has been implemented, while keeping the original proportion of vehicle types. The result was 3282 vehicles in ALA, 3760 in MGA, 3680 in STO, 4640 in BER, and 4560 in PAR.

7.4.2 Maximum 30 km/h (30km/h)

In this strategy the traffic demand is the same as in the expert's solution (number and types of vehicles and their routes). However, the maximum speed has been restricted to 30 km/h for all the vehicles.

7.4.3 HDV-LDV

Finally, the HDV-LDV strategy consists in replacing trucks, which have the worst emission class of all the vehicle types studied, with 1.5 light duty vehicles (sedan, van, and wagon). As HDV traffic (trucks) is 10% of our demand, after applying this strategy the number of vehicles in the case study *Alameda* (ALA) is: $4104 \times 0.9 = 3694$; $3694 + 4104 \times 0.15 = 4308$. Note that this represents an increase in demand of approximately 5% so that the number of vehicles in the rest of the case studies is 4930 in MGA, 4830 in STO, 6086 in BER, and 5985 in PAR.

7.5 Experimentation

First, several experiments were conducted to determine which metrics were best for inclusion in the evaluation function (Section 7.5.1). Second, the optimization of one case study was addressed, consisting of real traffic flows obtained from data published by the council of Malaga for the main streets included in the case study called *Alameda*.

Then, four other case studies were optimized, where vehicles used various, different routes between their origin and destination (Section 7.5.2). At this point, our proposal was compared with different, state of the art strategies where the behavior of GS when it is used after applying the other strategies was evaluated. This allowed us to know if they were compatible and if the metrics could be reduced even more (Section 7.5.3).

The best configuration obtained for GS in the previous experiments was tested in 500 unseen scenarios where vehicles either followed a number of different routes (more difficult to optimize) or just the fastest ones (a situation closer to reality). The other strategies were also included at this point and a combination of them were tested with GS in 1500 scenarios (Section 7.5.4). Finally, a study was done to analyze how GS behaves when only a certain percentage of people are using it (Section 7.5.5), followed by a discussion on the GrA performance (Section 7.5.6).

7.5.1 Metric Study

An initial series of tests were conducted to evaluate which emission metrics were more suitable for optimization. We exploited the case study called *Alameda* (ALA) because its size makes the analysis more affordable (in time) than the rest of the bigger scenarios. Moreover, we think the analysis of just two different scenarios of ALA still being valid and do not lose generality in the conclusions.

Four hundred and twenty runs were carried out, which lasted 31.2 hours on average. From the results it could be observed that EfRA was able to reach the same optimal configuration in each optimization process when using different metrics in the evaluation function (CO_2 , Fuel, $\text{CO}_2 + \text{Fuel}$, $\text{CO} + \text{CO}_2$, $\text{CO} + \text{HC}$, $\text{PM} + \text{HC}$, and $\text{CO} + \text{CO}_2 + \text{NO}_x$). Based on these results, CO_2 was chosen as the metric to be optimized, because not only is it a well-known gas causing global warming, but also because it keeps the evaluation function simple.

Figure 7.3 presents the graphs of the different metrics vs. CO_2 from 16416 vehicles (4 scenarios of *Alameda*) in order to visualize and confirm the similarities between them. The majority of the graphs show different slopes which correspond to the different emission classes of vehicles. Some of them are mostly coincident, especially in the case of the fuel consumption, where its linear relation with CO_2 is evident. This fact supports even further the decision made in respect to the variable (CO_2) evaluated to calculate the fitness value of our scenarios, as the rest of the metrics are reduced when reducing the CO_2 emissions.

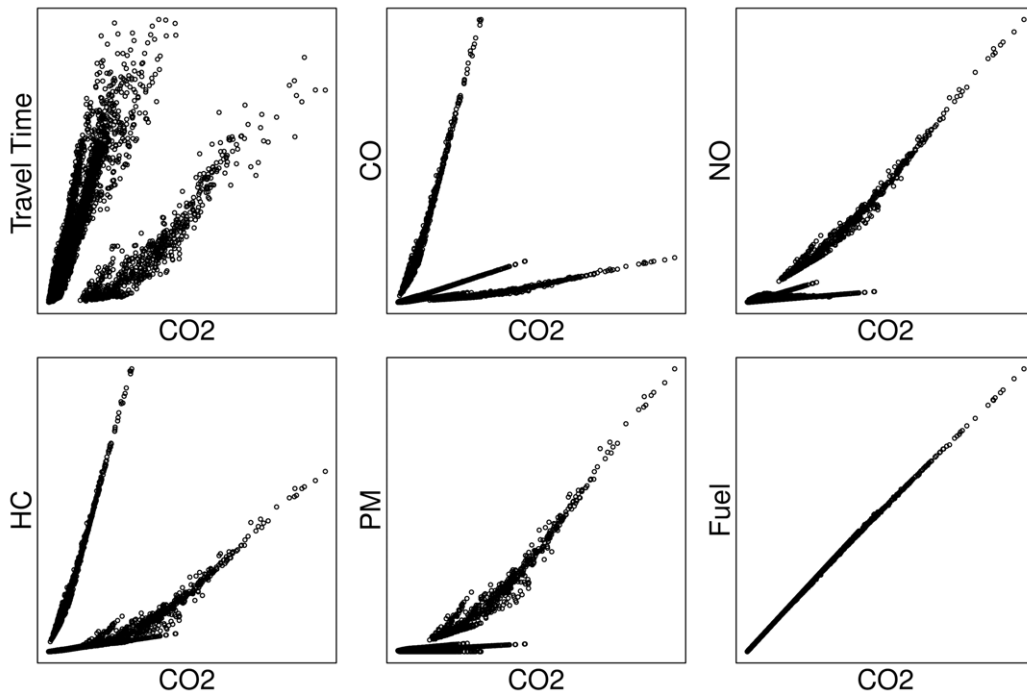


Figure 7.3: Similarities between CO_2 and the rest of the metrics. Different slopes correspond to the different emission classes of vehicles. Note that fuel consumption presents a linear relation with CO_2 .

7.5.2 Optimization

In this section four training scenarios are optimized for each one of the five case studies. Each scenario presents different traffic distributions to EfRA so that the optimization processes can produce robust solutions [203]. We ended up with vectors of 168 probability values in ALA, 840 in MGA, 1314 in STO, 450 in BER, and 732 in PAR, after generating the scenarios, which represents the high complexity of this problem. Thirty independent runs of EfRA were carried out to optimize each case study and the results are presented in Table 7.4 (GS strategy).

Table 7.4: Improvements in the experts' solution achieved by the strategies used to optimize our five case studies. We report here all the metrics despite having optimized only CO₂ to observe how they are reduced as well. These results correspond to the scenarios used during the optimization. The best performing strategies are in bold.

| Metric | Strategy | ALA | MGA | STO | BER | PAR |
|-----------------|----------|--------------|--------------|--------------|--------------|--------------|
| T.Time | GS | 69.7% | 18.7% | 41.7% | 19.0% | 10.2% |
| | -20% | 15.8% | 25.0% | 33.0% | 33.0% | 37.8% |
| | 30km/h | -5.0% | -12.5% | -10.3% | -12.8% | -22.7% |
| | HDV-LDV | -4.8% | 0.7% | -4.1% | -2.8% | -5.1% |
| CO | GS | 56.7% | 10.6% | 31.8% | 12.8% | 7.9% |
| | -20% | 11.3% | 15.3% | 25.0% | 23.9% | 23.6% |
| | 30km/h | 4.6% | 17.6% | 5.9% | 10.2% | 8.0% |
| | HDV-LDV | -14.3% | -6.2% | -15.8% | -7.9% | -13.7% |
| CO ₂ | GS | 36.6% | 5.3% | 15.1% | 5.2% | 3.6% |
| | -20% | 7.8% | 7.8% | 13.4% | 12.4% | 12.5% |
| | 30km/h | 6.0% | 10.2% | 6.4% | 7.2% | 9.3% |
| | HDV-LDV | 25.8% | 35.7% | 30.6% | 31.7% | 31.8% |
| HC | GS | 54.3% | 9.4% | 29.3% | 10.8% | 7.3% |
| | -20% | 10.3% | 13.5% | 23.2% | 21.2% | 22.0% |
| | 30km/h | 0.5% | 4.2% | 0.1% | 0.3% | -3.6% |
| | HDV-LDV | 1.9% | 1.1% | -1.5% | 2.6% | -2.4% |
| PM | GS | 47.6% | 8.0% | 24.6% | 8.7% | 5.7% |
| | -20% | 8.6% | 10.5% | 20.6% | 18.0% | 18.2% |
| | 30km/h | 2.1% | 8.1% | 4.2% | 4.4% | 3.6% |
| | HDV-LDV | 75.9% | 68.2% | 74.2% | 70.0% | 69.4% |
| NO _x | GS | 35.0% | 5.4% | 15.4% | 4.8% | 3.8% |
| | -20% | 6.3% | 7.3% | 13.4% | 11.8% | 12.1% |
| | 30km/h | 5.4% | 10.5% | 7.1% | 7.6% | 9.1% |
| | HDV-LDV | 66.7% | 63.5% | 65.6% | 63.2% | 63.2% |
| Fuel | GS | 36.3% | 5.2% | 14.8% | 5.1% | 3.6% |
| | -20% | 7.8% | 7.7% | 13.2% | 12.2% | 12.4% |
| | 30km/h | 6.1% | 10.2% | 6.5% | 7.3% | 9.5% |
| | HDV-LDV | 25.3% | 35.6% | 30.3% | 31.5% | 31.7% |

GS achieves improvements in all the metrics and in all the cities. The results are especially interesting in ALA, where there exists a real traffic challenge with a large number of vehicles in a reduced area. There, GS has shortened travel times by 70%, reduced CO emissions by 57%, CO₂ by 37%, and fuel consumption by 36% on average. In MAL, GS achieves 19% shorter travel times and a reduction in CO of 11%. Moreover, it can be seen that in STO there are important reductions in travel times (42%) and emissions (32% in CO and 29% in HC) when using GS. Vehicles driving through BER benefit from 19% shorter travel times when using GS and emit 13% less CO and 11% less HC in the atmosphere, on average. Finally, the best, improved metrics in PAR are travel times (10%), CO (8%), and HC (7%).

As a consequence of the rerouting strategy, some drivers have individually experienced longer travel times. Concretely, 25% of drivers have longer travel times in ALA, 38% in MAL, 39% in STO, and 47% in BER and PAR. This is a low price to pay for achieving global reductions of travel times and gas emissions in the city, especially if we take into account that it is not likely that the same drivers are penalized every day.

Next, the three competitor strategies were implemented as described in Section 7.4. Then, they were applied to our case studies (again the same four scenarios of each) to obtain improvements in each metric, also presented in Table 7.4. As can be seen, the improvements vary notably among the metrics and scenarios, which makes it difficult to conclude which strategy is the best one.

Nevertheless, looking at the different strategies it can be appreciated that a reduction in the number of vehicles (-20%) has a positive impact on travel times as there are fewer vehicles on the streets of our case studies. Reducing the number of vehicles has worked well, especially in the reduction of CO, HC, and PM emissions. This strategy seems to achieve similar results to GS: the former reduces the number of vehicles directly while the latter reroutes them via alternative streets without restricting the drivers.

Fixing the maximum speed at 30 km/h has turned out to be the least effective measure to reduce emissions, demonstrating the worst travel times as well. All in all, the reduction of emissions is quite low in most of the cases, except for the case study MAL. Paradoxically, this is the method applied by the majority of city authorities when the pollution levels are high [221]. Our conclusions in this matter are in keeping with those discussed in [109] where the authors illustrate the scientific uncertainties inherent in implementing speed management policies [120].

Replacing trucks with sedans, vans, and wagons (HDV-LDV) enables a huge reduction of CO₂, PM, and NO_x emissions, as they are the main gases emitted by trucks according to the HBEFA class selected for this type of vehicle (HDV_3_1). Furthermore, this strategy also reduces fuel consumption which is directly related to CO₂ emissions as we have stated in Section 7.5.1. The HDV-LDV strategy is a serious competitor to our system (emissions), but it definitely has a negative impact on the economy of the city, as it is difficult to implement, and will definitely incur protests. Our system is smoother and simultaneously more efficient with shorter travel times whereas the other strategies show longer ones (negative improvements).

Our conclusion after this study is that despite the fact that some competitor strategies perform better in some case studies, GS has competitive results. We must keep in mind that we do not restrict the number, type, or speed of vehicles which would not be desirable

or even viable in many cities in the world. Therefore, the next step taken was to optimize the same four training scenarios after applying the competitor strategies to know how GS behaves under these new conditions.

7.5.3 Green Swarm Combined with Other Strategies

In this section the combination of GS with other strategies is studied to discover not only if they are viable but also if the strategies achieve better results when are applied together.

We took the traffic distributions obtained when the -20%, 30km/h, and HDV-LDV strategies were applied in our training scenarios and applied GS as the optimization algorithm in order to analyze how they combine with each other and see if some metrics could be improved even further. After performing a further 30 independent runs of the EfRA in four scenarios of our five case studies (150 runs per strategy) GS demonstrated the relative improvements over the other strategies shown in Table 7.5. At first glance, the best improvements are made when applying GS after limiting the vehicles' maximum speed (30km/h+GS). However, the most important conclusion here is that all the metrics have been improved by complementing the competitors with GS which, in our opinion, validates our proposal as a promising solution for improving the city's streets reducing travel times, greenhouse gas emissions and fuel consumption. Focusing on the numbers, the maximum improvements are nearly 50% in travel times, 45% in CO, 30% in CO₂, 41% in HC, 38% in PM, 30% in NO_x, and 30% in fuel consumption.

The total number of runs performed in the optimization processes (GS alone and combined with other strategies) was 600 and the time spent on each of them was, on average, between 19 and 92 hours as shown in Table 7.6. Note that the diversity of values depends not only on the case study and the number of vehicles but also on the heterogeneity of the hardware used to conduct the experiments (Intel Core 2 Quad Q9400 @ 2.66 GHz, Core i7 920 @ 2.67 GHz, and Xeon E5-2670v3 @ 2.30 GHz).

7.5.4 Study on Unseen Scenarios

After the aforementioned optimization processes we wanted to test GS in unseen scenarios. With this in mind, 50 new scenarios were generated for each city, where the vehicles followed a variety of routes to their destination and another 50 in which they just flowed via the fastest routes (subscript TT which stands for travel time). Then, the seven optimization strategies on these scenarios (700 in total) were tested. The results are shown in Table 7.7 where the average improvements achieved by each strategy in each case study and metric are displayed. The GS configurations previously obtained were used here, so no extra optimization process was needed. Note that GS was working in the *Green Stage* during the experiments conducted in this section.

It can be seen in Table 7.7 that GS has improved the other strategies in this study as well, even turning some of their results that were worse than the experts' solution into actual improvements. Most of the best performing strategies in each metric involve GS, either alone or applied after another strategy. The HDV-LDV strategy shows the best reductions of PM

Table 7.5: Relative improvements achieved by using GS after the other competitor strategies. These results correspond to the training scenarios used in the initial optimization. The best improvements are in bold.

| Metric | Strategy | ALA | MGA | STO | BER | PAR |
|-----------------|------------|--------------|--------------|--------------|--------------|--------------|
| T.Time | -20%+GS | 43.1% | 6.4% | 31.9% | 12.8% | 7.4% |
| | 30km/h+GS | 49.1% | 15.1% | 37.7% | 14.1% | 7.0% |
| | HDV-LDV+GS | 49.8% | 18.8% | 42.3% | 16.8% | 10.3% |
| CO | -20%+GS | 33.4% | 2.0% | 20.6% | 6.0% | 4.7% |
| | 30km/h+GS | 45.4% | 11.9% | 33.9% | 11.5% | 6.9% |
| | HDV-LDV+GS | 40.8% | 10.6% | 33.7% | 9.6% | 8.9% |
| CO ₂ | -20%+GS | 20.0% | 1.9% | 8.2% | 2.1% | 1.8% |
| | 30km/h+GS | 30.4% | 5.9% | 14.6% | 3.5% | 3.3% |
| | HDV-LDV+GS | 30.4% | 5.9% | 19.3% | 4.1% | 4.6% |
| HC | -20%+GS | 32.4% | 1.8% | 18.2% | 4.4% | 4.3% |
| | 30km/h+GS | 40.9% | 8.4% | 27.7% | 8.6% | 5.9% |
| | HDV-LDV+GS | 36.6% | 8.2% | 27.3% | 5.6% | 7.2% |
| PM | -20%+GS | 28.0% | 2.9% | 13.5% | 4.8% | 3.3% |
| | 30km/h+GS | 38.1% | 9.0% | 23.8% | 7.3% | 5.9% |
| | HDV-LDV+GS | 20.1% | 3.3% | 10.0% | 1.1% | 2.4% |
| NO _x | -20%+GS | 19.5% | 2.1% | 7.8% | 2.5% | 2.0% |
| | 30km/h+GS | 29.6% | 6.3% | 14.2% | 3.3% | 3.8% |
| | HDV-LDV+GS | 18.3% | 2.9% | 9.8% | 0.6% | 2.5% |
| Fuel | -20%+GS | 19.8% | 1.9% | 8.0% | 2.1% | 1.8% |
| | 30km/h+GS | 30.2% | 5.8% | 14.4% | 3.4% | 3.2% |
| | HDV-LDV+GS | 30.4% | 5.9% | 19.3% | 4.1% | 4.6% |

Table 7.6: Average time spent by 30 independent runs in the optimization process of each case study.

| Strategy | Average time (hours) | | | | |
|------------|----------------------|------|-------|-------|------|
| | ALA | MGA | STO | BER | PAR |
| GS | 10.3 | 23.0 | 32.2 | 32.3 | 88.7 |
| -20%+GS | 8.0 | 11.5 | 31.1 | 27.2 | 19.0 |
| 30km/h+GS | 56.1 | 20.2 | 32.4 | 126.9 | 47.1 |
| HDV-LDV+GS | 79.9 | 88.3 | 110.6 | 134.5 | 46.7 |

and NO_x on average, -20%+GS reduces the most CO, HC, and travel times on average, and HDV-LDV+GS achieves the biggest reductions in CO₂ and fuel consumption on average.

In Figure 7.4 a graphical comparison is given between strategies in each case study over six graphs for each metric. There, GS clearly performs especially well in our realistic congested case study (ALA) and it always presents a consistent improvement in all metrics. However, HDV-LDV and 30km/h encounter problems when improving travel times and reducing HC. HDV-LDV alone or combined with GS demonstrates the biggest reductions of CO₂, NO_x, and PM. Finally, we have calculated the Wilcoxon p -value to be sure that the improvements reported on each metric are statistically significant. In all cases the p -value obtained was less than 0.01, that is, a confidence level greater than 99%.

Table 7.7: Average improvement achieved by applying the seven strategies analyzed to 50 unseen scenarios of each case study (500 scenarios in total) during the *Green Stage*. Note that we have included scenarios where vehicles follow a variety of routes to their destination (no subscript) and others in which they just drive via the fastest routes (subscript TT). The best performing strategies in each case study are in bold.

| Metric | Strategy | ALA | MGA | STO | BER | PAR | ALA _{TT} | MGA _{TT} | STO _{TT} | BER _{TT} | PAR _{TT} | Average |
|-----------------|------------|--------------|--------------|--------------|--------------|--------------|-------------------|-------------------|-------------------|-------------------|-------------------|--------------|
| T.Time | GS | 67.8% | 14.5% | 37.8% | 15.0% | 7.1% | 63.5% | 23.0% | 59.6% | 10.3% | 15.3% | 31.4% |
| | -20% | 21.3% | 23.9% | 38.1% | 32.8% | 37.0% | 15.5% | 21.0% | 31.1% | 37.4% | 36.1% | 29.4% |
| | 30km/h | -2.6% | -13.3% | -5.0% | -13.1% | -23.3% | -3.8% | -13.0% | -8.7% | -9.9% | -16.1% | -10.9% |
| | HDV-LDV | 0.0% | -1.0% | -3.9% | -1.9% | -4.0% | -1.2% | -0.2% | -1.8% | -4.0% | -0.8% | -1.9% |
| | -20%+GS | 54.1% | 27.8% | 53.2% | 39.4% | 39.4% | 52.8% | 32.4% | 67.0% | 39.7% | 43.9% | 45.0% |
| | 30km/h+GS | 48.4% | 0.5% | 31.2% | -1.0% | -16.9% | 45.3% | 10.4% | 50.2% | -3.9% | 1.6% | 16.6% |
| | HDV-LDV+GS | 49.8% | 15.3% | 38.2% | 11.0% | 0.0% | -5.4% | 32.8% | 60.3% | 7.8% | 16.0% | 22.6% |
| CO | GS | 56.1% | 7.4% | 28.6% | 10.3% | 5.5% | 51.9% | 14.2% | 48.2% | 5.9% | 11.3% | 23.9% |
| | -20% | 18.6% | 14.5% | 30.0% | 23.7% | 23.3% | 12.2% | 13.6% | 26.5% | 25.0% | 25.2% | 21.3% |
| | 30km/h | 6.0% | 17.1% | 10.5% | 9.1% | 8.0% | 5.6% | 15.0% | 6.2% | 12.0% | 12.1% | 10.2% |
| | HDV-LDV | -10.7% | -9.6% | -13.8% | -11.2% | -12.4% | -11.7% | -9.6% | -12.5% | -13.4% | -9.9% | -11.5% |
| | -20%+GS | 43.4% | 15.7% | 40.0% | 27.5% | 25.3% | 41.7% | 20.0% | 53.5% | 25.8% | 31.0% | 32.4% |
| | 30km/h+GS | 49.9% | 25.0% | 37.9% | 16.9% | 12.4% | 47.1% | 30.6% | 54.5% | 14.8% | 26.6% | 31.6% |
| | HDV-LDV+GS | 34.1% | 0.1% | 22.4% | -1.8% | -8.6% | -22.2% | 21.8% | 44.1% | -5.2% | 4.8% | 8.9% |
| CO ₂ | GS | 36.2% | 3.1% | 13.3% | 3.2% | 2.1% | 33.1% | 6.5% | 25.7% | 1.2% | 3.6% | 12.8% |
| | -20% | 12.0% | 7.5% | 15.8% | 12.2% | 12.4% | 8.1% | 7.2% | 15.8% | 13.5% | 14.0% | 11.8% |
| | 30km/h | 6.9% | 10.5% | 8.8% | 6.4% | 8.9% | 7.0% | 9.6% | 7.7% | 8.2% | 10.3% | 8.4% |
| | HDV-LDV | 28.2% | 35.4% | 30.2% | 32.6% | 32.3% | 27.9% | 34.6% | 28.9% | 31.6% | 32.0% | 31.4% |
| | -20%+GS | 28.5% | 7.8% | 20.0% | 12.5% | 12.7% | 27.2% | 9.7% | 29.9% | 12.0% | 15.4% | 17.6% |
| | 30km/h+GS | 35.8% | 13.1% | 19.6% | 7.8% | 10.7% | 33.8% | 16.4% | 30.6% | 6.9% | 16.2% | 19.1% |
| | HDV-LDV+GS | 49.7% | 38.3% | 43.0% | 34.0% | 33.2% | 25.0% | 12.4% | 51.8% | 32.7% | 36.6% | 35.7% |
| HC | GS | 53.7% | 6.5% | 25.7% | 8.4% | 5.0% | 49.4% | 12.3% | 44.3% | 4.6% | 10.2% | 22.0% |
| | -20% | 17.8% | 12.7% | 27.3% | 21.2% | 21.6% | 11.9% | 12.1% | 25.1% | 23.0% | 23.8% | 19.6% |
| | 30km/h | 2.2% | 3.5% | 3.4% | -0.9% | -3.7% | 1.7% | 2.5% | -0.2% | 1.3% | 0.3% | 1.0% |
| | HDV-LDV | 4.6% | -2.0% | -0.5% | -0.2% | -1.4% | 3.7% | -0.8% | 2.0% | -1.5% | 1.9% | 0.6% |
| | -20%+GS | 41.9% | 13.7% | 36.3% | 23.8% | 23.5% | 40.0% | 17.4% | 49.5% | 22.6% | 29.1% | 29.8% |
| | 30km/h+GS | 43.4% | 9.7% | 27.0% | 5.2% | 0.5% | 40.2% | 15.5% | 43.2% | 2.8% | 13.6% | 20.1% |
| | HDV-LDV+GS | 39.4% | 4.9% | 25.0% | 4.7% | 1.1% | -3.3% | 19.4% | 43.2% | 2.6% | 12.1% | 14.9% |
| PM | GS | 46.6% | 5.2% | 20.5% | 5.7% | 3.9% | 42.4% | 9.9% | 36.7% | 3.0% | 7.7% | 18.2% |
| | -20% | 15.5% | 10.1% | 22.1% | 17.6% | 17.8% | 10.6% | 9.9% | 21.6% | 19.4% | 20.3% | 16.5% |
| | 30km/h | 3.8% | 8.4% | 5.5% | 3.1% | 2.8% | 3.8% | 7.3% | 3.8% | 5.9% | 5.5% | 5.0% |
| | HDV-LDV | 76.9% | 68.5% | 73.0% | 71.3% | 69.5% | 76.4% | 69.4% | 74.5% | 71.1% | 70.6% | 72.1% |
| | -20%+GS | 36.8% | 10.9% | 29.1% | 19.3% | 19.0% | 35.1% | 13.9% | 41.5% | 18.8% | 24.1% | 24.9% |
| | 30km/h+GS | 40.9% | 12.9% | 24.3% | 7.3% | 6.5% | 37.8% | 17.4% | 38.6% | 6.1% | 16.0% | 20.8% |
| | HDV-LDV+GS | 81.4% | 69.1% | 75.5% | 71.1% | 69.6% | 76.5% | 16.3% | 79.3% | 70.9% | 71.4% | 68.1% |
| NO _x | GS | 34.3% | 3.0% | 12.4% | 2.7% | 2.2% | 31.1% | 6.0% | 24.0% | 0.8% | 3.5% | 12.0% |
| | -20% | 11.6% | 7.1% | 14.8% | 11.6% | 11.8% | 7.9% | 6.8% | 15.5% | 13.1% | 14.0% | 11.4% |
| | 30km/h | 6.6% | 10.8% | 8.2% | 6.4% | 8.4% | 6.9% | 9.9% | 7.5% | 8.5% | 9.6% | 8.3% |
| | HDV-LDV | 67.7% | 63.2% | 64.9% | 64.0% | 63.3% | 67.4% | 63.5% | 65.4% | 63.9% | 63.8% | 64.7% |
| | -20%+GS | 27.2% | 7.3% | 18.6% | 11.8% | 12.1% | 25.8% | 9.0% | 28.0% | 11.4% | 15.3% | 16.7% |
| | 30km/h+GS | 34.4% | 13.1% | 18.3% | 7.4% | 10.4% | 32.2% | 16.1% | 28.6% | 6.8% | 15.4% | 18.3% |
| | HDV-LDV+GS | 73.5% | 63.9% | 68.0% | 63.6% | 63.4% | 66.9% | 11.8% | 71.6% | 63.4% | 64.8% | 61.1% |
| Fuel | GS | 35.9% | 3.1% | 13.1% | 3.2% | 2.1% | 32.8% | 6.5% | 25.3% | 1.2% | 3.5% | 12.7% |
| | -20% | 11.9% | 7.5% | 15.6% | 12.0% | 12.2% | 8.0% | 7.1% | 15.7% | 13.4% | 13.8% | 11.7% |
| | 30km/h | 7.0% | 10.5% | 8.9% | 6.4% | 9.0% | 7.1% | 9.7% | 7.8% | 8.2% | 10.4% | 8.5% |
| | HDV-LDV | 27.7% | 35.4% | 29.9% | 32.4% | 32.1% | 27.4% | 34.5% | 28.5% | 31.4% | 31.8% | 31.1% |
| | -20%+GS | 28.3% | 7.7% | 19.8% | 12.3% | 12.6% | 27.0% | 9.6% | 29.5% | 11.9% | 15.2% | 17.4% |
| | 30km/h+GS | 35.6% | 13.1% | 19.5% | 7.8% | 10.8% | 33.6% | 16.3% | 30.3% | 6.9% | 16.1% | 19.0% |
| | HDV-LDV+GS | 49.4% | 38.2% | 42.7% | 33.8% | 33.0% | 24.5% | 12.3% | 51.5% | 32.5% | 36.3% | 35.4% |

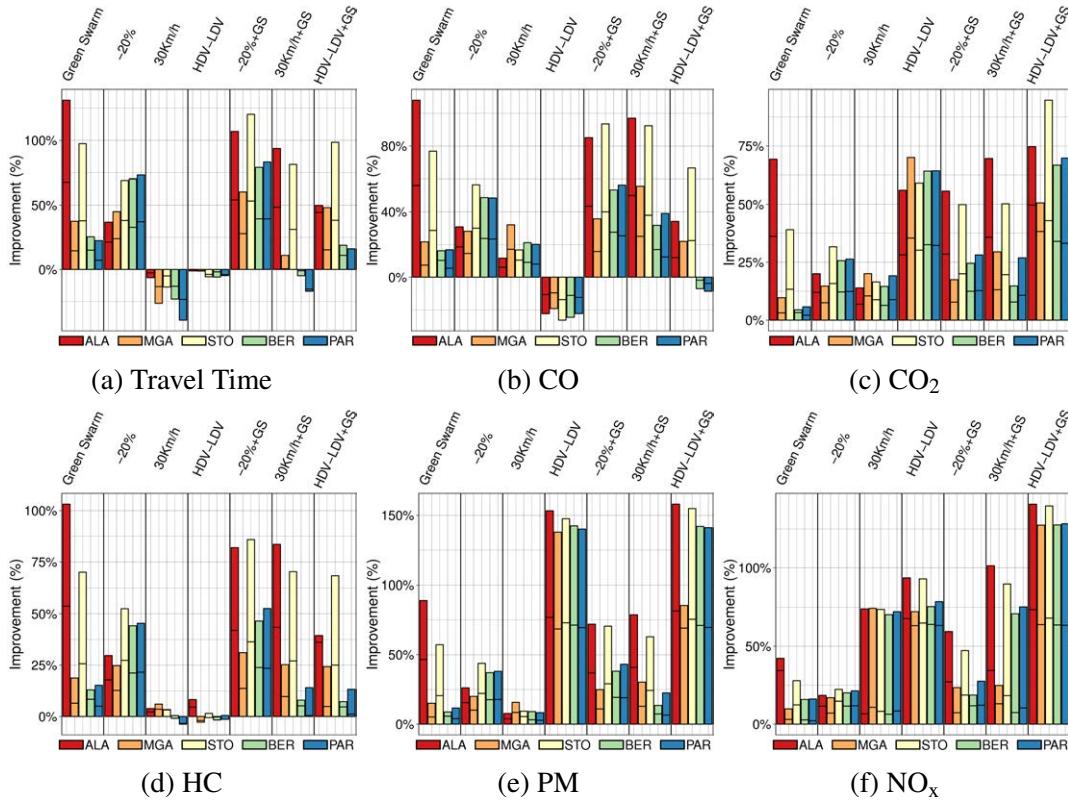


Figure 7.4: Average improvement of the strategies applied to 500 unseen scenarios (50 per case study). Note that the case studies of the same city are stacked in the same bar, e.g. ALA and ALA_{TT}, MGA and MGA_{TT}, etc.

7.5.5 Study of User Acceptance Rates

Since GS could be delivered as an app for smartphones, it is quite realistic to think that initially only a small number of drivers will have access to the system. Therefore, we have analyzed how the traffic behaves when just a subset of the vehicles use GS in our case studies.

In Figure 7.5 the graphs for the five case studies analyzed when the rates go from 10% to 100% in the best performing scenario are displayed. In the upper row, where the average improvement with respect to the experts' solution is plotted, it is clear that GS always reduces the average levels of gas emitted in each case study, even at low acceptance rates.

In the bottom row of Figure 7.5, the percentage of scenarios improved vs. GS acceptance rate is shown. The number of scenarios which are more eco-friendly when not all vehicles are using GS decreases, so that less use equals lower improvement, as one would expect. It is however noticeable that there is an average reduction in emissions in at least 48% of scenarios (the worst case: CO₂, *Malaga*), even when only 10% of drivers are using GS.

In addition, we observe that the behavior of GS in *Paris* has turned out to be a little different from the rest of the case studies. In figures 7.5e and 7.5j we can see that the metrics' variation for different usages is not as neat as in the rest of the case studies. This shows the different characteristics of *Paris*, especially its wide avenues and large roundabouts which leave little room for improvement.

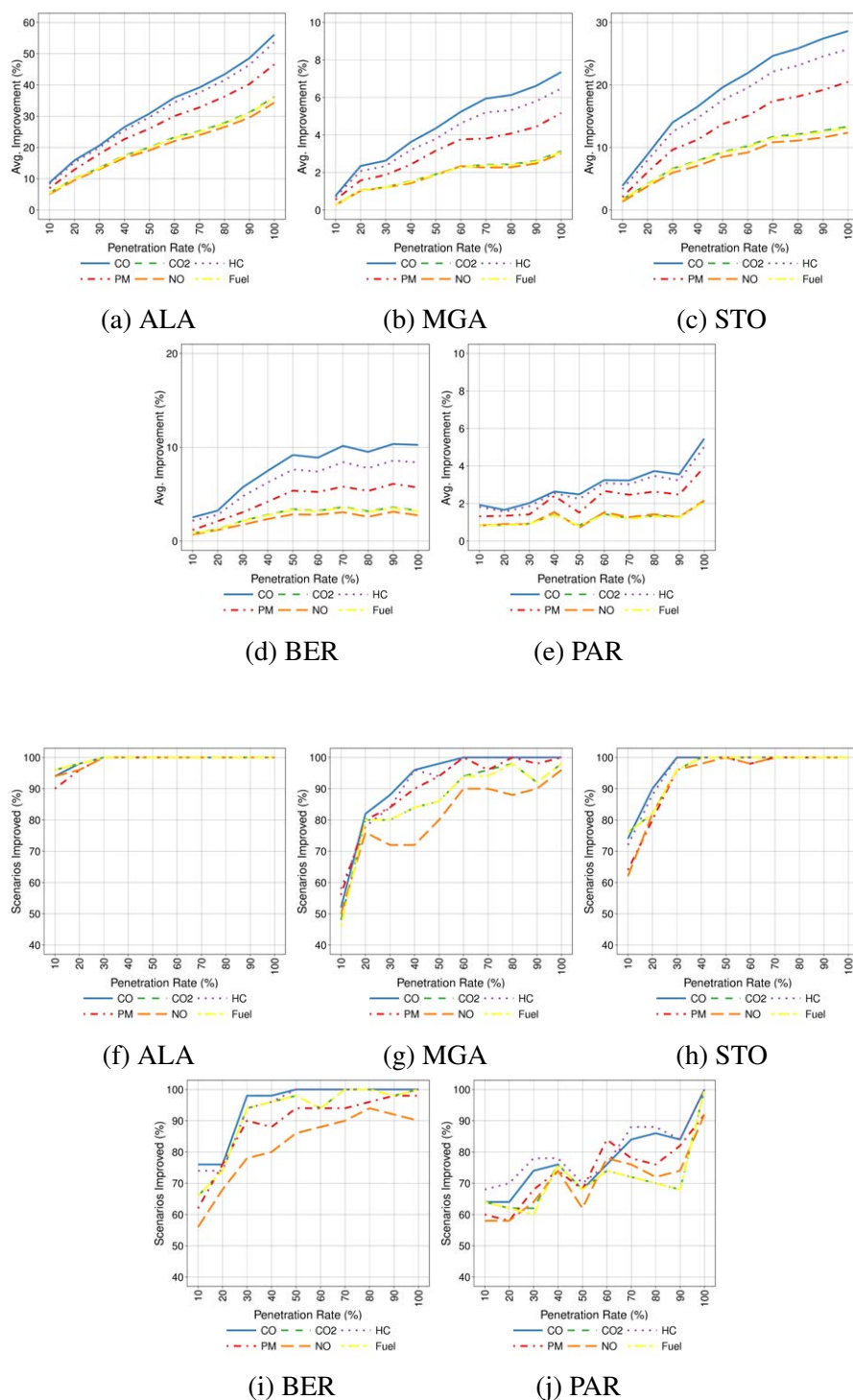


Figure 7.5: Graphs showing the average improvement achieved by GS for different user acceptance rates (upper rows) and the percentage of the 50 scenarios improved (lower rows) for the five case studies analyzed.

7.5.6 A Better Context for Understanding the Contributions of GS

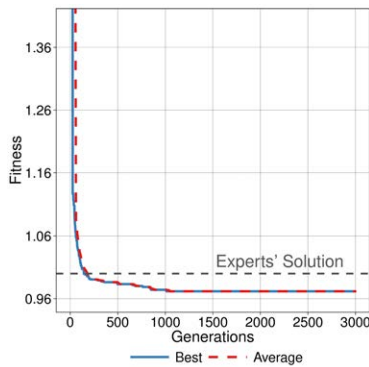
In this section a study of the internal performance of Green Swarm is addressed. Concretely, a comparison of the EfRA with a state of the art Genetic Algorithm (GA) [88, 102] (Section 3.2.1) and Simulated Annealing (SA) [35, 118] (Section 3.2.2) is presented, followed by a convergence analysis.

The competitor GA implemented is a steady state ($\mu = 10, \lambda = 2$), using Binary Tournament as selection operator, Uniform Crossover as recombination operator ($P_C = 0.6$ as in EfRA), VMO with probability $1/L$ as the mutation operator, and an elitist replacement. The SA selected is a well-known metaheuristic applicable to a wide range of problems. In this comparison we have used $\alpha = 0.9$ and generated 50 random neighbors before each temperature decrement. Thirty independent runs of each algorithm were made, stopping after 2000 evaluations to make a fair comparison, which amounts to 284 equivalent days.

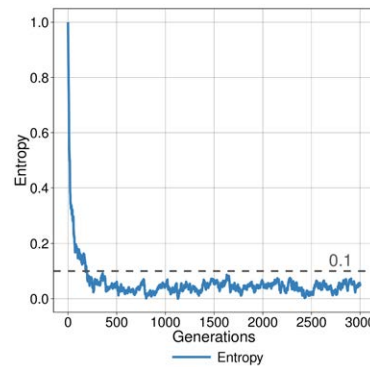
The objective of this study is to know how EfRA performs against its competitors and provide an internal statistical study [189] so as not to focus solely on the best fitness value. After testing the normality of the distributions using the Kolmogorov-Smirnoff test, we obtained p -values of 0.832 for the 30 runs of EfRA, 0.990 for GA, and 0.996 for SA. Consequently, non-parametric statistics (Friedman Rank and Wilcoxon) were used in the analysis. Table 7.8 shows the results of the comparison. EfRA achieved the best median value and was the best ranked algorithm. Additionally, the Wilcoxon test indicates that the differences between the results of the algorithms are statistically significant. We can therefore claim that our proposal overcomes existing results of the state of the art in the literature.

Table 7.8: EfRA compared with GA and SA.

| Alg. | Fitness | | Friedman Rank | Wilcoxon p -value |
|------|---------------|---------------|---------------|---------------------|
| | Median | Best | | |
| EfRA | 0.9625 | 0.9367 | 1.40 | — |
| GA | 1.0145 | 0.9783 | 2.93 | 0.000 |
| SA | 0.9779 | 0.9441 | 1.67 | 0.032 |



(a) Phenotype convergence of EfRA



(b) Genotype convergence of EfRA

Figure 7.6: Convergence of the EfRA over 3000 generations.

Moreover, a study on the EfRA fitness convergence over five independent runs (3000 generations, about 14 equivalent days) evaluating one instance of Malaga (MAL) was done. Figure 7.6a shows that after the 180th generation, the experts' solution has been improved by our proposal. After that point the entropy, which had been falling until this moment, begins to fluctuate below 0.1 (a very welcome exploration management of our algorithm) when the VMO changes the mutation probability from π_1 to π_2 to better exploit the solutions found (Figure 7.6b).

7.6 Discussion

In this chapter we have presented a system to reduce greenhouse gas emissions and used it to optimize road traffic in five cities in terms of not only emissions but also fuel consumption and travel times. After reducing travel times using Red Swarm, we have improved our architecture to address also gas emissions.

In a smart city context, our proposal represents an interesting working strategy focused on initially performing a micro analysis and then obtaining global results (bottom-up). It could also be implemented in other domains belonging to smart cities apart from Smart Mobility, such as Smart People and Smart Economy.

The results show that our system has been able to deliver reduced travel times (31.8% on average, 74.4% maximum), CO emissions (24% on average, 61% maximum), CO₂ (12.6% on average, 41.2% maximum), HC (22% on average, 58.5% maximum), PM (18% on average, 52% maximum), NO_x (11.8% on average, 39.1% maximum), and fuel consumption (12.5% on average, 40.9% maximum).

There is a negligible increase in route lengths (2% on average) which is a consequence of the eco-friendly rerouting of vehicles via alternative streets which are not included in the shortest path (the needed trade-off between the individual and the community). In spite of the variations observed in the results, which must be expected, as we are considering different cities (cultures, locations), we have improved all the metrics, even when only 10% of vehicles are using Green Swarm.

Our final reflexion about this study is that all the improvements achieved by GS are obtained without restricting the number of vehicles in the city, or type, weight or maximum speed, which the competitor strategies currently do. From a users' point of view we strongly believe that this is an added value as it does not debar anyone. In other words, everyone is able to travel through a city without traffic jams when it is being optimized by Green Swarm.

As a matter for future work, we are working on different strategies to implement the rerouting of vehicles by using city districts in order to be able to install GS throughout the entire city as well as address the optimization of harder scenarios (computation time and hardware requirements) involving hundreds of thousands of vehicles. We are currently working on different strategies to address unforeseen events such as accidents, fires, public demonstrations, which could suddenly close streets, turning open routes into invalid ones.

Chapter 8

Yellow Swarm: Low-Cost Infrastructure for the City

In this chapter Yellow Swarm architecture is proposed for reducing travel times, greenhouse gas emissions and fuel consumption of road traffic by using several LED panels to suggest changes in the direction of vehicles (detours) for different time slots. These time intervals are calculated using an evolutionary algorithm, specifically designed for our proposal, which evaluates many working scenarios based on real cities, imported from OpenStreetMap into the SUMO traffic simulator. Our results show an improvement in average travel times, emissions, and fuel consumption even when only a small percentage of drivers follow the indications provided by our panels.

8.1 Introduction

In the previous chapters we have been studying two architectures to prevent traffic jams and reduce congestion in cities. Despite the good results achieved, those proposals require that users need to have a device with Wi-Fi connectivity to use them. We propose in this chapter a new architecture, called Yellow Swarm [206, 208, 211], for redirecting road traffic by using LED (Light-Emitting Diode) panels, placed at strategic points of the city, to suggest possible changes in the direction drivers can take (continue straight on, turn left, turn right, etc.).

By using Yellow Swarm we are able to evenly spread the traffic throughout the city (without interfering too much with the drivers' itineraries), prevent traffic jams, and reduce travel times, greenhouse gas emissions and fuel consumption. As it uses LED panels to inform of possible changes in direction throughout the city, Yellow Swarm also reinforces the road safety and makes the system cheaper and easier to implement, develop and use. By extracting knowledge from real data of the city and using evolutionary computation to design the Yellow Swarm system, we hope to advance in the “smart” part of “smart cities” by providing numerical evidence that these kinds of intelligent algorithms are usable both in academia and industry.

8.2 The Yellow Swarm Architecture

The Yellow Swarm architecture, presented in Figure 8.1, has two stages: The *Offline* stage in which the system is configured, and the *Online* stage in which drivers are informed about the suggested detours. If we carry out the two contiguous stages with a given frequency we can improve the dynamics of the proposal and better fit changing scenarios in the city.

In the *Offline* stage, our Evolutionary Algorithm (EA) analyzes different scenarios (traffic distributions) of our case studies, using the SUMO traffic simulator [123] which is controlled by the TraCI module [237], in order to implement the decisions that drivers make during their journey. The urban maps used to build the case studies have been imported from OpenStreetMap so that we can test our system in realistic city districts including traffic lights, roundabouts, etc. The training carried out in this phase results in the configuration of the LED panels to be used in the next stage.

In the *Online* stage, the LED panels show the different detour options to drivers depending on the time slots calculated in the previous stage, using the Panel Manager. The possible signs are: go straight on, turn left, and turn right. However, their availability depends on the type of junction the vehicles are approaching, i.e. the possible detour options, and the street where the panel is placed. The first sign is visible during its previously calculated fixed time interval, after that the next sign in the sequence will be presented to the drivers. Once the cycle has finished it again starts with the first of the sequence. By using this strategy, Yellow Swarm is able to prevent possible traffic jams in the city as well as improve the use of most of the available secondary streets.

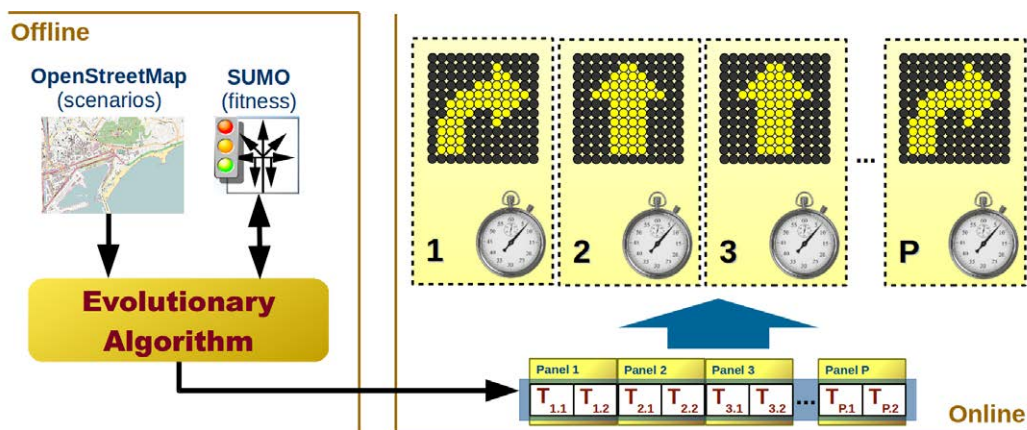


Figure 8.1: The Yellow Swarm architecture: In the *Offline* stage, it defines city scenarios and analyzes them with the EA, and in the *Online* stage, the previously calculated configuration is applied to the LED panels in order to suggest changes in the direction of the vehicles.

8.2.1 Evolutionary Algorithm (EA)

We have designed a (10+2)-EA (an elitist steady state EA with a population of ten individuals generating two new individuals at each step) in order to obtain the periods of time that each

sign is presented to drivers. We have used Binary Tournament as the selection operator, Uniform Crossover as the recombination operator, and elitism as the replacement policy. The mutation operator was designed especially for our problem as described later in Section 8.2.1.

Problem Representation

In Figure 8.2 we can see the status vector containing time values for each sign to be shown to drivers. The total number of panels (P) will depend on the street layout and the number of time slots, on the the junction characteristics.

The status vector for the case study *Malaga* consists of 16 integers which can take values between 30 and 300 as the number of seconds the corresponding sign is active. For *Madrid* we placed four LED panels consisting of two signs each (eight integers), and for *Quito*, there were some left turn signs (panels 2, 3, 5, and 7) which amounts to 24 integers in total.

| Malaga | | | | | | | | | | | | | | | |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| Panel 1 | Panel 2 | Panel 3 | Panel 4 | Panel 5 | Panel 6 | Panel 7 | Panel 8 | | | | | | | | |
| T _{1.1} | T _{1.2} | T _{2.1} | T _{2.2} | T _{3.1} | T _{3.2} | T _{4.1} | T _{4.2} | T _{5.1} | T _{5.2} | T _{6.1} | T _{6.2} | T _{7.1} | T _{7.2} | T _{8.1} | T _{8.2} |

| Madrid | | | | | | | |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| Panel 1 | Panel 2 | Panel 3 | Panel 4 | | | | |
| T _{1.1} | T _{1.2} | T _{2.1} | T _{2.2} | T _{3.1} | T _{3.2} | T _{4.1} | T _{4.2} |

| Quito | | | | | | | | | | | | | | | | | | | | | | | |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|-------------------|-------------------|
| Panel 1 | Panel 2 | Panel 3 | Panel 4 | Panel 5 | Panel 6 | Panel 7 | Panel 8 | Panel 9 | Panel 10 | | | | | | | | | | | | | | |
| T _{1.1} | T _{1.2} | T _{2.1} | T _{2.2} | T _{2.3} | T _{3.1} | T _{3.2} | T _{3.3} | T _{4.1} | T _{4.2} | T _{5.1} | T _{5.2} | T _{5.3} | T _{6.1} | T _{6.2} | T _{7.1} | T _{7.2} | T _{7.3} | T _{8.1} | T _{8.2} | T _{9.1} | T _{9.2} | T _{10.1} | T _{10.2} |

Figure 8.2: Status vectors of *Malaga*, *Madrid*, and *Quito*.

Evaluation Function

Two different evaluation criteria were followed depending on the optimization done. We wished to reduce the average travel times of vehicles in *Malaga* and *Madrid*, and then observe how this optimization affects the emissions and fuel consumption. To achieve our objective, we proposes the evaluation function presented in Equation 8.1 which is used to calculate the fitness of each individual. There, N is the total number of vehicles entering the city, n is the number of vehicles leaving the city during the period of time analyzed, and $travel\ time_i$ is the travel time spent by the vehicle i for its journey.

Moreover, α_1 and α_2 are the weights of each term calculated in order to normalize the fitness value calculated by the function. Note that the first term of the function penalizes individuals with one unit for each vehicle ($\alpha_1 = 1$) when there are still vehicles in the city at the end of the analysis period. Only when all vehicles end their journeys can we include the metrics from all of them in the fitness function computation, thereby the penalization term is not present ($N = n$). Additionally, we have calculated α_2 as described in Equation 8.2, so that the fitness of the experts' solution is equal to 1. There, λ is equal to four as we are evaluating four scenarios and averaging their fitness values during the optimization process in order to achieve more robust configurations to test in several unseen scenarios.

$$F = \alpha_1(N - n) + \alpha_2^{-1} \frac{1}{n} \sum_{i=1}^n travel\ time_i \quad (8.1)$$

$$\alpha_2 = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \frac{1}{n_i} \sum_{j=1}^{n_i} travel\ time_{ij} \quad (8.2)$$

Note that α_2 will be different for each case study as the average travel time of vehicles is different as well. It takes about one minute to calculate each fitness value as we need to test the configuration of the training scenarios using SUMO. As we are minimizing these variables, the lower the fitness value, the better.

On the other hand, we wished to reduce the number of vehicles in *Quito* in peak hours (i.e. foster their arrival at their desired destinations) so that we can reduce travel times, greenhouse gas emissions, and fuel consumption. The function presented in Equation 8.3 is meant to calculate the fitness of each individual and maximize the number of vehicles arriving at their destination when Yellow Swarm is active.

$$F(\vec{x}) = \alpha_3^{-1}(n_f - n_0), \quad (n_f, n_0) = Simulate(\vec{x}) \quad (8.3)$$

$$\alpha_3 = \frac{1}{\lambda} \sum_{k=1}^{\lambda} s_{k_f} - s_{k_0}, \quad \vec{s}_k = (s_{k_f}, s_{k_0}) = Simulate(scen_k) \quad (8.4)$$

Here, vector \vec{n} is obtained by evaluating the individual \vec{x} in SUMO. It consists of two components: n_f which is the number of running vehicles at the end of the optimization interval and n_0 which is the number of vehicles at the beginning of it. Additionally, α_3 is a coefficient that normalizes the fitness function calculated as shown in Equation 8.4. We calculate the differences between the number of vehicles running in the city during the optimization interval ($s_{k_f} - s_{k_0}$) when the Yellow Swarm is not being used. As we are using four training scenarios ($scen_k, k \in \{1 \dots \lambda\}$) in this study, we worked also with $\lambda = 4$.

After evaluating an individual, fitness values greater than 1 are improvements, as they represent a higher number of vehicles leaving the city at the end of the optimization interval. So, we want to maximize the fitness value meaning the higher, the better.

Operators

As we stated before, we have used Binary Tournament, Uniform Crossover, and elitism as the replacement policy. The specific mutation operator is described in Algorithm 8.1.

First, all panels from the individual are obtained. Second, some panel configurations are selected to be changed depending on the mutation probability (P_m). Third, all signs (all the different detours) from the panel selected are obtained and one of them is randomly chosen to have its time value incremented in τ_1 . The rest of the signs on the panel have their time value decremented in τ_2 . Note that the time values are kept in the range of 30 – 300 so that

Algorithm 8.1 Mutation Operator.

```

procedure MUTATE(individual)
  panelConfigs  $\leftarrow$  getPanels(individual)
  for all  $p \in \text{panelConfigs}$  do
    if random()  $< P_m$  then
      signs  $\leftarrow$  getAllSigns(p)
       $s \leftarrow \text{selectOneSignRND}(\text{signs})$  ▷ Selects one sign  $s$ 
      for all  $ss \in \text{signs}$  do
        if  $ss == s$  then
          incrementTime( $ss, \tau_1$ ) ▷ Increments time slot of  $s$ 
        else
          decrementTime( $ss, \tau_2$ ) ▷ Decrements the rest
        end if
      end for
    end if
  end for
  return individual
end procedure

```

the values can be truncated if necessary. Finally, when the main loop ends, the modified individual is returned.

In Figure 8.3 an example of the mutation of an individual is depicted. We can see that the third panel has been selected for mutation and its time values have been changed from (61, 274) to (56, 279). Additionally, we have experimentally found the values of the crossover probability ($P_c = 0.6$), the mutation probability ($P_m = 1/L, L = \text{status vector length}$), and $\tau_1 = \tau_2 = 5$.

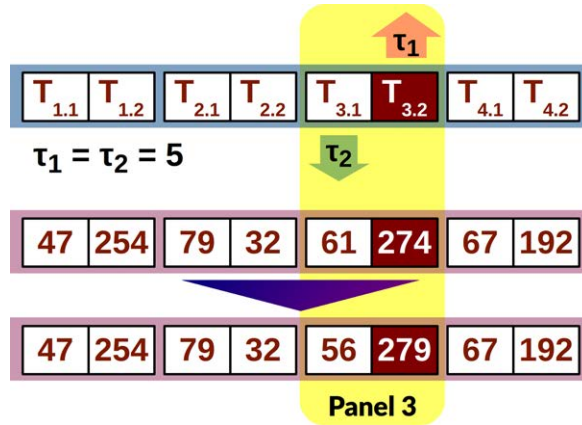


Figure 8.3: Example of the mutation operator applied to an individual. Note that $T_{3,2}$ is incremented in $\tau_1 = 5$ and $T_{3,1}$ is decremented in $\tau_2 = 5$.

8.2.2 Panel Manager

In the *Online* stage the Panel Manager selects the different detour options to be shown to drivers by each panel. Figure 8.4 shows the schema of the Panel Manager running in the panel i . Depending on the Yellow Swarm configuration, one sign is shown, e.g. turn right during $T_{i,1}$ seconds, followed by the next sign (going straight on in the example) during the following $T_{i,2}$ seconds. After that, the first sign is shown again, completing a cycle of period $T_{i,1} + T_{i,2}$. Note that each panel needs its own Panel Manager in order to follow the Yellow Swarm configuration.

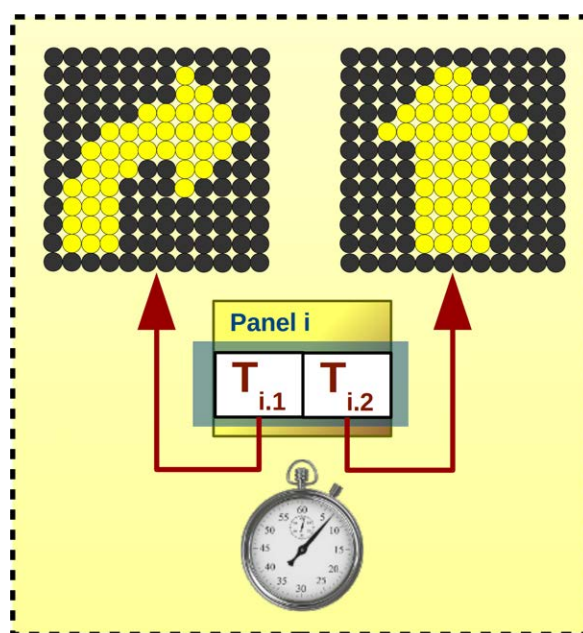


Figure 8.4: Panel Manager.

8.3 Case Studies

In this approach we have analyzed and optimized three important cities: Malaga and Madrid in Spain, and Quito in Ecuador, all of which suffer from traffic congestion at peak hours.

8.3.1 Malaga and Madrid

After importing the maps of Malaga and Madrid into SUMO from OpenStreetMap we solved several issues observed, such as those junctions, roundabouts, and traffic lights which tended to form traffic jams. Our objective was to build more robust scenarios, more difficult to optimize as they consist of a greater number of vehicles. Furthermore, we have defined two case studies for each city. One of them corresponds to the case in which the vehicles use various streets in the city to arrive at their destination, and the other (TT subscript) in which only faster routes are used (TT stands for travel time). All these flows were generated by the

Table 8.1: Type and characteristics of vehicles in *Malaga* and *Madrid*.

| Type | Arrival Prob. | MaxSpd. (km/h) | Accel. (m/s ²) | Decel. (m/s ²) | Length (m) | Emission class |
|-------|---------------|----------------|----------------------------|----------------------------|------------|----------------|
| truck | 0.10 | 40 | 0.6 | 3.5 | 6.0 | HDV_3_3 |
| wagon | 0.15 | 50 | 0.7 | 4.0 | 4.3 | P_14_12 |
| van | 0.25 | 100 | 0.8 | 4.5 | 4.2 | P_14_8 |
| sedan | 0.50 | 160 | 0.9 | 5.0 | 3.8 | P_7_7 |

DUAROUTER utility included in SUMO which uses different streets' weights such as travel time, emissions, and fuel consumption to build the routes (see Section 4.3.2). Consequently, we have named these scenarios as the experts' solution from SUMO in our experiments. By changing the random seed of a scenario we were able to modify the order and pace that vehicles enter an area in order to define multiple scenarios of the same case study.

In order to make our study more realistic, we have defined four different vehicle types: sedan, van, wagon, and truck. Each one has its own characteristics according to the class they belong to, e.g. trucks are longer, slower and more pollutant than sedans, etc. The rest of characteristics are the probability of arriving, maximum speed, acceleration, deceleration, length, and the emission class from the HBEFA database [96]. In Table 8.1 the type of vehicles and their characteristics are listed.

We have analyzed the behavior of 4500 vehicles in *Malaga* and 4840 in *Madrid* for 7200 seconds. In *Malaga* we placed eight LED panels but in *Madrid* only four were effective, especially due to the street layout of this capital city, e.g. wide avenues, huge roundabouts, lots of parallel streets. The localizations of the panels were mainly chosen by taking into account several hot spots in the cities where traffic jams were more likely to happen. Due to the streets distribution of *Madrid*, we have up to 1641 different routes in this case study, compared with 365 in *Malaga*. Table 8.2 shows the rest of the characteristics of the case studies.

The geographical area analyzed in the city of Malaga includes the following areas: *Centro Histórico*, *Olletas*, *Ciudad Jardín*, and *El Limonar*, which together encompass an area of about 10.7 km². The case study *Madrid* includes the areas *Barrio de la Latina*, *Parque del*

Table 8.2: Characteristics of the four case studies: *Malaga*, *Malaga_{TT}*, *Madrid*, and *Madrid_{TT}*.

| Case study | <i>Malaga</i> | <i>Malaga_{TT}</i> | <i>Madrid</i> | <i>Madrid_{TT}</i> |
|-------------------|---------------|----------------------------|---------------|----------------------------|
| Analysis time (s) | 7200 | | | |
| # Vehicles | 4500 | | 4840 | |
| # Traffic lights | 515 | 515 | 942 | 942 |
| # LED panels | 8 | 8 | 4 | 4 |
| # Vehicle types | 4 | 4 | 4 | 4 |
| # Source streets | 18 | 18 | 14 | 14 |
| # Destinations | 8 | 8 | 25 | 25 |
| # Vehicle routes | 365 | 134 | 1641 | 574 |



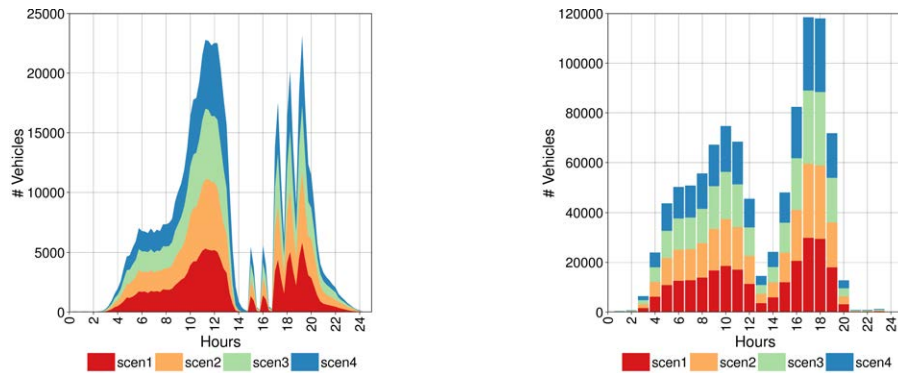
Figure 8.5: Case studies: *Malaga* and *Madrid*. Exported from OpenStreetMap (left) to SUMO (right).

Retiro, Salamanca, Chueca, Plaza de España, and Malasaña. The total area analyzed from Madrid is about 10.3 km². In Figure 8.5 both geographical areas including the locations of the LED panels are depicted. The maps from OpenStreetMap are in the left-hand column, and in the right-hand one, the corresponding models imported into SUMO.

8.3.2 Quito

We tried something different for this case study. Instead of generating data randomly or according to common sense social patterns, we directly use real demographic and geographical data as well as peak traffic hours in Quito, to confirm whether we can actually have an effect on real scenarios as well as to provide evidence of the benefits of Yellow Swarm.

The city of Quito (Ecuador) is a large city in which almost two million people commute every day for varied reasons such as work, study, leisure, and shopping, using private and public transportation. The area of the study embodies 14 neighborhoods that represent 30% of Quito's population with approximately 560.000 inhabitants [11]. It includes the business district, eight major universities, several hospitals, large malls, two large parks, and one major soccer stadium, covering approximately 40 km². The varied number of services and the abundant population mean that the present daily demand has outstripped the existing traffic infrastructure.



(a) # of vehicles in the city during the day. (b) # of vehicles arriving the city per hour.

Figure 8.6: Number of vehicles in the *Quito* case study.

In order to create a more realistic study, we have generated four different traffic scenarios (*scen₁*, *scen₂*, *scen₃*, and *scen₄*) by using the ACTIVITYGEN tool provided by the SUMO microsimulator. ACTIVITYGEN allows us to implement not only the characteristics of mobility around Quito such as peak hours, work areas, and residential neighborhoods, but also a little variability based on a random number seed in order to generate the aforementioned four realistic scenarios. Figure 8.6a shows the four distributions of traffic we have worked with over 24 hours in the city and Figure 8.6b the total number of vehicles arriving the scenario per hour.



(a) Quito in OpenStreetMap

(b) Quito in Google Maps™

Figure 8.7: Yellow Swarm panels placed in the business district of Quito.

The total number of journeys modeled for 24 h is approximately 245000 where 76% of the journeys are inside the zone being studied and, the reminder outside, which reflects the city behavior in that area. Additionally, we have placed ten panels in seven locations across the area as shown in Figure 8.7. Some locations include several panels controlling different flows of traffic. Their placement has been decided considering the behavior of the traffic in the city after identifying zones where jams are likely to happen and that the infrastructure permits a route that allows drivers to reach their destinations.

8.4 Experiments and Results

In the following sections we address the optimization of the case studies describe before. Different studies were done as each case study has its own interesting characteristics.

8.4.1 Malaga and Madrid

First, we have optimized *Malaga* and *Madrid* by performing 30 independent runs of our EA on four scenarios of each case study. Second, we have tested the best solution from each algorithm in 200 different scenarios (*Malaga*, *Malaga_{TT}*, *Madrid*, and *Madrid_{TT}*) in order to discover how robust and scalable the solutions are. Finally, we carried out a deep analysis of data and how Yellow Swarm behaves when the use rate is lower than 100%.

Training

We optimized four different scenarios of the case study *Malaga* (in total, 18000 vehicles) as well as four of *Madrid* (19360 vehicles) in order to achieve a robust configuration for the Yellow Swarm, using the EA. In previous similar approaches [203] this strategy has been used to achieve more robust solutions as it is better than using just only one. We completed 30 independent runs on each case study (the same four scenarios) and achieved the results presented in Table 8.3.

We can observe the mean values of not only travel times but also emissions, fuel consumption, and distances, as well as the standard deviation. We present the mean and standard deviation of the metrics from the experts' solution versus Yellow Swarm in both case studies, *Malaga* and *Madrid*, as well as the improvement achieved and the Wilcoxon p -value. The best improvements are observed in travel time values as this is the metric we are optimizing. However, we have also achieved a reduction of emissions and fuel consumption. Distances traveled are slightly longer when vehicles take the detours suggested (0.2% maximum). However, this was to be expected as they are driving through alternatives streets which are not part of the shortest path in order to achieve the global improvements discussed. Note that we have assumed, in this training, that drivers always follow the detours shown in the panels.

Furthermore, we have calculated the Wilcoxon p -value to be sure that the improvements reported on each metric are statistically significant. We can see that in *Malaga* all the results of the optimization have a confidence level of greater than 99%. However, in *Madrid* only travel times, CO₂, HC, and fuel consumption have that confidence level, which as we are

Table 8.3: Results of the optimization process of *Malaga* and *Madrid* (four training scenarios). Note that the improvements achieved are in bold.

| Malaga | | | | | | |
|----------------------|-------------------|--------|--------------|--------|--------------|-----------------------------|
| Metrics | Experts' solution | | Yellow Swarm | | Improvement | Wilcoxon <i>p</i> -value |
| | Avg. | StdDev | Avg. | StdDev | | |
| Travel Time (s) | 1903.2 | 2.3% | 1562.1 | 2.5% | 17.9% | 0.00 |
| CO (mg) | 15744.6 | 3.4% | 13829.2 | 2.5% | 12.1% | 0.00 |
| CO ₂ (mg) | 1418052.7 | 1.5% | 1332355.0 | 1.7% | 6.0% | 0.00 |
| HC (mg) | 2360.3 | 3.2% | 2103.4 | 3.4% | 10.9% | 0.00 |
| PM (mg) | 224.9 | 6.2% | 207.7 | 6.1% | 7.6% | 0.00 |
| NO _x (mg) | 8904.6 | 5.2% | 8483.0 | 5.5% | 4.7% | 0.00 |
| Fuel (ml) | 562.6 | 1.4% | 529.0 | 1.7% | 6.0% | 0.00 |
| Distance (m) | 3451.3 | 0.7% | 3457.2 | 0.3% | -0.2% | 0.00 |
| Madrid | | | | | | |
| Metrics | Experts' solution | | Yellow Swarm | | Improvement | Wilcoxon <i>p</i> -value |
| | Avg. | StdDev | Avg. | StdDev | | |
| Travel Time (s) | 1374.7 | 1.3% | 1318.5 | 2.9% | 4.1% | 0.00 |
| CO (mg) | 12144.2 | 2.0% | 11705.8 | 2.7% | 3.6% | 0.04 |
| CO ₂ (mg) | 1165631.8 | 1.0% | 1148906.4 | 1.8% | 1.4% | 0.00 |
| HC (mg) | 1828.7 | 1.5% | 1779.4 | 2.3% | 2.7% | 0.01 |
| PM (mg) | 172.4 | 3.1% | 171.4 | 2.6% | 0.6% | 0.15 |
| NO _x (mg) | 7188.5 | 2.2% | 7158.3 | 2.2% | 0.4% | 0.11 |
| Fuel (ml) | 463.1 | 1.0% | 456.5 | 1.7% | 1.4% | 0.00 |
| Distance (m) | 3096.3 | 0.2% | 3099.8 | 0.2% | -0.1% | 0.18 |

mainly optimizing travel times, is satisfactory in spite of the lower confidence level of the other metrics (96% in CO, 85% in PM, 89% in NO_x, and 82% in Distance).

Table 8.4 shows the values obtained as the configuration of Yellow Swarm for each case study. We can see that Panels 7 and 8 in *Malaga* and Panels 3 and 4 in *Madrid* present a highly asymmetric configuration (91%, 88%, 87%, and 86%, respectively). That is, the traffic which arrives at these panels is mainly being directed towards just one detour. It denotes that the other detours have almost been discarded by the EA because they produce worse traffic flows. Nevertheless, other time slots are more balanced, e.g. Panel 2 (48%) in *Malaga*, and Panel 2 (42%) in *Madrid*.

Test of 50 Scenarios

We have tested our Yellow Swarm on 50 new different scenarios of each case study (*Malaga* and *Madrid*) and a further 100 scenarios where vehicles only use the fastest routes (50 scenarios of *Malaga*_{TT} and 50 of *Madrid*_{TT}) which results in a total of 200 different scenarios.

Table 8.5 shows the average improvements obtained in the four case studies when we compare the experts' solution to Yellow Swarm. In addition, the best scenario and the percentage of scenarios improved are shown.

Table 8.4: Configuration of panels obtained by the EA.

| <i>Malaga</i> | | | | | | | | | | | | | | | |
|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Panel 1 | | Panel 2 | | Panel 3 | | Panel 4 | | Panel 5 | | Panel 6 | | Panel 7 | | Panel 8 | |
| $T_{1.1}$ | $T_{1.2}$ | $T_{2.1}$ | $T_{2.2}$ | $T_{3.1}$ | $T_{3.2}$ | $T_{4.1}$ | $T_{4.2}$ | $T_{5.1}$ | $T_{5.2}$ | $T_{6.1}$ | $T_{6.2}$ | $T_{7.1}$ | $T_{7.2}$ | $T_{8.1}$ | $T_{8.2}$ |
| 38 | 172 | 260 | 280 | 141 | 80 | 89 | 238 | 290 | 66 | 90 | 257 | 300 | 30 | 43 | 286 |

| <i>Madrid</i> | | | | | | | |
|---------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Panel 1 | | Panel 2 | | Panel 3 | | Panel 4 | |
| $T_{1.1}$ | $T_{1.2}$ | $T_{2.1}$ | $T_{2.2}$ | $T_{3.1}$ | $T_{3.2}$ | $T_{4.1}$ | $T_{4.2}$ |
| 194 | 30 | 215 | 294 | 30 | 228 | 46 | 286 |

In the case study *Malaga*, we can observe an average improvement of: 13.4% (18.4% max.) in travel times, 10.3% (12.9% max.) in CO emissions, 5% (7.4% max.) in CO₂ emissions, 9.5% (11.8% max.) in HC, 7.6% (10.6% max.) in PM, 4.9% (7.2% max.) in NO_x, and 4.9% (7.4% max.) in fuel consumption. Furthermore, the distance traveled by vehicles is 0.9% longer on average as we are suggesting detours that are not part of the shortest path. This, however, is negligible, especially when compared to the improvement achieved in travel times. Yellow Swarm was able to improve upon the experts' solution in all 50 scenarios tested (100%) in both travel times and emissions. In *Malaga_{TT}* we achieved even better results (vehicles' travel times are up to 32.3% shorter on average) as its routes involve only the fastest ones. Consequently, as traffic jams are more likely to happen in that situation, Yellow Swarm has turned out to be more efficient preventing them.

When we decided to include *Madrid* in this study, we wanted to compare two cities with different topologies. While the geographical area of *Malaga* consists of narrow streets and very few avenues and roundabouts, *Madrid* boasts wide avenues with up to ten lanes, huge

Table 8.5: Improvement achieved in the average vehicles' travel times, gas emissions, fuel consumption, and distance traveled in the four case studies.

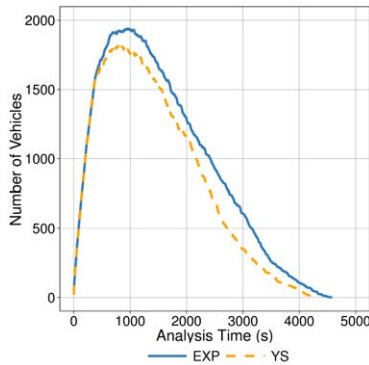
| | | T. Time | CO | CO ₂ | HC | PM | NO _x | Fuel | Distance |
|----------------------------|----------------------|---------|--------|-----------------|--------|--------|-----------------|--------|----------|
| <i>Malaga</i> | Average 50 Scenarios | 13.4% | 10.3% | 5.0% | 9.5% | 7.6% | 4.9% | 4.9% | -0.9% |
| | Best Scenario | 18.4% | 12.9% | 7.4% | 11.8% | 10.6% | 7.2% | 7.4% | -0.6% |
| | % Scenarios Improved | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 8.0% |
| <i>Malaga_{TT}</i> | Average 50 Scenarios | 22.2% | 17.9% | 9.8% | 16.2% | 13.1% | 9.0% | 9.6% | -2.6% |
| | Best Scenario | 32.3% | 25.3% | 16.5% | 23.3% | 22.9% | 16.6% | 16.4% | -1.1% |
| | % Scenarios Improved | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 100.0% | 2.0% |
| <i>Madrid</i> | Average 50 Scenarios | 2.1% | 1.5% | 0.8% | 1.3% | 1.1% | 0.7% | 0.8% | -0.5% |
| | Best Scenario | 8.1% | 10.1% | 3.2% | 8.9% | 3.7% | 2.5% | 3.2% | 0.5% |
| | % Scenarios Improved | 72.0% | 66.0% | 68.0% | 68.0% | 60.0% | 62.0% | 68.0% | 34.0% |
| <i>Madrid_{TT}</i> | Average 50 Scenarios | 2.3% | 1.7% | 0.8% | 1.6% | 1.4% | 0.8% | 0.8% | -0.4% |
| | Best Scenario | 9.1% | 7.5% | 3.8% | 6.4% | 3.9% | 2.9% | 3.8% | -0.2% |
| | % Scenarios Improved | 74.0% | 70.0% | 64.0% | 70.0% | 68.0% | 68.0% | 64.0% | 16.0% |

roundabouts, and square blocks. Although it allowed us to include more vehicles in the study of *Madrid*, it also made it more difficult to optimize. Regardless, our results show travel times of 2.1% shorter on average (8.1% max.) and reductions in the gases emitted between 0.7% and 1.5% (10.1% max.). Moreover, we were able to improve travel times in 72% of the 50 scenarios as well as emissions in more than 60% of them.

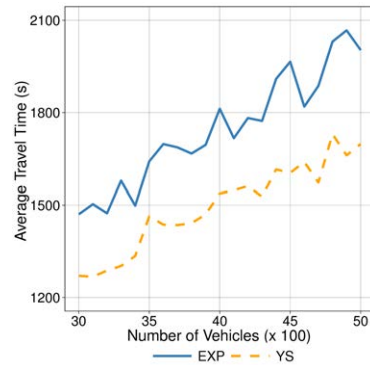
Finally, in *Madrid_{TT}* the results are better than in *Madrid* for the same reasons explained when comparing *Malaga* to *Malaga_{TT}*, i.e. vehicles are taking only faster routes.

Further Analysis

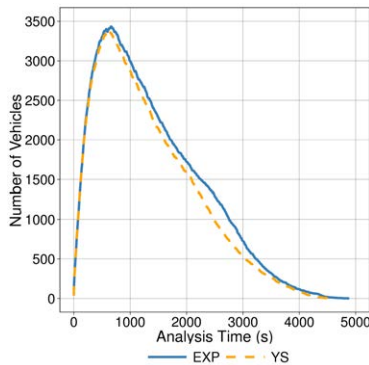
Our next step in our experimentation focused on analyzing the data collected even further. We have studied the traffic density and the number of vehicles in the city when vehicles follow the routes of the experts' solution and when they follow the detours proposed by Yellow Swarm. In Figure 8.8 we show plots on traffic density and number of vehicles in the city. Figure 8.8a shows how vehicles behave during the analysis in *Malaga* where we can see that Yellow Swarm keeps the number of vehicles to lower than the experts' solution and also how it ensures the last vehicle leaves the city earlier. Figure 8.8b shows how the system scales with respect to the number of vehicles in the city.



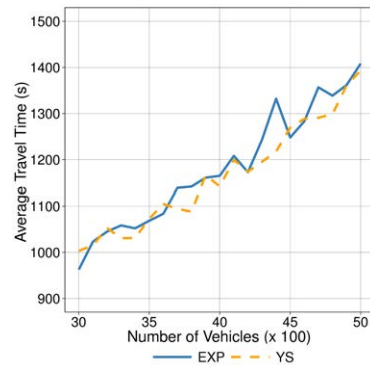
(a) Traffic density in *Malaga*



(b) Travel times in *Malaga*



(c) Traffic density in *Madrid*



(d) Travel times in *Madrid*

Figure 8.8: Traffic density and travel time vs. number of vehicles in *Malaga* and *Madrid*. We compare the values from the experts' solution to the Yellow Swarm ones.

We can see that the average travel time is always shorter when drivers follow the detours suggested by Yellow Swarm as it does not depend on the number of vehicles. Figures 8.8c and 8.8d present the same study for *Madrid*. In this case the curves are not so different although the behavior is still better when vehicles follow the signs from panels.

Our final step consists of testing Yellow Swarm when it is used by just a few vehicles. This situation makes common sense, as the directions are just suggestions and we cannot expect all drivers to follow them (confusions, lack of trust or information, etc.) We have tested it on 50 scenarios from a use rate of 10% to 90% (100% has been studied previously in this chapter). All in all, it adds up to 450 scenarios of *Malaga* and 450 of *Madrid* which have been used to produce the graphs presented in Figure 8.9. The average improvement in *Malaga* as well as the percentage of scenarios improved are depicted in Figures 8.9a and 8.9b respectively. As can be seen, Yellow Swarm always reduces the average travel times and levels of gas emitted in each case study, even at low penetration rates. This confirms our optimization strategy as well as the correct placing of the panels. Nevertheless, the number of scenarios improved when not all vehicles are following the detours decreases: fewer users, lower the improvement, as expected.

However, the behavior of Yellow Swarm in *Madrid* has turned out to be different from *Malaga*. In spite of the fact that the metrics are always improved for all the usage rates,

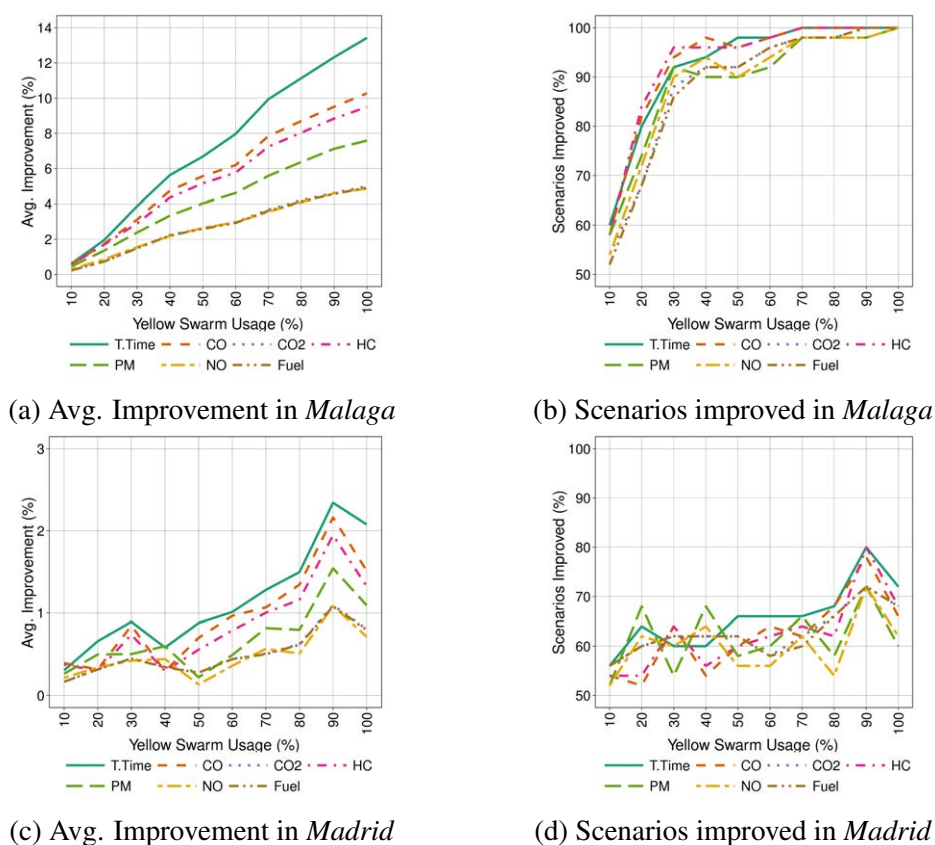


Figure 8.9: Average improvement and scenarios improved vs. panel use when a only percentage of the drivers decide to follow the detours. The greater the acceptance of drivers, the greater the sustainability of the city.

figures 8.9c and 8.9d show a behavior that is not as neat as in *Malaga*, especially due to the reasons we have discussed before. However, it is clear that there is an improvement in each metric when we use Yellow Swarm, even for penetration rates as low as 10%.

8.4.2 Quito

In the following sections we address the optimization of *Quito*. First, we calculate our optimization interval and conduct experiments in order to analyze the best strategy to maximize the number of vehicles that arrive at their destinations in this period. Second, after selecting the best strategy, we test the best configuration obtained in 30 different unseen scenarios so that we can measure the performance of the proposal presented here.

In both analyses we study not only the fitness value, but also travel times, gas emissions, fuel consumption, and distances traveled as complementary metrics, which we strongly believe are also important for this study.

8.4.3 Optimization Interval

In our experiments we wanted not only to optimize the city of Quito by using the Yellow Swarm architecture, but also select the best strategy for doing so. In accordance with the average number of vehicles in the city in a day, we set a threshold of 2000 vehicles to detect when traffic jams are likely to happen, so that we could switch the Yellow Swarm on only during these time intervals (Figure 8.10).

We observed that the threshold is crossed twice in a day (beginning at 8:30 and 16:30, respectively) when people are going to their workplaces and when they are returning home. In this first approach we decided to focus on the first one, so we divided the first time interval into four sub-intervals to be optimized. Consequently, we trained the EA independently for 25%, 50%, 75%, and 100% of the peak hour interval to collect the results and evaluate which was the best choice according to the quality of the solutions (fitness) and how long they took to be calculated (longer optimization periods imply longer simulation times in order to evaluate the panel configurations). The optimization sub-intervals are described in Table 8.6. All of them begin at 8:30, and end at 9:45, 11:00, 12:15, and 13:30, respectively.

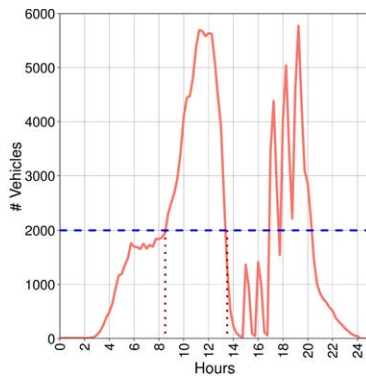


Figure 8.10: Average number of vehicles.

| Sub-interval | Begin | End | Duration (m) |
|--------------|-------|-------|--------------|
| 25% | 8:30 | 9:45 | 75 |
| 50% | 8:30 | 11:00 | 150 |
| 75% | 8:30 | 12:15 | 225 |
| 100% | 8:30 | 13:30 | 300 |

Table 8.6: Optimization sub-intervals.

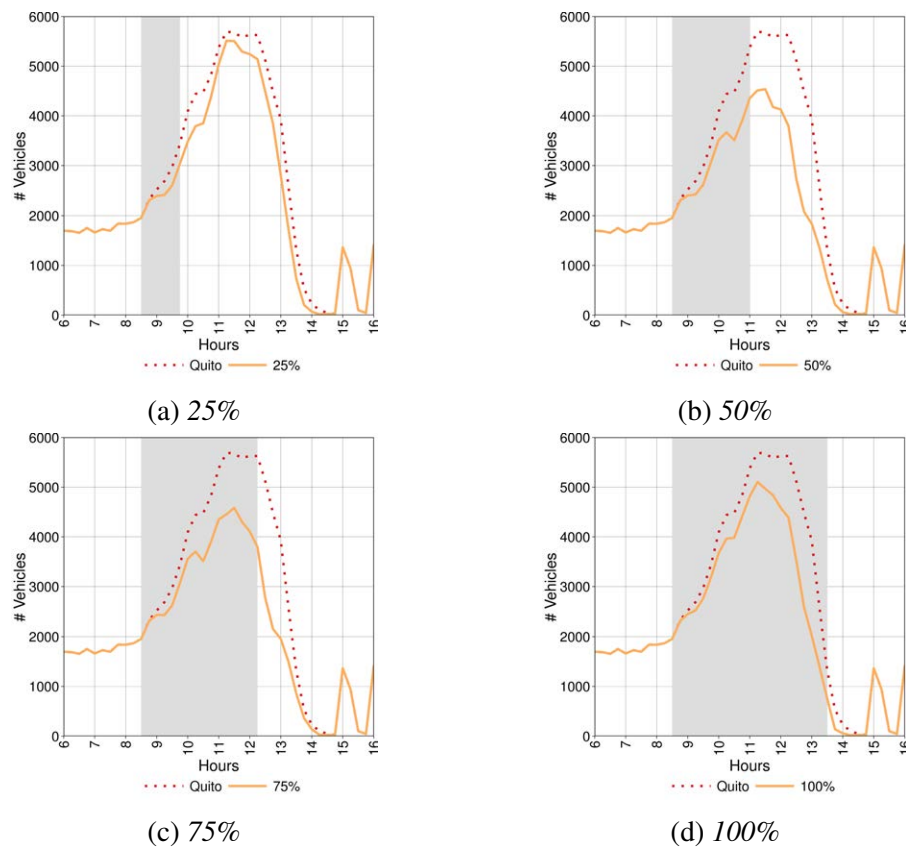


Figure 8.11: Number of vehicles during the analysis period in Quito before and after the optimization process using the four sub-intervals tested: 25%, 50%, 75%, and 100%.

After training our EA with four scenarios (more diversity implies more robust solutions [203]) over 30 independent runs, we obtained the evolutions in the number of vehicles shown in Figure 8.11. These graphs were obtained after using Yellow Swarm configured with the best solution (out of 30) for each optimization sub-interval. As we can see, in all the

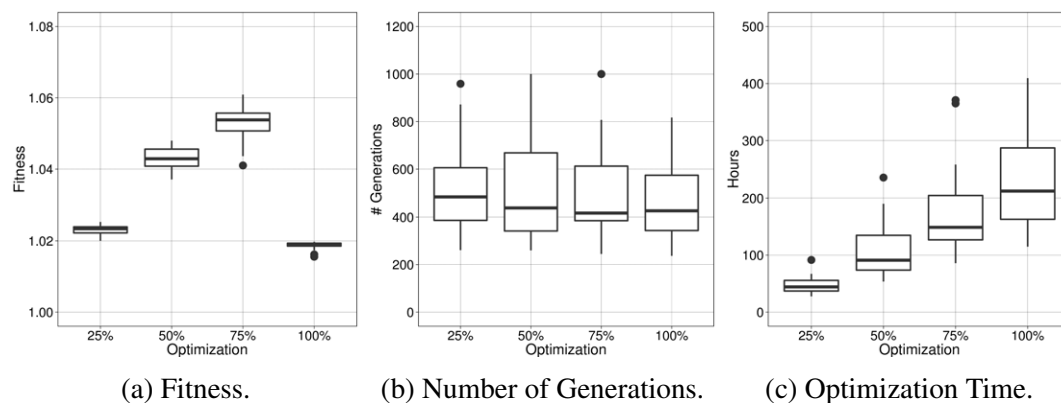


Figure 8.12: Fitness, number of generations, and optimization time of the four optimization intervals analyzed.

Table 8.7: Fitness values obtained from the four optimization process and statistical tests.

| Sub-interval | Fitness | | Friedman Rank | Wilcoxon p -value |
|--------------|--------------|--------|---------------|---------------------|
| | Average | StdDev | | |
| 25% | 1.023 | 0.15% | 2.00 | 0.00 |
| 50% | 1.043 | 0.27% | 3.10 | 0.00 |
| 75% | 1.053 | 0.44% | 3.90 | — |
| 100% | 1.019 | 0.10% | 1.00 | 0.00 |

sub-intervals the number of vehicles running between 8:30 and 13:30 has been reduced. The 50% and 75% sub-intervals demonstrate interesting outcomes since the maximum number of vehicles are the smallest. The optimization interval of 25% does not contain enough traffic information for the EA to learn the characteristics of the traffic and the 100% has not been the most successful which we can deduce from the results (it is even worse than using 25%).

In Figure 8.12 we can observe the box plots obtained from the 30 runs performed by the four sub-intervals (120 in total). The fitness distributions confirm what we deduced from the previous plots. Furthermore, the 75% sub-interval shows the best improvement of all, even though the optimization time spent is longer than in 50%. In fact, the optimization time scales with the optimization sub-interval used as expected, while the number of generations needed to converge is quite similar for all them.

It is interesting to discuss here the optimization time we spent on our experiments. As shown in Figure 8.12c, we spent 47 hours on average in optimizing the 25% sub-interval (30 runs), 108 in optimizing the 50% one, 173 in 75%, and 226 in the 100% sub-interval. These long times reported are mainly due to the evaluation time required as we needed to simulate the city, using SUMO.

The data presented in Table 8.7 confirm that 75% is the best ranked interval (3.9) according to the Friedman test, followed by the 50% (3.1), 25% (2.0), and 100% (1.0). Wilcoxon p -values were also calculated to ensure that these values are statistically significant.

Table 8.8: Fitness, travel times, gas emissions, and fuel consumptions obtained when using Yellow Swarm in the four training scenarios of Quito configured with the best solution calculated by the EA for the four proposed sub-intervals. We also include here the values of Quito without optimization.

| | Quito | 25% | | 50% | | 75% | | 100% | |
|----------------------|------------|--------|---------|---------------|----------------|--------------|----------------|--------|---------|
| Fitness | 1.000 | 1.025 | (2.5%) | 1.048 | (4.8%) | 1.061 | (6.1%) | 1.020 | (2.0%) |
| Travel Time (s) | 1067.4 | 1007.3 | (5.6%) | 934.6 | (12.4%) | 939.9 | (11.9%) | 973.9 | (8.8%) |
| CO (g) | 15.2 | 14.5 | (4.5%) | 13.7 | (9.8%) | 13.6 | (10.2%) | 14.3 | (6.0%) |
| CO ₂ (g) | 1686.1 | 1641.6 | (2.6%) | 1590.7 | (5.7%) | 1593.5 | (5.5%) | 1633.7 | (3.1%) |
| HC (mg) | 529.5 | 504.5 | (4.7%) | 474.7 | (10.4%) | 475.2 | (10.3%) | 493.2 | (6.9%) |
| PM (mg) | 156.2 | 152.4 | (2.4%) | 148.4 | (5.0%) | 147.9 | (5.3%) | 152.7 | (2.2%) |
| NO _x (mg) | 3129.8 | 3053.0 | (2.5%) | 2966.9 | (5.2%) | 2967.3 | (5.2%) | 3047.5 | (2.6%) |
| Fuel (ml) | 672.2 | 654.5 | (2.6%) | 634.2 | (5.7%) | 635.3 | (5.5%) | 651.4 | (3.1%) |
| Distance (km) | 5.6 | 5.7 | (-0.2%) | 5.7 | (-0.8%) | 5.7 | (-1.2%) | 5.8 | (-1.8%) |

Finally, we report the metrics when using Yellow Swarm in Quito, configured with the best solution achieved by the optimization process of each sub-interval. Table 8.8 shows the fitness and the average values of Travel Time, CO, CO₂, HC, PM, NO_x, Fuel, and Distance. We observe that, in spite of the best fitness values of 75%, 50% seems to be slightly better in most of the metrics. Based on the Friedman ranks we chose the configuration for Yellow Swarm obtained by the optimization using the 75% time interval to configure and test our proposal in 30 different unseen scenarios in Quito. However, 50% would have been a valid choice as well, especially if we wanted shorter optimization times.

8.4.4 Validation on 30 Unseen Scenarios

Our next step was to test the best configuration calculated by the EA in 30 unseen scenarios in Quito. We activated Yellow Swarm between 8:30 and 12:15 (the 75% sub-interval) so that the detours were presented to vehicles only during this interval, while the rest of the day, the panels were off. Additionally we collected the metrics of the traffic over an entire day to evaluate the impact of our proposal over 24 hours for 30 unseen traffic distributions (scenarios). Table 8.9 presents the improvements obtained, including the average of each metric, the standard deviation as a percentage of the mean value, minimum, and maximum values, for the city of Quito without Yellow Swarm.

We can observe that Yellow Swarm reduces each emission metric, travel times, and fuel consumption, even in the most difficult scenarios. The drivers following the detours signaled by the Yellow Swarm achieve travel times 11.9% shorter, reduced by 9.6% their CO emissions, 5.3% in CO₂, 10% in HC, 4.7% in PM, 4.9% in NO_x, and consumed 5.3% less fuel. Moreover, the distances traveled are negligibly longer (1.3% maximum) as we divert vehicles via alternative routes which are not part of the shortest path.

We report data from 30 different testing scenarios, also showing minimum and maximum improvements in Table 8.9 where the most remarkable improvements are travel times which are 28.4% shorter, a reduction of 13.2% in fuel consumption, and 23% in HC.

Table 8.9: Improvements achieved in the traffic of Quito city during an entire day when using Yellow Swarm just for 225 minutes (75% of the morning peak hours).

| Metrics | Improvement | | | |
|-----------------|-------------|--------|---------|--------------|
| | Average | StdDev | Minimum | Maximum |
| Travel Time | 11.9% | 0.4% | 5.3% | 28.4% |
| CO | 9.6% | 0.3% | 5.2% | 19.4% |
| CO ₂ | 5.3% | 0.4% | 2.4% | 13.2% |
| HC | 10.0% | 0.4% | 4.8% | 23.0% |
| PM | 4.7% | 0.3% | 2.4% | 9.1% |
| NO _x | 4.9% | 0.4% | 2.3% | 11.3% |
| Fuel | 5.3% | 0.4% | 2.4% | 13.2% |
| Distance | -1.2% | -0.1% | -1.3% | -1.0% |

8.5 Discussion

In this chapter we have studied the reduction of travel times, greenhouse gas emissions, and fuel consumption of road traffic in three big cities using the Yellow Swarm architecture.

We have designed an evolutionary algorithm to optimize the training scenarios by using a series of LED panels to propose detours to vehicles, and found solutions that improve upon the experts' ones during the optimization process with statistical significance. Two different approach were followed. First, we minimized travel times in *Malaga* and *Madrid* and observed also reductions in gas emissions and fuel consumption. Second, we maximized the vehicles throughput in peak hours of *Quito* to obtain also shorter travel times and less emissions. In both cases, we observed an increase in route lengths (2.6% max.) which is a consequence of detouring vehicles via alternative streets which are not included in the shortest path.

As we have considered different cities (topologies, avenues, roundabouts, intersections) we have observed several variations in the results. However, we have improved all the metrics, even when only 10% of vehicles are obeying the instructions of Yellow Swarm. This, and our tests in several different unseen scenarios confirm Yellow Swarm as a valid and inexpensive strategy to optimize road traffic. The use of Yellow Swarm in cities, would mean that people not only get to work early and not spend their precious time stuck in traffic jams, but also they would be healthier as they are breathing in cleaner air.



UNIVERSIDAD
DE MÁLAGA

Chapter 9

Smarter Routes for GPS Navigators

In this chapter a new way of calculating alternative routes for GPS navigators is proposed in order to foster a better use of the city's streets. The experimentation presented involves maps from OpenStreetMap, real road traffic, and the microsimulator SUMO, to reduce travel times, greenhouse gas emissions, and fuel consumption in the city. Additionally, an analysis of the sociological aspect of our proposal is done by observing the penetration (acceptance) rate which shows that our strategy is competitive even when just 10% of the drivers are using it.

9.1 Introduction

Global Positioning System (GPS) navigators are now present in most vehicles and smart-phones nowadays, as they are needed when driving through an unknown city or neighborhood. The usual goal of these navigators is to take the user in less time or distance to a destination.

Although some of them use data representing the current state of the road traffic to calculate the route shown to the driver, this kind of service is neither updated in real time nor available everywhere in the world. As a result, routes end up being calculated by Dijkstra [50] or A* [94] algorithms which only use the length of the streets and their average speed to find the best way to reach a destination (shortest path). Thus, the global use of navigators in a given city could lead to traffic jams as they have a highly biased preference for some streets.

In this chapter we present an alternative way of calculating routes [198] based on the concept of dynamic user equilibrium. The alternative routes can be provided (and updated) as a complement to the cartography so they can be used by GPS navigators to improve traffic flows when assigning routes to vehicles driving through a city. From a general point of view, spreading the traffic throughout the city could be a way of preventing jams and making a better use of public resources, reducing traffic jams, gas emissions and fuel consumption, and improving the quality of life of citizens.

9.2 Dynamic User Equilibrium (DUE)

The traffic assignment problem consists of assigning routes to vehicles which are moving from their origin to their destination, usually taking into account variables such as cost and benefits. It can be solved by calculating the user equilibrium route choice in which routes are assigned to vehicles so that an alternative assignation would have worsened travel times. According to the Wardrop's first principle [234], the user equilibrium is the state in which every driver chooses a route for which the travel time is minimal. Consequently, the resulting network state is in equilibrium, since nobody can improve his travel time by choosing a different route.

We have used an approach to the assignment model which is based on an iterated simulation [78] to calculate the dynamic user equilibrium (DUE) by using tools provided by SUMO. This model uses a probability distribution for the route choice so that a route is stochastically picked for each vehicle traveling from its original location to its destination.

Our proposal consists in calculating the dynamic user equilibrium and using the new routes generated to help a GPS navigator in rerouting drivers through different streets to reach their destination, instead of using the shortest path. Concretely, we divide the city into *ad hoc* zones and use the resulting input and output streets as origin and destination of the routes throughout the zone. Then, when a vehicle enters the zone with the intention of driving through it, it will follow one of the available routes according to a previously calculated probability. Note that local trips (i.e. those whose starting and destination points are within this area) are not considered by our proposal as our intention is to favor the traffic flows that are crossing the area (and the city).

An example of the route assignment process when a driver intends to cross a defined zone is given in Figure 9.1. The best route in terms of distance is obviously route A. Alternatively, there are two other routes, B and C, which despite being longer, may lead to a reduction of travel times for everyone, as possible congestions can be avoided by using them.

Our proposal involves not only calculating these routes but also testing three new strategies to obtain the probabilities of using them to drive through the analyzed zone, and prevent

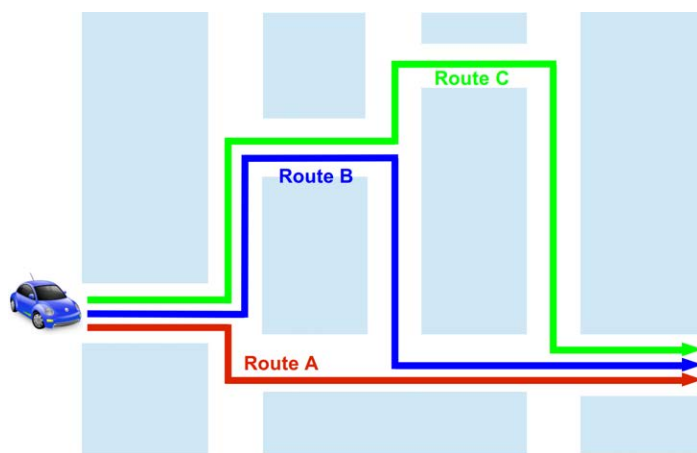


Figure 9.1: Possible routing example.

traffic jams. Of course, the probabilities not only depend on the streets' distributions (which is the reason we are using OpenStreetMap as the source for the maps), but also on the number and behavior of the vehicles involved (we use microsimulations and actual traffic data). In the following sections we describe this three new strategies.

9.2.1 DUE.r & DUE.rp

The traffic simulator SUMO (Simulation of Urban MObility) [123] includes a tool, written in Python, called *DualIterate*, which is used to calculate the dynamic user equilibrium as described in [78]. Using *DualIterate* we have calculated the DUE for our case study and extracted the different routes, whose origins and destinations are the input streets and exits, respectively of the area under analysis, as presented in Algorithm 9.1.

First, the initial trips from the case study (*malaga*) are obtained in order to maintain the initial demand when calculating the new routes. Second, by using *DualIterate* the probabilities are initialized and the first traffic simulation is carried out to assign routes to vehicles and obtain travel times. After each traffic simulation, the probabilities are updated according to the travel time values measured in the simulation so that the probability of assigning a route is higher for those with lower travel times. This process is repeated until the algorithm converges or the maximum number of steps is reached and *DualIterate* ends.

Finally, the routes resulting from the DUE process are used to build the *DUE.rp* (Dynamic User Equilibrium routes by probability) strategy, in which the probability of choosing a route from a starting point to a destination from those available, depends on how frequently it has been assigned by *DualIterate*. Additionally, the *DUE.r* (Dynamic User Equilibrium routes) assignation is obtained by keeping just the different routes (without repetition) so that all the routes from each origin to a destination are equiprobable.

We propose a third strategy to assign the routes included in *DUE.r*. Instead of assigning them according to how frequently they are used (*DUE.rp*) we propose an Evolutionary Algorithm (EA) to calculate the best probabilities for each route to minimize travel times. We have named our proposal *DUE.ea* and it is described in the next section.

Algorithm 9.1 DUE Routes.

```

procedure DUE ROUTES
  trips  $\leftarrow$  SUMO(malaga)                                ▷ OD Matrix
  Pd  $\leftarrow$  initializeProbabilities()                       ▷ begin DualIterate
  while not TerminationCondition() do
    travelTimes  $\leftarrow$  SUMO(malaga, trips, Pd)
    Pd  $\leftarrow$  updateProbabilities(travelTimes)
  end while                                                ▷ end DualIterate
  routes  $\leftarrow$  SUMO(malaga, trips, Pd)
  DUE.rp  $\leftarrow$  routes                                       ▷ DUE.rp
  DUE.r  $\leftarrow$  getUnique(routes)                             ▷ DUE.r
end procedure

```

9.2.2 DUE.ea

We have designed a (10+2)-EA (an elitist steady state evolutionary algorithm with a population of ten individuals generating two new individuals at each step) to calculate the probabilities of assigning one of the available routes to vehicles which are driving through the area under analysis. These routes were previously calculated by the aforementioned *DualIterate* utility. A diagram of the processes followed by *DUE.ea* is shown in Figure 9.2.

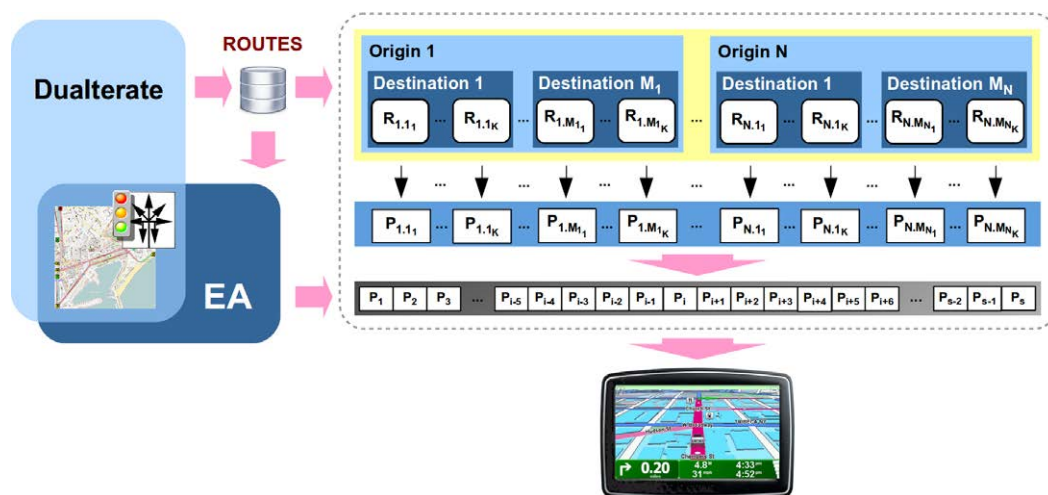


Figure 9.2: *DUE.ea* diagram and solution encoding. In our case study $N = 8$, $S = 121$.

Solution Encoding

The solution encoding consists of a numeric vector corresponding to the probabilities for the routes to be chosen. The probability values correspond to the different routes from the possible origins and their available destinations in area being analyzed.

Figure 9.2 shows the representation of the problem where N origins are arranged into blocks containing the M_n reachable destinations from each origin. Finally, each destination could be reached by $K_{n,m}$ routes which have an associated probability value and are restricted so that the sum of them in a destination block is equal to 1. Note that the number of possible destinations (M) and routes (K) for each origin is not always the same as it depends on the streets' connectivity where not all destinations can be reached from each origin in the area analyzed. Our case study (Figure 9.5) contains 121 routes between their eight origins and seven destinations, so that the problem representation is a vector of 121 probability values.

Fitness Function

We define the fitness function presented in Equation 9.1 to reduce travel times and later evaluate the rest of the metrics (gas emissions, fuel consumption, and route lengths) as a way of checking how robust our solution is and what relationships are observed.

$$F = \frac{1}{\alpha} \frac{1}{N} \sum_{i=1}^N travel\ time_i \quad (9.1)$$

The coefficient α is calculated as described in Equation 9.2. It is used to normalize the value returned by the fitness function so that the evaluation of each scenario (sc) of the case study is equal to 1.0. Consequently, fitness values lower than 1.0 indicate improvement in the average travel times as our aim is to minimize them, i.e. the lower, the better.

$$\alpha_{sc} = \frac{1}{N_{sc}} \sum_{i=1}^{N_{sc}} travel\ time(sc)_i \quad (9.2)$$

$$sc \in \{malaga_{WD}, malaga_{SAT}, malaga_{SUN}\}$$

Operators

The selection strategy implemented in the EA is Binary Tournament. We have used a standard two point crossover (Street Two Point Crossover) as the recombination operator where the crossing points are the origin blocks as shown in Figure 9.3. It exchanges entire block of probabilities between individuals, which gives the operator the ability to build new configurations at the block level.

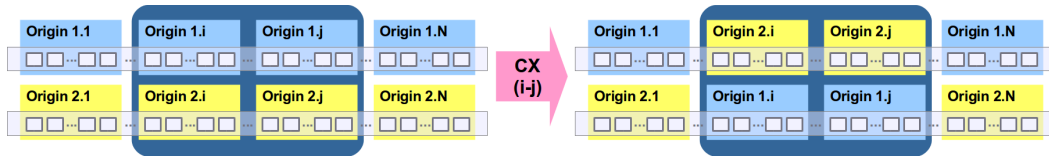


Figure 9.3: Crossover operator. Probabilities values for sensors i to j are selected to be exchanged.

Additionally, for the mutation operator, we have designed an operator that changes the probability values for the routes in a destination block by first selecting one of them, then incrementing its value, and finally decrementing the rest, in order to keep the sum total equal to 1 (Figure 9.4). In the example the destination j of the origin i has been selected for mutation. Then, probability P_{i,j_3} for route R_{i,j_3} is randomly selected to be incremented by 0.1 (probability increment). We can see in the resulting individual that not only has P_{i,j_3} been incremented, but also the probabilities for the rest of routes in destination j have been decremented to keep the sum total equal to 1.

We have experimentally set the crossover probability (P_c) to 0.9, the mutation probability (P_m) to 0.1, and the probability increment performed by the mutation operator was set to 0.1. Finally, we have performed an elitist replacement, so that the worst individuals of the population are replaced if they have a fitness value higher than the offspring produced in the current generation in order to build the next one.

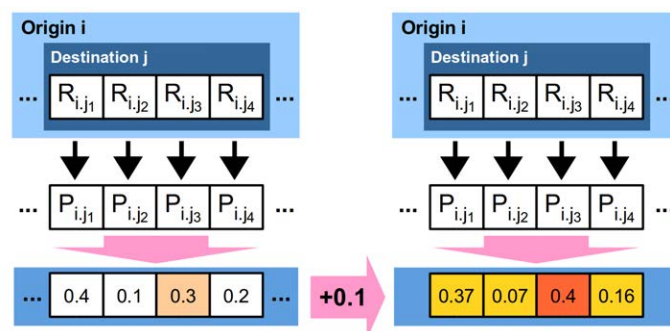


Figure 9.4: Mutation operator. Probability value of $P_{i,j3}$ has been selected to be increased from 0.3 to 0.4 according to the defined increment 0.1. The rest of values are proportionally decreased to keep the sum total equal to 1.

9.3 Case Study

We have chosen as our case study an area of the city center of Malaga (Spain), well-known for suffering from traffic jams. The geographical area studied is delimited to the north by *San Bartolomé* Street and *Ferrándiz* Street, to the west by the *Guadalmedina* River, to the east by *Keromnes* Street, and to the south by the Mediterranean Sea, which encompasses an area of about 3 km² in total.

We have imported the chosen area (shown in Figure 9.5) into the SUMO traffic microsimulator [123] from OpenStreetMap [169]. This allows us to work with a real scenario, e.g. streets, traffic lights, left turns, and roundabouts. The traffic flows for our case study were calculated using the method presented in Chapter 5 based on the Flow Generator Algorithm

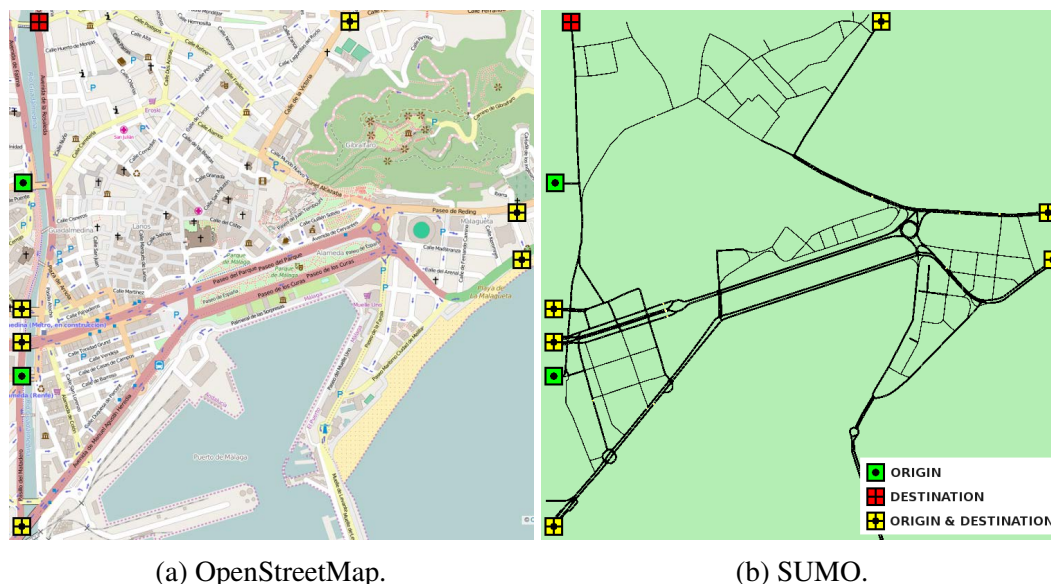


Figure 9.5: City center of Malaga. The original map (a) and how it looks after importing it into SUMO (b). Note that most of the missing streets correspond to pedestrian ways.

(FGA) [197, 201]. The algorithm assigns vehicles to the traffic flows generated by the program DUAROUTER included in the SUMO software package. This assignation adjusts the number of vehicles in the simulation to the values measured by real sensors in the city.

Using the data published by the local council of Malaga consisting of 12 sensors, we have obtained three different scenarios corresponding to average traffic per hour during working days ($malaga_{WD}$), Saturdays ($malaga_{SAT}$), and Sundays ($malaga_{SUN}$). The real number of vehicles in the city (Real), the value measured at each sensor when simulating the generated scenarios (FGA), and the difference percentage (Diff) are presented in Table 9.1.

Table 9.1: Real number of vehicles and the values measured at each sensor during the simulation for the three scenarios generated by the FGA.

| Sensor | $malaga_{WD}$ | | | $malaga_{SAT}$ | | | $malaga_{SUN}$ | | |
|--------|---------------|-------|-------------|----------------|-------|-------------|----------------|-------|-------------|
| | Real | FGA | Diff. | Real | FGA | Diff. | Real | FGA | Diff. |
| 5 | 1071 | 1069 | -0.2% | 900 | 900 | 0.0% | 803 | 803 | 0.0% |
| 6 | 347 | 348 | 0.3% | 273 | 276 | 1.1% | 227 | 229 | 0.9% |
| 7 | 279 | 278 | -0.4% | 246 | 246 | 0.0% | 208 | 208 | 0.0% |
| 8 | 254 | 251 | -1.2% | 239 | 240 | 0.4% | 212 | 212 | 0.0% |
| 9 | 256 | 256 | 0.0% | 248 | 248 | 0.0% | 222 | 221 | -0.5% |
| 10 | 644 | 646 | 0.3% | 641 | 640 | -0.2% | 584 | 584 | 0.0% |
| 13 | 229 | 230 | 0.4% | 214 | 214 | 0.0% | 172 | 171 | -0.6% |
| 14 | 479 | 479 | 0.0% | 444 | 443 | -0.2% | 348 | 352 | 1.1% |
| 15 | 631 | 633 | 0.3% | 566 | 567 | 0.2% | 467 | 469 | 0.4% |
| 16 | 518 | 518 | 0.0% | 420 | 422 | 0.5% | 359 | 358 | -0.3% |
| 17 | 839 | 854 | 1.8% | 684 | 683 | -0.1% | 617 | 619 | 0.3% |
| 18 | 600 | 602 | 0.3% | 466 | 469 | 0.6% | 437 | 440 | 0.7% |
| Avg: | 512.3 | 513.7 | 0.4% | 445.1 | 445.7 | 0.3% | 388.0 | 388.8 | 0.4% |

9.4 Results

We tested our proposal in our case study for one hour, to obtain not only travel times but also greenhouse gas emissions, fuel consumption, and distance traveled by vehicles. Additionally, we performed a penetration rate study to know if our proposal would be useful when is used by a little percentage of users.

9.4.1 Optimization

First, we took the three scenarios of our map ($malaga_{WD}$, $malaga_{SAT}$, and $malaga_{SUN}$) calculated by using the FGA as explained in Section 9.3. To achieve the desired precision (greater than 99.6% in all the scenarios) we performed 90 independent runs of the FGA (30 per scenario) which lasted 5.2, 3, and 2.6 hours, respectively.

Second, we obtained the Dynamic User Equilibrium routes ($DUE.r$), and the $DUE.rp$ (Dynamic User Equilibrium routes by probability) as explained in Section 9.2. This process took about 5 minutes to converge.

Then, we have tested the *DUE.r* and *DUE.rp* routes in our scenarios by making the GPS navigators to suggest these routes. Note that *DUE.r* routes are equiprobable, while in *DUE.rp*, the route probability depends on how much they have been assigned when calculating the user equilibrium.

Furthermore, we tested the Dijkstra shortest path algorithm [50] (*Dijkstra*) to include its results as we believe that it is the strategy most used by GPS devices nowadays. The implementation of this algorithm and the weight function used in it are provided by SUMO, which takes into account the travel time according to the street characteristics of the city.

Finally, we calculated new probabilities for the DUE routes using our EA (*DUE.ea*) and tested them in our scenarios, as well. We performed 30 independent runs of the EA on each scenario (90 runs) which lasted 3.5, 4, and 3 hours on average, respectively. Note that we used several machines to execute the 30 independent runs in parallel so that we just had to wait for the longest execution to get our results (8.3, 6, and 4.7 hours).

Table 9.2 shows the results obtained in terms of Travel Times (TT), Carbon Monoxide (CO), Carbon Dioxide (CO₂), Hydrocarbons (HC), Particulate Matter (PM), Nitrogen Oxides (NO_x), Fuel consumption (Fuel), and traveled distance (Distance). Note that we are supposing that all the drivers crossing the area have a GPS device and follow the indications given.

We can see that in spite of the reduced travel times (and emissions) produced by *DUE.r*, *DUE.rp* and even *Dijkstra*, the shortest travel times are obtained in the three scenarios when using *DUE.ea*. Our strategy also has the lowest emissions and fuel consumption as vehicles arrive at their destinations earlier, avoiding possible traffic jams. Differences in distances between the strategies are negligible (variations below 1%).

In Figure 9.6 the results obtained are presented as improvement percentages when vehicles are being routed according to the strategies analyzed here instead of following the flows obtained from the available real data. We can see that the greatest improvements

Table 9.2: Results obtained for the scenarios when vehicles are using the routes based on data publish by the Malaga local council (*Malaga*), shortest path (*Dijkstra*), Dynamic User Equilibrium (*DUE.r*), Dynamic User Equilibrium with probabilities obtained by rate of use (*DUE.rp*), and Dynamic User Equilibrium with probabilities obtained by our EA (*DUE.ea*). The best values are in bold.

| Scenario | Strategy | # Veh. | TT (s) | CO (mg) | CO ₂ (mg) | HC (mg) | PM (mg) | NO _x (mg) | Fuel (l) | Dist. (m) | Friedman Rank | Wilcoxon <i>p</i> -value |
|-----------------------------|----------|--------|--------------|---------------|----------------------|-------------|-------------|----------------------|--------------|---------------|---------------|--------------------------|
| <i>malaga_{WD}</i> | Malaga | 4883 | 351.6 | 1591.9 | 322840.7 | 88.6 | 20.7 | 554.1 | 128.7 | 1926.6 | 3.20 | 0.00 |
| | Dijkstra | 4883 | 297.3 | 1424.7 | 304507.5 | 79.6 | 19.9 | 526.7 | 121.4 | 1917.4 | 3.00 | 0.00 |
| | DUE.r | 4883 | 294.5 | 1401.5 | 302745.6 | 78.8 | 19.8 | 523.5 | 120.7 | 1924.6 | 2.98 | 0.01 |
| | DUR.rp | 4883 | 292.7 | 1390.5 | 301328.7 | 78.3 | 19.7 | 521.0 | 120.1 | 1924.1 | 2.93 | 0.09 |
| | DUE.ea | 4883 | 288.5 | 1374.9 | 299418.3 | 77.4 | 19.6 | 518.1 | 119.4 | 1922.3 | 2.90 | — |
| <i>malaga_{SAT}</i> | Malaga | 3961 | 344.1 | 1547.7 | 323919.4 | 87.1 | 20.9 | 557.0 | 129.1 | 2004.9 | 3.18 | 0.00 |
| | Dijkstra | 3961 | 324.7 | 1481.6 | 316290.6 | 83.6 | 20.5 | 545.3 | 126.1 | 2000.2 | 3.06 | 0.00 |
| | DUE.r | 3961 | 303.8 | 1399.7 | 309326.1 | 80.0 | 20.2 | 534.2 | 123.3 | 2008.0 | 2.95 | 0.00 |
| | DUR.rp | 3961 | 314.0 | 1421.3 | 310741.4 | 81.2 | 20.2 | 535.6 | 123.9 | 2003.5 | 2.97 | 0.00 |
| | DUE.ea | 3961 | 291.7 | 1363.9 | 305130.4 | 77.9 | 20.0 | 528.1 | 121.7 | 2011.0 | 2.84 | — |
| <i>malaga_{SUN}</i> | Malaga | 3679 | 279.6 | 1292.4 | 291131.9 | 74.0 | 19.1 | 503.9 | 116.1 | 1933.3 | 3.09 | 0.00 |
| | Dijkstra | 3679 | 275.7 | 1269.0 | 287901.5 | 72.9 | 18.9 | 498.4 | 114.8 | 1928.6 | 2.99 | 0.05 |
| | DUE.r | 3679 | 275.8 | 1261.6 | 288565.0 | 72.9 | 18.9 | 499.4 | 115.0 | 1945.0 | 3.04 | 0.02 |
| | DUR.rp | 3679 | 273.6 | 1248.3 | 286268.5 | 72.3 | 18.7 | 495.4 | 114.1 | 1937.9 | 2.96 | 0.03 |
| | DUE.ea | 3679 | 271.1 | 1232.5 | 284807.0 | 71.6 | 18.6 | 492.9 | 113.5 | 1940.3 | 2.92 | — |

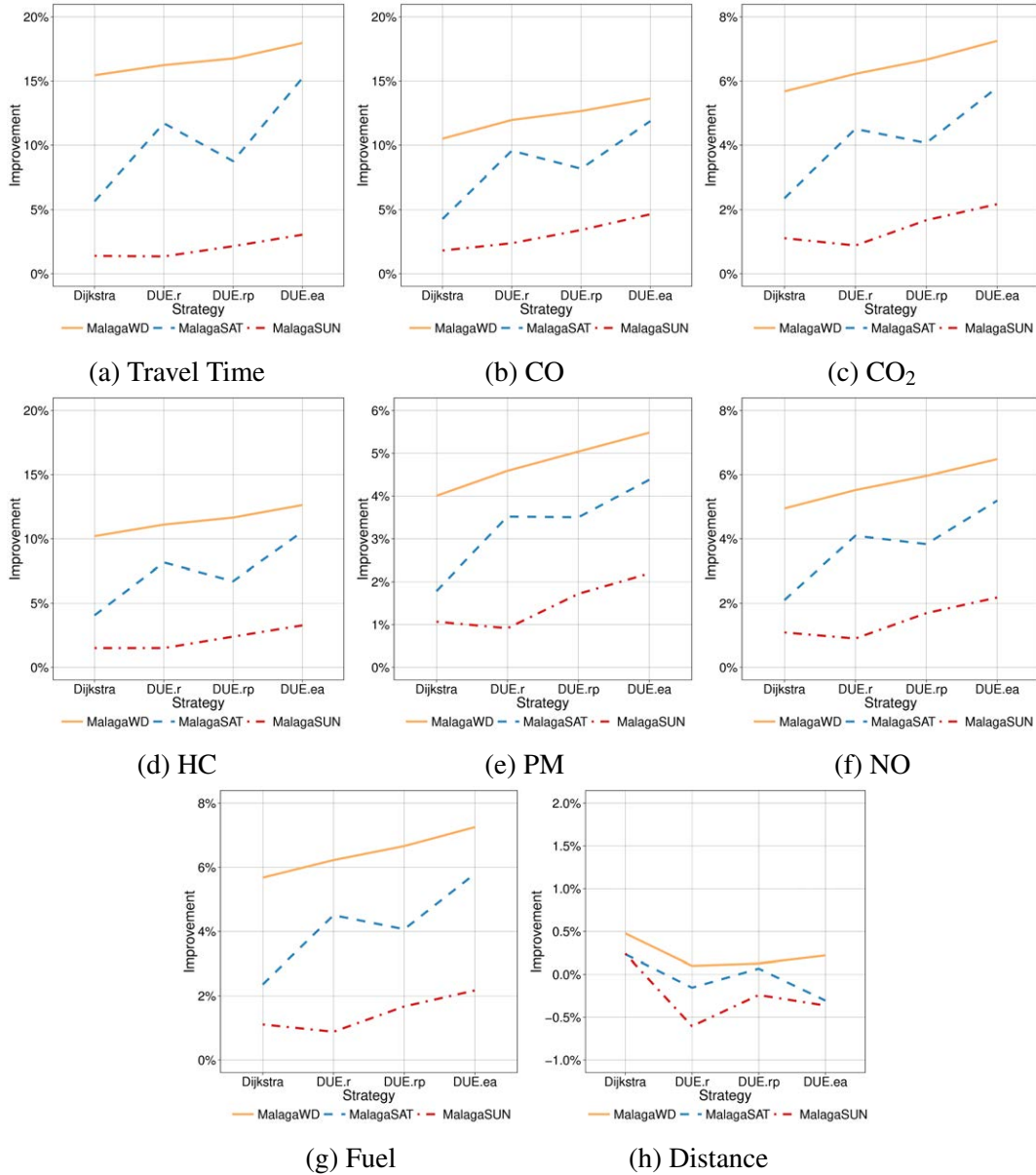


Figure 9.6: Improvements in the metrics for the three scenarios of our case study when using different strategies for routing vehicles instead of the routes obtained from the data published by the local council. Note that some of the scales used are different for better visualization.

are achieved when there are more vehicles in the area (working days) and that *DUE.r* and *DUE.rp* perform better than *Dijkstra* which was expected as they have more routes available for vehicles. However, *DUE.ea* outperforms all of them in all the scenarios and metrics, achieving improvements in travel times (up to 18%), CO (up to 14%), and fuel consumption (up to 7%). Additionally, a statistical analysis is provided (Friedman Ranks and Wilcoxon *p*-values) showing that our results are statistically significant.

By using *DUE.ea* in the GPS navigators the city's streets are exploited better by vehicles, with drivers leaving the analyzed zone, on average 63 seconds earlier. Although our initial concern was to shorten travel times, a better flow of vehicles preventing congestion has also reduced pollution levels as well as fuel consumption (9.3 liters per hour on average).

9.4.2 Penetration Rate

We also wished to know if our proposal would be useful when it is not being used by every single vehicle (a very real assumption). To answer this question, we tested the configuration (probabilities) achieved by *DUE.ea* when no one was using it (*Malaga* real traffic) and incrementing the penetration rate in steps of 10% until reaching a full usage (*DUE.ea* values previously reported in Table 9.2).

We present our penetration rate results in Figure 9.7. It can be seen that, despite some variations which make the increment inconsistent, all the metrics are improved with respect to *Malaga* (0% usage) as the penetration rate increased. There is at least a minimum improvement when just 10% of the drivers are using the routes and probabilities calculated by *DUE.ea* in their GPS device, which gets better as the penetration rate increases.

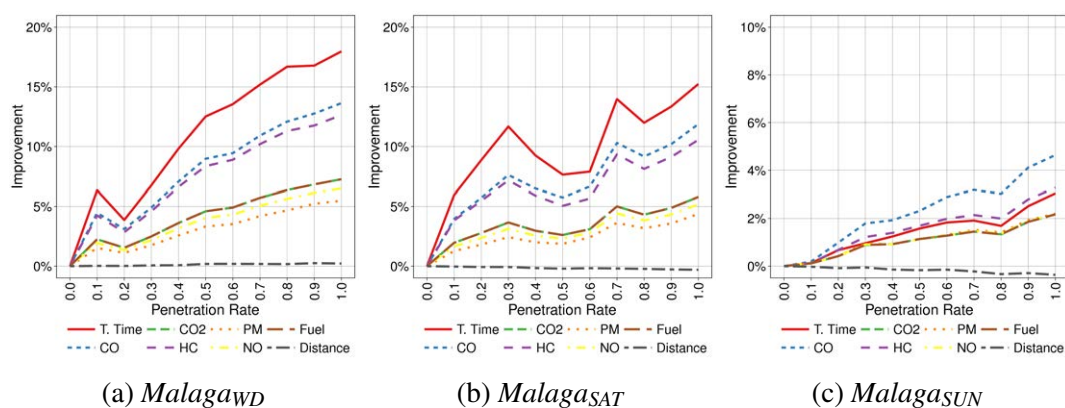


Figure 9.7: Penetration rate study for the three scenarios of our case study. Note that some of the scales used are different for better visualization. Even with a reduced use of our technique (10% of drivers), travel times and other metrics are improved.

9.5 Discussion

In this chapter we have proposed a way of calculating alternative routes to be used by a GPS navigator. Additionally, we have provided three different strategies to select which of these routes are presented to the drivers by the navigator, and compared them with each other and with the Dijkstra shortest path algorithm. Our results show that we have improved travel times (up to 18%), greenhouse gas emissions (up to 14%), and fuel consumption (up to 7%) in *Malaga* when using *DUE.ea*. Furthermore, we have also demonstrated that our proposal is viable even when just 10% of drivers are using it.

Chapter 10

Know Your City: Car Park Spots

In this chapter a study of parking occupancy data of Birmingham, Glasgow, Norfolk, and Nottingham in the U.K. is addressed. Different prediction strategies such as polynomial fitting, Fourier series, K -means clustering, and time series, are analyzed. Moreover, cross validation has been used to train the predictors and to test them with unseen occupancy data. Finally, a web service to visualize the current and historical parking data in a map is also presented. It allows users to consult the occupancy rate prediction in order to satisfy their parking needs.

10.1 Introduction

Finding an available parking space could be difficult in most cities, especially in the city center. Off-street car parks are a viable alternative, especially when the number of inhabitants in urban areas is increasing and expected to rise to 75% of the world's population by 2050 [17]. On-street parking spaces are quite limited and usually it is cheaper to find an off-street car park or pay and display bays rather than wasting time (and fuel) in finding a free space. Nevertheless, even paid spaces are scarce nowadays as, unfortunately, city infrastructures have not grown in line with population growth.

Smart cities initiatives are here to take care of this [158]. Whether it implies populating the city with sensors [68] or developing several apps [44] to encourage citizens not only to use them but also to take an active part in the developing of the future smart city, the final goal is to take advantage of the new technologies to improve our quality of life.

Despite how fast we can reach our destination by using different optimization strategies to prevent traffic jams [75], there is a need of finding an available parking space at the end of our trip. There are some studies which have already addressed this issue by using a time Markov model [119], regression trees, neural networks and support vector regression [253], or a real-time availability forecast algorithm [33], so that all the previous improvements do still apply.

Although monitoring single parking spaces may not be economically viable, it is possible to count the number of vehicles entering and leaving an off-street car park and make these data publicly available to help make decisions (and predictions) based on them. In this

chapter we address the study of parking occupancy data of Birmingham, Glasgow, Norfolk, and Nottingham in the U.K. We aim to test several prediction strategies such as polynomial fitting, Fourier series, K -means clustering, and time series, and analyze their results. We have used cross validation to train the predictors and then tested them with unseen occupancy data. Additionally, we have developed a web service to visualize the current and historical parking data in a map, allowing users to consult the occupancy rate forecast in order to satisfy their parking need up to one week in advance. We believe that the use of these accurate intelligent techniques conducts to final user services for citizens living in real smart cities as a way of improving their quality of life, shortening wait times, and reducing fuel consumption.

Concretely, our proposal is a system to collect public data of car park occupancy values, show them in a user-friendly web service, store them to be consulted as a historical archive (most existing data sources provide only values corresponding to the last measurements), and use these past data to predict the car parks' occupancy rate of the following week. Thus, citizens can use our proposed system to decide where to go, where to park and to know when is the best moment to go there if they wish to find an available parking space.

10.2 Car Park Occupancy Prediction

Our main objective was to predict the future occupancy rate of car parks by using the previous occupancy data collected from different data sources [34, 209, 210]. In addition, we wished to analyze and compare our six different predictors taking into account not only their accuracy, but also the number of parameters (complexity) needed to model a car park.

The architecture developed to predict car park occupancy rates [209, 210] is shown in Figure 10.1. Currently, there are four downloaders which obtain occupancy values from the different data sources (Birmingham, Glasgow, Norfolk, and Nottingham in this study). Then, data is parsed by the data parser which put them in a common format to be stored in the database. Finally, these data can be obtained from the database anytime to be shown to the users as well as to be used by the predictors to obtain future occupancy values. This process could be used to study any car park dataset available online as it does not depend on the codification or format.

We have analyzed six predictors as shown in Figure 10.2. Some of them are well-known prediction techniques; however, each one presents different characteristics to be exploited.

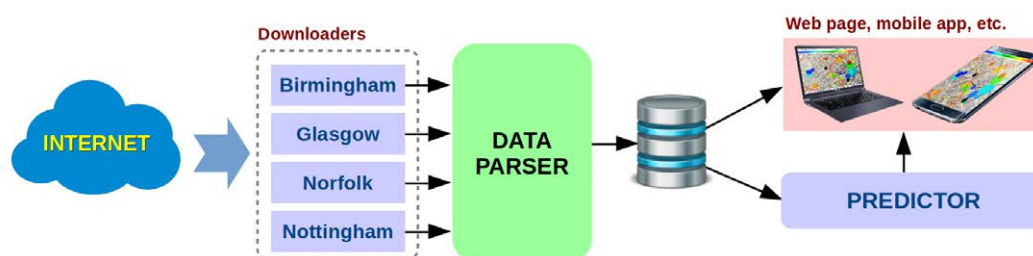


Figure 10.1: Schema of the prediction system architecture where the relationship between the downloaders, the data parser, the data storage, the predictor and the web prototype, are shown.

Our aim was not only to find the more accurate predictor, but also keep the model simple so that we could represent each car park and weekday by using the minimum amount of data. Each predictor is described as follows:

10.2.1 Polynomial Fitting (P)

This predictor consists in a polynomial fitted to each car park and weekday. A polynomial describes a continuous function composed of terms of different degree. We have used it to fit the several points that represent the occupancy values. We studied different polynomial degrees to find which value presented a more accurate prediction keeping a reduced number of parameters (its degree plus one) to represent each car park and weekday.

10.2.2 Fourier Series (F)

This predictor consists in fitting a Fourier series to each car park and weekday. Formally, Fourier series decomposes a periodic function into the sum of sinuses and cosines which can be used to fit a curve. In our case we considered different numbers of components (predictor's parameters) as alternatives which are always odd numbers, because an extra constant term.

10.2.3 K-Means (KM)

Clustering by using *K*-Means is a method that allows grouping pairs of car parks and weekdays in different clusters whose centroid represents the whole set of occupancy measures in the group. This technique was initially used in signal theory, however, nowadays is also popular for cluster analysis and data mining. We wish to describe sets of car parks behaving similarly during several weekdays by using the corresponding centroid as occupation rate values instead of the individual ones. By using KM we can describe a set of car parks and weekdays by using only the number of parameters which correspond with the number of values in the centroid.

10.2.4 KM-Polynomial (KP)

This predictor fits a polynomial to the existing centroid points of each cluster calculated by *K*-Means. This step was necessary to improve the accuracy of the predictions by interpolating a polynomial to the points in each centroid as they are spaced according to the frequency of the measures (30 minutes for Birmingham and 15 for Glasgow, Norfolk and Nottingham).

10.2.5 Shift & Phase (SP)

To improve the accuracy of the prediction even further, we defined a new predictor which uses the KM-Polynomials described in the previous section and adds two new parameters to the existing coefficients, in order to modify the shift (*y* axis) and the phase (*x* axis) of the original polynomial. By doing so, we are adding a little complexity to the predictor to customize it to each car park and weekday.

10.2.6 Time Series (TS)

Time Series required a different approach. This is a well-known predictor which consists of a series of values which are stored in time order to be statistically analyzed in order to predict future values. This predictor needs the largest number of parameters for each car park, growing in size as we include more training data.

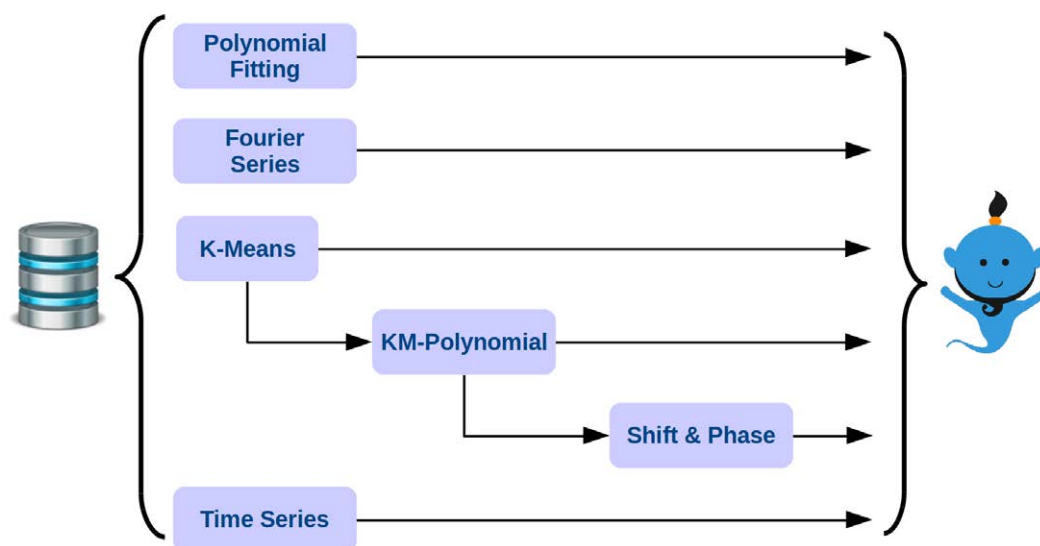


Figure 10.2: Predictors analyzed and their relationship.

10.3 Case Studies

Several cities and counties in the United Kingdom have been publishing their open data to be used, not only by researchers and companies, but also for citizens for better know the place where they live. They are published under the U.K. Open Government Licence (OGL) [222] or even Creative Commons Attribution [168], what allowed us to conduct this study.

Concretely, we have used data from the cities of Birmingham, Nottingham, Glasgow, and the county of Norfolk, (Figure 10.3), all in the U.K. We have selected them for this study, not only because the availability of data, but also because the number of car parks and values they offer. Each region is describe as follows:

10.3.1 Birmingham

This is a major city in the West Midlands of England, standing on the small River Rea. It is the largest and most populous British city outside London, with an estimated population of 1,124,569 as of 2016 [164]. We have analyzed valid data of 22 car parks in Birmingham after filtering the original dataset of 33 [26].

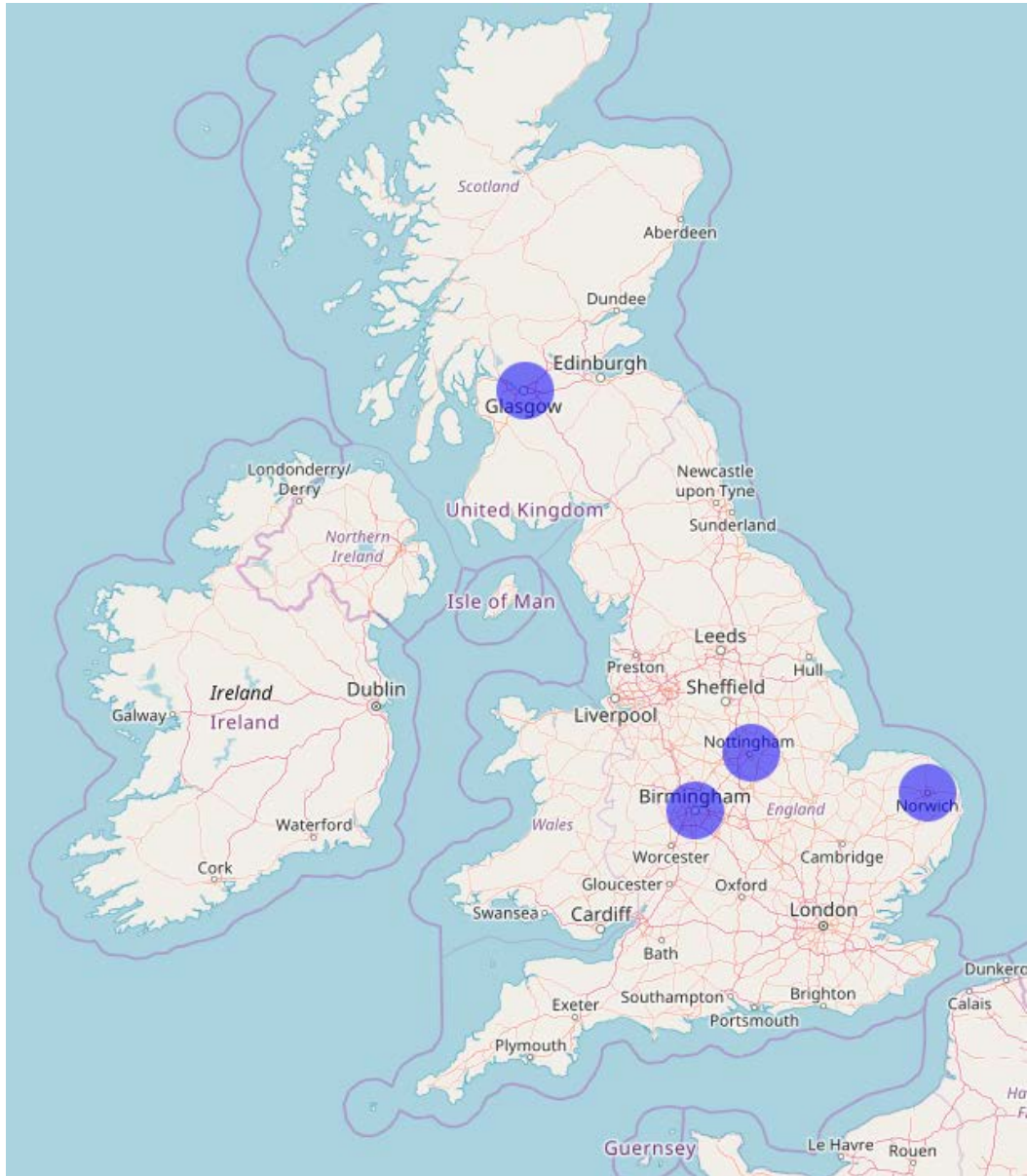


Figure 10.3: Case studies: Birmingham, Nottingham, Glasgow, and Norfolk.

10.3.2 Glasgow

Being the largest city of Scotland, and the third one in the U.K, Glasgow is situated on the River Clyde in the West Central Lowlands of the country. The estimated number of inhabitants in the Greater Glasgow urban area is 1,209,143 [156]. The Glasgow city council has published a dataset consisting of 18 car parks [82] many of which have been removed after applying our quality filters. Thus, we ended up with just five valid car parks to be analyzed in Glasgow.

10.3.3 Norfolk

This is a county in England whose county town is Norwich. Its boundaries are Lincolnshire to the west and north-west, Cambridgeshire to the west and southwest, and Suffolk to the south. The northern and eastern limits are the North Sea and, The Wash, in the north-west. Norfolk is a largely rural county with a population of 892,870 [164]. The available car park dataset of Norfolk is composed of 18 car parks [161], although only eight of them presented valid data.

10.3.4 Nottingham

This city of the East Midlands in England, belongs to Nottinghamshire. According to [164] Nottingham had an estimated population of 325,282 in 2016. Additionally, its urban area is the largest in the east Midlands and the second-largest in the Midlands. Nottingham city council has published a dataset including 47 car parks [162] which after being thoroughly filtered contains just 12 valid ones.

10.4 Training

Before using our predictors they needed to be trained. By doing so, we let them to create a model of the occupancy of each car park and weekday by providing a training dataset. The training dataset passed through a previous filtering stage of the available input data so that each car park did not present duplicates, its occupancy values were not constant, there were at least 75% of the number of expected values, and each car park had data for each weekday.

After that, the missing values (if any) in the occupancy dataset were completed by using the average value of the four previous weekdays when an entire day was missing, or by replicating the previous value when just one single measurement was needed. For example, if one car park's data for Monday 31st was missing (dataset sources usually stop working or individual sensors become temporarily faulty) the occupancy values for that day would be generated by using the average values of that car park for Mondays 3rd, 10th, 17th, and 24th.

To test how well a predictor behaved, i.e. how accurate it was, we decided to use the average Mean Squared Error (MSE) [129] of all the predictions done so that the lower this value, the better. To improve the training process we have used *K*-Fold cross validation as we have done in [210]. This method consists of dividing the entire dataset into subsets (folds, weeks in our study), choosing one as the testing set and using the rest as the training set.

For example, in Birmingham we had 193 training days giving us 27 weeks for training (it begins on Tuesday and ends on Wednesday). They allowed us to perform 27 training processes for each car park in that city, the first one using weeks 2nd to 27th and testing with the 1st, etc. After calculating the MSE for the 27 training processes we have chosen the training set with the lower MSE value.

We set up a training process for each predictor by creating training sets for each weekday. As each dataset has a different number of available days, we ended up with 28 folds for Birmingham and Glasgow, 57 for Norfolk, and 26 for Nottingham. This training process was

Table 10.1: Characteristics of the available datasets before and after applying our training filter.

| Dataset | Date Range | Available Car Parks | Valid Car Parks | Values Per Day | Training Values | Training Days | Training Sets |
|------------|----------------------------|---------------------|-----------------|----------------|-----------------|---------------|---------------|
| Birmingham | 04-Oct-2016 05-Apr-2017 | 32 | 22 | 18 | 95,733 | 193 | 27 |
| Glasgow | 27-Oct-2016 07-May-2017 | 9 | 5 | 96 | 227,275 | 193 | 28 |
| Norfolk | 27-Oct-2016 23-Nov-2017 | 15 | 8 | 64 | 388,908 | 393 | 57 |
| Nottingham | 27-Oct-2016 24-Apr-2017 | 22 | 12 | 96 | 633,926 | 181 | 26 |

useful not only for selecting the best training set (the one which minimized the prediction errors) but also for selecting the best parameters for each predictor (polynomial degrees, Fourier series' components, etc.), so that it is accurate enough with a minimum complexity.

Table 10.1 summarizes the characteristics of the datasets corresponding to the case studies analyzed including the number of training days and folds. Note that despite the number of training days, the number of training values depends also on the number of measurements per day which differs between cities. Additionally, the SP and TS predictors do not have any parameters to be set (the former depends on the KP's degree and the latter does not need any parameter). SP was also trained using the different folds to obtain the best, accurate results. We have used a different approach for TS, incrementing the number of weekdays in each iteration and testing with the following one as the training dataset has to be time ordered.

After training our predictors we have selected the parameters for P, F, KM, and KP, by using the *elbow* method [214]. We dealt with a trade-off between accuracy and number of parameters in our study so that we tried to keep the number of parameters as low as possible, as shown in Table 10.2.

Additionally, we present the distribution of car parks and weekdays in different clusters (KM predictor) as a result of the training process in Figure 10.4. We can see that in Birmingham (Figure 10.4a) some car parks were included in more than one cluster (variability), while in Glasgow (Figure 10.4b) the first cluster is for car park *CPG21C* and the rest of car parks were assigned to the second one. Figure 10.4c shows the cluster distribution for Norfolk where half of car parks are in both clusters (different weekdays), and finally, in Figure 10.4d we can see the only existing cluster (according to this predictor all the occupancy data follow

Table 10.2: Parameters for the predictors calculated by using *K*-Fold cross validation.

| Dataset | Polynomials (P) (degree) | Fourier Series (F) (# components) | <i>K</i> -Means (KM) (# clusters) | KM Polynomials (KP) (degree) |
|------------|-----------------------------|--------------------------------------|--------------------------------------|---------------------------------|
| Birmingham | 2 | 3 | 3 | 2 |
| Glasgow | 4 | 3 | 2 | 4 |
| Norfolk | 3 | 3 | 2 | 3 |
| Nottingham | 4 | 3 | 1 | 4 |

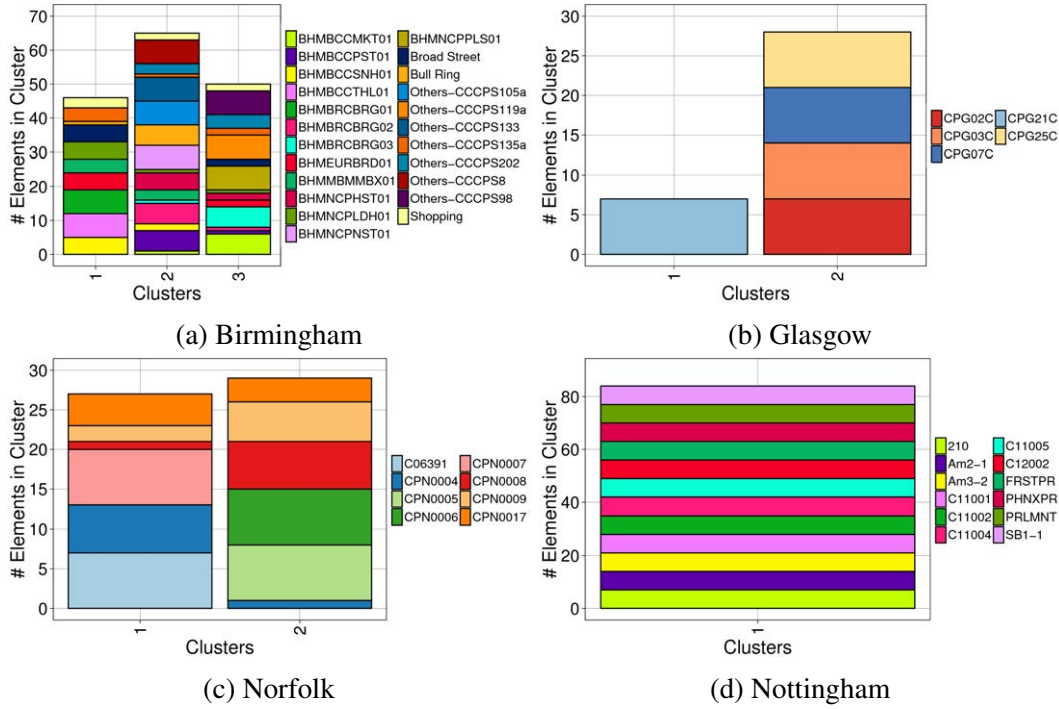


Figure 10.4: Car parks in each cluster for Birmingham, Glasgow, Norfolk, and Nottingham, when using k -means to group car parks and weekdays.

a similar pattern throughout the weekdays) for Nottingham where all the car parks and week days were included.

Furthermore, we present the training of the SP predictor by using K -Fold cross validation in the datasets of Birmingham (Figure 10.5a), Glasgow (Figure 10.5b), Norfolk (Figure 10.5c), and Nottingham (Figure 10.5d). We can see there that the average MSE varies considerably depending on the chosen training subset (n -th fold). All in all, the more accurate predictions were achieved for the 8th fold in Birmingham, the 2nd in Glasgow, the 53rd in Norfolk, and the 14th in Nottingham.

In Figure 10.6 we present the training of the TS predictor. As was mentioned before, the number of days in each training process is increased keeping the right time order of the values. We can see that despite some anomalous fluctuations, TS is quite accurate, presenting the lowest MSE values after the third weekday of training in the four case studies.

10.5 Testing

After training our predictors it was time to test how well they performed when used to predict an entire unseen week. To do that, we used the last week of data (not seen in the training process) to compare their real occupancy values to the ones provided by our six predictors, P, F, KM, KP, SP, and TS.

Figure 10.7 shows the box plots corresponding to the results achieved by our predictors in Birmingham, Glasgow, Norfolk, and Nottingham. Note that we are plotting the MSE so

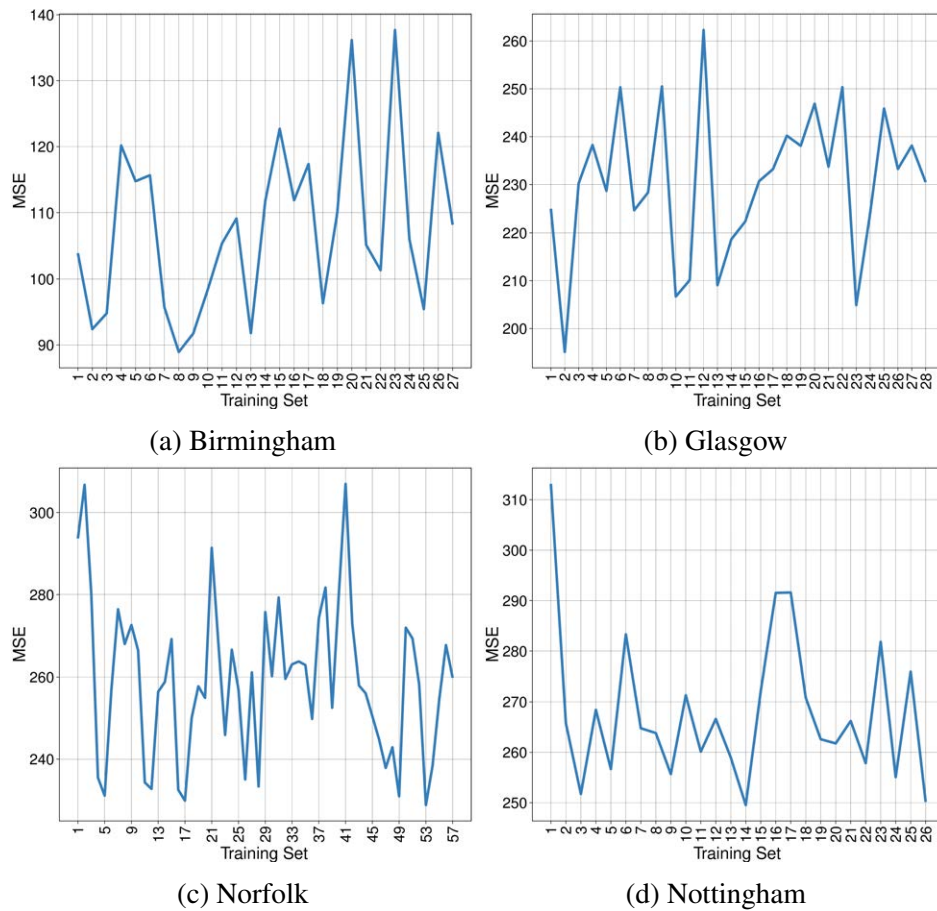


Figure 10.5: Training of Shift + Phase (SP) by using K -Fold cross validation with our datasets.

that lower values are better, and that in order to favor the graphical comparison between predictors, the graphs do not include most of the few existing outliers. We can see that in Birmingham (Figure 10.7a), TS has achieved the most accurate predictions, especially on Mondays, Thursdays, and Fridays, while P, F, and SP were quite accurate as well, although their MSE values were bigger. The accuracy values of the predictions done for Glasgow are shown in Figure 10.7b. We can see that TS presented the lowest MSE, except by Mondays, where there was a bigger variability in its results. P and F have shown good results too, while

Table 10.3: Average MSE values achieved after testing our predictors on an unseen week and comparing the predicted values against the real ones.

| Dataset | Polynomials (P) | Fourier Series (F) | K -Means (KM) | KM Polynomials (KP) | Shift & Phase (SP) | Time Series (TS) |
|------------|--------------------|-----------------------|--------------------|------------------------|-----------------------|---------------------|
| Birmingham | 42.3 | 63.1 | 89.7 | 94.9 | 46.0 | 29.8 |
| Glasgow | 106.8 | 127.5 | 286.7 | 286.2 | 167.8 | 38.7 |
| Norfolk | 171.1 | 255.6 | 340.1 | 334.9 | 276.0 | 85.6 |
| Nottingham | 89.8 | 104.8 | 374.9 | 374.6 | 223.3 | 34.7 |

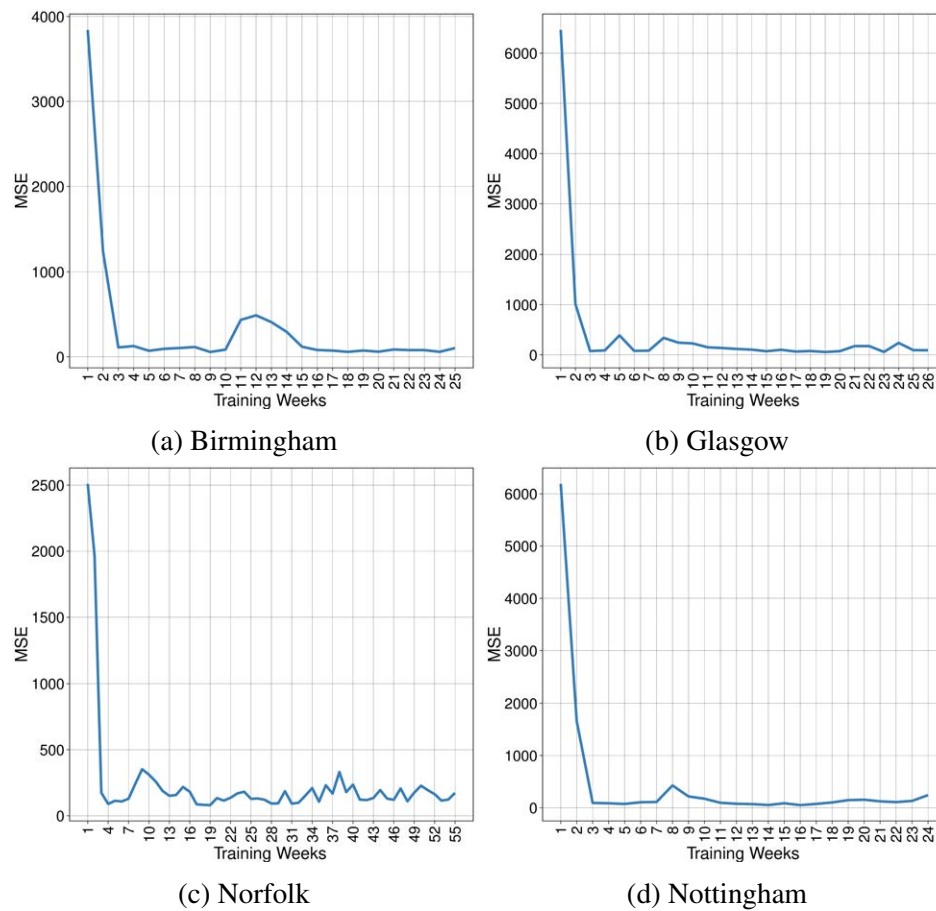


Figure 10.6: Training of Time Series (TS) with the datasets for our case studies.

KM and KP were far away of the desired accuracy. Note that, as KP interpolates the KM points, their results were similar for most weekdays and scenarios. Figure 10.7c shows the MSE values for our six predictors in Norfolk. Again TS presented the most accurate values, this time far better than the rest of the predictors.

Finally, in Figure 10.7d we can see the accuracy values achieved for the car parks in Nottingham. This last experiment confirmed that TS was the best predictor of our comparison, followed by P and F. These results are also presented in Table 10.3, where we can see that SP has always improved KM and KP which was its main aim. Nevertheless, its accuracy was lower than TS to be considered as an option.

10.6 Web Prototype

As a way of bringing all these prediction techniques to the inhabitants of a smart city we have developed a web service prototype. In Figure 10.8a we show the web page where the past, current and future occupancy rates of a car park can be seen. The region, day and hour can be selected and if it is a future date, the corresponding prediction done by the TS predictor is

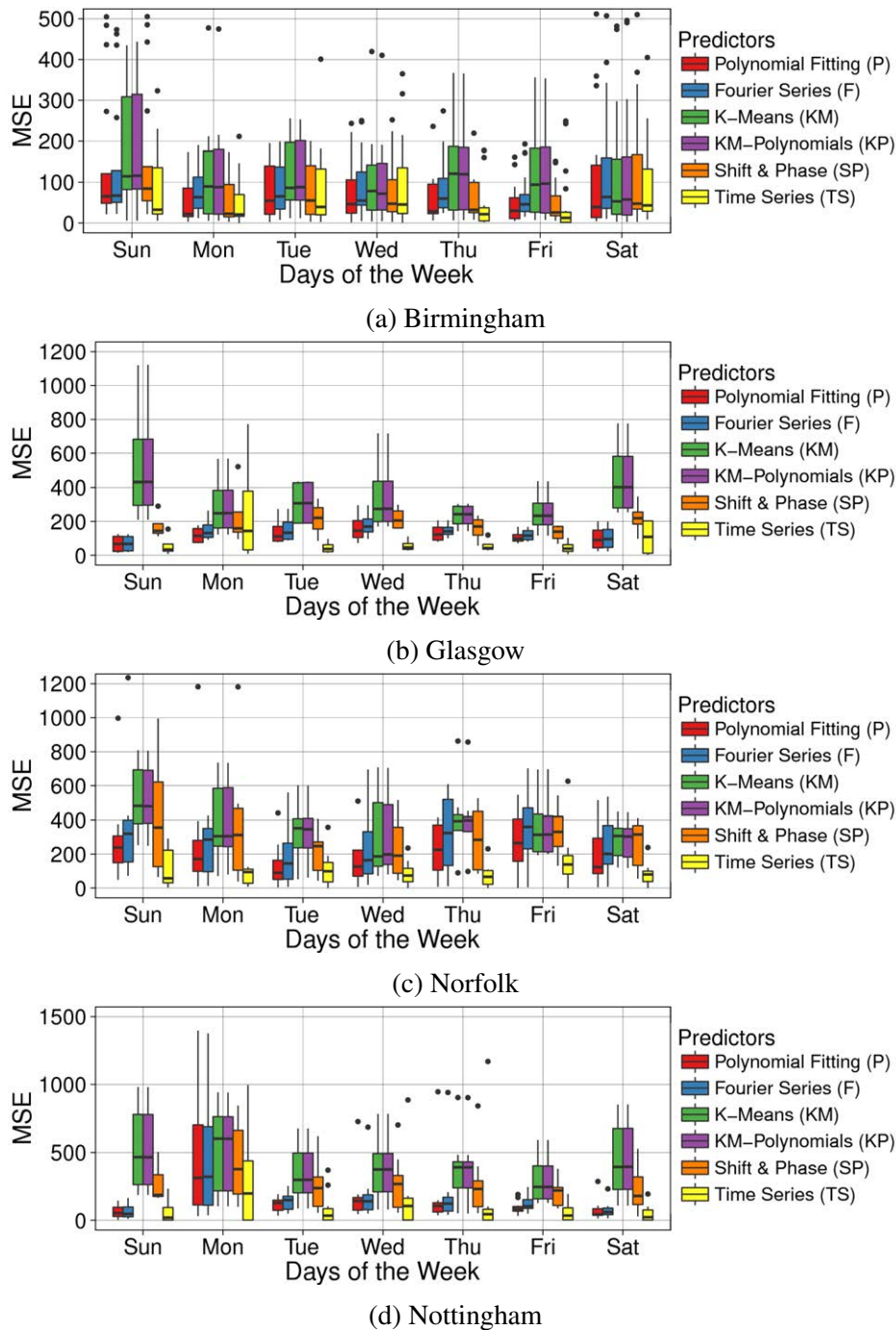
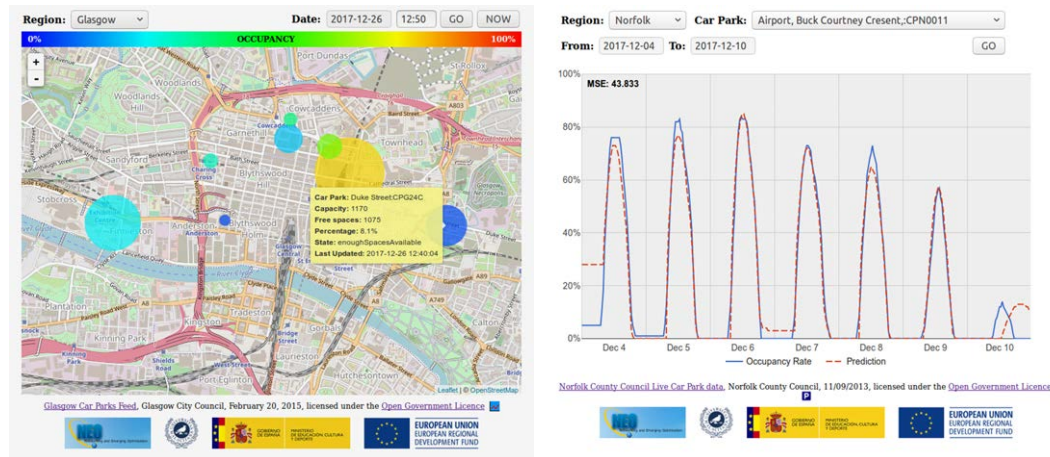


Figure 10.7: Comparison of the accuracy of our predictors for our case studies over the weekdays.

shown (up to 7 days in advance). By using this service users could plan their trips, looking for available parking spaces in the future to decide when is the best moment for going to downtown according to their needs.

Furthermore, in Figure 10.8b the prediction stats page is shown. It can be accessed from the main page to see how accurate the predictions have been in the past for each car park and



(a) Real-time occupancy.

(b) One-week prediction.

Figure 10.8: Snapshots of our web prototype showing the current occupancy of the car parks of Glasgow (a) and the predictions done (red dashed line) compared to the real values (blue continuous line) for one car park of Norfolk (b)

region. At the top of the page the region, car park, and date range can be selected, while in the central graph a continuous blue line indicates the real occupancy rate and a dashed red line, the values predicted by the TS predictor. Finally, in the upper left corner of the graph the calculated MSE value for the selected data range is shown.

10.7 Discussion

In this chapter we have presented six very accurate predictors for forecasting car park occupancy rates in Birmingham, Glasgow, Norfolk, and Nottingham. We have trained them by using real data published by local councils and presented the results obtained after testing them with one week of unseen parking data. Our results show that TS turned out to be the most accurate predictor although it required the larger amount of data to represent each car park and weekday. Polynomials and Fourier series also performed quite well, the former needed between two and four parameters to represent each car park and weekday, while the latter always needed three. Hence, their use can be interesting in some applications where the size of the model is more relevant than an extreme accuracy. Moreover, our proposal includes a novel web service that can be used for real, despite the fact that there are web pages offering information on car park's occupancy rates, they rarely make predictions of the next day's state nor offer historical data. The use of our proposal in a smart city would mean that people are not only wasting their precious time looking for a parking space, but also they would be healthier as they are living a happier, less stressed life, breathing in a cleaner air.

Part III

New Intelligent Algorithms



UNIVERSIDAD
DE MÁLAGA

Chapter 11

New Bio-inspired Algorithms

This chapter presents a new set of ideas on how to build bio-inspired algorithms based on the new field of epigenetics. By analyzing this domain and extracting working computational ideas we want to offer a set of tools for the future creation of representations, operators, and search techniques that can competitively solve complex problems. To illustrate this, we describe an epiGenetic Algorithm, analyze its behavior and solve a set of instances of the multidimensional knapsack problem. Since we are in some measure opening a new line of research, we include a description of epigenetics and computational search, show their working principles and show an example algorithm solving a real problem. Our aim is to offer ideas as well as put them to work, to show that they are actually competitive, not just a nice new inspiration.

11.1 Introduction

According to the Lamarckian inheritance, organisms are able to transmit a number of characteristics acquired during its lifetime to their offspring. Later, Darwin in his theory of evolution by natural selection, rejected Lamarck theories as it explains the existing variations as random mutations that arise in the genome of an individual which are passed to offspring. After that, Mendelian inheritance was set in a form of three laws which explained the inheritance in terms of genes, which are passed from one generation to the next based on rules of probability. This led to a general neglect of the Lamarckian theory of evolution in biology. Recently, epigenetics has turned up the interest in Lamarckism, as it involves the possible inheritance of behavioral traits acquired by the previous generation from the expression of only one allele in the same nuclear environment [157].

Darwin, and many others before (e.g., Owen) and after him (e.g., Weismann), contributed to dismissing some of the ideas of J.B. Lamarck [151]. In fact, several of these ideas were clearly, correctly discredited, such as the linear concept of evolution (known as *recapitulation*). The inheritance of acquired characters from Lamarck was also dismissed at this moment until now [192]. *Epigenetics* (relating heritable traits that respond to the environment and cannot be explained by changes in DNA sequences) is here to stay, after so much evidence of its existence and so many applications in biology, pharmacology and medicine [62, 66, 191].

It seems that we *have to* consider a second “cock of the walk” for a better understanding of nature, and that is what we do in this study: we combine the ideas of Darwin and Lamarck into one single computational algorithm, obviously within the niche of research into nature-inspired algorithms. In addition to a nice, appealing inspiration for future work, we of course aim for a competitive evaluation of the resulting techniques, to prove that not only are epigenetic algorithms new as a class (not a renaming of an existing algorithm!), but also that they are useful for solving combinatorial and other kinds of problems in modern research.

Metaheuristics for combinatorial optimization problems [27] are frequently inspired by natural processes such as Darwin’s theory of evolution: *evolutionary algorithms* are today a classic example [14]. Some algorithms work better with specific types of problems and perform worse over others [243]. This first study is targeted to develop a new bio-inspired algorithm family based on epigenetics, that can be later adapted to different problems. This is possible not just in the traditional manner (as with EAs) but also by using a variety of epigenetic mechanisms that we successfully translate from biology to computer optimization.

In summary, this chapter studies the diverse epigenetic mechanisms controlled by specific DNA methylation as a method of modifying DNA expression, that may be reversible and inheritable. From this study, we define a methodology to generate EAs that efficiently solve many different problems by using epigenetic concepts on information representation, such as histones, nucleosomes, and chromatin, and epigenetic operations such as genomic imprinting, reprogramming, paramutation, position effect, X-inactivation, bookmarking, and gene silencing. As initially stated, we move from the pure inspiration to the traditional optimization analysis, to ensure that this idea is noteworthy to know and usable in the future of bio-inspired computing.

11.2 Background

Epigenetics have inspired several articles in computer science in the last decade, we comment on some of them here. Even if the term has been mentioned in the past, nothing similar to a methodology for building efficient algorithms has been developed in these few papers.

In [172] the authors describe the optimization strategies that bio-molecules utilize and propose an intragenerational epigenetic algorithm based on them. The authors also present an agent-based cell modeling and simulation environment, called SwarmCell, whose model has been built as an autopoietic system that represents a minimal biological cell. Then, they implement the epigenetic strategies in the model to better clarify the disease mechanisms at the sub cellular level. This strategy’s proposed study cancer development patterns in different cell types which have been differentiated by various trans-generational epigenetic mechanisms. The authors state that their epigenetic algorithm can prove to be a fundamental extension to existing evolutionary systems and swarm intelligence models. They discuss improving problem-solving capabilities by implementing epigenetic strategies in their model. Finally, for future work, they intend to develop a trans-generational epigenetic algorithm to demonstrate how the internal organization of a system can pass on its traits to the next generation. Although epigenetic techniques are also proposed in their work, our study focuses

on a new set of algorithms based on natural evolution rather than autopoietic systems and swarm models.

An epigenetic approach in artificial life (ALife) is presented in [195] where the model proposed (EpiAL) uses a dynamic environment to influence the regulation of organisms and the possible inheritance of epigenetic acquired marks. The objective of the EpiAL model is to study the plausibility for the existence of epigenetic phenomena and its relevance to an evolutionary system, from an ALife point of view. Therefore, each agent is able to modify its phenotypic expression due to environment conditions, pass on epigenetic marks between generations enabling the existence of acquired traits which can be transmitted through consecutive generations of agents. The experimentation performed with the EpiAL model in order to study the mechanisms that influence the evolution of the agents shows that the epigenetic populations are able to regulate themselves for dynamic conditions, while the non epigenetic populations find it hard to prosper in dynamic environments. The authors plan a future development of the model with a focus on both, biological knowledge (developmental biology) and possible problem solving techniques (dynamic environments). This epigenetic approach is focused on the evolution of epigenetic agents to gain more knowledge about this field while we propose an algorithm and solve problems with it.

The authors in [219] incorporate an explicitly controlled gene expression through histone modification in strongly-typed genetic programming (STGP) and call it, epigenetic programming. They propose a double cell representation of the simulated individuals represented by their respective chromatin structures. The authors view their proposed approach of epigenetic programming as a form of epigenetic learning (EL) incorporated into genetic programming via the beneficial modifications of histone code, which take place within the life cycle of evolved simulated organisms. They achieve phenotypic diversity of genotypically similar individuals by using the cumulative effect of polyphenism. They preserve individuals from the destructive effects of crossover by silencing genotypic combinations and explicitly activate them when it is more beneficial. Based on the empirically obtained results, the authors indicate that epigenesis contributes to a 2.1-fold improvement in the computational effort of genetic programming when it is used to evolve the social behavior of predator agents in the predator-prey pursuit problem.

Although these three approaches use concepts related to epigenetic theory, to the best of our knowledge, none of them have proposed an epigenetic algorithm to solve problems related to combinatorial optimization as we do in this PhD thesis. We believed that the epigenetic model, including problem representation and operators, has not been comprehensively described and exploited to build a working search algorithm, especially in those studies concerned with solving complex optimization problems. Additionally, we compare our proposal to other well-known models and evaluate their numeric results to see whether our results are competitive or even can improve upon theirs. We first present the natural processes explaining epigenetics.

11.3 Epigenetics From an EA Representation

A DNA molecule consists of two strands coiled around each other forming a double helix. Each strand is composed of *nucleotides* containing *nucleobases* such as guanine (G), adenine (A), thymine (T), and cytosine (C) [63]. DNA is organized into long structures called chromosomes (23 pairs in humans, consisting of approximately 25000 genes) which are duplicated during cell division. Inside each chromosome, proteins such as *histones* compact and organize DNA to guide the interactions with other proteins, controlling which genes are expressed. DNA molecule carries genetic information that can be passed from one generation to the next [12]. This is the concept that we can find in current EAs, where the chromosome is a vector of symbols representing DNA genes, usually in a haploid manner (although some diploid representations were once proposed in [88] and [194]).

In contrast to the classic Mendelian inheritance of phenotypic traits, caused by mutations of the DNA sequence, under the natural selection explained by Darwin's theory of evolution, epigenetic changes are long-term alterations in the transcriptional potential of a cell, due to the activation of certain genes, that are not necessarily heritable [5].

Epigenetics is the study of the biological mechanisms which cause longterm alterations in the transcriptional potential of cells (first step of gene expression, in which a particular segment of DNA is copied into RNA) during their development without changing the DNA sequence, i.e. it does not involve mutations of the DNA itself [23]. These alterations can be heritable and, perhaps, not visible in the next generation but in a generation after. The gene expression process might also be modified by environmental factors [193], diet, personal habits, aging, or random changes, which may contribute to the development of abnormal phenotypes [112]. In addition, epigenetic marks between generations can be reset, and the genome reverted to its original state [249]. Epigenetic processes are essential for development and differentiation, but they can also be present in mature humans as well.

In the nucleus of *eukaryotes* (organisms whose cells contain a nucleus enclosed within membranes), DNA is packaged into a smaller volume so that it can fit in the cell. This combination of DNA and proteins is called *chromatin* (Figure 11.1), which also prevents DNA damage, strengthens the DNA to allow mitosis (cell duplication where the cell nucleus is separated into two identical sets of chromosomes), and controls gene expression and DNA

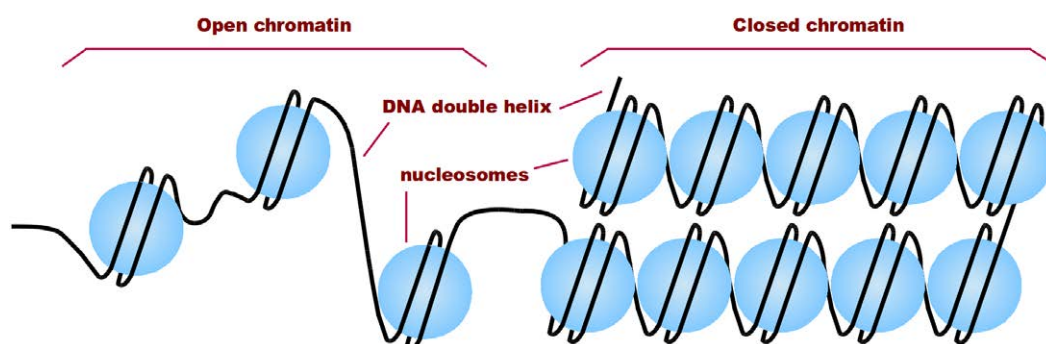


Figure 11.1: DNA packaged by the chromatin in eukaryotic cells.

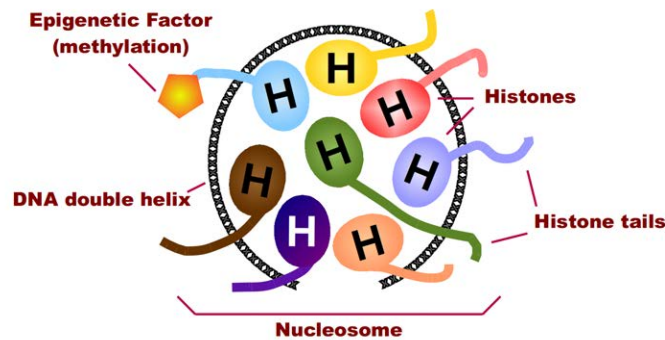


Figure 11.2: An epigenetic factor, i.e. methylation, bound to a histone in a nucleosome.

replication. During the metaphase (the most condensed and coiled stage) the structure of chromatin is optimized for physical strength and manageability, forming a chromosome structure to prevent shear damage to the DNA when the chromosomes are separated. Epigenetic chemical modification of the structural proteins in chromatin also alters its local structure.

The primary protein components of chromatin are *histones* [23], in which eukaryotic DNA is wrapped to form *nucleosomes* (Figure 11.2). Nucleosomes are the fundamental unit into which DNA and histones are packaged. They are the basic components of a chromosome where the DNA helix is wrapped around to form a series of beads compacting the DNA. Each nucleosome consists of eight histones called the histone octamer. Histones have long tails protruding from the nucleosome, which can be modified by methylation, acetylation, etc.

DNA methylation is an epigenetic factor which is recognized as the main contributor to the stability of gene expression states through mitotic cell division [104] because it establishes a silent chromatin state that modifies nucleosomes [242]. Epigenetic mechanisms constrain expression by adapting regions of the genome to maintain either gene silencing or gene activity [21]. This is achieved through direct chemical modification of the DNA region itself and by the modification of proteins that are closely associated with the location of each gene [112]. Additionally, DNA methylation and histone modification serve as epigenetic marks for active or inactive chromatin, and such epigenetic marks can be heritable [130].

Epigenetic regulation of gene expression can occur when DNA methylation is lost to allow active or inactive genetic states to be potentially reversible. If methylation fails to be maintained during multiple rounds of DNA replication, a passive loss occurs. On the other hand, active demethylation takes place in non-dividing cells and requires enzymatic activities [112].

11.4 Epigenetic operators

Epigenetic Mechanisms [5] are the temporal and spatial controllers of gene activity during the development of complex organisms [104]. DNA methylation and histone modification are clear examples of epigenetic mechanisms [182], all of which can affect long-term gene expression, which constitutes the basis for the accurate execution of developmental programs and the maintenance of the cell types over subsequent cell divisions [117].

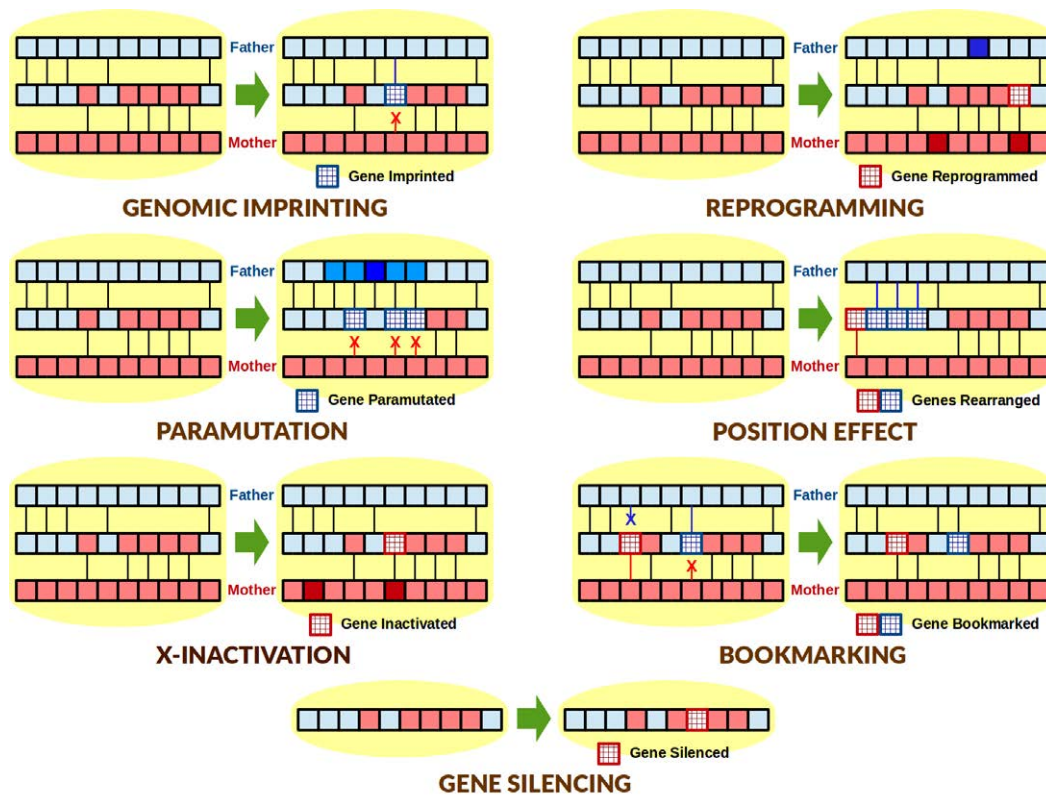


Figure 11.3: Schema of each epigenetic mechanism and the modifications made to the cell's DNA through methylation.

Epigenetic Mechanisms can be used as operators to modify the solution of a problem represented as a chromosome following the epigenetic methylation rules. Most of the mechanisms use references to the cell's parents to calculate new values for the chromosome as depicted in Figure 11.3. In the following sections we analyze seven epigenetic mechanisms.

11.4.1 Genomic Imprinting

Genome Imprinting [37] is a non-Mendelian phenomenon by which a gene expression depends on whether its origin is paternal or maternal [166, 242]. Mammals are diploid organisms whose cells have two matched sets of chromosomes, one inherited from the mother and one from the father. Therefore, mammals have two copies of each gene with the same potential to be active in any cell. Genome Imprinting changes this potential by restricting the expression of a gene to one of the parental chromosomes. If an allele inherited from the father is imprinted, only the allele inherited from the mother will be expressed, and vice versa.

11.4.2 Reprogramming

Epigenetic reprogramming [179] is an important aspect of normal mammalian development. Several changes to DNA methylation and histones are imposed on the two parental genomes during cell division, differentiation and other stages of vertebrate development. Many environmental factors, stochastic events, diet, and early experiences may contribute to the variations in the epigenome [9].

11.4.3 Paramutation

Paramutation [36] is the epigenetic alteration of one allele induced by the other one in the same location. It occurs when certain alleles impose an epigenetic imprint on the susceptible ones. The paramutated allele could be inherited in traits of later generations even if the gene behind those traits is absent. Paramutation violates Mendel's first law as alleles do not remain unchanged, which differs from the expected classical Mendelian inheritance patterns [46].

11.4.4 Position Effect

Position Effect [20] consists in the juxtaposition of genes with *heterochromatin* (a tightly packed form of DNA) either by rearrangement or by transposition, resulting in a variation of the phenotype to indicate that the gene has been silenced in cells where it is usually active.

11.4.5 X-Inactivation

Human DNA is packed into 23 pairs of chromosomes (22 pairs of autosomes and one pair of sex chromosomes) of varying size. One chromosome of each pair is inherited from the individual's father and the other from his mother. The sex chromosomes differ between the sexes so that females have two copies of the X chromosome (XX) and males have one X and one copy of the Y chromosome (XY). One of the mechanisms to compensate this difference between members of the same species is switching off genes on one of the female Xs [181]. In some cells it is the paternal X, in others it is the maternal X, but once inactive, all of the clonal descendants of the cell have the same inactive X [111].

11.4.6 Bookmarking

Gene bookmarking is an epigenetic mechanism that controls cell fate and lineage commitment as cells must propagate the gene pattern through mitosis, to daughter cells [187]. It is believed that this pattern of gene activity is somehow marked before mitosis to avoid suffering modifications and let daughter cells know how to reassemble the transcription machinery on the promoters of these genes once mitosis is completed. Bookmarking describes the retention of phenotype-specific transcription factors on mitotic chromosomes, allowing the necessary information to be conveyed to progeny cells by inheritable histone marks and DNA methylation [250].

11.4.7 Gene Silencing

Gene silencing describes the epigenetic mechanism of gene regulation by “switching off” a gene without using genetic mutation. Transcriptional gene silencing is the result of histone modifications and post-transcriptional gene silencing is the result of the messenger RNA (mRNA) destruction of a particular gene [177]. This epigenetic mechanism plays a central role in the regulation of gene expression, genome stability and is involved in defense against changes in position of a DNA sequence and RNA viruses [173].

11.5 The epiGenetic Algorithm (epiGA)

Now we have briefly presented the basis of epigenetic chromosomes and operations on them, we start the pure, computational part where we show different ideas to build algorithms based on the previous concepts.

Our novel proposal, the epiGenetic Algorithm (epiGA) [200], consists of a set of strategies, based on evolutionary computation, inspired in nature, especially in epigenetics, with the aim of solving complex combinatorial problems. The foundation of epiGA is epigenesis. We are interested in how the DNA and histones are collapsed to form nucleosomes, how this affects the gene replication during reproduction, and how the epigenetic mechanisms modify the gene expression through methylation, all of them in order to build the bio-inspired operators of our algorithm. We think this is a way of building our algorithm that, while different from known models, remains close to a standard GA which will make it easier for other authors to adopt it.

In Figure 11.4 we present the block diagram of the epiGA where the epigenetic operations are highlighted. During the *Population Initialization*, new individuals containing cells are created. The *Nucleosome Generation* creates the nucleosome structure where the DNA is collapsed and made inaccessible during reproduction. The *Nucleosome Based Reproduction* operator is where the most promising cells combine with each other following epigenetic rules. Finally, the block called *Epigenetic Mechanisms* is the place in which those mechanisms are applied according to DNA methylation and the surrounding environment.

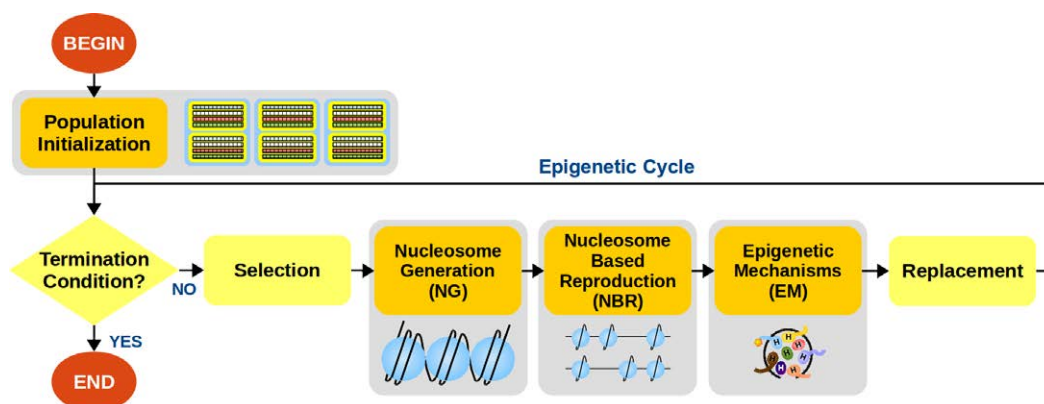


Figure 11.4: epiGenetic Algorithm (epiGA). Note that the specially built epigenetic blocks are highlighted.

Algorithm 11.1 epiGenetic Algorithm (epiGA).

```

procedure EPIGA( $N_i, N_c, P_e, P_n, R, Mechanisms, Environment$ )
   $t \leftarrow 0$ 
   $P(0) \leftarrow PopulationInitialization(N_i, N_c)$  ▷ P = population
   $Q(0) \leftarrow \emptyset$  ▷ Q = auxiliary population
  while not TerminationCondition() do
     $Q(t) \leftarrow Selection(P(t))$ 
     $Q(t) \leftarrow NucleosomeGeneration(Q(t), P_n, R)$  ▷ NG
     $Q(t) \leftarrow NucleosomeBasedReproduction(Q(t))$  ▷ NBR
    for all  $m \in Mechanisms$  do
       $Q(t) \leftarrow EpigeneticMechanisms(Q(t), P_e, m, Environment)$  ▷ EM
    end for
     $P(t+1) \leftarrow Replacement(P(t), Q(t))$ 
     $t \leftarrow t + 1$ 
  end while
end procedure

```

The pseudocode of the epiGA is described in Algorithm 11.1. There are a few necessary parameters such as these: N_i : number of individuals; N_c : number of cells; P_e : epigenetic probability; P_n : nucleosome probability; R : nucleosome radius; *Mechanisms*: set of epigenetic mechanisms to be applied; *Environment*: epigenetic environment rules. Most of these parameters depend on the problem to be solved, consequently their values will be studied in Section 12.3.

First, the number of steps t , the population $P(0)$, and the auxiliary population $Q(0)$ are initialized. Then, the main loop is executed until the termination condition holds. In the main loop, the auxiliary population Q is filled with individuals from the population P by using the *Selection* operator, as in standard GA. Next, the nucleosome chain is generated for each cell belonging to the individuals in Q . Then, the offspring is obtained after reproduction taking into account the nucleosomes previously generated.

The cells of the offspring in Q are then exposed to the epigenetic environment while the different epigenetic mechanisms are applied. Then the modified chromosome of each cell is evaluated to obtain their new fitness values. Finally, the new population $P(t+1)$ replaces the current one ($P(t)$) with the individuals of the auxiliary population $Q(t)$ by using the *Replacement* operator. This implies that the individuals of $P(t)$ with the worst fitness values are replaced by the individuals of $Q(t)$ if and only if the new ones have better fitness values (lower values if we are minimizing and vice versa). Each part of the algorithm, including its operators, are explained as follows:

11.5.1 Population Initialization

The initial population could be randomly initialized or could be created by seeding according to the problem knowledge or by a greedy algorithm. In this study we address, without loss of generality, the solution of combinatorial optimization problems represented as a binary

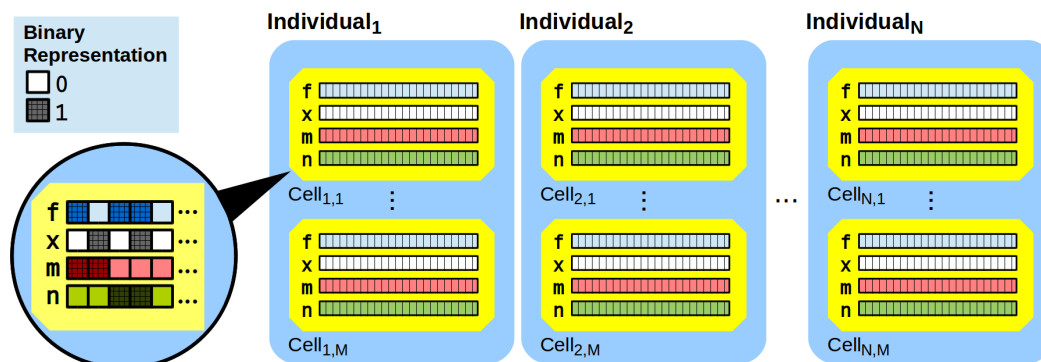


Figure 11.5: Population of the epiGenetic Algorithm.

vector. Consequently, we present in Figure 11.5 the structure of the population for epiGA. Each individual in the population made of N individuals contains M cells which can represent different solutions of the problem. Inside each cell we find four binary vectors of the same size of the chromosome (problem representation). They are the x vector (chromosome) where the solution is encoded, the f and m vectors which contain the chromosomes of the parents of the cell (f stands for father and m for mother), and finally the n vector where the binary mask (nucleosome mask) representing the nucleosome structure is stored.

The pseudocode of the *Population Initialization* is presented in Algorithm 11.2. The parameters N_i and N_c are the number of individuals and cells that will be in the population. Each new cell is generated by the constructor function, *Cell*, which fills the x vector with a new solution that can be either randomly generated or made especially, depending on the problem's characteristics. After that, the cell is evaluated to obtain its fitness value according to its chromosome and the problem being solved. The result of this operator is the set of N_i individuals, I , containing N_c cells each, which will become the algorithm's initial population.

Algorithm 11.2 Population Initialization.

```

function POPULATIONINITIALIZATION( $N_i, N_c$ )
  for all  $i \in N_i$  do                                     ▷  $N_i$  individuals
    for all  $c \in N_c$  do                                     ▷  $N_c$  cells
       $C(c) \leftarrow \text{Cell}()$                                ▷ Constructs a new cell
       $\text{Evaluate}(C(c))$ 
    end for
     $I(i) \leftarrow \text{Individual}(C)$                          ▷ set of individuals
  end for
  return  $I$                                                  ▷ set of  $N_i$  individuals, each one containing  $N_c$  cells
end function

```

11.5.2 Selection

If the termination condition is not satisfied, the algorithm enters a new iteration whose first step is the *Selection* operator. In this study we have used the well-known binary tournament [87] as the selection operator although a different selection strategy could be used here. As a result, the auxiliary population P' is obtained and the epigenetic cycle of epiGA continues.

11.5.3 Nucleosome Generation (NG)

The next operation is the *Nucleosome Generation* (NG) which generates a new nucleosome vector as a mask for each cell in the individuals of the population as shown in Algorithm 11.3.

Each position of the nucleosome vector n in each cell is likely to be a center of a nucleosome in the vector. Depending on the nucleosome probability P_n new nucleosomes will be generated, centered in the chosen position j , with a radius R , representing a collapsed DNA region. At the end of this operation all the cells in the population have a new nucleosome mask in n .

Algorithm 11.3 Nucleosome Generation (NG).

```

function NUCLEOSOMEGENERATION( $P, P_n, R$ )
  for all  $i \in P$  do                                     ▷ Each individual  $i$ 
    for all  $c \in i$  do                                       ▷ Each cell  $c$ 
       $n \leftarrow \text{getNucleosome}(c)$ 
      for all  $j \in n$  do
        if  $\text{rnd}() < P_n$  then                               ▷ Chromosome is probabilistically collapsed
          Collapse( $n, R$ )                                     ▷  $n[j-R] - n[j+R] = 1$ 
        end if
      end for
    end for
  end for
  return  $P$ 
end function

```

11.5.4 Nucleosome Based Reproduction (NBR)

The *Nucleosome Based Reproduction* (NBR) operator (Algorithm 11.4 and Figure 11.6) uses the nucleosome mask generated by the previous operator in the epigenetic cycle to guide the recombination of the solutions stored in each cell.

First, an empty population P' is initialized. Second, two individuals (i_1 and i_2) are taken from the current working population P . Third, the best cells (c_1 and c_2) from i_1 and i_2 are extracted as well as their chromosomes x_1 and x_2 and nucleosomes n_1 and n_2 . Next, a new nucleosome mask N is calculated by carrying out the boolean operation OR between the n_1 and n_2 . After that, the contents of the chromosomes are swapped only where the DNA is not

Algorithm 11.4 Nucleosome Based Reproduction (NBR).

```

function NUCLEOSOMEBASEDREPRODUCTION( $P$ )
   $P' \leftarrow \emptyset$ 
  for all  $\{i_1, i_2\} \in P$  do                                ▷ Each two individuals
     $c_1 \leftarrow \text{getBestCell}(i_1)$ 
    Let  $c_1, c_2$  be the best cells from  $i_1$  and  $i_2$             ▷ According to fitness value
    Let  $x_1, x_2$  be the solution vectors from  $c_1$  and  $c_2$ 
    Let  $n_1, n_2$  be the nucleosome vectors from  $c_1$  and  $c_2$ 
     $N \leftarrow n_1$  OR  $n_2$                                 ▷ New nucleosome mask
    while  $j < \text{size}(N)$  do
      if  $N(j)$  then                                       ▷ Collapsed DNA zones ( $N(j) = 1$ ) do not change
         $x_1'(j) \leftarrow x_1(j)$ 
         $x_2'(j) \leftarrow x_2(j)$ 
      else                                                 ▷ Non-collapsed DNA zones do change
         $x_1'(j) \leftarrow x_2(j)$ 
         $x_2'(j) \leftarrow x_1(j)$ 
      end if
       $j \leftarrow j + 1$ 
    end while
     $c_1' \leftarrow \text{Cell}(x_1', x_1, x_2, N)$                 ▷ New cells with new parents and nucleosomes
     $c_2' \leftarrow \text{Cell}(x_2', x_1, x_2, N)$ 
     $i_1' \leftarrow \text{replaceWorst}(i_1, c_1')$                 ▷ New individuals based on the former ones
     $i_2' \leftarrow \text{replaceWorst}(i_2, c_2')$ 
     $P' \leftarrow P' \cup \{i_1', i_2'\}$ 
  end for
  return  $P'$ 
end function

```

collapsed, i.e. where the nucleosome mask value is zero, and stored in two new chromosomes x_1' and x_2' . At the end of the loop, two new cells c_1' and c_2' are created by using the new chromosomes x_1' and x_2' , the former ones x_1 and x_2 and the new nucleosome mask N . Note that the former chromosomes are needed to store the father f and mother m in the new cell (Figure 11.6), as they could be needed when applying Epigenetic Mechanisms later. Finally, two new individuals i_1' and i_2' are created by replacing the worst cell in them by the new cells c_1' and c_2' , respectively. Then i_1' and i_2' are added to the population P' . This operation is especially useful to preserve the diversity of the cells in the individual and thus in the population. At the very end of the code, the resulting population P' is returned to be used by the next operator in the epigenetic cycle of the epiGA.

11.5.5 Epigenetic Mechanisms (EM)

This operator applies to each cell of the population P , the epigenetic mechanism specified in the parameter m with a probability P_e and under the effects of the epigenetic environment

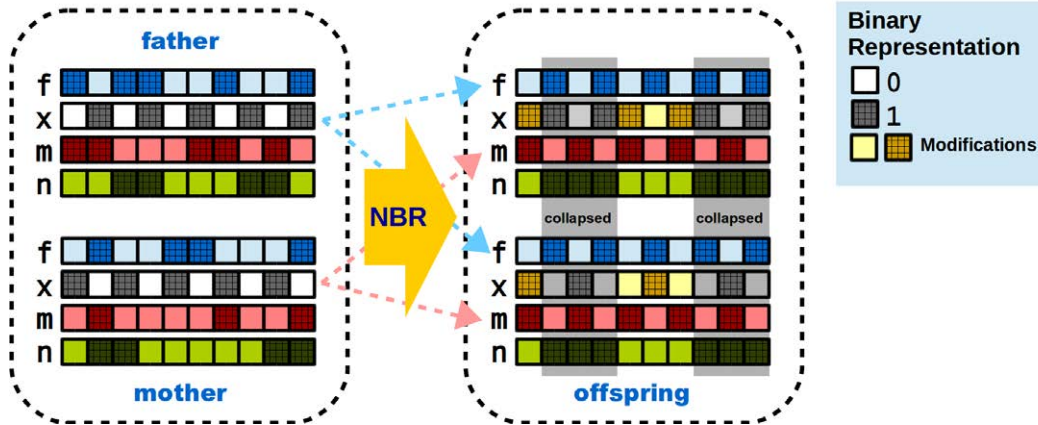


Figure 11.6: Nucleosome Based Reproduction (NBR). Both vectors x change only where the DNA is not collapsed (positions of vector n with zeros). New vectors f and m are taken from the parents of the new cell. Offspring's vectors n are calculated by applying the boolean OR to the parents' vector n .

rules contained in *Environment*, as shown in Algorithm 11.5. After that, the new content of the cell, i.e. the x chromosome, is evaluated. Although we discussed seven different epigenetic mechanisms in Section 11.4 and showed them in Figure 11.3, in this first approach we are only studying the Gene Silencing mechanism. Not only does it frequently appear in biology, it is also easy to implement as follows.

Algorithm 11.5 Epigenetic Mechanisms (EM).

```

function EPIGENETICMECHANISMS( $P, P_e, m, Environment$ )
  for all  $i \in P$  do                                     ▷ Each individual  $i$ 
    for all  $c \in i$  do                                     ▷ Each cell  $c$ 
      ApplyMechanisms( $m, c, P_e, Environment$ )
      Evaluate( $c$ )
    end for
  end for
  return  $P$ 
end function

```

Gene Silencing (GeS)

The pseudocode of *Gene Silencing* is presented in Algorithm 11.6. It receives cell c , epigenetic probability P_e , and environment E , as parameters. As we have mentioned, only collapsed DNA is likely to be changed by methylation. In GeS, the probability of methylation is provided by the environment (a vector of probabilities) for each gene (position of vector x).

First, the chromosome x and the nucleosome mask n is obtained from the cell c . Second, each position of the chromosome x is selected and, if the corresponding position in the nucleosome mask n indicates that the DNA is collapsed, the value of $x(j)$ may change according to the probability P_e and the environment E . Here we are supposing a binary

Algorithm 11.6 Gene Silencing (GeS).

procedure GENESILENCING(c, P_e, E)

 $x \leftarrow \text{getSolution}(c)$
 $n \leftarrow \text{getNucleosome}(c)$
while $j < \text{size}(x)$ **do**

 if $n(j)$ **then**

 if $\text{rnd}() < P_e$ **then**

 $x(j) \leftarrow \text{rnd}() < E(j)$

 end if

 end if

 $j \leftarrow j + 1$
end while
end procedure

▷ Only Collapsed DNA methylates

▷ Epigenetic probability

▷ Influence of the environment

problem, as a result, each position of x is a binary value, and E is a vector of probabilities that modifies the likelihood that each value x has of being “1”.

By using the nucleosome mask (n), we control which parts of the solution are likely to change, i.e. the ones that were not modified by the reproduction. Moreover, the epigenetic environment allows us to define a different probability distribution for each position of the solution vector. It could be just a floating point number if we have a binary vector as the solution, or a more complex expression for a solution comprising integers, for example. Note that we can prescind from the environment’s influence if we set 0.5 as the probability value for every position of the solution, making them all equiprobable.

In Figure 11.7 we present an example of *Gene Silencing* applied to chromosome x , according to the nucleosome mask n . It can be seen that the positions 1, 2, 3, 7, 8, and 9, correspond to a collapsed DNA (vector n). After using the random number generator and comparing the result with P_e , only positions 2, 7, and 9, are candidates for changing their value. Finally, the influence of the epigenetic environment changes positions 2 and 7 to “1”,

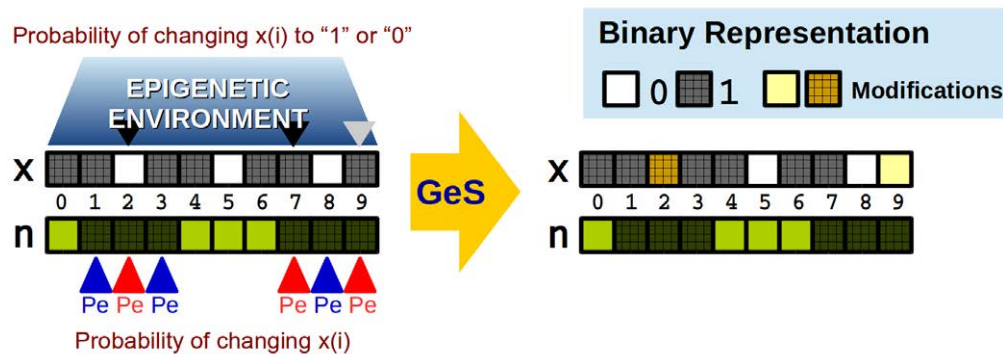


Figure 11.7: Gene Silencing (GeS). Only collapsed DNA is likely to be methylated. The methylation probability is given by the environment.

and position 9 to “0”. Note that this is not just a bit flip operator as the value of position 7 has not been changed this time despite having been initially selected.

11.5.6 Replacement

The last operator in the epigenetic cycle is the *Replacement* operator. Here, we have used an elitist replacement [86] although another replacement operator could be used instead. In doing so, we have selected the new working population in an elitist way, copying the best individuals to it. Note that the concept of best individuals depends on their fitness values and the problem that is being solved, e.g. if we are minimizing, the lower, the better.

11.6 Discussion

Through this chapter we have introduced epigenetics and its mechanisms with the aim of using it to build new computational algorithms. Our goal was to understand a biological domain that could represent an interesting source of inspiration on building new algorithms. These new algorithms will not only have different operators with respect to the standard ones, but a different representation that takes ideas from nature itself. They are used to evolve more complex structures while expressing different relationships between genes beyond simple Mendelian ideas. The additional interest in learning from the environment and acting on inherited chromosomes is also a particular way of thinking that could be exploited later by other researchers.

We strongly believe that the epiGenetic Algorithm presented in this PhD thesis could be a powerful tool for solving combinatorial optimization problems as it can be tuned to different types of problems by using specially made representations, environments, and choosing the right set of epigenetic mechanisms to explore the solution space.



UNIVERSIDAD
DE MÁLAGA

Chapter 12

Solving Problems with epiGA

In this chapter our epiGenetic Algorithm (epiGA) is used to solve the Multidimensional Knapsack Problem (MKP) which has been selected to test our proposal. Additionally, four strategies are proposed as competitors. Finally, the parameterization and evaluation of epiGA is done plus a convergence analysis to better know the behavior of this new algorithm.

12.1 Multidimensional Knapsack Problem (MKP)

The Multidimensional Knapsack Problem (MKP) is a well-known NP-Hard combinatorial problem [76] that has been studied for decades since it first appeared in [93, 138].

We have chosen this problem because it is a highly complex binary problem and because there are several studies available to be compared to our results. Different methods have been used to solve the MKP [175], many of them are based on kernel search [8], multi-level search strategy [30], Particle Swarm Optimization (PSO) [39], Genetic Algorithms (GA) [42], branch and bound techniques [71], and greedy techniques [152]. Our goal here is to add experimentation to our proposed algorithm based on epigenesis, test how it performs against the state of the art, and validate the use of the epigenetic operators as a viable way of solving hard combinatorial problems.

The MKP consists of n items and m different knapsacks of capacity c_i , $i \in \{1, \dots, m\}$. Each item j , $j \in \{1, \dots, n\}$, has an associated profit p_j and consumes a quantity w_{ij} from the knapsack i , if the item has been selected through the variable x_j by setting it to 1, otherwise 0. The objective is to maximize the profit of the items in the m knapsacks (Equation 12.1) without exceeding the maximum capacity of each one (c_i), according to the constraints described in equations 12.2 and 12.3.

$$\text{Maximize } z = \sum_{j=1}^n p_j x_j \quad (12.1)$$

$$\text{Subject to } \sum_{j=1}^n w_{ij} x_j \leq c_i, \quad i = 1, 2, \dots, m \quad (12.2)$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \dots, n \quad (12.3)$$

To solve the MKP we have defined the fitness function as presented in Equation 12.4.

$$F(\vec{x}) = \begin{cases} \sum_{j=1}^n p_j x_j & \text{if } \sum_{j=1}^n w_{ij} x_j \leq c_i, \forall i \\ -1.0 & \text{otherwise} \end{cases} \quad (12.4)$$

There, the fitness value is the sum of the profits corresponding to the objects included in all the knapsacks ($x_j = 1$) providing they do not exceed the maximum capacity of each one (c_i). If this happens, the result is a negative value, e.g. -1, so that the *Evaluate* function can repair the solution as described in Algorithm 12.1.

First, solution x is obtained from cell c . Second, the fitness value is calculated according to Equation 12.4. Third, if the fitness value is less than 0 a random position of x where its value is 1 is changed to 0 (an item is removed from the knapsacks). After that, the fitness value is calculated again and the internal loop is repeated until the fitness value is greater or equal to 0. Finally, the solution in cell c is replaced by the new one and the fitness value is returned. Note that we have preferred to use a simple repairing technique instead of a greedy one [42] or just a penalization term [52].

We have adapted the environment to the MKP so that it represents the probability of including an item in the knapsacks. Instead of being equiprobable, all the available items have a probability bias which depends on the relation between its profit and weight in all the knapsacks (Equation 12.5).

Algorithm 12.1 Evaluate.

```

function EVALUATE( $c$ )
   $x \leftarrow \text{getSolution}(c)$ 
   $f \leftarrow \text{Fitness}(x)$  ▷ Evaluates solution  $x$ 
  while  $f < 0$  do ▷ Is invalid?
     $i \leftarrow \text{findRndOne}(x)$ 
     $x(i) \leftarrow 0$  ▷ Removes an item randomly
     $f \leftarrow \text{Fitness}(x)$  ▷ And reevaluates  $x$ 
  end while
   $\text{setSolution}(c, x)$  ▷ Updates the solution in the cell  $c$ 
  return  $f$ 
end function

```

$$E(j) = \frac{p_j}{\sum_{i=1}^m w_{ij}} \quad (12.5)$$

We calculate the environment (E) for all the items at the very beginning of the algorithm, when the characteristics of the available items are known. The values are normalized so that the probabilities $E(j)$ are between $\frac{1}{3}$ for the item j with worst relation value (Equation 12.5), and $\frac{2}{3}$ for the best one. Although we have tested with other limits, $\frac{1}{3}$ has worked the best.

We have chosen the OR-Library [19] to evaluate epiGA. This is a well-known set of instances of the MKP consisting of problems with $n = 100, 250$, and 500 variables and $m = 5, 10$, and 30 constraints. There are 30 instances of each combination of n and m . We have named them according to the pattern: $n.m_i$, so that the instance 250.5_1 corresponds to the first instance of MKP problem with 250 variables and 5 constraints. In this approach we have addressed the optimization of 30 instances of the following problems: 100.5, 500.5, 100.10, and 250.10, i.e. 120 instances in total. In the next section we describe the selected competitors for epiGA.

12.2 Competitors

Even if our goal is to create a new methodology and contribute new ideas to the domain, we are aware of the advantages of showing from the very beginning that the resulting algorithms could actually work versus published competitors in the literature.

We have chosen several competitors for epiGA, some of which are taken from the state of the art, providing their results are available and can be compared to ours. Specifically, the SACRO-PSO algorithms and Resolution Search + Branch & Bound are included in our comparison. In addition, we have included an exact optimizer (CPLEX), and two well-known metaheuristics for solving combinatorial problems, Simulated Annealing (SA) and a Genetic Algorithm (GA). These five algorithms help us to prove performance for new ideas, even if this PhD thesis is just devoted to introducing them for future exploitation.

12.2.1 IBM ILOG CPLEX

IBM ILOG CPLEX [108] is a commercial software developed by ILOG and currently owned by IBM, based on linear programming and the simplex method [155]. We have formalized each working instance as a linear programming problem to be solved by CPLEX as a maximization task (Equation 12.6) subject to the restrictions defined in each instance (Equation 12.7).

$$\text{Maximize } obj = p_0 \cdot x_1 + p_1 \cdot x_2 + \dots + p_n \cdot x_n \quad (12.6)$$

$$\text{Subject to } c_1 = w_{1,1} \cdot x_1 + w_{1,2} \cdot x_2 + \dots + w_{1,n} \cdot x_n \quad (12.7)$$

$$c_2 = w_{2,1} \cdot x_1 + w_{2,2} \cdot x_2 + \dots + w_{2,n} \cdot x_n$$

$$\dots$$

$$c_m = w_{m,1} \cdot x_1 + w_{m,2} \cdot x_2 + \dots + w_{m,n} \cdot x_n$$

To make a fair comparison we have restricted the execution of CPLEX (version 12.6.2.0) to one CPU and thread. Furthermore, we have set the relative and absolute MIP gap tolerance (*mipgap* and *absmipgap*) to 0 in order to improve the precision when finding the maximum.

12.2.2 SACRO-PSO Algorithms

In [39] the author presents a novel Self-Adaptive Check and Repair Operator (SACRO) combined with particle swarm optimization (PSO) to solve the MKP. SACRO is based on the check and repair operator (CRO) explained in [42]. However, in SACRO the profit/weight utility and profit density are used as alternative pseudo-utility ratios.

In the two resulting algorithms, based on the existing BPSO-TVAC and CBPSO-TVAC [40], the values for all particles are randomly generated and evaluated to obtain the corresponding fitness value. If the constraints of the knapsack are not satisfied, the infeasible solutions are converted into feasible ones, using SACRO. The repair of the infeasible solutions is based on alternative pseudo-utility ratios which varies the approach directions allowing the particles to visit different feasible regions of the search space.

In our study we have used the results of SACRO-BPSO-TVAC and SACRO-CBPSO-TVAC published in [39] corresponding to 100 runs and 20.000 iterations.

12.2.3 Resolution Search + Branch & Bound (RS + B&B)

An exact method based on a multi-level search strategy for solving the MKP is proposed in [30]. First, the top level branches are enumerated by using Resolution Search Strategy, in which the authors proposed an improvement of the waning phase in the resolution search principle [43].

Second, the middle level branches are solved by using Branch & Bound [231]. In this stage, the algorithm first searches in the most promising parts of the search tree. In the implementation of the algorithm proposed in [231], done by the same authors, the specific reduced cost propagation was removed to save time.

Third, the lower level branches are enumerated according to a simple Depth First Search enumeration brute force. The branching strategy in this phase consists in fixing the first free variable to 0 and then to 1.

12.2.4 Genetic Algorithm (GA)

In addition to these competitors, we want to compare our results to a well-known implementation of a Genetic Algorithm (GA) [86, 102], since our epiGA could easily be seen as the next step in the evolution of algorithms like GA, when epigenetics are used. This is a metaheuristic inspired by nature, specifically by natural selection and genetics, which has been described in Section 3.2.1.

We have designed an elitist, generational GA [15], and used it to solve the selected instances of the MKP after performing a first phase of parameterization and population size studies, in order to achieve a fair comparison afterwards. The pseudocode of the GA was already presented in Algorithm 3.1. We have used Binary Tournament [87] as selection operator, Uniform Crossover [87] as recombination operator, Bit Flip Mutation, and, the new population $P(t+1)$ has been obtained in an elitist way, as in epiGA.

12.2.5 Simulated Annealing (SA)

Finally, one last competitor is presented in this section. It consists in an implementation of Simulated Annealing (SA) [35, 118] a well-known metaheuristic applicable to a wide range of problems which has been described in Section 3.2.2.

We have used five parameters in our implementation (Algorithm 12.2): N is the number of iterations in the same temperature (internal loop), T_0 is the initial temperature, T_{min} is the minimum (final) temperature, and P_{min} and P_{max} are two extra parameters set up for initially exploring the search space and lately exploiting the best solutions found.

Algorithm 12.2 Simulated Annealing (SA).

```

procedure SA( $N, T_0, T_{min}, P_{min}, P_{max}$ )
   $X \leftarrow \text{GenerateInitialState}()$ 
  repeat
    for  $i = 0$  to  $N$  do
       $Y \leftarrow \text{Generate}(X, T_k)$                                 ▷ Generates a new solution
      if  $\text{Accept}(X, Y, T_k)$  then                                ▷ New solution acceptance
         $X \leftarrow Y$ 
      end if
    end for
     $T_{k+1} \leftarrow \text{Update}(T_k)$                                 ▷ Temperature decrement
     $k \leftarrow k + 1$ 
  until Termination Condition
end procedure

```

First, the SA algorithm generates an initial solution X as done in GA and epiGA. Second, the main loop begins until the *Termination Condition* is met. Third, the internal loop begins performing N iterations. Each iteration generates a new solution Y (Algorithm 12.3) which is accepted depending on the current temperature T_k (Equation 12.10). After that, the current

temperature T_k is updated following Equation 12.8 as used in the Fast Simulated Annealing (FSA) [218]. Finally, if the *Termination Condition* is not fulfilled a new iteration begins.

$$T(k) = \frac{T_0}{1+k} \quad (12.8)$$

Algorithm 12.3 presents the pseudocode of the function used to generate a new solution. There, the positions (bits, as we are working with binary representations) of the current solution X are visited and their values are flipped depending on probability p . In order to reduce the number of changes made to the solution following the reduction of the temperature, the value of p decreases following a line defined by slope m and intercept value h .

Algorithm 12.3 Generate Function.

```

procedure GENERATE( $X, T_k$ )
   $p \leftarrow m * T_k + h$  ▷ Calculates probability  $p$ 
  while  $j < \text{size}(X)$  do
    if  $\text{rnd}() < p$  then
       $X(j) \leftarrow \text{not } X(j)$  ▷ Changes  $j$ -th value
    end if
  end while
end procedure

```

Equation 12.9 shows the calculation of both parameters based on the initial temperature T_0 , the minimum one T_{min} and the parameters P_{max} and P_{min} which were obtained during the parameterization of the SA we have done (Section 12.3). By including these parameters, we are tuning the SA to efficiently solve the MKP in order to be a better competitor of epiGA.

$$m = \frac{P_{max} - P_{min}}{T_0 - T_{min}}; \quad h = P_{min} - m \quad (12.9)$$

Finally, the acceptance of a new generated solution Y is defined in Equation 12.10 and is known as the Metropolis probability [118]. There, the energy function c is the fitness of the solutions X and Y , which is calculated as in epiGA. Furthermore, we have included the process that fixes the invalid solutions in the fitness calculation to keep the comparison fair.

$$\text{Accept}(X, Y, T_k) = \min\{1, e^{-\frac{c(Y) - c(X)}{T_k}}\} \quad (12.10)$$

12.3 Parameterization

In this section we address not only the parameterization of the epiGA, but also the GA and SA in order to improve their performance when solving the MKP and foster a fair result comparison later. All the runs performed in this section were limited to max 1,000,000 evaluations and executed by the same hardware.

12.3.1 epiGA

The epiGA has three main parameters: the epigenetic probability P_e , the nucleosome probability P_n , and the nucleosome radius R . Additionally, we have included in the parameterization two different population sizes.

First, we have performed 30 independent runs of epiGA with a different combination of the parameters in the same instance of MKP (one of the hardest available so that the maximum should not be found during the optimization time). We have tested $P_e, P_n \in \{0.01, 0.02, \dots, 0.10\}$ and $R \in \{1, 2, \dots, 10\}$ which accounts for 1,000 combinations, i.e. 30,000 runs. On the left of Table 12.1 we present the 30 best results achieved (according to the Friedman Rank), the standard deviation, the Friedman Rank itself, and the Wilcoxon p -value for the selected combinations of the parameters. There are two solutions which are

Table 12.1: Parameterization of the epiGA. The left of the table shows the parameter tuning (30 best ranked combinations) while the second experiment, (population sizes of 200 and 400), is shown on the right.

| P_e | P_n | R | Fitness | | Friedman Rank | Wilcoxon p -value | Instance | Average Fitness | | Wilcoxon p -value |
|-------|-------|-----|-----------------|--------|----------------|---------------------|----------|-----------------|-----------------|---------------------|
| | | | Average | StdDev | | | | 200 | 400 | |
| 0.01 | 0.03 | 3 | 114966.2 | 123.8 | 996.75 | 0.028 | 1 | 115337.6 | 115577.9 | 0.000 |
| 0.01 | 0.03 | 4 | 114974.9 | 188.6 | 986.35 | 0.084 | 2 | 114170.8 | 114461.3 | 0.000 |
| 0.01 | 0.03 | 5 | 114999.2 | 131.1 | 1013.95 | 0.162 | 3 | 116055.2 | 116280.3 | 0.000 |
| 0.01 | 0.03 | 6 | 114977.3 | 178.9 | 989.72 | 0.086 | 4 | 114701.4 | 114961.7 | 0.000 |
| 0.01 | 0.03 | 7 | 114983.4 | 155.5 | 997.73 | 0.202 | 5 | 115862.3 | 116061.3 | 0.000 |
| 0.01 | 0.04 | 4 | 115032.6 | 149.8 | 1024.25 | 0.604 | 6 | 115049.3 | 115348.0 | 0.000 |
| 0.01 | 0.04 | 5 | 114985.5 | 159.8 | 1001.35 | 0.150 | 7 | 113463.1 | 113678.5 | 0.000 |
| 0.01 | 0.04 | 6 | 114981.0 | 135.1 | 997.68 | 0.047 | 8 | 113599.1 | 113873.7 | 0.000 |
| 0.01 | 0.05 | 2 | 114996.2 | 158.8 | 1009.07 | 0.459 | 9 | 114761.4 | 114971.8 | 0.000 |
| 0.01 | 0.05 | 3 | 115006.2 | 177.5 | 1006.45 | 0.355 | 10 | 116383.8 | 116616.1 | 0.000 |
| 0.01 | 0.05 | 4 | 114985.8 | 173.7 | 993.80 | 0.241 | 11 | 217648.7 | 217839.6 | 0.000 |
| 0.01 | 0.06 | 2 | 115020.2 | 134.7 | 1019.98 | 0.399 | 12 | 214049.5 | 214272.5 | 0.000 |
| 0.01 | 0.06 | 3 | 115054.3 | 177.9 | 1025.62 | 0.951 | 13 | 215404.8 | 215617.2 | 0.000 |
| 0.01 | 0.07 | 2 | 114971.8 | 141.0 | 995.82 | 0.026 | 14 | 217361.0 | 217593.3 | 0.000 |
| 0.01 | 0.07 | 3 | 114983.1 | 172.5 | 993.67 | 0.100 | 15 | 215164.7 | 215382.2 | 0.000 |
| 0.01 | 0.08 | 1 | 114962.1 | 149.8 | 988.78 | 0.053 | 16 | 215319.6 | 215489.4 | 0.000 |
| 0.01 | 0.08 | 2 | 114962.7 | 185.1 | 985.00 | 0.074 | 17 | 215459.2 | 215676.3 | 0.000 |
| 0.01 | 0.09 | 1 | 114997.8 | 169.2 | 1004.78 | 0.399 | 18 | 215950.6 | 216179.8 | 0.000 |
| 0.01 | 0.10 | 1 | 115051.6 | 154.6 | 1028.90 | — | 19 | 216871.6 | 217073.0 | 0.000 |
| 0.01 | 0.10 | 2 | 115023.2 | 141.2 | 1020.02 | 0.497 | 20 | 214209.3 | 214448.2 | 0.000 |
| 0.02 | 0.02 | 5 | 114976.3 | 180.2 | 992.13 | 0.206 | 21 | 301207.0 | 301384.7 | 0.000 |
| 0.02 | 0.04 | 1 | 114966.7 | 169.3 | 986.25 | 0.074 | 22 | 299644.1 | 299811.3 | 0.000 |
| 0.02 | 0.04 | 2 | 115008.3 | 211.7 | 1001.72 | 0.376 | 23 | 304751.7 | 304876.9 | 0.000 |
| 0.02 | 0.04 | 3 | 114960.1 | 136.0 | 989.23 | 0.048 | 24 | 301554.4 | 301728.8 | 0.000 |
| 0.02 | 0.05 | 3 | 114993.5 | 170.3 | 998.97 | 0.171 | 25 | 304021.4 | 304219.3 | 0.000 |
| 0.02 | 0.06 | 2 | 114998.1 | 119.3 | 1013.88 | 0.109 | 26 | 296549.0 | 296755.2 | 0.000 |
| 0.02 | 0.07 | 2 | 114975.0 | 184.7 | 988.93 | 0.079 | 27 | 302981.1 | 303122.5 | 0.000 |
| 0.02 | 0.08 | 1 | 114964.3 | 170.3 | 984.52 | 0.018 | 28 | 306585.8 | 306771.8 | 0.000 |
| 0.02 | 0.09 | 1 | 115012.1 | 138.0 | 1016.67 | 0.258 | 29 | 302782.0 | 302947.3 | 0.000 |
| 0.02 | 0.10 | 1 | 114994.9 | 193.4 | 997.33 | 0.202 | 30 | 300114.7 | 300302.7 | 0.000 |

statistically equivalent: $P_e = 0.01$, $P_n = 0.10$, $R = 1$ and $P_e = 0.01$, $P_n = 0.06$, and $R = 3$. If we compare the existing Wilcoxon p -value between these two combinations (0.951) we can see that it is high enough to allow us to use any of them. Figures 12.1a, 12.1b, and 12.1c show the parameterization performed and how the fitness evolves for the different combinations of values when we fix the selected ones. It can be seen that the lower the P_e and R , the better, while P_n tends to depend on the value of R for this problem.

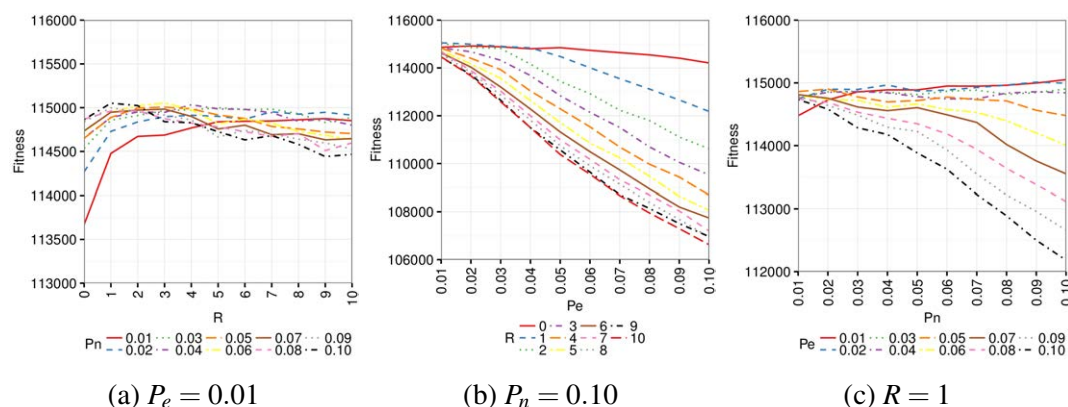


Figure 12.1: Fitness variation of epiGA when we set one of the parameters to the best value according to the parameterization done.

Second, we performed 30 independent runs of epiGA optimizing 30 different instances, using two different population sizes: 200 and 400, which amounts to 1800 runs. We have used as parameters the best combination obtained in the previous experiment ($P_e = 0.01$, $P_n = 0.10$, and $R = 1$). On the right of Table 12.1 we present the results obtained from the parameterization of the population size performed. We can see that epiGA always obtains a better average fitness when using a population (μ) of 400 individuals to solve the MKP. Additionally, we provide the Wilcoxon p -value of each comparison which shows that they are statistically significant.

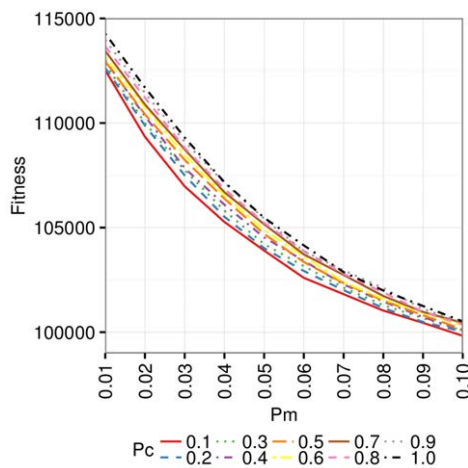
12.3.2 GA and SA

The left of Table 12.2 shows the parameterization of GA for different combinations of its two parameters, the crossover probability P_c , and the mutation probability P_m . We have performed 30 independent runs for values of $P_c \in \{0.1, 0.2, \dots, 1.0\}$ and $P_m \in \{0.01, 0.02, \dots, 0.10\}$ which accounts for 100 combinations, i.e. 3,000 runs. We can see that the combination of $P_c = 1.0\}$ and $P_m = 0.01$ is clearly the best for solving the MKP with a confidence interval greater than 99% according to the Wilcoxon p -value calculated.

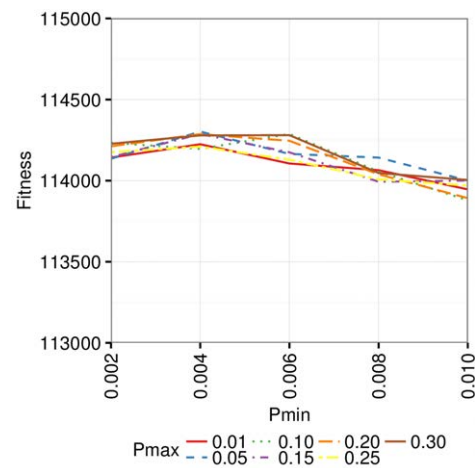
Furthermore, we have tested two population sizes as in epiGA by performing 30 independent runs on the 30 different instances as in epiGA (900 runs). The center columns of Table 12.2 show that the average fitness values are always better when using a population of 400 individuals (as in epiGA) and that the results are statistically significant for all the instances (above 97%). The parameterization of GA (Figure 12.2a) shows that the fitness values improve as the mutation probability decreases and the crossover probability increases.

Table 12.2: Parameterization of the GA and SA (30 best ranked combinations). We tested different values for the parameters for GA (P_m and P_c) and two population sizes ($\mu = 200$ and $\mu = 400$). Additionally, we tested two parameters for SA (P_{min} and P_{max}). We provide the Friedman Rank and the Wilcoxon p -value as well.

| GA | | | | | | | | | SA | | | | |
|-------|-------|----------------|---------------|---------------------|------|-----------------|-----------------|---------------------|-----------|-----------|----------------|---------------|---------------------|
| P_c | P_m | Fitness (Avg.) | Friedman Rank | Wilcoxon p -value | Ins. | Average Fitness | | Wilcoxon p -value | P_{min} | P_{max} | Fitness (Avg.) | Friedman Rank | Wilcoxon p -value |
| | | | | | | 200 | 400 | | | | | | |
| 0.10 | 0.01 | 112528.6 | 92.10 | 0.000 | 1 | 115294.8 | 115416.2 | 0.001 | 0.002 | 0.01 | 114142.3 | 19.12 | 0.054 |
| 0.10 | 0.02 | 109344.5 | 80.13 | 0.000 | 2 | 114157.9 | 114297.0 | 0.000 | 0.002 | 0.05 | 114135.3 | 18.37 | 0.044 |
| 0.20 | 0.01 | 112638.2 | 92.92 | 0.000 | 3 | 115971.2 | 116174.0 | 0.000 | 0.002 | 0.10 | 114233.3 | 20.43 | 0.365 |
| 0.20 | 0.02 | 109880.0 | 82.27 | 0.000 | 4 | 114621.9 | 114795.4 | 0.000 | 0.002 | 0.15 | 114144.7 | 18.83 | 0.049 |
| 0.20 | 0.03 | 107497.9 | 71.93 | 0.000 | 5 | 115801.9 | 115956.8 | 0.000 | 0.002 | 0.20 | 114211.8 | 20.33 | 0.225 |
| 0.30 | 0.01 | 112826.4 | 93.57 | 0.000 | 6 | 114968.7 | 115165.2 | 0.000 | 0.002 | 0.25 | 114175.8 | 19.13 | 0.064 |
| 0.30 | 0.02 | 110002.7 | 83.23 | 0.000 | 7 | 113429.9 | 113543.0 | 0.001 | 0.002 | 0.30 | 114227.5 | 20.70 | 0.510 |
| 0.30 | 0.03 | 107749.3 | 72.90 | 0.000 | 8 | 113546.7 | 113712.4 | 0.000 | 0.004 | 0.01 | 114224.5 | 19.83 | 0.382 |
| 0.40 | 0.01 | 112990.6 | 94.27 | 0.000 | 9 | 114625.8 | 114813.7 | 0.000 | 0.004 | 0.05 | 114302.5 | 23.40 | — |
| 0.40 | 0.02 | 110328.6 | 84.40 | 0.000 | 10 | 116323.1 | 116493.3 | 0.000 | 0.004 | 0.10 | 114195.8 | 19.33 | 0.285 |
| 0.40 | 0.03 | 107845.1 | 73.37 | 0.000 | 11 | 217599.7 | 217690.0 | 0.001 | 0.004 | 0.15 | 114287.9 | 21.87 | 0.902 |
| 0.50 | 0.01 | 112950.9 | 94.00 | 0.000 | 12 | 213993.6 | 214164.6 | 0.000 | 0.004 | 0.20 | 114286.8 | 22.72 | 0.537 |
| 0.50 | 0.02 | 110414.8 | 84.47 | 0.000 | 13 | 215408.3 | 215526.0 | 0.000 | 0.004 | 0.25 | 114209.7 | 19.50 | 0.202 |
| 0.50 | 0.03 | 108239.4 | 75.00 | 0.000 | 14 | 217298.4 | 217488.6 | 0.000 | 0.004 | 0.30 | 114279.0 | 23.03 | 0.781 |
| 0.60 | 0.01 | 113173.8 | 95.12 | 0.000 | 15 | 215107.7 | 215289.1 | 0.000 | 0.006 | 0.01 | 114105.9 | 16.80 | 0.014 |
| 0.60 | 0.02 | 110753.0 | 85.97 | 0.000 | 16 | 215241.5 | 215359.6 | 0.000 | 0.006 | 0.05 | 114164.9 | 18.63 | 0.088 |
| 0.60 | 0.03 | 108481.9 | 76.53 | 0.000 | 17 | 215481.5 | 215551.3 | 0.013 | 0.006 | 0.10 | 114287.8 | 22.33 | 0.491 |
| 0.70 | 0.01 | 113448.7 | 96.80 | 0.000 | 18 | 215928.1 | 216048.2 | 0.000 | 0.006 | 0.15 | 114174.7 | 19.77 | 0.102 |
| 0.70 | 0.02 | 110886.3 | 86.67 | 0.000 | 19 | 216825.4 | 216950.4 | 0.000 | 0.006 | 0.20 | 114245.7 | 21.08 | 0.572 |
| 0.70 | 0.03 | 108739.9 | 77.50 | 0.000 | 20 | 214177.3 | 214289.6 | 0.001 | 0.006 | 0.25 | 114129.7 | 18.23 | 0.086 |
| 0.80 | 0.01 | 113668.2 | 97.60 | 0.000 | 21 | 301204.8 | 301317.8 | 0.000 | 0.006 | 0.30 | 114280.5 | 20.97 | 0.789 |
| 0.80 | 0.02 | 111209.3 | 87.70 | 0.000 | 22 | 299656.5 | 299738.7 | 0.000 | 0.008 | 0.01 | 114064.6 | 15.50 | 0.006 |
| 0.80 | 0.03 | 108817.1 | 77.80 | 0.000 | 23 | 304702.9 | 304774.6 | 0.002 | 0.008 | 0.05 | 114142.1 | 18.75 | 0.040 |
| 0.90 | 0.01 | 113974.0 | 98.70 | 0.000 | 24 | 301527.0 | 301632.3 | 0.000 | 0.008 | 0.10 | 114056.2 | 15.00 | 0.003 |
| 0.90 | 0.02 | 111435.8 | 88.73 | 0.000 | 25 | 303996.0 | 304099.6 | 0.000 | 0.008 | 0.15 | 113993.2 | 15.32 | 0.013 |
| 0.90 | 0.03 | 109108.8 | 79.10 | 0.000 | 26 | 296499.3 | 296625.5 | 0.000 | 0.008 | 0.20 | 114039.1 | 14.98 | 0.001 |
| 1.00 | 0.01 | 114278.7 | 99.77 | — | 27 | 302960.5 | 303007.4 | 0.024 | 0.008 | 0.25 | 114008.0 | 15.27 | 0.000 |
| 1.00 | 0.02 | 111697.3 | 89.60 | 0.000 | 28 | 306571.6 | 306673.2 | 0.000 | 0.008 | 0.30 | 114047.1 | 16.73 | 0.015 |
| 1.00 | 0.03 | 109307.8 | 80.13 | 0.000 | 29 | 302778.7 | 302860.5 | 0.001 | 0.010 | 0.15 | 114000.6 | 14.90 | 0.001 |
| 1.00 | 0.04 | 107172.6 | 69.87 | 0.000 | 30 | 300122.5 | 300206.0 | 0.000 | 0.010 | 0.25 | 113970.0 | 14.67 | 0.001 |



(a) GA



(b) SA

Figure 12.2: Parameterization of the GA and SA.

SA has just two parameters to tune as this type of algorithm does not include a population. These parameters are used for adjusting the decrement rate of the probability of accepting a solution. We have performed 30 independent runs for 35 combinations of $P_{min} \in \{0.002, 0.004, \dots, 0.010\}$ and $P_{max} \in \{0.01, 0.05, \dots, 0.30\}$ which accounts for 1,050 runs. Table 12.2 (right) presents the 30 best ranked parameter combinations for SA. We can see that the combination $P_{min} = 0.004$ and $P_{max} = 0.05$, presents the best average fitness and is also the best ranked one. There are also other best ranked combinations that might be used as well, as they present a *p-value* high enough to be considered equivalent.

The parameterization of SA can be seen in Figure 12.2b. It is noticeable that the resulting fitness values do not show a big sensitivity to the two parameters (i.e. very robust behavior). Finally, Table 12.3 presents the best values for the parameters of epiGA, GA, and SA, obtained after the parameterization described.

Table 12.3: Configuration of epiGA, GA, and SA.

| | | | | | | | |
|-------|------------------------------|------------------------|----|------------------------------|-------------|----|--|
| epiGA | $P_e = 0.01$ $P_n = 0.10$ | $R = 1$ $\mu = 400$ | GA | $P_c = 1.00$ $P_m = 0.01$ | $\mu = 400$ | SA | $P_{min} = 0.004$ $P_{max} = 0.050$ |
|-------|------------------------------|------------------------|----|------------------------------|-------------|----|--|

12.4 Evaluating epiGA

In this section we evaluate epiGA on 120 instances (four different MKP types) and compare its results to the selected competitors when possible. We wished to know if epiGA is capable of solving these instances and also to compare how competitive it could be. Finally, we address a convergence analysis where we compare the behavior of the epiGA, GA, and SA.

The instances of the MKP are of four different types obtained from the OR-Library [19]: 100 variables and 5 constraints (100.5), 500 variables and 5 constraints (500.5), 100 variables and 10 constraints (100.10), 250 variables and 10 constraints (250.10). In our experimentation we conducted one run of the CPLEX algorithm (it is deterministic) and 30 independent runs of GA, SA, and epiGA, per instance and problem, which amounts to 10,920 runs. Additionally, we have included the results from two SACRO-PSO algorithms [39] (SACRO-BT and SACRO-CBT) and Resolution Search + Branch & Bound [30] (RS + B&B) for the instances in which they were available.

We wished to test if epiGA was able to solve the different instances but also to know if its results were competitive not only against the standard GA and SA, but also against the state-of-the-art algorithms included in our study. We have replicated the results already published according to the experimentation done on SACRO-PSO and RS + B&B algorithms so that we have more data to compare. The author of the SACRO-PSO algorithms carried out 100 runs, optimizing each instance for 20,000 iterations. RS + B&B, like CPLEX, needed just one run. We set up the conditions for CPLEX, GA, SA, and epiGA, equally, i.e. one execution core and thread, 2 Gigabytes of RAM, and one hour as maximum execution time.

In the following tables we show the results obtained from our experiments consisting of the best fitness (profit) found by the algorithms and also whether they are better or worse compared to the epiGA's results (negative percentages mean worse values, and vice versa).

Table 12.4 shows the results of the optimization of 30 instances of the MKP with 100 variables and 5 constraints done by CPLEX, SACRO-BT, SACRO-CBT, GA, SA, and epiGA. We can see that CPLEX, GA, and epiGA have reached the best profit over the 30 instances. In fact, those are the best-known values for these instances. SACRO based algorithms, in turn, have found the best result only in roughly 50% of the instances (16 SACRO-BT and 15 SACRO-CBT), and SA only in 18 of them. It is good to see that the idea of epigenetics, put to work in an algorithm is able to beat published results and, in this case, completely accurately for all 30 instances.

The average execution time of each algorithm when the best value was found was 241 seconds for CPLEX, 5 seconds for GA, 240 seconds for SA and 3 seconds for epiGA. It was

Table 12.4: Accuracy of the algorithms on 30 instances of the 100.5 MKP (100 variables and 5 constraints).

| Instance | CPLEX | | SACRO-BT | | SACRO-CBT | | GA | | SA | | epiGA |
|----------|----------------|-------|--------------|--------|--------------|--------|----------------|-------|--------------|--------|----------------|
| | Best | % | Best | % | Best | % | Best | % | Best | % | Best |
| 100.5_1 | 24381 | 0.00% | 24343 | -0.16% | 24343 | -0.16% | 24381 | 0.00% | 24381 | 0.00% | 24381 |
| 100.5_2 | 24274 | 0.00% | 24274 | 0.00% | 24274 | 0.00% | 24274 | 0.00% | 24274 | 0.00% | 24274 |
| 100.5_3 | 23551 | 0.00% | 23538 | -0.06% | 23538 | -0.06% | 23551 | 0.00% | 23538 | -0.06% | 23551 |
| 100.5_4 | 23534 | 0.00% | 23527 | -0.03% | 23527 | -0.03% | 23534 | 0.00% | 23527 | -0.03% | 23534 |
| 100.5_5 | 23991 | 0.00% | 23991 | 0.00% | 23966 | -0.10% | 23991 | 0.00% | 23966 | -0.10% | 23991 |
| 100.5_6 | 24613 | 0.00% | 24601 | -0.05% | 24601 | -0.05% | 24613 | 0.00% | 24601 | -0.05% | 24613 |
| 100.5_7 | 25591 | 0.00% | 25591 | 0.00% | 25591 | 0.00% | 25591 | 0.00% | 25591 | 0.00% | 25591 |
| 100.5_8 | 23410 | 0.00% | 23410 | 0.00% | 23410 | 0.00% | 23410 | 0.00% | 23410 | 0.00% | 23410 |
| 100.5_9 | 24216 | 0.00% | 24204 | -0.05% | 24216 | 0.00% | 24216 | 0.00% | 24216 | 0.00% | 24216 |
| 100.5_10 | 24411 | 0.00% | 24399 | -0.05% | 24411 | 0.00% | 24411 | 0.00% | 24399 | -0.05% | 24411 |
| 100.5_11 | 42757 | 0.00% | 42705 | -0.12% | 42705 | -0.12% | 42757 | 0.00% | 42757 | 0.00% | 42757 |
| 100.5_12 | 42545 | 0.00% | 42494 | -0.12% | 42471 | -0.17% | 42545 | 0.00% | 42510 | -0.08% | 42545 |
| 100.5_13 | 41968 | 0.00% | 41959 | -0.02% | 41959 | -0.02% | 41968 | 0.00% | 41946 | -0.05% | 41968 |
| 100.5_14 | 45090 | 0.00% | 45090 | 0.00% | 45090 | 0.00% | 45090 | 0.00% | 45090 | 0.00% | 45090 |
| 100.5_15 | 42218 | 0.00% | 42218 | 0.00% | 42218 | 0.00% | 42218 | 0.00% | 42192 | -0.06% | 42218 |
| 100.5_16 | 42927 | 0.00% | 42927 | 0.00% | 42927 | 0.00% | 42927 | 0.00% | 42886 | -0.10% | 42927 |
| 100.5_17 | 42009 | 0.00% | 42009 | 0.00% | 42009 | 0.00% | 42009 | 0.00% | 42009 | 0.00% | 42009 |
| 100.5_18 | 45020 | 0.00% | 45010 | -0.02% | 45020 | 0.00% | 45020 | 0.00% | 45000 | -0.04% | 45020 |
| 100.5_19 | 43441 | 0.00% | 43441 | 0.00% | 43381 | -0.14% | 43441 | 0.00% | 43441 | 0.00% | 43441 |
| 100.5_20 | 44554 | 0.00% | 44554 | 0.00% | 44529 | -0.06% | 44554 | 0.00% | 44554 | 0.00% | 44554 |
| 100.5_21 | 59822 | 0.00% | 59822 | 0.00% | 59822 | 0.00% | 59822 | 0.00% | 59799 | -0.04% | 59822 |
| 100.5_22 | 62081 | 0.00% | 62081 | 0.00% | 62081 | 0.00% | 62081 | 0.00% | 62081 | 0.00% | 62081 |
| 100.5_23 | 59802 | 0.00% | 59802 | 0.00% | 59754 | -0.08% | 59802 | 0.00% | 59802 | 0.00% | 59802 |
| 100.5_24 | 60479 | 0.00% | 60478 | 0.00% | 60478 | 0.00% | 60479 | 0.00% | 60479 | 0.00% | 60479 |
| 100.5_25 | 61091 | 0.00% | 61055 | -0.06% | 61079 | -0.02% | 61091 | 0.00% | 61079 | -0.02% | 61091 |
| 100.5_26 | 58959 | 0.00% | 58959 | 0.00% | 58937 | -0.04% | 58959 | 0.00% | 58959 | 0.00% | 58959 |
| 100.5_27 | 61538 | 0.00% | 61538 | 0.00% | 61538 | 0.00% | 61538 | 0.00% | 61538 | 0.00% | 61538 |
| 100.5_28 | 61520 | 0.00% | 61489 | -0.05% | 61520 | 0.00% | 61520 | 0.00% | 61520 | 0.00% | 61520 |
| 100.5_29 | 59453 | 0.00% | 59453 | 0.00% | 59453 | 0.00% | 59453 | 0.00% | 59453 | 0.00% | 59453 |
| 100.5_30 | 59965 | 0.00% | 59960 | -0.01% | 59960 | -0.01% | 59965 | 0.00% | 59965 | 0.00% | 59965 |
| Average: | 42640.4 | 0.00% | 42630.7 | -0.02% | 42626.9 | -0.03% | 42640.4 | 0.00% | 42632.1 | -0.02% | 42640.4 |

also a good finding to see our epiGA needed the lowest running time out of all the techniques: this at least suggests that its complexity is average-to-low, which is again good news.

Table 12.5 shows the results of the optimization of 30 instances of the MKP with 500 variables and 5 constraints for all the algorithms. In this second set of larger instances, CPLEX and RS + B&B have found the best profit for all the 30 instances. Our proposal, epiGA, has found the best values for nine instances and its average results are almost the same than CPLEX and RS + B&B (lower than 0.01% on average, 0.04% max). GA, SA, and SACRO algorithms have found values which are far worse than the RS + B&B, never achieving the highest values, although GA's results are 0.02% under epiGA on average.

CPLEX took 35 minutes on average, GA and SA consumed their 60 minutes without finding any best value, and epiGA needed 22 minutes on average to find nine. On these instances, epiGA had a better run time than a standard GA and an SA, only improved by CPLEX and RS + B&B. Note that when an algorithm does not find the optimum, it keeps running until reaching the maximum execution time, i.e. one hour. We can see that values for the 15th instance of SACRO algorithms are missing in the original paper [39].

Table 12.6 shows the results of the optimization of 30 instances of the MKP with 100 variables and 10 constraints by CPLEX, SACRO-BT, SACRO-CBT, GA, SA, and epiGA. It

Table 12.5: Accuracy of the algorithms on 30 instances of the 500.5 MKP (500 variables and 5 constraints).

| Instance | CPLEX | | RS + B&B | | SACRO-BT | | SACRO-CBT | | GA | | SA | | epiGA |
|----------|-----------------|-------|-----------------|-------|----------|--------|-----------|--------|----------|--------|----------|--------|---------------|
| | Best | % | Best | % | Best | % | Best | % | Best | % | Best | % | Best |
| 500.5_1 | 120148 | 0.03% | 120148 | 0.03% | 119867 | -0.20% | 120009 | -0.08% | 120050 | -0.05% | 119438 | -0.56% | 120107 |
| 500.5_2 | 117879 | 0.00% | 117879 | 0.00% | 117681 | -0.17% | 117699 | -0.15% | 117843 | -0.03% | 117037 | -0.71% | 117879 |
| 500.5_3 | 121131 | 0.01% | 121131 | 0.01% | 120951 | -0.14% | 120923 | -0.16% | 121050 | -0.05% | 120218 | -0.74% | 121116 |
| 500.5_4 | 120804 | 0.02% | 120804 | 0.02% | 120450 | -0.28% | 120563 | -0.18% | 120720 | -0.05% | 119875 | -0.75% | 120783 |
| 500.5_5 | 122319 | 0.01% | 122319 | 0.01% | 122037 | -0.22% | 122054 | -0.20% | 122248 | -0.04% | 121538 | -0.62% | 122302 |
| 500.5_6 | 122024 | 0.01% | 122024 | 0.01% | 121918 | -0.08% | 121901 | -0.09% | 121960 | -0.04% | 121241 | -0.63% | 122011 |
| 500.5_7 | 119127 | 0.00% | 119127 | 0.00% | 118771 | -0.30% | 118846 | -0.24% | 119055 | -0.06% | 118371 | -0.63% | 119126 |
| 500.5_8 | 120568 | 0.00% | 120568 | 0.00% | 120364 | -0.17% | 120376 | -0.16% | 120486 | -0.07% | 119744 | -0.68% | 120568 |
| 500.5_9 | 121586 | 0.03% | 121586 | 0.03% | 121201 | -0.29% | 121185 | -0.30% | 121504 | -0.04% | 120732 | -0.67% | 121552 |
| 500.5_10 | 120717 | 0.04% | 120717 | 0.04% | 120471 | -0.17% | 120453 | -0.18% | 120665 | -0.01% | 119934 | -0.61% | 120674 |
| 500.5_11 | 218428 | 0.00% | 218428 | 0.00% | 218291 | -0.06% | 218269 | -0.07% | 218347 | -0.03% | 217748 | -0.31% | 218419 |
| 500.5_12 | 221202 | 0.01% | 221202 | 0.01% | 221025 | -0.07% | 221007 | -0.08% | 221130 | -0.03% | 220490 | -0.32% | 221188 |
| 500.5_13 | 217542 | 0.01% | 217542 | 0.01% | 217337 | -0.09% | 217398 | -0.06% | 217470 | -0.02% | 216815 | -0.33% | 217524 |
| 500.5_14 | 223560 | 0.00% | 223560 | 0.00% | 223429 | -0.06% | 223450 | -0.05% | 223513 | -0.02% | 222925 | -0.28% | 223558 |
| 500.5_15 | 218966 | 0.00% | 218966 | 0.00% | N/A | — | N/A | — | 218962 | 0.00% | 218304 | -0.30% | 218966 |
| 500.5_16 | 220530 | 0.00% | 220530 | 0.00% | 220337 | -0.09% | 220428 | -0.04% | 220490 | -0.02% | 220034 | -0.22% | 220527 |
| 500.5_17 | 219989 | 0.00% | 219989 | 0.00% | 219686 | -0.14% | 219734 | -0.12% | 219982 | 0.00% | 219349 | -0.29% | 219989 |
| 500.5_18 | 218215 | 0.00% | 218215 | 0.00% | 218094 | -0.06% | 218096 | -0.05% | 218175 | -0.02% | 217647 | -0.26% | 218215 |
| 500.5_19 | 216976 | 0.00% | 216976 | 0.00% | 216785 | -0.09% | 216851 | -0.06% | 216967 | 0.00% | 216316 | -0.30% | 216976 |
| 500.5_20 | 219719 | 0.00% | 219719 | 0.00% | 219561 | -0.07% | 219549 | -0.08% | 219675 | -0.02% | 219082 | -0.29% | 219717 |
| 500.5_21 | 295828 | 0.00% | 295828 | 0.00% | 295346 | -0.16% | 295309 | -0.18% | 295790 | -0.01% | 295429 | -0.13% | 295828 |
| 500.5_22 | 308086 | 0.00% | 308086 | 0.00% | 307666 | -0.14% | 307808 | -0.09% | 308054 | -0.01% | 307581 | -0.16% | 308083 |
| 500.5_23 | 299796 | 0.00% | 299796 | 0.00% | 299292 | -0.17% | 299393 | -0.13% | 299788 | 0.00% | 299298 | -0.16% | 299788 |
| 500.5_24 | 306480 | 0.00% | 306480 | 0.00% | 305915 | -0.18% | 305992 | -0.16% | 306441 | -0.01% | 305932 | -0.18% | 306476 |
| 500.5_25 | 300342 | 0.00% | 300342 | 0.00% | 299810 | -0.18% | 299947 | -0.13% | 300301 | -0.01% | 299957 | -0.13% | 300342 |
| 500.5_26 | 302571 | 0.00% | 302571 | 0.00% | 302132 | -0.15% | 302156 | -0.14% | 302536 | -0.01% | 302194 | -0.12% | 302571 |
| 500.5_27 | 301339 | 0.00% | 301339 | 0.00% | 300905 | -0.14% | 300854 | -0.16% | 301305 | -0.01% | 300832 | -0.16% | 301329 |
| 500.5_28 | 306454 | 0.01% | 306454 | 0.01% | 306132 | -0.10% | 306069 | -0.12% | 306433 | 0.00% | 305948 | -0.16% | 306430 |
| 500.5_29 | 302828 | 0.01% | 302828 | 0.01% | 302436 | -0.12% | 302447 | -0.12% | 302788 | -0.01% | 302318 | -0.16% | 302809 |
| 500.5_30 | 299910 | 0.00% | 299910 | 0.00% | 299456 | -0.15% | 299558 | -0.12% | 299881 | -0.01% | 299510 | -0.13% | 299904 |
| Average: | 214168.8 | 0.00% | 214168.8 | 0.00% | 213701.6 | -0.21% | 213735.3 | -0.20% | 214120.3 | -0.02% | 213527.9 | -0.29% | 214158.6 |

can be seen that CPLEX, GA, and epiGA have found the best results for all the instances of 100.10 MKP. SACRO-BT has found 18 optimums and SACRO-CBT, 17 of them, while SA has only found the optimum for 7 instances. This again represents a nice endorsement of the recently born epiGA, as we are more convinced that it is not just a new nice inspiration, but an accurate and efficient technique. The average execution time was 386 seconds for CPLEX, 75 seconds for GA, 557 seconds for SA, and 59 seconds for epiGA. Authors of RS + B&B algorithm have not included these instances into their experimentation.

Our last experiment consists in the optimization of 30 instances of MKP with 250 variables and 10 constraints. The results achieved by CPLEX, RS + B&B, GA, SA, and epiGA are presented in Table 12.7. We can see that again RS + B&B has achieved the best results, followed by CPLEX, failing to achieve them in only two instances (just because of

Table 12.6: Accuracy of the algorithms on 30 instances of the 100.10 MKP (100 variables and 10 constraints).

| Instance | CPLEX | | SACRO-BT | | SACRO-CBT | | GA | | SA | | epiGA |
|-----------|----------------|-------|--------------|--------|--------------|--------|----------------|-------|--------------|--------|----------------|
| | Best | % | Best | % | Best | % | Best | % | Best | % | Best |
| 100.10_1 | 23064 | 0.00% | 23064 | 0.00% | 23064 | 0.00% | 23064 | 0.00% | 23055 | -0.04% | 23064 |
| 100.10_2 | 22801 | 0.00% | 22739 | -0.27% | 22750 | -0.22% | 22801 | 0.00% | 22739 | -0.27% | 22801 |
| 100.10_3 | 22131 | 0.00% | 22131 | 0.00% | 22131 | 0.00% | 22131 | 0.00% | 22081 | -0.23% | 22131 |
| 100.10_4 | 22772 | 0.00% | 22772 | 0.00% | 22717 | -0.24% | 22772 | 0.00% | 22650 | -0.54% | 22772 |
| 100.10_5 | 22751 | 0.00% | 22751 | 0.00% | 22751 | 0.00% | 22751 | 0.00% | 22697 | -0.24% | 22751 |
| 100.10_6 | 22777 | 0.00% | 22725 | -0.23% | 22716 | -0.27% | 22777 | 0.00% | 22614 | -0.72% | 22777 |
| 100.10_7 | 21875 | 0.00% | 21875 | 0.00% | 21875 | 0.00% | 21875 | 0.00% | 21785 | -0.41% | 21875 |
| 100.10_8 | 22635 | 0.00% | 22551 | -0.37% | 22542 | -0.41% | 22635 | 0.00% | 22476 | -0.70% | 22635 |
| 100.10_9 | 22511 | 0.00% | 22511 | 0.00% | 22438 | -0.32% | 22511 | 0.00% | 22511 | 0.00% | 22511 |
| 100.10_10 | 22702 | 0.00% | 22702 | 0.00% | 22702 | 0.00% | 22702 | 0.00% | 22561 | -0.62% | 22702 |
| 100.10_11 | 41395 | 0.00% | 41395 | 0.00% | 41388 | -0.02% | 41395 | 0.00% | 41354 | -0.10% | 41395 |
| 100.10_12 | 42344 | 0.00% | 42344 | 0.00% | 42344 | 0.00% | 42344 | 0.00% | 42227 | -0.28% | 42344 |
| 100.10_13 | 42401 | 0.00% | 42350 | -0.12% | 42350 | -0.12% | 42401 | 0.00% | 42347 | -0.13% | 42401 |
| 100.10_14 | 45624 | 0.00% | 45585 | -0.09% | 45511 | -0.25% | 45624 | 0.00% | 45479 | -0.32% | 45624 |
| 100.10_15 | 41884 | 0.00% | 41799 | -0.20% | 41833 | -0.12% | 41884 | 0.00% | 41884 | 0.00% | 41884 |
| 100.10_16 | 42995 | 0.00% | 42995 | 0.00% | 42995 | 0.00% | 42995 | 0.00% | 42941 | -0.13% | 42995 |
| 100.10_17 | 43574 | 0.00% | 43497 | -0.18% | 43517 | -0.13% | 43574 | 0.00% | 43553 | -0.05% | 43574 |
| 100.10_18 | 42970 | 0.00% | 42970 | 0.00% | 42970 | 0.00% | 42970 | 0.00% | 42914 | -0.13% | 42970 |
| 100.10_19 | 42212 | 0.00% | 42212 | 0.00% | 42212 | 0.00% | 42212 | 0.00% | 42212 | 0.00% | 42212 |
| 100.10_20 | 41207 | 0.00% | 41123 | -0.20% | 41134 | -0.18% | 41207 | 0.00% | 41050 | -0.38% | 41207 |
| 100.10_21 | 57375 | 0.00% | 57375 | 0.00% | 57375 | 0.00% | 57375 | 0.00% | 57375 | 0.00% | 57375 |
| 100.10_22 | 58978 | 0.00% | 58922 | -0.09% | 58978 | 0.00% | 58978 | 0.00% | 58975 | -0.01% | 58978 |
| 100.10_23 | 58391 | 0.00% | 58391 | 0.00% | 58391 | 0.00% | 58391 | 0.00% | 58370 | -0.04% | 58391 |
| 100.10_24 | 61966 | 0.00% | 61966 | 0.00% | 61966 | 0.00% | 61966 | 0.00% | 61903 | -0.10% | 61966 |
| 100.10_25 | 60803 | 0.00% | 60803 | 0.00% | 60803 | 0.00% | 60803 | 0.00% | 60803 | 0.00% | 60803 |
| 100.10_26 | 61437 | 0.00% | 61368 | -0.11% | 61368 | -0.11% | 61437 | 0.00% | 61336 | -0.16% | 61437 |
| 100.10_27 | 56377 | 0.00% | 56377 | 0.00% | 56377 | 0.00% | 56377 | 0.00% | 56353 | -0.04% | 56377 |
| 100.10_28 | 59391 | 0.00% | 59332 | -0.10% | 59391 | 0.00% | 59391 | 0.00% | 59391 | 0.00% | 59391 |
| 100.10_29 | 60205 | 0.00% | 60205 | 0.00% | 60205 | 0.00% | 60205 | 0.00% | 60165 | -0.07% | 60205 |
| 100.10_30 | 60633 | 0.00% | 60629 | -0.01% | 60629 | -0.01% | 60633 | 0.00% | 60633 | 0.00% | 60633 |
| Average: | 41606.0 | 0.00% | 41582.0 | -0.06% | 41580.8 | -0.06% | 41606.0 | 0.00% | 41547.8 | -0.14% | 41606.0 |

the time restriction, with enough time the optimum will of course appear). Additionally, epiGA has reached the best results in 17 instances and its average best profit is otherwise just 0.01% around RS + B&B and CPLEX. On the other hand, GA found the optimums in only 11 instances while SA found none, actually it presents the worst values (0.3% below the epiGA ones). SACRO's values are not reported here because its authors have not experimented with these more complex instances.

Now, we aim to clarify the internal behavior of epiGA and its relationship with GA and SA. Figure 12.3 shows the convergence analysis of epiGA for the first instance of the four instance sets addressed. The graphs, which correspond to the best run (out of 30), show that SA converges very slowly compared to the other two algorithms. Additionally, epiGA hits the best value before GA and usually it is a better (higher) value.

Table 12.7: Accuracy of the algorithms on 30 instances of the 250.10 MKP (250 variables and 10 constraints).

| Instance | CPLEX | | RS + B&B | | GA | | SA | | epiGA |
|-----------|---------------|-------|-----------------|-------|---------------|--------|----------|--------|---------------|
| | Best | % | Best | % | Best | % | Best | % | Best |
| 250.10_1 | 59187 | 0.00% | 59187 | 0.00% | 59187 | 0.00% | 58859 | -0.55% | 59187 |
| 250.10_2 | 58781 | 0.13% | 58781 | 0.13% | 58705 | 0.00% | 58390 | -0.54% | 58705 |
| 250.10_3 | 58097 | 0.01% | 58097 | 0.01% | 58094 | 0.00% | 57625 | -0.81% | 58094 |
| 250.10_4 | 61000 | 0.02% | 61000 | 0.02% | 60957 | -0.05% | 60763 | -0.37% | 60989 |
| 250.10_5 | 58092 | 0.00% | 58092 | 0.00% | 58070 | -0.04% | 57728 | -0.63% | 58092 |
| 250.10_6 | 58824 | 0.00% | 58824 | 0.00% | 58765 | -0.10% | 58325 | -0.85% | 58824 |
| 250.10_7 | 58704 | 0.00% | 58704 | 0.00% | 58618 | -0.15% | 58174 | -0.90% | 58704 |
| 250.10_8 | 58933 | 0.00% | 58936 | 0.01% | 58933 | 0.00% | 58547 | -0.65% | 58933 |
| 250.10_9 | 59387 | 0.01% | 59387 | 0.01% | 59381 | -0.01% | 59056 | -0.55% | 59384 |
| 250.10_10 | 59208 | 0.00% | 59208 | 0.00% | 59208 | 0.00% | 58777 | -0.73% | 59208 |
| 250.10_11 | 110913 | 0.02% | 110913 | 0.02% | 110875 | -0.02% | 110542 | -0.32% | 110894 |
| 250.10_12 | 108715 | 0.01% | 108717 | 0.01% | 108689 | -0.01% | 108317 | -0.35% | 108702 |
| 250.10_13 | 108932 | 0.00% | 108932 | 0.00% | 108932 | 0.00% | 108581 | -0.32% | 108932 |
| 250.10_14 | 110086 | 0.00% | 110086 | 0.00% | 110037 | -0.04% | 109687 | -0.36% | 110081 |
| 250.10_15 | 108485 | 0.00% | 108485 | 0.00% | 108458 | -0.02% | 108209 | -0.25% | 108485 |
| 250.10_16 | 110845 | 0.00% | 110845 | 0.00% | 110821 | -0.02% | 110452 | -0.35% | 110845 |
| 250.10_17 | 106077 | 0.00% | 106077 | 0.00% | 106075 | 0.00% | 105797 | -0.26% | 106075 |
| 250.10_18 | 106686 | 0.00% | 106686 | 0.00% | 106686 | 0.00% | 106380 | -0.29% | 106686 |
| 250.10_19 | 109829 | 0.00% | 109829 | 0.00% | 109825 | 0.00% | 109518 | -0.28% | 109825 |
| 250.10_20 | 106723 | 0.00% | 106723 | 0.00% | 106723 | 0.00% | 106502 | -0.21% | 106723 |
| 250.10_21 | 151809 | 0.01% | 151809 | 0.01% | 151801 | 0.00% | 151639 | -0.11% | 151801 |
| 250.10_22 | 148772 | 0.00% | 148772 | 0.00% | 148772 | 0.00% | 148545 | -0.15% | 148772 |
| 250.10_23 | 151909 | 0.00% | 151909 | 0.00% | 151900 | -0.01% | 151765 | -0.09% | 151909 |
| 250.10_24 | 151324 | 0.03% | 151324 | 0.03% | 151269 | 0.00% | 151035 | -0.16% | 151275 |
| 250.10_25 | 151966 | 0.01% | 151966 | 0.01% | 151948 | 0.00% | 151694 | -0.17% | 151948 |
| 250.10_26 | 152109 | 0.00% | 152109 | 0.00% | 152109 | 0.00% | 151795 | -0.21% | 152109 |
| 250.10_27 | 153131 | 0.00% | 153131 | 0.00% | 153131 | 0.00% | 152884 | -0.16% | 153131 |
| 250.10_28 | 153578 | 0.00% | 153578 | 0.00% | 153578 | 0.00% | 153383 | -0.13% | 153578 |
| 250.10_29 | 149160 | 0.00% | 149160 | 0.00% | 149160 | 0.00% | 148879 | -0.19% | 149160 |
| 250.10_30 | 149704 | 0.00% | 149704 | 0.00% | 149704 | 0.00% | 149474 | -0.15% | 149704 |
| Average: | 106365.5 | 0.01% | 106365.7 | 0.01% | 106347.0 | -0.01% | 106044.1 | -0.30% | 106358.5 |

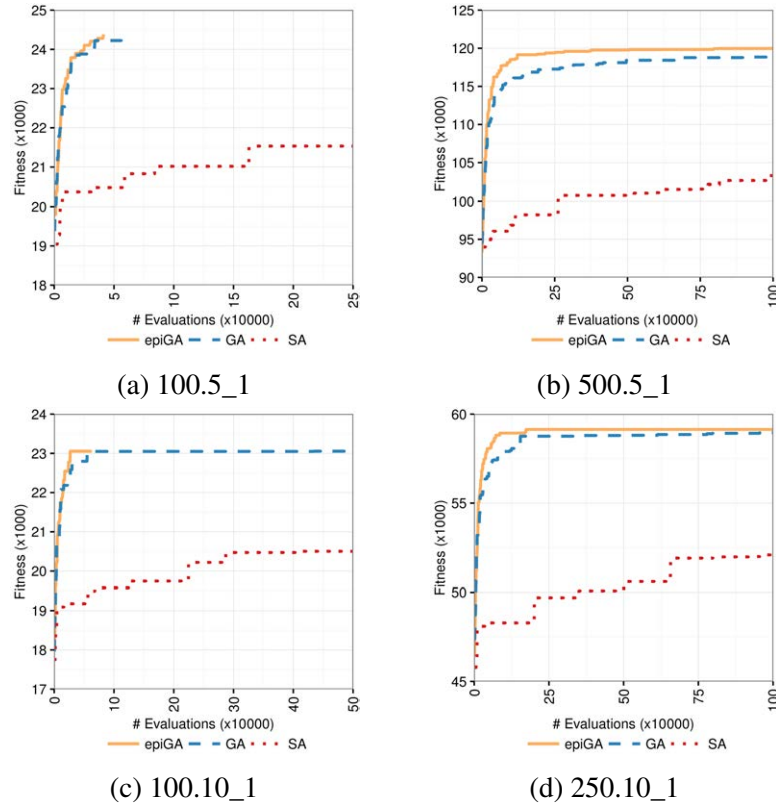


Figure 12.3: Convergence analysis of epiGA, GA, and SA.

12.5 Discussion

In this chapter we have introduced four state-of-the-art competitors to the epiGA, IBM ILOG CPLEX, SACRO-PSO algorithms (SACRO-BT and SACRO-CBT), and Resolution Search + Branch & Bound. Moreover, we have implemented two well-known metaheuristics, namely Genetic Algorithm and Simulated Annealing, as complementary competitors.

We have moved from explanation/construction in the previous chapter to actual evaluation. For this, we have parameterized all the algorithms implemented and tested them on 120 instances of the Multidimensional Knapsack Problem extracted from the OR-Library.

Although our goal in this study is not to improve upon the state of the art, we did perform similarly or better than published results in the literature. In general, our actual goal is to provide the basis for a versatile customizable tool, despite our results showing that not only the values obtained are similar to the state of the art algorithms ones but also our execution times are very competitive.



UNIVERSIDAD
DE MÁLAGA

Chapter 13

Bio-inspired Computing and Smart Mobility

In this chapter we are revisiting the Yellow Swarm architecture to reduce travel times in a big area of the city of Malaga by using the epiGenetic Algorithm, finalizing in this way the research work proposed in this PhD thesis, i.e. solving Smart Mobility problems with our new bio-inspired proposal.

13.1 Introduction

Having presented the Yellow Swarm Architecture in Chapter 8 and the epiGenetic Algorithm (epiGA) in Chapter 11 we present in this chapter the integration of our research work by optimizing a big geographical area of Malaga, featuring realistic traffic flows calculated by our Flow Generator Algorithm (FGA) (Chapter 5). After the optimization, we compare the epiGA's performance with the Evolutionary Algorithm (EA) formerly used in Yellow Swarm.

13.2 Yellow Swarm Revisited

We have selected the Yellow Swarm architecture [206, 208, 211] to test our epiGenetic Algorithm (epiGA) [200] in a realistic map of the city of Malaga.

Yellow Swarm suggests possible detours to vehicles by showing different cyclical indications to drivers using LED panels. We have modified the fitness function in this new study to take into account the maximum travel time of vehicles. Initially, we have just reduced average travel times (Equation 8.1) and the vehicle flowing throughout the city (Equation 8.3). However, we wish now to sacrifice a little of the average improvement to keep every driver under a maximum travel time, so that everyone is not penalized.

13.2.1 Evaluation Function

Equation 13.1 presents the evaluation function. It consists in calculating the average travel time of vehicles for a panel configuration, \vec{X} , plus a penalization term (α_2 , Equation 13.3) which depends on the observed maximum travel time, compared to the maximum travel time observed in Malaga (the scenario without panels). Finally, the entire expression is normalized by α_1^{-1} so that the fitness value of Malaga without panels is 1.0 (Equation 13.2). Note that fitness values under 1.0 represent improvements as we are minimizing travel times.

$$F(\vec{X}) = \frac{1}{\alpha_1} \left[\left(\frac{1}{n} \sum_{i=1}^n TravelTime_i(\vec{X}) \right) + \max\{0, \alpha_2\} \right] \quad (13.1)$$

$$\alpha_1 = \sum_{i=1}^n TravelTime_i(city) \quad (13.2)$$

$$\alpha_2 = \max_i (TravelTime_i(\vec{X})) - \max_i (TravelTime_i(city)) \quad (13.3)$$

13.2.2 Problem Representation

We have placed eight LED panels in our case study whose configuration is represented by the vector of 19 integer values shown in Figure 13.1. They are the time slot for each possible detour and panel. Panels 6, 7, and 8, have three possible detours while the rest has just two, mainly due to the city's street layout.

| Panel 1 | | Panel 2 | | Panel 3 | | Panel 4 | | Panel 5 | | Panel 6 | | | Panel 7 | | | Panel 8 | | |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| T _{1.1} | T _{1.2} | T _{2.1} | T _{2.2} | T _{3.1} | T _{3.2} | T _{4.1} | T _{4.2} | T _{5.1} | T _{5.2} | T _{6.1} | T _{6.2} | T _{6.3} | T _{7.1} | T _{7.2} | T _{7.3} | T _{8.1} | T _{8.2} | T _{8.3} |

Figure 13.1: Representation of the panel configuration (19 integer values).

13.2.3 The epiGenetic Algorithm (epiGA)

We propose our new epiGA to optimize the road traffic by calculating the optimal time slots for each sign of Yellow Swarm. This version of epiGA has a population of 28 individuals, uses Binary Tournament for selection, Nucleosome Based Reproduction, Gene Silencing as epigenetic mechanism, and an elitist replacement.

Gene Silencing was adapted to work with integer values in this case, modifying the solution vector by increasing/decreasing the current values by a fixed value. The decision on the operation was set to be equiprobable by using the Epigenetic Environment. The pseudocode of this operator was presented in Algorithm 11.6.

13.2.4 Evolutionary Algorithm (EA)

Previously, in Chapter 8, we have developed an Evolutionary Algorithm to calculate the configuration of the Yellow Warm panels. It consisted of a (10+2)-EA, featuring Binary

Tournament as the selection operator, Uniform Crossover as the recombination operator, and elitism as the replacement policy. The mutation operator used was already explained in Section 8.2.1. We have used this EA again in this study to compare its results with epiGA to see if they can be improved.

13.3 Case Study

As our case study, we have used the big map calculated by the FGA [197, 201] in Chapter 5, comprising an area of about 32 km² as described in Section 5.3. We have selected the scenarios called 23.2015.WD and 23.2015.SAT, both presenting realistic road traffic during working days and Saturdays in the city of Malaga.

13.4 epiGA vs. EA

We have optimized the proposed scenarios by conducting 30 independent runs of epiGA and EA (120 runs in total). As we wanted a fair comparison, we have set 10000 evaluations as the stop condition for both algorithms. The results of the optimization process is presented in Table 13.1, where we can see that both algorithms have improved each scenario (fitness values under 1.0).

Table 13.1: Comparison of the results obtained by EA and epiGA when optimizing our scenarios.

| Scenario | Algorithm | Fitness | | | Travel Time | |
|----------|-----------|----------------|----------------|----------------|--------------|-------------|
| | | Avg. | StdDev | Min | Avg. | Max. |
| 2015.WD | Malaga | 1.00000 | — | 1.00000 | 580.9 | 1735 |
| | EA | 0.98528 | 0.00438 | 0.97902 | 557.0 | 1686 |
| | epiGA | 0.97377 | 0.00183 | 0.97063 | 563.9 | 1502 |
| 2015.SAT | Malaga | 1.00000 | — | 1.00000 | 574.3 | 1436 |
| | EA | 0.99326 | 0.01576 | 0.98066 | 565.2 | 1346 |
| | epiGA | 0.98169 | 0.00220 | 0.97762 | 561.5 | 1342 |

Additionally, epiGA has turned out to be the most accurate, presenting a better (lower) standard deviation. Regarding the vehicles' travel times, epiGA achieved the lower values and it has also reduced the maximum travel time.

Figure 13.2 presents the boxplots representing the distribution of the fitness values from the algorithms' runs. We can see there evidences which confirm that epiGA has outperformed the EA's results. Moreover, the algorithm comparisons are statistically significant (Wilcoxon's p -value < 0.00).

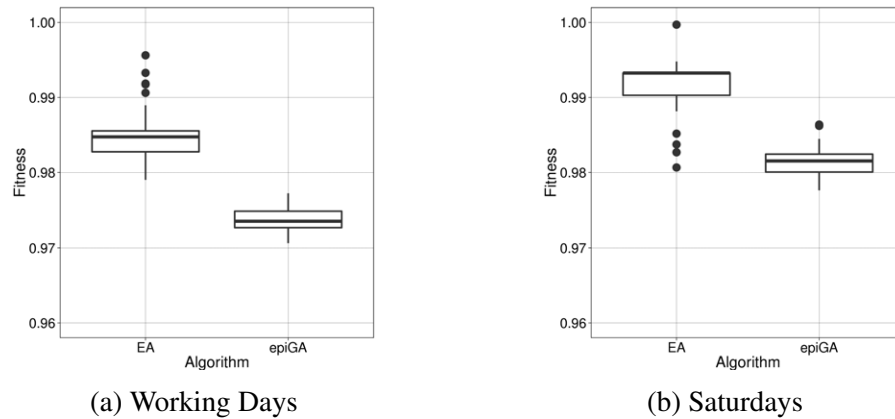


Figure 13.2: Comparison of the algorithms' fitness distributions.

13.5 Discussion

In this chapter we have closed the circle of our research work. We have selected one of the Smart Mobility problems studied and solved it using our new proposal of bio-inspired algorithm. By using an adapted epiGenetic Algorithm we have optimized a realistic scenario generated by our FGA, using the Yellow Swarm panels. Our results have shown an improvement on the performance of EA which represents shorter travel times in the city. This confirms that epiGA is a viable alternative to the traditional optimization algorithms, not only for combinatorial optimization problems, but also for continuous ones.

Part IV

Conclusions and Future Lines of Research



UNIVERSIDAD
DE MÁLAGA

Chapter 14

Conclusions and Future Work

This chapter contains a general review of this PhD thesis and the main conclusions drawn from our studies and experimentation. Finally, future lines of research are discussed.

14.1 Global Conclusions

In this PhD thesis we have defined a set of Smart Mobility problems and how they affect the citizens living in cities around the world. We have described the tools used in our studies, i.e. traffic simulators, especially SUMO, and existing metaheuristics, especially the bio-inspired ones. Moreover, we have built a new tool to generate realistic vehicular flows based on incomplete traffic data published by local councils and called it Flow Generator Algorithm.

Then, we have presented our Smart Mobility proposals, three new architectures: Red Swarm, Green Swarm, and Yellow Swarm, with the aim of reducing travel times, greenhouse gas emissions, and fuel consumption of vehicles in urban areas. Additionally, a method for calculating alternative GPS routes was also proposed. Finally, the last Smart Mobility approach was the analysis of six different predictors to forecast car park occupancy rates and a web prototype as an example of use.

All in all, we have presented solutions to very important Smart Mobility problems: long travel times and high gas emissions due to traffic jams. Our proposals have been tested in different geographical areas of European cities such as Malaga and Madrid in Spain, Stockholm in Sweden, Berlin in Germany, Paris in France, and Quito in Ecuador.

The last part of this PhD thesis was focused on new intelligent algorithms. Specifically, a new family of algorithms based on epigenesis was proposed to solve hard computational problems. We have tested a specific implementation of the epiGenetic Algorithm to solve the Multidimensional Knapsack Problem and compared our competitive results with other state-of-the-art algorithms. Finally, we have solved one of our smart mobility proposals using an epiGenetic Algorithm improving the results of an EA.

Using the Flow Generator Algorithm we have faced incomplete data when building our maps. By using our (10+2)-EA, we have generated traffic flows based on data from sensors and obtained accurate results presenting errors below 10%.

The Red Swarm architecture configured by our EA to suggest alternative routes customized to drivers helped them avoid traffic jams and find a quicker way to reach their destination. Red Swarm based on probabilistic routes, also reduced traffic densities in the most likely traffic situations in a modern city. We have shortened travel times by 18.8% by using Red Swarm.

One step further was the Green Swarm architecture where we have taken advantage of our previous work, improving our design to address also the reduction of greenhouse gas emissions and fuel consumption. Not only have we compared our results with other strategies but also combined Green Swarm with them to improve the city metrics even more. Green Swarm achieved shorter travel times (68%), reduced gas emissions (56% CO, 36% CO₂, 54% HC, 47% PM, 34% NO_x), and saved up to 36% of fuel.

At that point, we realized that despite being very useful and interesting proposals, Red Swarm and Green Swarm required using a terminal such as a smart phone or an on-board unit to inform of new routes. As we wished to present a proposal even easier to use, especially in less developed countries, we designed the Yellow Swarm architecture, a system based on LED panels which can be seen by users while driving throughout the city. Using Yellow Swarm, drivers are able to discover alternative routes to their destination by following the detours suggested by its panels. Yellow Swarm was found to be useful to reduce travel times (up to 32%) and gas emissions (18% CO, 10% CO₂, 16% HC, 13% PM, 9% NO_x), helping users to save fuel (up to 10%).

The last Smart Mobility proposal based on new routes was the calculation of alternative routes for GPS navigators based on the Dynamic User Equilibrium. Again, we have obtained results that encourage exploring new ways that vehicles can reach destination instead of following the same congested routes. We have observed improvements in travel times (up to 18%), CO (up to 14%), CO₂ (up to 7%), HC (up to 13%), PM (up to 5%), and NO_x (up to 7%), and fuel consumption (up to 7%).

There was a negligible increment in route lengths in all our proposals which is a consequence of rerouting vehicles via alternative streets which do not belong to the shortest path. In spite of the variations observed in the results, which must be expected as we are considering different cities (topologies, avenues, roundabouts, intersections), all the metrics were improved, even when only 10% of vehicles were using one of our systems, according to the user acceptance analyses done.

Another different approach was followed to enhance the citizens' quality of life in modern cities: the prediction of car park occupancy rates. We have analyzed six very accurate predictors for forecasting car park occupancy rates in Birmingham, Glasgow, Norfolk, and Nottingham, in the U.K. We have trained them by using real data published by open data initiatives and found that Time Series turned out to be the most accurate predictor although it required the larger amount of data to represent each car park and weekday. Furthermore, our proposal includes a novel web prototype that offers information on real time occupancy rates and historical data.

The use of our proposals by citizens would mean that they are not spending their precious time looking for a parking space or stuck in a traffic jam. They will be happier and healthier,

living a less stressed life, breathing in a cleaner air. Perhaps the city of the future will resemble that of our childhood dreams.

Bio-inspired algorithms are an efficient way of solving problems using models and techniques inspired by nature. We believe in the wisdom of nature and natural selection, so that after studying the epigenetic theory and its mechanisms we have used them to build new computational algorithms. New operators, an environment that can influence chromosomes, DNA methylation, histones and nucleosomes were used in our experiments for solving combinatorial optimization problems. We have used our epiGenetic Algorithm to solve 120 instances of the Multidimensional Knapsack Problem. Although our goal in that study was not to improve upon the state of the art, epiGA did perform similarly or better than published results in the literature.

Finally, we revisited the Yellow Swarm architecture, optimizing the panel's time slots by using epiGA, and compared its performance to the formerly used EA. According to our results, epiGA has outperformed our EA achieving shorter travel times in the scenarios tested.

14.2 Future Lines of Research

As a matter of future work we hope to test epiGA in other, different problems. Also, we will expand its capabilities by including several cells into each individual, which will allow us to implement a parallel version of it. Since we have not tested all the existing epigenetic mechanisms, we plan to do so in the near future to further improve the searching process, adapting it to each problem's characteristics and environments.

Regarding our Smart Mobility architectures, we wished to extend the geographical area to be analyzed to include entire cities. It will imply working on different strategies to implement the rerouting of vehicles by using city districts in order to be able to address the optimization of harder scenarios (computation time and hardware requirements) involving hundreds of thousands of vehicles. Another interesting aspect to take into account is unforeseen events such as accidents, fires, demonstrations, which could suddenly close streets turning open routes into invalid ones. Possible strategies to address this could be switching off rerouting nodes or dynamically updating their configuration.

As part of our future work, we want also to test different strategies to optimally place LED panels throughout the city. Moreover, we want to improve upon our results, especially in the harder scenarios, extend our study to the entire city (slower evaluations as SUMO needs more time and the search space is bigger), and include actual traffic distributions at different hours of the day in order to adapt our system to the many possible variations

The optimization of the afternoon traffic in Quito is another pending study, as there exists a second road traffic peak in the day. Moreover, we need to study the scalability of Yellow Swarm not only in the number of panels, but also in the size of the city we wish to analyze. Furthermore, we want to combine Yellow Swarm with other optimization techniques such as traffic light cycle optimization, maximum speed reduction, and vehicle type constraints. By doing so, we hope to create a holistic approach to improve mobility in smart cities, as well as to define new problems and challenges for intelligent algorithms.

Urban traffic is an actual matter of concern for governments and citizens. Whether for reasons of health, efficiency, or fuel saving, everyone should be worried about it and bio-inspired algorithms can really help to improve the daily trips in the city as we have shown in this PhD thesis. Our proposals have not only proven to be efficient and adaptable to many different situations and domains but can also be implemented with little effort and budget.

Eventually, urban traffic will be improved to zero emissions and traffic jams will be a thing of the past. Our intention was to contribute to those goals, and it was only possible because we were standing on the shoulders of giants.

Part V

Appendices



UNIVERSIDAD
DE MÁLAGA

Appendix A

List of Publications Supporting this PhD Thesis

In this appendix we summarize the set of articles published in indexed journals (4) and the research works presented in Core A international conferences (5), lecture notes (4), and national conferences (1). A summary of the publications supporting this PhD thesis can be seen in Table A.1.

Table A.1: Publications supporting this PhD thesis.

| | |
|--|---|
| ISI JCR Indexed Journals: | 4 |
| CORE A Ranked Intentional Conferences: | 5 |
| International Conferences of Lecture Notes in Computer Science series: | 4 |
| National Conferences: | 1 |
| Book Chapters: | 1 |

ISI JCR Indexed Journals

1. Daniel H Stolfi and Enrique Alba. Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms. *Applied Soft Computing Journal*, 24:181–195, 2014.
doi> [10.1016/j.asoc.2014.07.014](https://doi.org/10.1016/j.asoc.2014.07.014)
 - Impact factor: 2.810 (2014)
 - Category:
 - COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE: Q1
 - COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS: Q1
 - COMPUTER SCIENCE: Q1
2. Daniel H Stolfi and Enrique Alba. Epigenetic algorithms: A New way of building GAs based on epigenetics. *Information Sciences*, 424(Supplement C):250–272, 2018.
doi> [10.1016/j.ins.2017.10.005](https://doi.org/10.1016/j.ins.2017.10.005)

- Impact factor: 4.305 (2017)
 - Category:
 - COMPUTER SCIENCE, INFORMATION SYSTEMS: Q1
 - COMPUTER SCIENCE: Q1
3. Daniel H Stolfi and Enrique Alba. Green Swarm: Greener Routes with Bio-inspired Techniques. *Applied Soft Computing Journal*, 71:952–963, 2018.
doi> [10.1016/j.asoc.2018.07.032](https://doi.org/10.1016/j.asoc.2018.07.032)
- Impact factor: 3.907 (2017)
 - Category:
 - COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE: Q1
 - COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS: Q1
 - COMPUTER SCIENCE: Q1
4. Daniel H Stolfi and Enrique Alba. Generating Realistic Urban Traffic Flows with Evolutionary Techniques. *Engineering Applications of Artificial Intelligence*, 75:36–47, 2018. doi> [10.1016/j.engappai.2018.07.009](https://doi.org/10.1016/j.engappai.2018.07.009)
- Impact factor: 2.819 (2017)
 - Category:
 - AUTOMATION & CONTROL SYSTEMS: Q1
 - COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE: Q1
 - ENGINEERING, MULTIDISCIPLINARY: Q1
 - ENGINEERING, ELECTRICAL & ELECTRONIC: Q2

CORE A Ranked International Conferences

1. Daniel H Stolfi and Enrique Alba. Red Swarm: Smart Mobility in Cities With EAs. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 1373–1380, New York, NY, USA, 2013. ACM.
2. Daniel H. Stolfi and Enrique Alba. Eco-friendly reduction of travel times in european smart cities. In *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*, GECCO '14, pages 1207–1214, New York, NY, USA, 2014. ACM.
3. Daniel H Stolfi and Enrique Alba. Smart Mobility Policies with Evolutionary Algorithms. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, pages 1287–1294, New York, NY, USA, 2015. ACM.
4. Daniel H Stolfi, Rolando Armas, Enrique Alba, Hernan Aguirre, and Kiyoshi Tanaka. Fine Tuning of Traffic in Our Cities with Smart Panels: The Quito City Case Study. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, GECCO '16, pages 1013–1019, New York, NY, USA, 2016. ACM.

5. Daniel H Stolfi and Enrique Alba. Computing New Optimized Routes for GPS Navigators Using Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17*, pages 1240–1247, New York, NY, USA, 2017. ACM.

International Conferences of Lecture Notes in Computer Science series

1. Daniel H Stolfi and Enrique Alba. Reducing Gas Emissions in Smart Cities by Using the Red Swarm Architecture. In Concha Bielza, Antonio Salmerón, Amparo Alonso-Betanzos, J. Ignacio Hidalgo, Luis Martínez, Alicia Troncoso, Emilio Corchado, and JuanM. Corchado, editors, *Advances in Artificial Intelligence*, volume 8109 of *Lecture Notes in Computer Science*, pages 289–299. Springer Berlin Heidelberg, 2013.
2. Daniel H Stolfi and Enrique Alba. An Evolutionary Algorithm to Generate Real Urban Traffic Flows. In José M Puerta, José A Gámez, Bernabe Dorronsoro, Edurne Barrenechea, Alicia Troncoso, Bruno Baruque, and Mikel Galar, editors, *Advances in Artificial Intelligence*, volume 9422 of *Lecture Notes in Computer Science*, pages 332–343. Springer International Publishing, 2015.
3. Christian Cintrano, Daniel H Stolfi, Jamal Toutouh, Francisco Chicano, and Enrique Alba. CTPATH: A Real World System to Enable Green Transportation by Optimizing Environmentally Friendly Routing Paths. In Enrique Alba, Francisco Chicano, and Gabriel Luque, editors, *Smart Cities: First International Conference, Smart-CT 2016, Málaga, Spain, June 15-17, 2016, Proceedings*, pages 63–75. Springer International Publishing, Cham, 2016.
4. Daniel H. Stolfi, Enrique Alba, and Xin Yao. Predicting Car Park Occupancy Rates in Smart Cities. In Enrique Alba, Francisco Chicano, and Gabriel Luque, editors, *Smart Cities: Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings*, pages 107–117. Springer International Publishing, Cham, 2017.

National Conferences

1. Daniel H Stolfi and Enrique Alba. Un Algoritmo Evolutivo para la Reducción de Tiempos de Viaje y Emisiones Utilizando Paneles LED. In *X Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, MAEB 2015, pages 27–34, Mérida - Almendralejo, 2015.

Book Chapters

1. Daniel H Stolfi and Enrique Alba. Sustainable Road Traffic Using Evolutionary Algorithms. In *Sustainable Transportation and Smart Logistics*, In press. Elsevier, 2018.



UNIVERSIDAD
DE MÁLAGA

Appendix B

Resumen en Español

B.1 Introducción

Las ciudades hoy día evolucionan rápidamente. Un gran número de personas residen o están contemplando la posibilidad de trasladar su domicilio habitual a una gran ciudad, aumentando así la demanda de servicios, lo que supone una nueva fuente de problemas complejos [230].

Se observa un notable aumento en el número de traslados que los ciudadanos realizan así como su duración [224]. Estos viajes son con frecuencia para ir a los lugares de trabajo, de estudio, o para llevar los niños al colegio, situaciones que ocurren usualmente en los mismos momentos del día. Otras fuentes de tráfico rodado en la ciudad incluyen personas visitando hospitales, yendo de compras, o realizando desplazamientos para encontrarse con sus familiares o amistades [215].

Estos problemas se hacen evidente en forma de atascos [215], los cuales han ido incrementando su frecuencia en las últimas décadas, convirtiéndose en un problema serio para los residentes de las ciudades. Como resultado, el viajar en coche se ha vuelto más lento, siendo la causa más común para los retrasos, pérdidas económicas y estrés, debido al efecto negativo que una ciudad congestionada tiene sobre las horas de ocio y trabajo de las personas.

Otra consecuencia es la cantidad de gases de efecto invernadero emitidas a las atmósfera. Los vehículos, moviéndose ahora a muy baja velocidad o detenidos en un atasco, emiten aún más gases provenientes de su motor [101]. Entre las emisiones más comunes se encuentran el dióxido de carbono (CO_2), monóxido de carbono (CO), partículas (PM), óxidos de nitrógeno (NO_x), hidrocarburos (HC), metano (CH_4) y ozono de superficie (O_3).

Varias propuestas se han realizado para prevenir atascos y reducir las emisiones [70, 133]. Algunas de ellas se basan en la simulación de tráfico a nivel microscópico (microsimulación) en donde cada vehículo se modela como un agente sujeto a un modelo de movilidad, que se desplaza por una ciudad construida a partir de mapas realistas [212]. Estos estudios tan detallados tienen como contrapartida los largos tiempos de computación necesarios para simular muchos vehículos en escenarios grandes. Como ejemplo, una hora de simulación puede tranquilamente ser equivalente a varios minutos de tiempo real utilizando los ordenadores más recientes. Esto es algo a ser tenido en cuenta cuando se realizan estudios y optimizaciones que requieren evaluar (simular) muchas configuraciones para obtener su valor

de *fitness*. Además, la alta complejidad de los problemas de movilidad, en especial los que presentan muchas rutas que elegir basándose en un criterio de optimización, hace que sean muy difícil de resolver utilizando una heurística exacta (determinista).

El uso de Tecnologías de Información (TI) para resolver problemas de una ciudad del siglo XXI [207] hace más fácil combinar varias técnicas para recolectar datos así como el uso de algoritmos inteligentes basados en metaheurísticas [27]. Estas metaheurísticas están con frecuencia inspiradas en procesos naturales como la teoría de evolución de Darwin, cuyo clásico ejemplo son los Algoritmos Evolutivos [14].

Esta tesis doctoral se encuentra centrada en el diseño de nuevos algoritmos basados en la epigénesis y en la aplicabilidad de sus resultados para mejorar el tráfico rodado en las ciudades. El trabajo de investigación se ha realizado en conexión con los siguientes proyectos de investigación orientados a aplicaciones del mundo real, inteligencia holística y movilidad inteligente: roadME [183], MAXCT [145], moveON [153], CI-RTI [184] y 6city [1].

B.2 Bases Tecnológicas y Científicas

B.2.1 Problemas de Movilidad Inteligente

Un 50% de los europeos utilizan el coche cada día [224], mientras que el 38% de ellos se encuentran con problemas de movilidad mientras viajan por las calles de la ciudad. Algunos de los problemas más importantes que preocupan a nuestra sociedad son la salud de las personas, el desarrollo económico, el consumo de energía, los atascos de tráfico, el precio del combustible, polución, y gestión de residuos.

Estos problemas, relacionados con el crecimiento y desarrollo, representan un desafío para las autoridades de la ciudad si desean gestionarlos de forma inteligente. Aquí es donde la investigación en ciudades inteligentes en conjunto con los sistemas de transporte inteligente se convierte en algo obligatorio, tal como ha sido ya reportado por una infinidad de agencias a lo largo y ancho del planeta [55, 159].

Esta tesis doctoral se concentra en problemas de movilidad tales como tiempos de viaje largos, altas emisiones de gases de efecto invernadero y consumo de combustible, y su optimización utilizando algoritmos bioinspirados. Además se aborda la predicción de plazas libres de aparcamientos, ya que de nada sirve optimizar rutas si cuando se llega a destino se pierde tiempo dando vueltas en busca de un sitio para aparcar el coche, emitiéndose incluso más gases y consumiendo un valioso combustible.

B.2.2 Computación Bioinspirada

Llamamos metaheurísticas [84, 178] a una familia de algoritmos aproximados que son capaces de encontrar buenas soluciones (frecuentemente la mejor) a problemas de optimización complejos, los cuales no pueden resolverse utilizando las técnicas exactas tradicionales, dado que éstas necesitarían tiempos de cómputo extremadamente largos y/o altos requisitos de memoria. Las metaheurísticas presentan dos estrategias de búsqueda: basada en trayectoria o en población.

Por un lado, los algoritmos basados en trayectoria exploran un solo elemento del espacio de soluciones a la vez. Estos algoritmos utilizan algún mecanismo para escapar de los óptimos locales dentro de sus métodos de exploración. Como ejemplo, se pueden nombrar: *Simulated Annealing* (SA) [118], *Tabu Search* (TS) [84], *Greedy Randomized Adaptive Search Procedures* (GRASP) [67], *Variable Neighborhood Search* (VNS) [150], *Iterated Local Search* (ILS) [85], y *Multiple Trajectory Search* (MTS) [228].

Por otro lado, los algoritmos basados en población, trabajan sobre un conjunto de soluciones, por ejemplo una población, colonia, o enjambre. Existe un factor de aprendizaje en estos algoritmos ya que los mismos intentan identificar regiones del espacio de búsqueda conteniendo soluciones de alta calidad mediante el uso de su población. Podemos decir entonces, que estos métodos realizan un muestreo sesgado del espacio de búsqueda. Los algoritmos evolutivos (*Evolutionary Algorithms*, EA) [85], *Scatter Search* (SS) [83], *Estimation of Distribution Algorithms* (EDA) [139], *Differential Evolution* (DE) [213], *Ant Colony Optimization* (ACO) [57], *Artificial Bee Colony* (ABC) [115], y *Particle Swarm Optimization* (PSO) [116], son todos ejemplos de algoritmos poblacionales.

Esta tesis doctoral se enfoca en el uso de computación bioinspirada [143, 171], la cual se inspira en la naturaleza para el diseño de algoritmos capaces de resolver problemas de alta complejidad. Desde el modelo matemático de una neurona [146] hasta el uso de feromonas por hormigas para recoger alimentos de forma más eficiente [114], los algoritmos bioinspirados se agrupan en tres tipos principales, atendiendo a la fuente de inspiración:

1. Computación Evolutiva (*Evolutionary Computing*, EC): Ideas tomadas de la biología evolutiva para diseñar algoritmos evolutivos.
2. Inteligencia de Enjambre (*Swarm Intelligence*, SI): Algoritmos en los cuales un conjunto de agentes sencillos se comportan como organismos sociales.
3. Sistemas Inmunitarios Artificiales (*Artificial Immune Systems*, AIS): Los modelos que siguen los sistemas inmunitarios se utilizan para desarrollar herramientas computacionales.

En esta tesis doctoral se utilizarán variantes de algoritmos evolutivos (*Evolutionary Algorithm*) y genéticos (*Genetic Algorithm*), *Simulated Annealing* y *Ant Colony Optimization*, no sólo como base para el desarrollo de nuevos algoritmos, si no que también para comparar los resultados obtenidos con los del estado del arte.

B.2.3 Microsimulación

Experimentar sobre una ciudad real implicando vehículos y personas reales resulta muy complicado y es casi imposible, en especial si el área de estudio es más grande que una intersección o dos. Ni hablar si el estudio se pretende realizar en tiempo real lo que requeriría extenderlo por semanas o incluso meses.

La simulación por computador [241] ha sido utilizada por muchas disciplinas desde los comienzos de la informática. Sin importar si se trata de la simulación de objetos físicos, procesos químicos, fenómenos atmosféricos, mercados económicos, astrofísica, etc., siempre

es necesario contar con un modelo que represente el comportamiento real del sistema bajo estudio, el cual puede estar sujeto a simplificaciones y generalizaciones.

Los simuladores de tráfico [18, 25] han probado ser una herramienta muy útil para representar todos los factores involucrados en un escenario real, donde las calles de una ciudad, flujos vehiculares, e incluso peatones pueden analizarse *in vitro*, empleando tiempos abordables y usualmente con un alto grado de realismo. Al finalizar la simulación, se dispone de un completo conjunto de valores para analizar el desempeño de los vehículos y sugerir mejoras para la ciudad real (*in silico*), lo que de otro modo sería imposible.

Los simuladores de tráfico implementan diferentes modelos para los flujos vehiculares [32, 95, 140] para definir las reglas del movimiento de los vehículos, cambios de carril, velocidad máxima, etc. Según su grado de granularidad, los simuladores se categorizan como macroscópicos, mesoscópicos, o microscópicos [18].

En los trabajos de investigación presentados en esta tesis doctoral se ha utilizado el simulador microscópico SUMO (Simulation for Urban MObility) [123, 216] desarrollado por el Centro Aeroespacial Alemán (German Aerospace Center, DLR) [53]. El mismo incluye varios programas dentro del mismo paquete para visualización, generación de rutas, importación de mapas, procesamiento de los resultados, etc.

SUMO implementa varios modelos de movilidad [81, 121, 125], pudiéndose definir también muchas características de los vehículos. Además, los escenarios de simulación pueden definirse manualmente o importarse desde OpenStreetMap [169], por ejemplo. SUMO incluso puede ser controlado externamente utilizando una conexión por *socket* y la interfaz TraCI (Traffic Control Interface) [236].

B.2.4 Mapas y Datos de Tráfico Incompletos

Hay varias maneras de enfrentarse a un problema del mundo real. Algunas se basan en modelos matemáticos para generar soluciones candidatas posibles y otras en evaluar un conjunto de soluciones [217].

Entre los modelos generativos, se encuentran los modelos matemáticos del tipo programación lineal, modelos de flujo, dinámica de fluidos, algoritmos genéticos y teoría de juegos; mientras que los modelos evaluativos comprenden teoría de colas, redes de Petri, simulaciones y análisis de perturbaciones [196]. Dada la complejidad de los problemas que se resuelven en esta tesis doctoral, se utilizarán técnicas evolutivas en conjunto con entornos de simulación para la realización de experimentos. Las primeras con el objeto de abordar problemas compuestos por cientos de variables, mientras que los segundos como un medio de inmersión en un mundo virtual para evaluar distintas configuraciones.

Los simuladores de tráfico se han utilizado con frecuencia en la última décadas para validar distintos trabajos de investigación, y no sólo los relacionados con vehículos. Un escenario de movilidad se encuentra compuesto principalmente por el mapa de la ciudad (incluyendo calles, rotondas, restricciones de giro, semáforos, etc.) y sus flujos vehiculares. Estos flujos se obtienen a partir de una matriz de origen-destino (*OD-matrix*) donde se especifica la demanda de viajes de los vehículos entre su punto de partida y de llegada. Dada

la imposibilidad de obtener estos datos para una ciudad grande al completo y así estimar la matriz, los flujos vehiculares pueden generarse a partir de mediciones realizadas por sensores.

En este estudio se presenta una nueva metodología para generar flujos de tráfico realistas, basándose en técnicas evolutivas, los cuales podrán ser incluidos en escenarios de movilidad, utilizando mapas importados desde OpenStreetMap [169] y datos de vehículos obtenidos desde sensores colocados en las calles de la ciudad. Con estas entradas se alimenta nuestro *Flow Generation Algorithm* (FGA) [197, 201] el cual usando un Algoritmo Evolutivo (EA) y un simulador de tráfico (SUMO [123] en nuestro caso), obtiene un modelo de simulación realista (Figura B.1). Este modelo contiene flujos de tráfico calculados de acuerdo a una matriz de origen-destino de forma tal que el número de vehículos en cada punto de medición de ajusta al real de la ciudad. Luego el mismo podrá ser utilizado por investigadores para experimentar con sus propuestas de movilidad inteligente así como otros trabajos de investigación que incluyan simulaciones de tráfico y requieran mapas realistas.

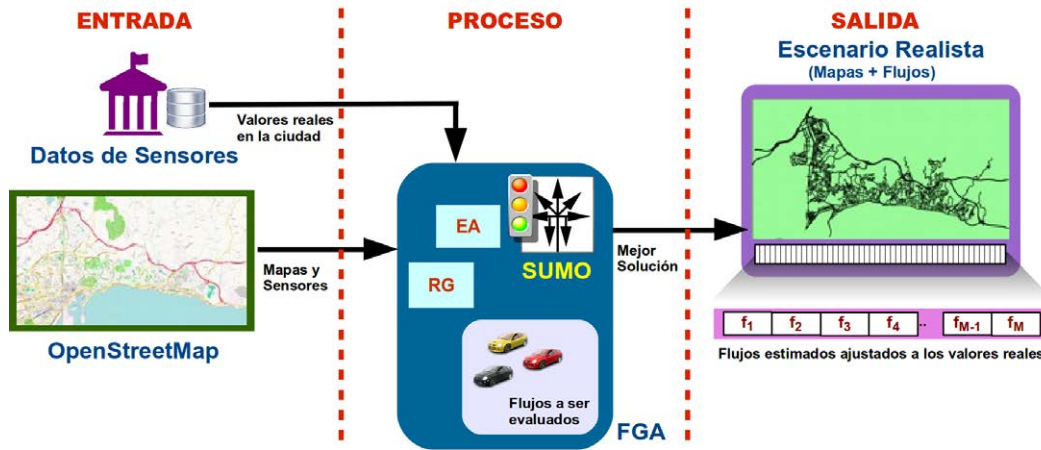


Figura B.1: La arquitectura del Flow Generator Algorithm (FGA).

Formalmente, sea $\vec{v}^* = (v_1^*, \dots, v_N^*)$ un vector que contiene los valores recolectados desde N sensores en la ciudad real, y $\vec{v} = (v_1, \dots, v_N)$ un vector que contiene los valores obtenidos en este caso de la evaluación del modelo de la ciudad en el simulador. Se busca minimizar el error $\vec{e}_i = |\vec{v}_i^* - \vec{v}_i|, i \in \{1, \dots, N\}$ a través de modificaciones de los flujos vehiculares $f = (f_1, \dots, f_M)$.

Resumiendo, buscando flujos apropiados (variables de decisión) se computan flujos estimados para el simulador con el objeto de que se aproximen a los reales medidos en la ciudad. Este conjunto de flujos vehiculares también contendrá un subconjunto que cubre las calles que no disponen de sensores, permitiendo así que los investigadores estudien la ciudad completa, disponiendo de datos para prácticamente todas las calles.

Se experimentó con el FGA sobre 12 escenarios de la ciudad de Málaga utilizándose datos de hasta 23 sensores, publicados por el área de movilidad del ayuntamiento de la ciudad [13]. Tras la optimización se consiguieron resultados (mapas + flujos) con una precisión mayor al 90% en todos los puntos de medición.

B.3 Modelado y Resolución de Problemas

B.3.1 Red Swarm: Reducción de Tiempos de Viaje

En esta sección se propone un nuevo sistema llamado Red Swarm (RS) [203, 204, 205] para la optimización del tráfico rodado en toda la ciudad con el objetivo de reducir los tiempos de viaje. Esto implica un intercambio constante y distribuido de datos entre los vehículos y nodos que nos permite utilizar un algoritmo inteligente para computar segmentos de ruta optimizados y personalizados para cada conductor en la ciudad.

La arquitectura Red Swarm consiste en:

1. Varios nodos distribuidos por la ciudad, instalados en los semáforos, que hacen uso de conexión Wi-Fi para sugerir nuevas rutas a los vehículos.
2. El algoritmo de cambio de rutas (*Rerouting Algorithm*, RA), que selecciona la ruta para cada conductor basándose en su destino y en la configuración del sistema.
3. El Algoritmo Evolutivo (*Evolutionary Algorithm*, EA), que computa la configuración del sistema.
4. Las terminales de usuario (*User Terminal Units*, UTU), que pueden ser teléfonos inteligentes o tabletas, que se utilizan para comunicarse con los nodos, enviar datos y recibir las nuevas rutas.

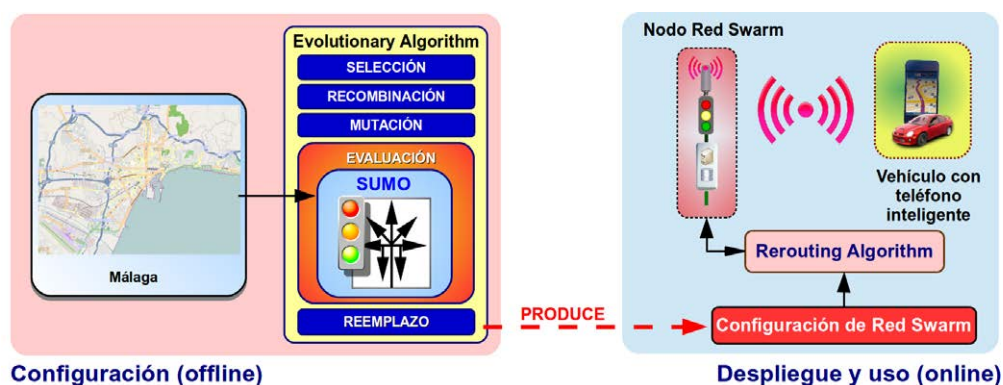


Figura B.2: La arquitectura Red Swarm (RS).

La arquitectura Red Swarm se encuentra dividida en dos etapas: i) la etapa de configuración, y ii) la etapa de despliegue y uso (Figura B.2). En la etapa de configuración, el EA calcula la configuración para los nodos utilizando el simulador SUMO [122] para evaluar cada solución. En la etapa de despliegue y uso, el RA utiliza la configuración de los nodos calculada en la etapa anterior, para sugerir nuevas rutas a los vehículos que se acercan a una intersección controlada por un nodo Red Swarm, mediante un enlace Wi-Fi.

Si bien la configuración no se recalcula en esta segunda fase, las rutas son personalizadas para poder dispersar el tráfico por diversas rutas que benefician tanto a los conductores individualmente como a todo el tráfico en su conjunto. Respecto a las comunicaciones, éstas son posibles en un radio medio de 77 metros tal como se demuestra en [225].

En este estudio se utilizan simulaciones realistas mediante el simulador SUMO y la interfaz TraCI [237] que permite controlar externamente al simulador, implementando así el RA en cada uno de los nodos. Cuando la simulación termina, se obtienen y procesan los datos de cada vehículo tales como tiempo de partida, tiempo de viaje, emisiones, etc.

Red Swarm ha sido probado en escenarios de la ciudad de Málaga mediante el uso de diez nodos. Los resultados obtenidos luego de analizar el comportamiento de hasta 1200 vehículos, presentan reducciones medias en los tiempos de viajes del 9%, llegándose a alcanzar hasta un 19% en el mejor escenario.

B.3.2 Green Swarm: Menos Emisiones de Gases

Otra fuente de problemas en las ciudades grandes es la polución, y el tráfico rodado es una fuente claramente establecida de emisiones de gases de efecto invernadero en áreas urbanas [101]. Siendo conscientes de ello y habiendo observado el comportamiento de los vehículos durante el desarrollo de la arquitectura Red Swarm, se propone aquí una nueva arquitectura, llamada Green Swarm (GS) [199, 202], como una evolución de la primera sujeta a un rediseño y adaptación para reducir ahora no sólo tiempos de viaje, si no que también emisiones de gases de efecto invernadero, y además ahorrar combustible.

La arquitectura Green Swarm sigue una línea de investigación diferente en la cual se abordan escenarios con un número mayor de vehículos, presentando las siguientes nuevas contribuciones:

1. GS utiliza una nueva función de optimización para medir la calidad de las soluciones.
2. Los algoritmos utilizados han sido revisados, mejorando su desempeño para conseguir mejores resultados en tiempos más cortos.
3. Se han optimizado cuatro ciudades diferentes (Málaga, Estocolmo, Berlín y París) y un escenario extra (Alameda) que utiliza flujos vehiculares realistas.
4. Se ha incrementado considerablemente el número de vehículos analizados lo que implica un realismo mayor pero tiempos de cómputo también mayores.
5. Se ha realizado además un estudio de aceptación ya que no todo el mundo va a estar interesado en esta propuesta inicialmente.

Green Swarm, al igual que Red Swarm, puede instalarse en las ciudades modernas con una inversión mínima ya que aprovecha la infraestructura existente consistente en semáforos controlados por ordenador, conectividad Wi-Fi, teléfonos inteligentes y tabletas.

Esta arquitectura se puede ver representada en la Figura B.3. La misma se encuentra dividida en dos etapas: una etapa *offline* llamada la etapa de configuración y una etapa *online* llamada la etapa verde. En la primera el algoritmo EfRA (*Eco-friendly Route Algorithm*) computa la configuración de los nodos y en la segunda, los vehículos que se conectan a un nodo Green Swarm reciben una ruta alternativa hacia su destino facilitada por el GrA (*Green Algorithm*). La ubicación de los nodos se ha realizado manualmente para este estudio ya que esto representa un desafío por si mismo y requeriría un artículo científico dedicado.

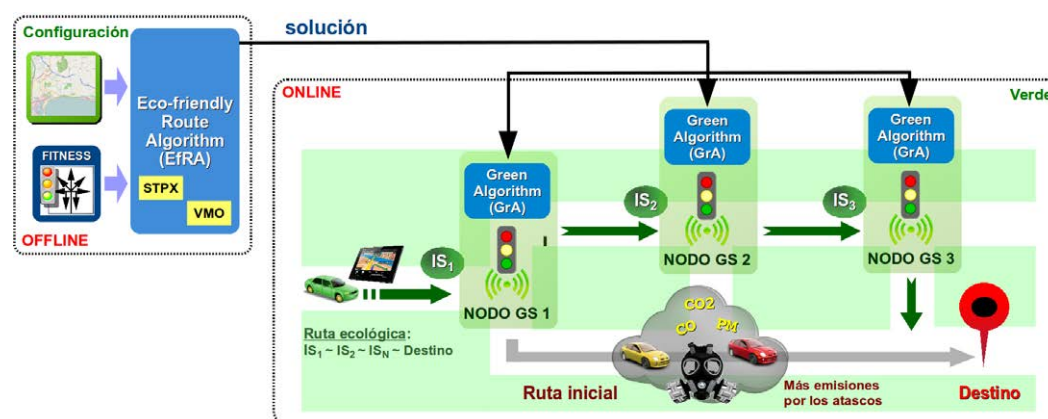


Figura B.3: La arquitectura Green Swarm (GS).

Tras utilizar GS un vehículo viajará probablemente una distancia mayor que cuando seguía su ruta original (usualmente el camino más corto), pero dado que esta es la elección primera de la mayoría de los conductores, el uso de rutas alternativas evitará la formación de atascos favoreciendo una la conducción ecológica. Como resultado, menos emisiones globales y tiempos de viaje más cortos.

Nuevamente los escenarios se evalúan utilizando el simulador SUMO [123] y los cambios de ruta están implementados haciendo uso de TraCI [237].

Green Swarm ha sido probado en escenarios de la ciudad de Málaga, Estocolmo, Berlín y París, habiéndose utilizado siete, seis, seis, y cuatro nodos, respectivamente. Los resultados obtenidos luego de analizar el comportamiento de hasta 4700 vehículos en una hora, presentan reducciones medias en los tiempos de viajes del 31%, un 24% menos emisiones, y un 13% de ahorro de combustible. Además el estudio de porcentajes de utilización realizado ha demostrado que se obtienen mejoras incluso cuando sólo un 10% de los conductores utiliza Green Swarm.

B.3.3 Yellow Swarm: Infraestructura de Bajo Coste Para la Ciudad

Tras estudiar las dos previas arquitecturas y obtener resultados prometedores, se propone aquí un nuevo enfoque que no requiere que los usuarios utilicen dispositivo alguno. Esta nueva propuesta, la arquitectura Yellow Swarm [206, 208, 211], utiliza paneles LED (Light-Emitting Diode) los cuales ubicados en puntos estratégicos de la ciudad sugieren posibles cambios de ruta a los conductores, con el afán de utilizar mejor las calles de la ciudad y prevenir atascos.

Al utilizar una indicación visual (girar a la izquierda, girar a la derecha, o continuar hacia adelante), Yellow Swarm colabora con la seguridad vial evitando distracciones, siendo a la vez una propuesta fácil de implementar y de utilizar, manteniendo los costes muy bajos.

La arquitectura Yellow Swarm, presentada en la Figura B.4, dispone de dos etapas: una *Offline* donde se obtiene la configuración de los paneles mediante el Evolutionary Algorithm (EA), y otra *Online*, en la que los paneles informan a los conductores de los posibles desvíos. Ejecutando iterativamente ambas etapas a una frecuencia dada, es posible mejorar la dinámica

de esta propuesta para que se adapte a los escenarios cambiantes que se pueden encontrar en una ciudad medianamente grande.

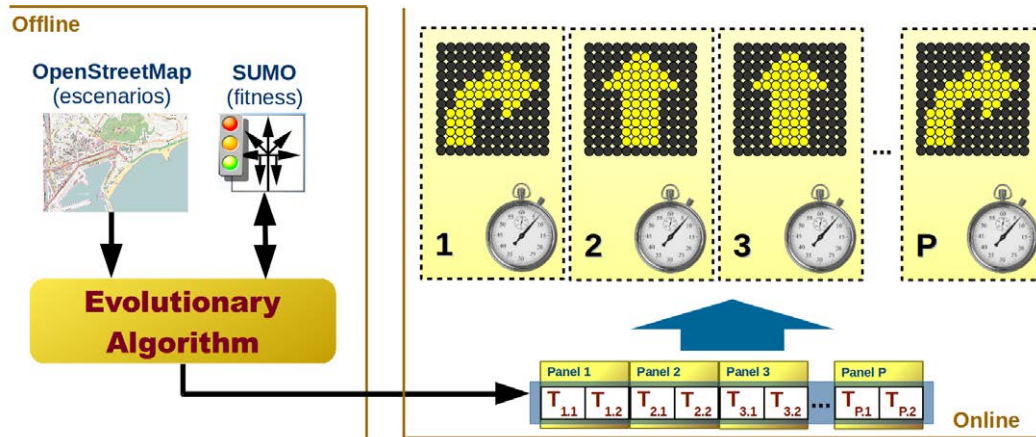


Figura B.4: La arquitectura Yellow Swarm (YS).

Los mapas urbanos utilizados para crear los casos de estudio han sido importados desde OpenStreetMap para experimentar sobre distritos reales de las ciudades utilizando el simulador SUMO [123] y la interfaz TraCI [237] como en los estudios anteriores.

La configuración del sistema consiste en los distintos intervalos de tiempo para cada señal a visualizar en cada panel, de esta manera, ajustando estos tiempos se controla como los vehículos se distribuyen por las calles de la ciudad. Esto disminuye la congestión de las calles, acortando los tiempos de viaje y reduciendo además las emisiones de gases de efecto invernadero y el consumo de combustible.

Yellow Swarm ha sido probado en escenarios de la ciudad de Málaga, Madrid y Quito, habiéndose instalado ocho, cuatro y diez paneles, respectivamente. Los resultados obtenidos luego de analizar el comportamiento de hasta 4840 vehículos en dos horas (Málaga y Madrid) o bien escenarios completos de 24 horas (Quito), presentan tiempos de viajes hasta 32% más cortos, 25% menos emisiones, y un 16% de ahorro de combustible. El estudio de porcentajes de utilización realizado, ha demostrado que también se obtienen mejoras incluso cuando sólo un 10% de los conductores utiliza Yellow Swarm.

B.3.4 Rutas más Inteligentes Para Navegadores GPS

Hoy en día podemos encontrar navegadores GPS en la mayoría de vehículos y teléfonos inteligentes, siendo muy utilizados cuando viajamos por ciudades o vecindarios que nos son desconocidos. El objetivo principal de estos navegadores es dar al usuario las indicaciones correspondientes para alcanzar el destino final de su viaje en un tiempo menor, tomando el camino más rápido.

Si bien muchos de estos dispositivos conocen el estado actual del tráfico y lo utilizan para calcular la ruta ofrecida al conductor, generalmente este servicio no se actualiza en tiempo real y tampoco se encuentra disponible en cualquier parte del mundo. Entonces las rutas terminan calculándose utilizando el algoritmo de Dijkstra [50] o A* [94] que sólo utilizan la

longitud de las calles y a lo sumo el tiempo medio de viaje por ellas, para calcular el mejor camino para llegar a destino. Como consecuencia el uso de navegadores GPS en una ciudad dada podría conducir a la formación de atascos y congestiones de tráfico por el uso preferente de ciertas calles en detrimento de otras.

Nuestra propuesta consiste en el cálculo de rutas alternativas [198] basadas en el concepto del equilibrio dinámico por usuario (Dynamic User Equilibrium, DUE). Estas rutas luego son provistas junto con la cartografía de los navegadores y pueden ser usadas para mejorar las condiciones de tráfico al ser asignadas a los vehículos que atraviesan la ciudad aprovechando mejor las vías disponibles, reduciendo los tiempos de viajes y emisiones, y mejorando la calidad de vida de las personas.

El problema de la asignación de tráfico consiste en asignar rutas a los vehículos que viajan desde su origen a destino, teniéndose en cuenta diversas variables como el coste y beneficio. Se suele resolver calculando una situación de equilibrio en la selección de rutas en la cual una asignación alternativa sólo puede conducir a una situación peor (tiempos de viaje más largos, por ejemplo). Según el primer principio de Wardrop [234], el estado de equilibrio es aquel en cual cada conductor escoge una ruta para la cual el tiempo de viaje es mínimo. Por lo tanto, la red resultante se encuentra en equilibrio dado que nadie puede mejorar su tiempo de viajes escogiendo una ruta diferente.

Nuestra propuesta utiliza el método de simulaciones iterativas [78] para calcular el DUE utilizando SUMO como medio de cálculo de un conjunto de rutas. Éstas se utilizarán luego en conjunción con un navegador GPS para proveer rutas alternativas a los conductores diferentes al camino más corto (*shortest path*). Un ejemplo de uso puede verse en la Figura B.5. La mejor ruta en términos de distancia es claramente la ruta A. Las rutas alternativas, B y C, a pesar de ser más largas, podrían conducir a una reducción de los tiempos de viaje para el tráfico de la ciudad en su conjunto, al evitarse situaciones de congestión.

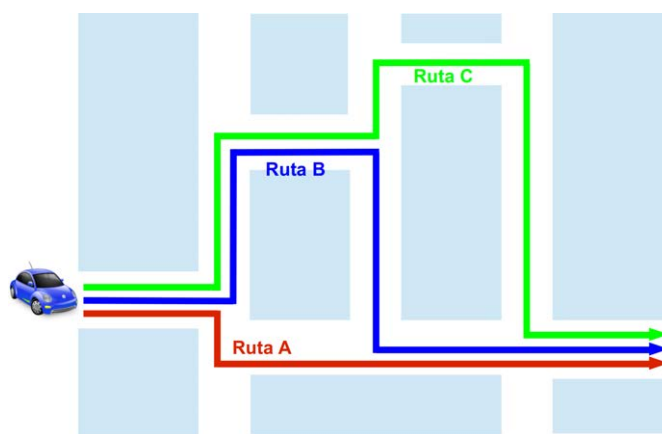


Figura B.5: Ejemplo de cambio de ruta.

La experimentación realizada comprende el área centro de la ciudad de Málaga en la cual el tráfico rodado se ha modelado de forma realista utilizando el FGA antes descrito. Los resultados muestran reducciones en el tiempo de viaje medio del 18%, se emiten un 14% menos gases de efecto invernadero, y también se observa un ahorro de combustible del 7%.

Las rutas alternativas suponen un aumento medio del 3% en las distancias recorridas, muy por debajo de las ventajas obtenidas a cambio.

B.3.5 Conoce tu Ciudad: Plazas de Aparcamiento

Dado que de nada sirve llegar pronto a destino si luego no es posible aparcar al encontrarse todas las plazas ocupadas, se realizó también una propuesta de predicción de plazas de aparcamiento libres. En concreto, nos enfocamos en los aparcamientos cerrados, usualmente subterráneos, que disponen de sensores y que publican esa información libremente en Internet.

La arquitectura desarrollada para la predicción de las plazas libres [209, 210] se muestra en la Figura B.6. La misma utiliza la información sobre los aparcamientos, públicamente disponible, para evaluar seis predictores diferentes: Polinomios; Series de Fourier; *K-Means clustering*; Polinomios ajustados a centroides; Polinomios basados en los anteriores, adaptados a cada aparcamiento; y Series Temporales. En cada caso se propone un predictor para cada aparcamiento y día de la semana para aumentar la precisión de las predicciones y aislar los diferentes comportamientos que suelen observarse entre los días laborables y fines de semana.



Figura B.6: Esquema de la arquitectura del sistema de predicción.

Para el entrenamiento se utilizaron conjuntos de datos de las ciudades de Birmingham (22 aparcamientos, 95733 mediciones), Glasgow (5 aparcamientos, 227275 mediciones), el condado de Norfolk (8 aparcamientos, 388908 mediciones), y la ciudad de Nottingham (12 aparcamientos, 633926 mediciones), todas pertenecientes al Reino Unido. Estos datos se encuentran publicados bajo la licencia *U.K. Open Government Licence* (OGL) [222] o *Creative Commons Attribution* [168].

Como estrategia de entrenamiento se ha utilizado validación cruzada (*K-folds*) para los cinco primeros predictores, mientras que las Series Temporales se entrenaron incrementalmente. Dado que los datos con los que se ha trabajado no estaban completos debido a sensores intermitentemente defectuosos u otros motivos, antes del entrenamiento se realizó el filtrado de los datos inválidos, mientras que los valores faltantes se completaron utilizando el promedio de los cuatro valores previos correspondientes.

Luego de su entrenamiento y configuración, los predictores se pusieron a prueba comparando sus resultados con una nueva semana de valores no vista durante el entrenamiento. En la comparación se utilizó el error cuadrático medio (Mean Squared Error, MSE) para medir la precisión de cada predictor, observándose que las Series Temporales alcanzaron los mejores resultados, seguidas de los Polinomios y las Series de Fourier.

B.4 Nuevos Algoritmos Bioinspirados

Una molécula de ADN consiste en dos hebras entrelazadas una sobre la otra formando una doble hélice. Cada hebra se compone de nucleótidos conteniendo bases nitrogenadas como la guanina (G), adenina (A), timina (T), y citosina (C) [63]. El ADN se organiza en estructuras largas llamada cromosomas (23 pares en humanos que consisten a aproximadamente 25000 genes) los cuales se duplican durante la división celular. Dentro de cada cromosoma, unas proteínas llamadas histonas compactan y organizan el ADN para guiar las interacciones de este con otras proteínas, controlando así como los genes se expresan.

La molécula del ADN porta información genética que puede pasarse desde una generación a la siguiente [12], un concepto que puede encontrarse en los actuales algoritmos evolutivos donde un cromosoma se representa como un vector de símbolos, correspondiéndose cada uno con un gen. Por lo general esta representación se realiza en forma haploide, aunque también se han utilizado diploides para ello [88, 194].

Contrastando con la herencia Mendeliana clásica de los rasgos fenotípicos, provocada por las mutaciones en la secuencia del ADN, bajo la selección natural explicada en la teoría de la evolución de Darwin, los cambios epigenéticos son alteraciones a largo plazo del potencial de transcripción de una célula, debido a la activación de ciertos genes, los cuales no son necesariamente heredables [5].

La Epigenética es el estudio de los mecanismos biológicos que causan alteraciones a largo plazo en el potencial de transcripción de células (primer paso de la expresión génica en la cual un segmento particular del ADN se copia en el ARN) durante su desarrollo sin cambiar la secuencia del ADN, o sea que no implica mutaciones en el ADN en sí mismo [23]. Estas alteraciones pueden ser heredables y no visibles en la generación siguiente, si no en la generación posterior. La expresión génica también puede ser modificada por factores medioambientales [193] como la dieta, hábitos personales, envejecimiento, o cambios aleatorios, que podrían contribuir al desarrollo de fenotipos anormales [112]. Además, las marcas epigenéticas entre generaciones pueden restaurarse, revirtiendo el genoma a su estado original [249].

En el núcleo de los organismos eucariotas (organismos cuya células contienen un núcleo rodeado de membranas), el ADN se compacta en un volumen pequeño para que quepa dentro de la célula. Esta combinación de ADN y proteínas se denomina cromatina (Figura B.7), la cual también protege al ADN, lo refuerza para facilitar la mitosis (duplicación celular donde el núcleo se separa en dos conjuntos de cromosomas iguales), y controla la expresión génica y replicación del ADN. Durante la metafase (la etapa más comprimida) la estructura de la cromatina forma la estructura del cromosoma para prevenir que el ADN sufra cualquier daño cuando los cromosomas se separan. Las modificaciones químicas epigenéticas de las proteínas estructurales en la cromatina también alteran su estructura local.

Los componentes principales de la cromatina son las histonas [23], en las cuales se enrolla el ADN eucariota para formar los nucleosomas (Figura B.8). Los nucleosomas son la unidad fundamental dentro de la cual se empaquetan las histonas y el ADN, para formar una serie de cuentas, como en un collar, compactando así el ADN. Cada nucleosoma consta de ocho histonas las cuales poseen largas colas proteicas que pueden ser modificadas por metilación, acetilación, etc.

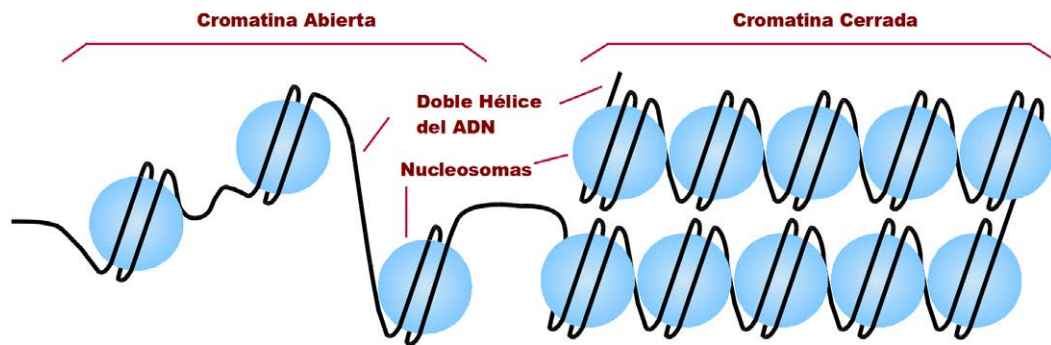


Figura B.7: ADN empaquetado por la cromatina en las células eucariotas.

La metilación del ADN es un factor epigenético reconocido como el principal contribuyente a la estabilidad de los estados de expresión génica en la división celular por mitosis [104] dado que el mismo establece un estado silencioso en la cromatina que modifica los nucleosomas [242]. Los mecanismos epigenéticos delimitan la expresión, adaptando regiones del genoma para mantener el silenciado o la expresión génica [21]. Esto se consigue mediante modificaciones químicas directamente sobre la región del ADN junto con la modificación de proteínas que están asociadas con la ubicación de cada gen [112]. Además, la metilación del ADN y la modificación de histonas sirven como marcas epigenéticas para la cromatina (activa o inactiva), y pueden ser heredables [130].

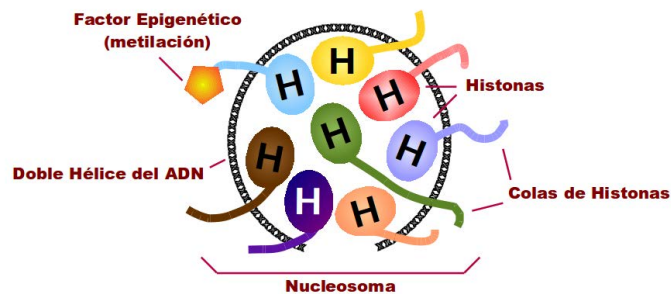


Figura B.8: Factor epigenético: metilación, en una de las histonas del nucleosoma.

Los mecanismos epigenéticos [5], como la impronta genética (*Genomic Imprinting*), pueden ser utilizados como base para la construcción de operadores, que modifican la solución a un problema representado por un cromosoma siguiendo las reglas de la metilación.

El Algoritmo Epigenético (*epiGenetic Algorithm*, epiGA) [200] es una nueva propuesta que consiste en un conjunto de estrategias, basadas en la computación evolutiva, inspiradas en la naturaleza, en especial en la epigenética, con el fin de resolver problemas combinatorios complejos. La base de epiGA es la epigenesis. Nuestro principal interés se halla en como las histonas y el ADN se colapsan para formar nucleosomas, como esto afecta la replicación génica durante la reproducción, y como los mecanismos epigenéticos modifican la expresión génica a través de la metilación, todo ello para construir los operadores bioinspirados de nuestro algoritmo. Creemos que esta es una forma de construir algoritmos diferente a

las existentes en los modelos conocidos que no pierde de vista a los algoritmos genéticos estándares, lo que la hace más fácil de adoptar por otros autores e investigadores.

En la Figura B.9 se presenta el diagrama en bloques del epiGA en donde los operadores epigenéticos se encuentran resaltados.

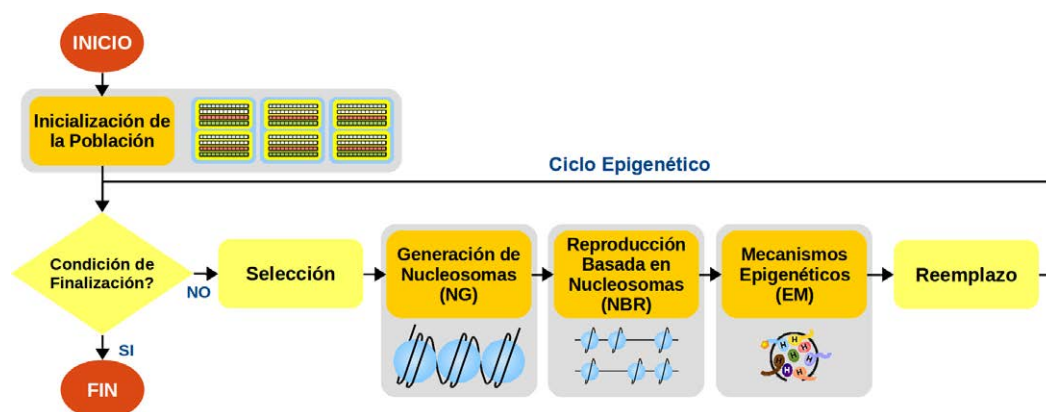


Figura B.9: Diagrama del Algoritmo epiGenético (epiGA).

Durante la Inicialización de la Población se generan nuevos individuos conteniendo células, luego se realiza la Selección (usualmente torneo binario) para obtener una población temporal de trabajo. El siguiente paso consiste en la Generación de Nucleosomas (*Nucleosome Generation*, NG) en los cuales el ADN se colapsa haciéndose inaccesible durante la siguiente etapa, la Reproducción Basada en Nucleosomas (*Nucleosome Based Reproduction*, NBR). Habiéndose generado los descendientes de la población temporal del actual ciclo epigenético, se aplican los Mecanismos Epigenéticos a los mismos según la metilación del ADN y un entorno previamente definido. Por último el ciclo termina, reemplazándose los individuos peor valuados (*fitness*) por los mejores, usualmente en un modo elitista.

Utilizando epiGA se han resuelto 120 instancias (*OR-Library* [19]) del problema de la mochila multidimensional (*Multidimensional Knapsack Problem*, MKP) [93, 138] de distinta complejidad, y se han comparado los resultados obtenidos con otros algoritmos del estado del arte (IBM ILOG CPLEX [108], SACRO-PSO [39], *Resolution Search + Branch & Bound* (RS + B&B) [30], *Genetic Algorithm* (GA) [86, 102] y *Simulated Annealing* (SA) [35, 118]. Los resultados obtenidos muestran que epiGA presenta un mejor comportamiento que GA, SA, y SACRO-PSO en todas las instancias y un comportamiento similar CPLEX y RS + B&B con diferencias menores al 0.2% en los valores máximos obtenidos.

Además se ha puesto a prueba epiGA, un algoritmo bioinspirado (*Bio-inspired Computing*), resolviendo un problema de movilidad inteligente (*Smart Mobility*). En concreto se han optimizado dos escenarios de la arquitectura Yellow Swarm, en los cuales los tiempos para las indicaciones visualizadas en los paneles LED se calcularon utilizando un Algoritmo epiGenético. Como referencia, se realizó también la optimización utilizando el algoritmo evolutivo que formaba parte de la propuesta inicial para la arquitectura, obteniéndose mejoras de hasta el 3% sobre los tiempos de viaje en la ciudad de Málaga y un 13% sobre el tiempo de viaje más largo. Respecto al algoritmo evolutivo, las mejoras fueron del 1% y 11%, respectivamente, todas ellas con significancia estadística (test de Wilcoxon).

B.5 Conclusiones y Trabajo Futuro

En esta tesis doctoral se han definido una serie de problemas de movilidad inteligente, se han descrito las herramientas existentes a utilizar y se han propuesto diferentes soluciones para resolverlos. Además se ha presentado un nuevo algoritmo inteligente, basado en la epigenética, para resolver problemas de optimización, entre ellos, los incluidos en este trabajo de investigación.

Todas las arquitecturas propuestas se han mostrado eficientes y competitivas a la hora de optimizar el tráfico rodado de la ciudad, reduciendo tiempos de viaje y emisiones. Mientras que el nuevo algoritmo propuesto, el Algoritmo epiGenético, ha igualado o superado a los otros algoritmos del estado del arte en la mayoría de las instancias optimizadas.

Como trabajo futuro resta por estudiar el funcionamiento del epiGA sobre otros problemas así como implementar más operadores basados en los mecanismos epigenéticos. En cuanto a las arquitecturas de movilidad inteligente, nuestro deseo es extender las áreas analizadas a ciudades completas que incluyan cientos de miles de vehículos, lo que requiere afrontar nuevos desafíos en términos de potencia de cómputo requerida, eficiencia de los algoritmos, nivel de paralelismo, etc.



UNIVERSIDAD
DE MÁLAGA

List of Figures

| | | |
|------|---|----|
| 2.1 | The six main axes of smart cities. | 10 |
| 3.1 | Classification of metaheuristics. | 20 |
| 3.2 | Statistical validation of results in metaheuristics. | 25 |
| 4.1 | Snapshot of the TRANSIMS's visualization component. | 29 |
| 4.2 | Unity Interface to PTV Vissim. | 30 |
| 4.3 | Traffic and Public Transit in Berlin. | 31 |
| 4.4 | SUMO's GUI (Graphical User Interface). | 32 |
| 4.5 | Some of the features of SUMO. | 34 |
| 4.6 | Scenario building schema. | 36 |
| 5.1 | Architecture of FGA. | 40 |
| 5.2 | Flow Generator Algorithm (FGA). | 41 |
| 5.3 | Different phases when adding routes. | 42 |
| 5.4 | Problem Representation. | 45 |
| 5.5 | Two case studies in Malaga. | 49 |
| 5.6 | Evolution of δ vs. the fitness value of the best individual in the population. . | 51 |
| 6.1 | The Red Swarm architecture. | 58 |
| 6.2 | Red Swarm spots rerouting vehicles. | 59 |
| 6.3 | Rerouting of a vehicle through Red Swarm spots toward its final destination. | 60 |
| 6.4 | Schematic representation of the configuration of the Red Swarm. | 60 |
| 6.5 | Red Swarm's recombination operators. | 62 |
| 6.6 | Variable Mutation Operator (VMO). | 63 |
| 6.7 | Parallel Evolutionary Algorithm (pEA). | 64 |
| 6.8 | Flow chart describing the Rerouting Algorithm. | 65 |
| 6.9 | Area of Malaga Park imported from OpenStreetMap into SUMO. | 66 |
| 6.10 | Traffic density and travel times comparison for the best scenario of z_8 | 73 |
| 6.11 | Traffic density and travel times comparison for the best scenario of z_{12} . . . | 73 |
| 6.12 | Traffic density and travel times comparison for the best scenario of z_{12} (pEA). | 75 |
| 7.1 | Green Swarm Architecture. | 81 |
| 7.2 | Case studies: <i>Alameda, Malaga, Stockholm, Berlin, and Paris</i> | 87 |
| 7.3 | Similarities between CO_2 and the rest of the metrics. | 91 |
| 7.4 | Average improvement of the strategies applied to 500 unseen scenarios. . . | 97 |
| 7.5 | Average improvement achieved by GS for different user acceptance rates. . | 98 |

| | | |
|------|---|-----|
| 7.6 | Convergence of the EfRA over 3000 generations. | 99 |
| 8.1 | The Yellow Swarm architecture. | 102 |
| 8.2 | Status vectors of <i>Malaga</i> , <i>Madrid</i> , and <i>Quito</i> | 103 |
| 8.3 | Example of the mutation operator applied to an individual. | 105 |
| 8.4 | Panel Manager. | 106 |
| 8.5 | Case studies: <i>Malaga</i> and <i>Madrid</i> | 108 |
| 8.6 | Number of vehicles in the <i>Quito</i> case study. | 109 |
| 8.7 | Yellow Swarm panels placed in the business district of Quito. | 109 |
| 8.8 | Traffic density and travel time vs. number of vehicles in <i>Malaga</i> and <i>Madrid</i> | 113 |
| 8.9 | Average improvement and scenarios improved vs. panel use. | 114 |
| 8.10 | Average number of vehicles. | 115 |
| 8.11 | Number of vehicles during the analysis period in Quito. | 116 |
| 8.12 | Fitness, number of generations, and optimization time. | 116 |
| 9.1 | Possible routing example. | 122 |
| 9.2 | <i>DUE.ea</i> diagram and solution encoding. | 124 |
| 9.3 | Crossover operator. | 125 |
| 9.4 | Mutation operator. | 126 |
| 9.5 | City center of Malaga. | 126 |
| 9.6 | Improvements in the metrics for the three scenarios of our case study. | 129 |
| 9.7 | Penetration rate study for the three scenarios of our case study. | 130 |
| 10.1 | Schema of the prediction system architecture. | 132 |
| 10.2 | Predictors analyzed and their relationship. | 134 |
| 10.3 | Case studies: Birmingham, Nottingham, Glasgow, and Norfolk. | 135 |
| 10.4 | Car parks in each cluster. | 138 |
| 10.5 | Training of Shift + Phase (SP) by using <i>K</i> -Fold cross validation. | 139 |
| 10.6 | Training of Time Series (TS) with the datasets for our case studies. | 140 |
| 10.7 | Comparison of the accuracy of our predictors. | 141 |
| 10.8 | Snapshots of our web prototype and the predictions done. | 142 |
| 11.1 | DNA packaged by the chromatin in eukaryotic cells. | 148 |
| 11.2 | An epigenetic factor, i.e. methylation, bound to a histone in a nucleosome. | 149 |
| 11.3 | Schema of each epigenetic mechanism and the modifications made to DNA. | 150 |
| 11.4 | epiGenetic Algorithm (epiGA). | 152 |
| 11.5 | Population of the epiGenetic Algorithm. | 154 |
| 11.6 | Nucleosome Based Reproduction (NBR). | 157 |
| 11.7 | Gene Silencing (GeS). | 158 |
| 12.1 | Fitness variation of epiGA. | 168 |
| 12.2 | Parameterization of the GA and SA. | 169 |
| 12.3 | Convergence analysis of epiGA, GA, and SA. | 175 |
| 13.1 | Representation of the panel configuration (19 integer values). | 178 |
| 13.2 | Comparison of the algorithms' fitness distributions. | 180 |

List of Tables

| | | |
|------|--|-----|
| 5.1 | Fine tuning of the Uniform Crossover Operator. | 50 |
| 5.2 | Fine tuning of the Mutation Operator. | 50 |
| 5.3 | Parameters of FGA. Brief description and values. | 51 |
| 5.4 | Optimization of both case studies using FGA (<i>Setup Stage</i>). | 52 |
| 5.5 | Optimization of 12 scenarios using FGA (<i>Optimization Stage</i>). | 53 |
| 6.1 | Characteristics of the two case studies. | 67 |
| 6.2 | Type and characteristics of vehicles. | 67 |
| 6.3 | Parameter tuning of the ACO algorithm. | 70 |
| 6.4 | Tuning of EA's recombination operator. | 71 |
| 6.5 | Tuning of EA's VMO. | 71 |
| 6.6 | Parameters of EA. | 71 |
| 6.7 | Fitness, number of iterations and statistical tests of one scenario. | 72 |
| 6.8 | Fitness comparative and statistical test of 30 scenarios. | 73 |
| 6.9 | Average fitness, average number of iterations and statistical tests (pEA). | 74 |
| 6.10 | Fitness comparative and statistical test of 30 scenarios of z_8 and z_{12} | 75 |
| 6.11 | Results of the optimization of the vehicles' average travel time. | 76 |
| 6.12 | Average execution times and speedup of ten independent runs. | 76 |
| 7.1 | Parameters of the EfRA. | 84 |
| 7.2 | Characteristics of the four types of vehicles. | 86 |
| 7.3 | Characteristics of <i>Alameda</i> , <i>Malaga</i> , <i>Stockholm</i> , <i>Berlin</i> , and <i>Paris</i> | 88 |
| 7.4 | Improvements in the experts' solution achieved by the strategies. | 92 |
| 7.5 | Relative improvements achieved by using GS after the other strategies. | 95 |
| 7.6 | Average time spent by 30 independent runs in the optimization process. | 95 |
| 7.7 | Average improvement achieved in 50 unseen scenarios. | 96 |
| 7.8 | EfRA compared with GA and SA. | 99 |
| 8.1 | Type and characteristics of vehicles in <i>Malaga</i> and <i>Madrid</i> | 107 |
| 8.2 | Characteristics of <i>Malaga</i> , <i>Malaga_{TT}</i> , <i>Madrid</i> , and <i>Madrid_{TT}</i> | 107 |
| 8.3 | Results of the optimization process of <i>Malaga</i> and <i>Madrid</i> | 111 |
| 8.4 | Configuration of panels obtained by the EA. | 112 |
| 8.5 | Improvement achieved in the metrics of the four case studies. | 112 |
| 8.6 | Optimization sub-intervals. | 115 |
| 8.7 | Fitness values obtained from the four optimization process and statistical tests. | 117 |
| 8.8 | Metrics obtained when using YS in the four training scenarios of Quito. | 117 |

| | | |
|------|--|-----|
| 8.9 | Improvements achieved in the traffic of Quito during an entire day. | 118 |
| 9.1 | Real number of vehicles and the values measured at each sensor. | 127 |
| 9.2 | Results obtained for the scenarios when using the different strategies analyzed. | 128 |
| 10.1 | Characteristics of the datasets before and after applying our training filter. . | 137 |
| 10.2 | Parameters for the predictors calculated by using <i>K</i> -Fold cross validation. . | 137 |
| 10.3 | Average MSE values achieved after testing our predictors on an unseen week. | 139 |
| 12.1 | Parameterization of the epiGA. | 167 |
| 12.2 | Parameterization of the GA and SA. | 169 |
| 12.3 | Configuration of epiGA, GA, and SA. | 170 |
| 12.4 | Accuracy of the algorithms on 30 instances of the 100.5 MKP. | 171 |
| 12.5 | Accuracy of the algorithms on 30 instances of the 500.5 MKP. | 172 |
| 12.6 | Accuracy of the algorithms on 30 instances of the 100.10 MKP. | 173 |
| 12.7 | Accuracy of the algorithms on 30 instances of the 250.10 MKP. | 174 |
| 13.1 | Comparison of the results obtained by EA and epiGA. | 179 |
| A.1 | Publications supporting this PhD thesis. | 189 |

List of Algorithms

| | | |
|------|--|-----|
| 3.1 | Pseudocode of Genetic Algorithm (GA). | 22 |
| 3.2 | Pseudocode of Simulated Annealing (SA). | 23 |
| 3.3 | Pseudocode of Ant Colony Optimization (ACO). | 24 |
| 5.1 | Route Generator (RG). | 43 |
| 5.2 | Evolutionary Algorithm (EA). | 44 |
| 5.3 | Blind Mutation (BM). | 46 |
| 5.4 | Flow Focused Mutation (FFM). | 47 |
| 5.5 | Sensor Focused Mutation (SFM). | 47 |
| 6.1 | Variable Mutation Operator (VMO). | 63 |
| 6.2 | Rerouting Algorithm (RA). | 64 |
| 7.1 | Eco-friendly Route Algorithm (EfRA). | 83 |
| 7.2 | Green Algorithm (GrA). | 85 |
| 8.1 | Mutation Operator. | 105 |
| 9.1 | DUE Routes. | 123 |
| 11.1 | epiGenetic Algorithm (epiGA). | 153 |
| 11.2 | Population Initialization. | 154 |
| 11.3 | Nucleosome Generation (NG). | 155 |
| 11.4 | Nucleosome Based Reproduction (NBR). | 156 |
| 11.5 | Epigenetic Mechanisms (EM). | 157 |
| 11.6 | Gene Silencing (GeS). | 158 |
| 12.1 | Evaluate. | 162 |
| 12.2 | Simulated Annealing (SA). | 165 |
| 12.3 | Generate Function. | 166 |



UNIVERSIDAD
DE MÁLAGA

Index

- A*, 21, 33, 121
ACTIVITYGEN, 33, 38, 109
Alameda Principal, 48, 88
ANOVA Test, 24
Ant Colony Optimization, 3, 11, 20, 23, 68, 70
Ant System, 68
Artificial Bee Colony, 20
Artificial Immune System, 21
Artificial Neural Network, 21

Bat-inspired Algorithm, 21
Bee Colony Optimization, 21
Berlin, 89
Binary Tournament, 44, 84, 99, 104, 125, 155, 165, 178, 179
Bio-inspiration, 3
Birmingham, 134
Bit Flip Mutation, 165
Blind Mutation, 46
Bluetooth, 39
Bookmarking, 151
Branch & Bound, 161, 164

CH₄, 2
Chromatin, 148
Chromosomes, 148
CO, 2, 91, 93, 111, 128
CO₂, 2, 13–15, 80, 84, 91, 93, 110, 128
Convergence, 47, 72, 100, 174
CPLEX, 163
Creative Commons Attribution, 134
Cuckoo Search, 21

Deoxyribonucleic Acid, 148
Destination Crossover, 61, 70

DFROUTER, 33, 38
Differential Evolution, 20
Dijkstra, 14, 33, 36, 40, 42, 67, 121, 128
Distance, 28, 35, 75, 82, 111, 124, 127, 128
Distance Vector, 68
DualIterate, 123
DUAROUTER, 33, 36, 42, 65, 67, 86, 107, 127
Dynamic User Assignment, 33, 34
Dynamic User Equilibrium, 34, 122

Elitist Replacement, 44, 63, 84, 99, 104, 125, 159, 165, 178, 179
Emissions, 2, 4, 13, 28, 35, 36, 79, 101, 103, 124, 127
Entropy, 100
epiGenetic Algorithm, 152, 177
Epigenetic Mechanisms, 149, 156
Epigenetic Probability, 167
Epigenetics, 148
Estimation of Distribution Algorithms, 20
Eukaryotes, 148
Evolutionary Algorithm, 20, 21, 43, 44, 59, 82, 102, 124, 178
Evolutionary Computing, 20
Experts' Solution, 36, 65, 88, 107
Extensible Markup Language, 35, 59

Firefly Algorithm, 21
First Wardrop's principle, 29, 122
Flow Focused Mutation, 46
Flow Generator Algorithm, 39, 127, 179
Fourier Series, 133
Friedman Test, 24, 48, 69, 72, 99, 117, 129, 167

- Fuel, 4, 35, 36, 80, 91, 101, 103, 110, 124, 127, 128
- Gawron Algorithm, 14, 34, 122
- Gene Silencing, 152, 157, 178
- Genetic Algorithm, 3, 21, 99, 161, 165
- Genome Imprinting, 150
- Glasgow, 135
- Global Positioning System, 11, 12, 121
- Greedy Randomized Adaptive Search Procedures, 20
- Greedy Techniques, 161
- Green Swarm, 80
- Harmonoise, 35
- Harmony Search, 21
- HBEFA, 35, 86, 93, 107
- HC, 2, 91, 93, 110, 128
- HDV-LDV, 90
- Heavy Duty Vehicles, 89
- Histones, 149
- Intelligent Transportation Systems, 9
- Intelligent Water Drops, 21
- Iterated Local Search, 20
- Java OpenStreetMap, 35
- JOSM, 86
- JTRROUTER, 33
- K-Fold Cross Validation, 136
- K-Means, 133
- Kernel Search, 161
- KM-Polynomial, 133
- Kolmogorov-Smirnov Test, 24, 99
- Krill Herd, 21
- LED Panels, 101, 177
- Light Duty Vehicles, 89
- Macroscopic Simulators, 28
- Madrid, 106
- Magnetic Sensors, 39
- Malaga, 48, 65, 88, 106, 126, 179
- MAROUTER, 33
- MATSim, 30, 34
- Maximum 30 km/h, 90
- Mean Squared Error, 136
- Mesoscopic Simulators, 28
- Metaheuristics, 19
- Metaphase, 149
- Methylation, 149
- Microscopic Simulators, 28
- Minus 20%, 90
- Mitosis, 148
- Model Predictive Control, 12, 14
- Monkey Search, 21
- Multi-level Search Strategy, 161
- Multidimensional Knapsack Problem, 161
- Multiple Trajectory Search, 20
- Mutation Operator, 104, 125
- NETCONVERT, 33, 36, 48, 86
- NETEDIT, 33
- NETGENERATE, 33
- NO_x, 2, 91, 93, 111, 128
- Noise, 9, 36, 43
- Norfolk, 136
- Nottingham, 136
- Nucleobases, 148
- Nucleosome Based Reproduction, 155, 178
- Nucleosome Generation, 155
- Nucleosome Probability, 167
- Nucleosome Radius, 167
- Nucleosomes, 149
- O₃, 2
- OD-matrix, 33, 38
- OD2TRIPS, 33
- On Board Units, 58, 80
- Open Government Licence, 134
- OpenDRIVE, 34
- OpenStreetMap, 12–14, 35, 39, 48, 65, 86, 106, 126
- Optimization Problem, 19
- OR-Library, 170
- Parallel Evolutionary Algorithm, 63
- Paramutation, 151
- Paris, 89
- Particle Swarm Optimization, 3, 20, 161

- PHEM, 35
- PM, 2, 91, 93, 111, 128
- POLYCONVERT, 33
- Polynomial Fitting, 133
- Population Based Algorithms, 20
- Position Effect, 151
- Quito, 108
- Red Swarm, 58
- Reprogramming, 151
- Rerouting Algorithm, 64
- Ribonucleic acid, 148
- Route Generator, 40, 42
- SACRO-PSO, 164
- Scatter Search, 20
- Sensor Focused Mutation, 47
- Shift & Phase, 133
- Simulated Annealing, 3, 14, 20, 22, 99, 165
- Smart City, 9, 77, 100, 131, 140
- Smart Economy, 10, 100
- Smart Environment, 10, 13
- Smart Governance, 10
- Smart Living, 10
- Smart Mobility, 3, 5, 10, 13, 27, 38, 57, 100, 177
- Smart People, 10, 100
- Stockholm, 89
- Street Two Point Crossover, 61, 70, 84, 125
- Student's *t*-test, 24
- SUMO, 13, 14, 31–33, 38, 40, 48, 58, 65, 82, 86, 102, 104, 107, 109, 123, 126
- SUMO-GUI, 33
- Swarm Intelligence, 21
- Tabu Search, 20
- Termite Colony Optimization, 21
- Thesis Contributions, 5
- Thesis Objectives, 3
- Thesis Phases, 4
- Time Series, 134
- TraCI, 34, 58, 82, 102
- Traffic Simulators, 27
- Trajectory Based Algorithms, 20
- TRANSIMS, 29
- Travel Time, 4, 11, 28, 35, 36, 43, 61, 71, 80, 93, 94, 101, 103, 110, 122, 124, 127, 128, 177, 179
- Uniform Crossover, 44, 48, 99, 104, 165, 179
- Variable Mutation Operator, 62, 70, 84, 99
- Variable Neighborhood Search, 20
- Vehicle to Infrastructure, 14, 57
- Vehicle to Vehicle, 12, 14
- Vehicular Ad-Hoc Networks, 12, 15
- VISSIM, 30, 34
- VISUM, 34
- Wi-Fi, 39, 58, 80, 101
- Wilcoxon Test, 24, 48, 69, 72, 95, 99, 110, 117, 129, 167, 168, 179
- Wireless Sensor Networks, 11, 39
- X-Inactivation, 151
- Yellow Swarm, 101, 177



UNIVERSIDAD
DE MÁLAGA

References

- [1] 6city. *6city: Cross-Intelligence for Smart Cities*. 2017. URL: <http://6city.lcc.uma.es/> (visited on Mar. 16, 2018).
- [2] E. Alba. “Parallel Evolutionary Algorithms can Achieve Super-linear Performance”. In: *Information Processing Letters* 82.1 (Apr. 2002), pp. 7–13.
- [3] E. Alba, G. Luque, and S. Nesmachnow. “Parallel metaheuristics: recent advances and new trends”. In: *International Transactions in Operational Research* 20.1 (Jan. 2013), pp. 1–48.
- [4] E. Alba, A. Nakib, and P. Siarry. *Metaheuristics for dynamic optimization*. Springer Publishing Company, Incorporated, 2012.
- [5] C. D. Allis, T. Jenuwein, D. Reinberg, and M.-L. Caparros. *Epigenetics*. Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY: 2007.
- [6] M. Alsabaan, K. Naik, T. Abdelkader, T. Khalifa, and A. Nayak. “Geocast Routing in Vehicular Networks for Reduction of CO2 Emissions”. In: *Information and Communication on Technology for the Fight against Global Warming*. Ed. by D. Kranzlmüller and A. Toja. Vol. 6868. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 26–40.
- [7] H. Amer, N. Salman, M. Hawes, M. Chaqfeh, L. Mihaylova, and M. Mayfield. “An Improved Simulated Annealing Technique for Enhanced Mobility in Smart Cities”. In: *Sensors* 16.7 (June 2016), p. 1013.
- [8] E. Angelelli, R. Mansini, and M. Grazia Speranza. “Kernel search: A general heuristic for the multi-dimensional knapsack problem”. In: *Computers and Operations Research* 37.11 (2010), pp. 2017–2026.
- [9] M. D. Anway, A. S. Cupp, M. Uzumcu, and M. K. Skinner. “Epigenetic transgenerational actions of endocrine disruptors and male fertility”. In: *Science* 308.5727 (2005), pp. 1466–1469.
- [10] R. Armas, H. Aguirre, F. Daolio, and K. Tanaka. “Traffic signal optimization and coordination using neighborhood mutation”. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. 2016, pp. 395–402.
- [11] R. Armas, H. Aguirre, S. Zapotecas-Martinez, and K. Tanaka. “Traffic Signal Optimization: Minimizing Travel Time and Fuel Consumption”. In: *Artificial Evolution*. Springer. 2015, pp. 29–43.
- [12] O. T. Avery, C. M. MacLeod, and M. McCarty. “Studies on the chemical nature of the substance inducing transformation of pneumococcal types induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III”. In: *The Journal of experimental medicine* 79.2 (1944), pp. 137–158.

- [13] Ayuntamiento de Malaga. *Area de Movilidad*. 2018. URL: <http://movilidad.malaga.eu/> (visited on Mar. 16, 2018).
- [14] T. Bäck. *Evolutionary algorithms in theory and practice*. Oxford Univ. Press, 1996.
- [15] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.
- [16] S.-H. Bae, T.-Y. Heo, and B.-Y. Ryu. “An Evaluation of the Ramp Metering Effectiveness in Reducing Carbon Dioxide Emissions”. In: *Simulation* 88.11 (Nov. 2012), pp. 1368–1378.
- [17] T. Bakici, E. Almirall, and J. Wareham. “A Smart City Initiative: the Case of Barcelona”. In: *Journal of the Knowledge Economy* 4.2 (2013), pp. 135–148.
- [18] J. Barceló. *Fundamentals of traffic simulation*. Vol. 145. Springer, 2010.
- [19] J. E. Beasley. “OR-Library: Distributing Test Problems by Electronic Mail”. In: *Journal of the Operational Research Society* 41.11 (1990), pp. 1069–1072.
- [20] E. S. Belyaeva, O. V. Demakova, G. H. Umbetova, and I. F. Zhimulev. “Cytogenetic and molecular aspects of position-effect variegation in *Drosophila melanogaster*”. In: *Chromosoma* 102.8 (1993), pp. 583–590.
- [21] J. Bender. “DNA methylation and epigenetics”. In: *Annual Review of Plant Biology* 55 (2004), pp. 41–68.
- [22] S. Bera and K. V. K. Rao. “Estimation of origin-destination matrix from traffic counts: The state of the art”. In: *European Transport - Trasporti Europei* 49.49 (2011), pp. 3–23.
- [23] J. M. Berg, J. L. Tymoczko, and L. Stryer. *Biochemistry*. 6th. WH Freeman & Company Limited, 2006.
- [24] D. Bertsekas and R. Gallager. *Data Networks*. 2nd. Vol. 2. Prentice-hall Englewood Cliffs, NJ, 1987.
- [25] M. Bierlaire. “Simulation and optimization: A short review”. In: *Transportation Research Part C: Emerging Technologies* 55 (2015), pp. 4–13.
- [26] Birmingham City Council. *Parking in Birmingham*. 2018. URL: <https://data.birmingham.gov.uk/dataset/birmingham-parking/> (visited on Mar. 16, 2018).
- [27] C. Blum and A. Roli. “Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison”. In: *Computing* 35 (2003), pp. 268–308.
- [28] P. M. Boesch, F. Ciari, and K. W. Axhausen. “Autonomous Vehicle Fleet Sizes Required to Serve Different Levels of Demand”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2542 (Jan. 2016), pp. 111–119.
- [29] I. Boussaïd, J. Lepagnot, and P. Siarry. “A survey on optimization metaheuristics”. In: *Information Sciences* 237 (2013), pp. 82–117.
- [30] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon. “A multi-level search strategy for the 0-1 Multidimensional Knapsack Problem”. In: *Discrete Applied Mathematics* 158.2 (2010), pp. 97–109.

- [31] B. Bowerman, J. Braverman, J. Taylor, H. Todosow, and U. von Wimmersperg. “The vision of a smart city”. In: *2nd International Life Extension Technology Workshop*. Paris, 2000.
- [32] W. Burghout. “Hybrid microscopic-mesoscopic traffic simulation”. PhD thesis. KTH, 2004.
- [33] F. Caicedo, C. Blazquez, and P. Miranda. “Prediction of parking space availability in real time”. In: *Expert Systems with Applications* 39.8 (2012), pp. 7281–7290.
- [34] A. Camero, J. Toutouh, D. H. Stolfi, and E. Alba. “Evolutionary Deep Learning for Car Park Occupancy Prediction in Smart Cities”. In: *International Conference on Learning and Intelligent Optimization*. Springer. 2018, In press.
- [35] V. Černý. “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm”. In: *Journal of Optimization Theory and Applications* 45.1 (1985), pp. 41–51.
- [36] V. L. Chandler and M. Stam. “Chromatin conversations: mechanisms and implications of paramutation”. In: *Nature Reviews Genetics* 5.7 (2004), pp. 532–544.
- [37] H. S. Chandra and V. Nanjundiah. “The evolution of genomic imprinting”. In: *Development* 108.Supplement (1990), pp. 47–53.
- [38] S. Y. Cheung, S. C. Ergen, and P. Varaiya. “Traffic surveillance with wireless magnetic sensors”. In: *Proceedings of the 12th ITS world congress*. Vol. 1917. 2005, p. 173181.
- [39] M. Chih. “Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem”. In: *Applied Soft Computing Journal* 26 (2015), pp. 378–389.
- [40] M. Chih, C. J. Lin, M. S. Chern, and T. Y. Ou. “Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem”. In: *Applied Mathematical Modelling* 38.4 (2014), pp. 1338–1350.
- [41] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl. “Understanding Smart Cities: An Integrative Framework”. In: *2012 45th Hawaii International Conference on System Sciences* (Jan. 2012), pp. 2289–2297.
- [42] P. C. Chu and J. E. Beasley. “A Genetic Algorithm for the Multidimensional Knapsack Problem”. In: *Journal of Heuristics* 4.1 (1998), pp. 63–86.
- [43] V. Chvátal. “Resolution search”. In: *Discrete Applied Mathematics* 73.1 (1997), pp. 81–99.
- [44] C. Cintrano, D. H. Stolfi, J. Toutouh, F. Chicano, and E. Alba. “CTPATH: A Real World System to Enable Green Transportation by Optimizing Environmentally Friendly Routing Paths”. In: *Smart Cities: First International Conference, Smart-CT 2016, Málaga, Spain, June 15-17, 2016, Proceedings*. Ed. by E. Alba, F. Chicano, and G. Luque. Cham: Springer International Publishing, 2016, pp. 63–75.
- [45] Z. Cong, B. De Schutter, and R. Babuška. “Ant Colony Routing algorithm for freeway networks”. In: *Transportation Research Part C: Emerging Technologies* 37 (Dec. 2013), pp. 1–19.
- [46] J. Craig and N. C. Wong. *Epigenetics: a reference manual*. Horizon Scientific Press, 2011.

- [47] L. N. De Castro and J. Timmis. *Artificial immune systems: a new computational intelligence approach*. Springer Verlag, 2002.
- [48] M. De Felice, E. Cerqueira, A. Melo, M. Gerla, F. Cuomo, and A. Baiocchi. “A distributed beaconless routing protocol for real-time video dissemination in multimedia VANETs”. In: *Computer Communications* 58 (Mar. 2015), pp. 40–52.
- [49] J. Derrac, S. García, D. Molina, and F. Herrera. “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18.
- [50] E. W. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. In: *Numerische mathematik* (1959), pp. 269–271.
- [51] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy. “A Communications-Oriented Perspective on Traffic Management Systems for Smart Cities: Challenges and Innovative Approaches”. In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 125–151.
- [52] F. Djannaty and S. Doostdar. “A hybrid genetic algorithm for the multidimensional knapsack problem”. In: *International Journal of Contemporary Mathematical Sciences* 3.9 (2008), pp. 443–456.
- [53] DLR. *German Aerospace Center (DLR)*. 2018. URL: <http://www.dlr.de/> (visited on Mar. 16, 2018).
- [54] G. Dodig-Crnkovic. “Scientific Methods in Computer Science”. In: *In Proc. PROMOTE IT 2002, 2nd Conference for the Promotion of Research in IT at New Universities and at University Colleges in*. 2002, pp. 22–24.
- [55] D. Donnerer. *Horizon 2020 and Smart Cities & Communities: Our recommendations for the work programmes 2018-2020*. Tech. rep. Energy Cities - The European Association of Local Authorities in Energy Transition, 2017.
- [56] R. Doolan and G. Muntean. “VANET-enabled Eco-friendly Road Characteristics-aware Routing for Vehicular Traffic”. In: *IEEE Vehicular Technology Conference (VTC)*. Desden, Germany, 2013.
- [57] M. Dorigo. “Optimization, learning and natural algorithms”. In: *Ph. D. Thesis, Politecnico di Milano, Italy* (1992).
- [58] M. Dorigo, M. Birattari, T. Stützle, M. Birattari, and T. Stützle. *Ant colony optimization*. Vol. 1. 4. MIT Press, Nov. 2006, pp. 28–39.
- [59] M. Dorigo, V. Maniezzo, and A. Colomi. “Ant System: Optimization by a Colony of Cooperating Agents”. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 26.1 (1996), pp. 29–41.
- [60] P. P. Dubey and P. Borkar. “Review on techniques for traffic jam detection and congestion avoidance”. In: *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*. IEEE. Feb. 2015, pp. 434–440.
- [61] K. Duncan and C. A. Brebbia. *Disaster management and human health risk: Reducing risk, improving outcomes*. Vol. 110. WIT Press, 2009.
- [62] G. Egger, G. Liang, A. Aparicio, and P. A. Jones. “Epigenetics in human disease and prospects for epigenetic therapy”. In: *Nature* 429.6990 (2004), pp. 457–463.

- [63] M. Egli and W. Saenger. *Principles of Nucleic Acid Structure*. 1st ed. New York: Springer-Verlag, 1984, p. 556.
- [64] European Commission. *Europe 2020: A Strategy for Smart, Sustainable and Inclusive Growth: Communication from the Commission*. Publications Office, 2010.
- [65] European Commission. *Transport & Environment*. 2018. URL: <https://web.archive.org/web/20121111033537/http://ec.europa.eu/environment/air/transport/road.htm> (visited on Mar. 16, 2018).
- [66] A. P. Feinberg and B. Vogelstein. “Hypomethylation distinguishes genes of some human cancers from their normal counterparts”. In: *Nature* 301.5895 (1983), pp. 89–92.
- [67] T. A. Feo and M. G. C. Resende. “Greedy Randomized Adaptive Search Procedures”. In: *Journal of Global Optimization* 6 (1995), pp. 109–133.
- [68] A. Fernández-Ares, A. M. Mora, M. G. Arenas, P. García-Sánchez, G. Romero, V. Rivas, P. A. Castillo, and J. J. Merelo. “Studying real traffic and mobility scenarios for a Smart City using a new monitoring and tracking system”. In: *Future Generation Computer Systems* 76 (2017), pp. 163–179.
- [69] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, and P. Pucci. “Smart City: An Event Driven Architecture for Monitoring Public Spaces with Heterogeneous Sensors”. In: *2010 Fourth International Conference on Sensor Technologies and Applications* (July 2010), pp. 281–286.
- [70] R. Florin and S. Olariu. “A survey of vehicular communications for traffic signal optimization”. In: *Vehicular Communications* 2.2 (Apr. 2015), pp. 70–79.
- [71] A. S. Fukunaga. “A branch-and-bound algorithm for hard multiple knapsack problems”. In: *Annals of Operations Research* 184.1 (Apr. 2011), pp. 97–119.
- [72] R. Gakenheimer. “Urban mobility in the developing world”. In: *Transportation Research Part A: Policy and Practice* 33.7-8 (1999), pp. 671–689.
- [73] H.-S. Gan, K.-F. Richter, M. Shi, and S. Winter. “Integration of simulation and optimization for evacuation planning”. In: *Simulation Modelling Practice and Theory* 67 (Sept. 2016), pp. 59–73.
- [74] A. H. Gandomi and A. H. Alavi. “Krill herd: A new bio-inspired optimization algorithm”. In: *Communications in Nonlinear Science and Numerical Simulation* 17.12 (2012), pp. 4831–4845.
- [75] M. T. Garip, M. E. Gursoy, P. Reiher, and M. Gerla. “Scalable reactive vehicle-to-vehicle congestion avoidance mechanism”. In: *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, Jan. 2015, pp. 943–948.
- [76] M. R. Gary and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. 1979.
- [77] H. G. Gauch. *Scientific method in practice*. Cambridge University Press, 2003.
- [78] C. Gawron. “Simulation-based traffic assignment”. PhD thesis. University of Cologne, 1998.

- [79] A. Gibaud, P. Thomin, and Y. Sallez. “Foresee, a Fully Distributed Self-organized Approach for Improving Traffic Flows”. In: *Simulation Modelling Practice and Theory* 19.4 (Apr. 2011), pp. 1096–1117.
- [80] R. Giffinger, C. Fertner, H. Kramar, and R. Kalasek. *Smart cities Ranking of European medium-sized cities*. Tech. rep. October. 2007, pp. 13–18.
- [81] P. G. Gipps. “A behavioural car-following model for computer simulation”. In: *Transportation Research Part B: Methodological* 15.2 (1981), pp. 105–111.
- [82] Glasgow City Council. *Glasgow Car Parks*. 2018. URL: <https://data.glasgow.gov.uk/dataset/glasgow-car-parks-feed/> (visited on Mar. 16, 2018).
- [83] F. Glover. “A Template For Scatter Search And Path Relinking”. In: *Computer* (1998), pp. 13–54.
- [84] F. Glover. “Future paths for integer programming and links to artificial intelligence”. In: *Computers & Operations Research* 13 (1986), pp. 533–549.
- [85] F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media, 2006.
- [86] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [87] D. E. Goldberg and K. Deb. “A comparative analysis of selection schemes used in genetic algorithms”. In: *Foundations of genetic algorithms*. Vol. 1. Elsevier, 1991, pp. 69–93.
- [88] D. E. Goldberg, E. Smith, and R. E. Smith. “Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy.” In: *International Conference on Genetic Algorithms*. 1987, pp. 59–68.
- [89] D. J. Griggs and M. Noguer. “Climate change 2001: the scientific basis. Contribution of working group I to the third assessment report of the intergovernmental panel on climate change”. In: *Weather* 57.8 (2002), pp. 267–269.
- [90] C. Guerreiro, F. de Leeuw, and V. Foltescu. *Air quality in Europe - 2013 report*. Tech. rep. European Environment Agency, 2013.
- [91] M. Haklay and P. Weber. “OpenStreetMap: User-Generated Street Maps”. In: *IEEE Pervasive Computing* 7.4 (Oct. 2008), pp. 12–18.
- [92] C. Harrison, B. Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, and P. Williams. “Foundations for smarter cities”. In: *IBM Journal of Research and Development* 54.4 (2010), pp. 1–16.
- [93] A. S. M. Harry M. Markowitz. “On the Solution of Discrete Programming Problems”. In: *Econometrica* 25.1 (1957), pp. 84–110.
- [94] P. E. Hart, N. J. Nilsson, and R. Bertram. “A formal basis for the heuristic determination of minimum cost paths”. In: *Systems Science and Cybernetics, IEEE Transactions on* 4.2 (1968), pp. 100–107.
- [95] H. Hartenstein and K. Laberteaux. *VANET: Vehicular Applications and Inter-networking Technologies*. Ed. by L. John Wiley & Sons. Vol. 1. Chichester, UK: Wiley Online Library, 2009.

- [96] S. Hausberger, M. Rexeis, M. Zallinger, and R. Luz. *Emission Factors from the Model PHEM for the HBEFA Version 3*. Tech. rep. I. 2009, p. 679.
- [97] M. L. Hazelton. “Statistical inference for time varying origin–destination matrices”. In: *Transportation Research Part B: Methodological* 42.6 (2008), pp. 542–552.
- [98] HBEFA. *HBEFA - The Handbook Emission Factors for Road Transport*. 2018. URL: <http://www.hbefa.net/> (visited on Mar. 16, 2018).
- [99] R. Hedayatizadeh, F. A. Salmassi, M. Keshtgari, R. Akbari, and K. Ziarati. “Termite colony optimization: A novel approach for optimizing continuous problems”. In: *Electrical Engineering (ICEE), 2010 18th Iranian Conference on* (2010), pp. 553–558.
- [100] D. Helbing and P. Molnár. “Social force model for pedestrian dynamics”. In: *Phys. Rev. E* 51.5 (May 1995), pp. 4282–4286.
- [101] O. Hertel, S. S. Jensen, M. Hvidberg, M. Ketzel, R. Berkowicz, F. Palmgren, P. Wåhlin, M. Glasius, S. Loft, P. Vinzents, O. Raaschou-Nielsen, M. Sørensen, and H. Bak. “Assessing the Impacts of Traffic Air Pollution on Human Exposure and Health”. In: *Road Pricing, the Economy and the Environment*. Advances in Spatial Science. Springer Berlin Heidelberg, 2008, pp. 277–299.
- [102] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control and artificial intelligence*. MIT press, 1992.
- [103] R. G. Hollands. “Will the real smart city please stand up?” In: *City* 12.3 (Dec. 2008), pp. 303–320.
- [104] R. Holliday and J. E. Pugh. “DNA modification mechanisms and gene activity during development”. In: *Science* 187.4173 (1975), pp. 226–232.
- [105] A. Horni, K. Nagel, and K. Axhausen, eds. *Multi-Agent Transport Simulation MAT-Sim*. London: Ubiquity Press, Aug. 2016, p. 618.
- [106] H. S. Hosseini. “The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm”. In: *International Journal of Bio-Inspired Computation* 1.1/2 (2009), p. 71.
- [107] C.-L. Hwang and K. Yoon. *Multiple attribute decision making: methods and applications a state-of-the-art survey*. Vol. 186. Springer Science & Business Media, 2012.
- [108] IBM. *IBM ILOG CPLEX Optimization Studio*. 2018. URL: <https://www.ibm.com/products/ilog-cplex-optimization-studio> (visited on Mar. 16, 2018).
- [109] L. Int Panis, C. Beckx, S. Broekx, I. De Vlieger, L. Schrooten, B. Degraeuwe, and L. Pelkmans. “PM, NO_x and CO₂ emission reductions from speed management policies in Europe”. In: *Transport Policy* 18.1 (Jan. 2011), pp. 32–37.
- [110] M. R. Jabbarpour, R. M. Noor, and R. H. Khokhar. “Green vehicle traffic routing system using ant-based algorithm”. In: *Journal of Network and Computer Applications* 58 (2015), pp. 294–308.
- [111] E. Jablonka and M. J. Lamb. “Epigenetic inheritance in evolution”. In: *Journal of Evolutionary Biology* 11.2 (1998), pp. 159–183.

- [112] R. Jaenisch and A. Bird. “Epigenetic regulation of gene expression: how the genome integrates intrinsic and environmental signals”. In: *Nature genetics* 33 Suppl.march (Mar. 2003), pp. 245–54.
- [113] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen. “A Survey on Platoon-Based Vehicular Cyber-Physical Systems”. In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 263–284.
- [114] S. Johnson. *Emergence: The connected lives of ants, brains, cities, and software*. Simon and Schuster, 2002.
- [115] D. Karaboga and B. Basturk. “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm”. In: *Journal of global optimization* 39.3 (2007), pp. 459–471.
- [116] J. Kennedy. “Particle Swarm Optimization”. In: *Encyclopedia of Machine Learning*. Ed. by C. Sammut and G. I. Webb. Boston, MA: Springer US, 2010, pp. 760–766.
- [117] J. K. Kim, M. Samaranayake, and S. Pradhan. “Epigenetic mechanisms in mammals”. In: *Cellular and molecular life sciences* 66.4 (2009), pp. 596–612.
- [118] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. In: *Science* 220.4598 (May 1983), pp. 671–680.
- [119] A. Klappenecker, H. Lee, and J. L. Welch. “Finding available parking spaces made easy”. In: *Ad Hoc Networks* 12.1 (2014), pp. 243–249.
- [120] A. D. Kontogianni, E. I. Papageorgiou, and C. Tourkolias. “How do you perceive environmental change? Fuzzy Cognitive Mapping informing stakeholder analysis for environmental policy making and non-market valuation”. In: *Applied Soft Computing* 12.12 (2012), pp. 3725–3735.
- [121] D. Krajzewicz. “Kombination von taktischen und strategischen Einflüssen in einer mikroskopischen Verkehrsflusssimulation”. In: *Fahrermodellierung in Wissenschaft und Wirtschaft, 2. Berliner Fachtagung für Fahrermodellierung* (2009), pp. 104–115.
- [122] D. Krajzewicz. “Traffic Simulation with SUMO – Simulation of Urban Mobility”. In: *Fundamentals of Traffic Simulation*. Ed. by J. Barceló. Vol. 145. International Series in Operations Research & Management Science. Springer New York, 2010, pp. 269–293.
- [123] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. “Recent Development and Applications of SUMO - Simulation of Urban MObility”. In: *International Journal On Advances in Systems and Measurements* 5.3 (2012), pp. 128–138.
- [124] D. Krajzewicz and P. Wagner. “Large-Scale Vehicle Routing Scenarios Based on Pollutant Emissions”. In: *Advanced Microsystems for Automotive Applications*. Ed. by G. Meyer and J. Valldorf. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 237–246.
- [125] S. Krauß. “Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics”. PhD thesis. Deutsches Zentrum fuer Luft- und Raumfahrt. Forschungsberichte, 1998.
- [126] T. Kurczveil, P. Á. López, and E. Schnieder. “Implementation of an Energy Model and a Charging Infrastructure in SUMO”. In: *Simulation of Urban Mobility*. Ed. by M. Behrisch, D. Krajzewicz, and M. Weber. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 33–43.

- [127] S. Kwatirayo, J. Almhana, and Z. Liu. “Adaptive Traffic Light Control using VANET: A case study”. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. July 2013, pp. 752–757.
- [128] T. Larsson, J. T. Lundgren, and A. Peterson. “Allocation of Link Flow Detectors for Origin-Destination Matrix Estimation—A Comparative Study”. In: *Computer-Aided Civil and Infrastructure Engineering* 25.2 (2010), pp. 116–131.
- [129] E. L. Lehmann and G. Casella. *Theory of point estimation*. Springer Science & Business Media, 1998.
- [130] Li En. “Chromatin modification and epigenetic reprogramming in mammalian development”. In: *Nat Rev Genet* 3.9 (Sept. 2002), pp. 662–673.
- [131] B. Li. “Bayesian inference for origin-destination matrices of transport networks using the EM algorithm”. In: *Technometrics* 47.4 (2005).
- [132] C. Li and S. Shimamoto. “A real time traffic light control scheme for reducing vehicles CO2 emissions”. In: *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*. 2011, pp. 855–859.
- [133] L. Li, D. Wen, and D. Yao. “A Survey of Traffic Control With Vehicular Communications”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.1 (Feb. 2014), pp. 425–432.
- [134] S. Lin, B. De Schutter, Y. Xi, and H. Hellendoorn. “Integrated Urban Traffic Control for the Reduction of Travel Delays and Emissions”. In: *Intelligent Transportation Systems, IEEE Transactions on* 14.4 (Dec. 2013), pp. 1609–1619.
- [135] B. Liu, D. Ghosal, C.-N. Chuah, and H. M. Zhang. “Reducing Greenhouse Effects via Fuel Consumption-Aware Variable Speed Limit (FC-VSL)”. In: *Vehicular Technology, IEEE Transactions on* 61.1 (Jan. 2012), pp. 111–122.
- [136] H.-P. Lo and C.-P. Chan. “Simultaneous estimation of an origin–destination matrix and link choice proportions using traffic counts”. In: *Transportation Research Part A: Policy and Practice* 37.9 (2003), pp. 771–788.
- [137] D. Loomis, Y. Grosse, B. Lauby-Secretan, F. El Ghissassi, V. Bouvard, L. Benbrahim-Tallaa, N. Guha, R. Baan, H. Mattock, and K. Straif. “The carcinogenicity of outdoor air pollution”. In: *The Lancet Oncology* 14.13 (2013), pp. 1262–1263.
- [138] J. H. Lorie and L. J. Savage. “Three problems in rationing capital”. In: *The journal of business* 28.4 (1955), pp. 229–239.
- [139] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a new evolutionary computation: advances on estimation of distribution algorithms*. Vol. 192. Springer, 2006.
- [140] M. Maciejewski. “A comparison of microscopic traffic flow simulation systems for an urban area”. In: *Transport Problems* 5.4 (2010).
- [141] M. Maciejewski, J. Bischoff, and K. Nagel. “An Assignment-Based Approach to Efficient Real-Time City-Scale Taxi Dispatching”. In: *IEEE Intelligent Systems* 31.1 (Jan. 2016), pp. 68–77.
- [142] M. Mahmood, B. van Arem, R. Pueboobpaphan, and R. de Lange. “Reducing local traffic emissions at urban intersection using ITS countermeasures”. In: *Intelligent Transport Systems, IET* 7.1 (Mar. 2013), pp. 78–86.

- [143] D. Mange and M. Tomassini. *Bio-inspired computing machines: Towards novel computational architectures*. PPUR presses polytechniques, 1998.
- [144] MATSim. *MATSim - Multi-agent Transport Simulation*. 2018. URL: <https://www.matsim.org/> (visited on Mar. 16, 2018).
- [145] maxCT. *maxCT: Imagine a new era in Smart Mobility*. 2014. URL: <http://maxct.lcc.uma.es/> (visited on Mar. 16, 2018).
- [146] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133.
- [147] D. Mckenney and T. White. “Distributed and Adaptive Traffic Signal Control Within a Realistic Traffic Simulation”. In: *Engineering Applications of Artificial Intelligence* 26.1 (Jan. 2013), pp. 574–583.
- [148] C. Menelaou, P. Kolios, S. Timotheou, and C. Panayiotou. “A congestion-free vehicle route reservation architecture”. In: *2016 18th Mediterranean Electrotechnical Conference (MELECON)*. April. IEEE, Apr. 2016, pp. 1–6.
- [149] H. Mizuta, Y. Yamagata, and H. Seya. “Large-scale Traffic Simulation for Low-carbon City”. In: *Proceedings of the Winter Simulation Conference*. WSC ’12. Winter Simulation Conference, 2012, 162:1–162:12.
- [150] N. Mladenović and P. Hansen. “Variable neighborhood search”. In: *Computers & operations research* 24.11 (1997), pp. 1097–1100.
- [151] J.-B. Monet de Lamarck. *Recherches sur l’organisation des corps vivants: précédé du Discours d’ouverture du cours de zoologie donné dans la Muséum d’histoire naturelle*. Fayard, 1802.
- [152] R. J. Moraga, G. W. Depuy, and G. E. Whitehouse. “Meta-RaPS approach for the 0-1 multidimensional knapsack problem”. In: *Computers and Industrial Engineering* 48.1 (2005), pp. 83–96.
- [153] moveON. *moveON: Metaheuristics, Holistic Intelligence, and Smart Mobility*. 2015. URL: <http://moveon.lcc.uma.es/> (visited on Mar. 16, 2018).
- [154] A. Mucherino and O. Seref. “Monkey search: A novel metaheuristic search for global optimization”. In: *AIP Conference Proceedings* 953.2007 (2007), pp. 162–173.
- [155] K. G. Murty. *Linear programming*. John Wiley & Sons, Inc., New York, 1983, pp. xix+482.
- [156] National Statistics of Scotland. *Mid-Year Population Estimates Scotland*. Tech. rep. National Statistics publication for Scotland (U.K. 2017), 2017.
- [157] D. Nätt, N. Lindqvist, H. Stranneheim, J. Lundeberg, P. A. Torjesen, and P. Jensen. “Inheritance of acquired behaviour adaptations and brain gene expression in chickens”. In: *PLoS ONE* 4.7 (2009).
- [158] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano. “Current trends in Smart City initiatives: Some stylised facts”. In: *Cities* 38 (June 2014), pp. 25–36.
- [159] J. New, D. Castro, and M. Beckwith. *How National Governments Can Help Smart Cities Succeed*. Tech. rep. Washington, DC, USA: Center for Data Innovation, 2017.

- [160] Y. M. Nie and H. M. Zhang. “A variational inequality formulation for inferring dynamic origin–destination travel demands”. In: *Transportation Research Part B: Methodological* 42.7 (2008), pp. 635–662.
- [161] Norfolk County Council. *Norfolk County Council Live Car Park data*. 2018. URL: <https://data.gov.uk/dataset/norfolk-county-council-live-car-park-data/> (visited on Mar. 16, 2018).
- [162] Nottingham City Council. *Open Data Nottingham – Car park occupancy*. 2018. URL: <http://www.opendatanottingham.org.uk/dataset.aspx?id=55> (visited on Mar. 16, 2018).
- [163] OECD/IEA. *CO2 Emissions from Fuel Combustion 2012*. Tech. rep. International Energy Agency, 2012.
- [164] Office for National Statistics. *Population Estimates for UK, England and Wales, Scotland and Northern Ireland*. Tech. rep. U.K. 2016, 2016.
- [165] M. O’Grady and G. O’Hare. “How Smart Is Your City?” In: *Science* 335.6076 (2012), pp. 1581–1582.
- [166] R. Ohlsson, K. Hall, and M. Ritzén. *Genomic imprinting: causes and consequences*. Cambridge University Press, 1995.
- [167] J. G. J. Olivier, G. Janssens-Maenhout, M. Muntean, and J. A. H. W. Peters. *Trends in global co2 emissions 2013 report*. Tech. rep. The Hague: Netherlands Environmental Assessment Agency, Institute for Environment and Sustainability (IES) of the European Commission’s Joint Research Centre, 2013.
- [168] Open Knowledge International. *Creative Commons Attribution License (cc-by)*. 2018. URL: <https://opendefinition.org/licenses/cc-by/> (visited on Mar. 16, 2018).
- [169] OpenStreetMap Foundation. *OpenStreetMap*. 2018. URL: <http://www.openstreetmap.org/> (visited on Mar. 16, 2018).
- [170] L. G. Papaleondiou and M. D. Dikaiakos. “TrafficModeler: A Graphical Tool for Programming Microscopic Traffic Simulators through High-Level Abstractions”. In: *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*. Apr. 2009, pp. 1–5.
- [171] R. Paton. *Computing with biological metaphors*. Chapman & Hall, 1994.
- [172] S. Periyasamy, A. Gray, and P. Kille. “The Epigenetic Algorithm”. In: *2008 IEEE Congress on Evolutionary Computation*. 2008, pp. 3228–3236.
- [173] R. H. A. Plasterk. “RNA silencing: the genome’s immune system”. In: *Science* 296.5571 (2002), pp. 1263–1265.
- [174] PTV. *PTV Group*. 2018. URL: <https://www.ptvgroup.com/> (visited on Mar. 16, 2018).
- [175] J. Puchinger, G. R. Raidl, and U. Pferschy. “The multidimensional knapsack problem: Structure and algorithms”. In: *Informatics Journal on Computing* 22.2 (2010), pp. 250–265.
- [176] J. B. Rawlings and D. Q. Mayne. *Postface to "Model Predictive Control: Theory and Design"*. Nob Hill Pub., 2012.
- [177] G. W. Redberry. *Gene silencing: new research*. Nova Publishers, 2006.

- [178] C. R. Reeves. “Modern heuristic techniques for combinatorial problems. Advanced topics in computer science”. In: *Modern Heuristic Techniques for Combinatorial Problems: Advanced Topics in Computer Science* (1995).
- [179] W. Reik, W. Dean, and J. Walter. “Epigenetic reprogramming in mammalian development”. In: *Science* 293.5532 (2001), pp. 1089–1093.
- [180] D. Rieck, B. Schünemann, and I. Radusch. “Advanced Traffic Light Information in OpenStreetMap for Traffic Simulations”. In: *Modeling Mobility with Open Data: 2nd SUMO Conference 2014 Berlin, Germany, May 15-16, 2014*. Ed. by M. Behrisch and M. Weber. Cham: Springer International Publishing, 2015, pp. 25–34.
- [181] A. D. Riggs. “X inactivation, differentiation, and DNA methylation”. In: *Cold Spring Harbor Monograph Archive* 32 (1996), pp. 646–662.
- [182] A. D. Riggs and T. N. Porter. “Overview of epigenetic mechanisms”. In: *Cold Spring Harbor Monograph Archive* 32 (1996), pp. 29–45.
- [183] roadME. *roadME: Fundamentals for Real World Applications of Metaheuristics: The vehicular case*. 2011. URL: <http://roadme.lcc.uma.es/> (visited on Mar. 16, 2018).
- [184] CI-RTI. *Research Thematic Network on Smart Cities*. 2018. URL: <http://www.cirti.es/> (visited on Mar. 16, 2018).
- [185] T. Saber, A. Ventresque, and J. Murphy. “ROThAr: Real-Time On-Line Traffic Assignment with Load Estimation”. In: *Proceedings of the 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications. DS-RT '13*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 79–86.
- [186] E. Salomons, D. van Maercke, J. Defrance, and F. de Roo. “The Harmonoise Sound Propagation Model”. In: *Acta Acustica united with Acustica* 97 (2011), pp. 62–74.
- [187] K. D. Sarge and O.-K. Park-Sarge. “Mitotic bookmarking of formerly active genes: keeping epigenetic memories from fading”. In: *Cell cycle* 8.6 (2009), pp. 818–823.
- [188] M. Shahidehpour, Z. Li, S. Bahramirad, and A. Khodaei. “Optimizing Traffic Signal Settings in Smart Cities”. In: *IEEE Transactions on Smart Grid* 3053.4 (2016), pp. 1–12.
- [189] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.
- [190] F. Simonelli, V. Marzano, A. Papola, and I. Vitiello. “A network sensor location procedure accounting for o–d matrix estimate variability”. In: *Transportation Research Part B: Methodological* 46.10 (2012), pp. 1624–1638.
- [191] L. Simó-Riudalbas and M. Esteller. “Targeting the histone orthography of cancer: drugs for writers, erasers and readers”. In: *British journal of pharmacology* 172.11 (2015), pp. 2716–2732.
- [192] M. K. Skinner. “A new kind of inheritance”. In: *Scientific American* 311.2 (2014), pp. 44–51.
- [193] M. K. Skinner, M. Manikkam, and C. Guerrero-Bosagna. “Epigenetic transgenerational actions of environmental factors in disease etiology”. In: *Trends in Endocrinology & Metabolism* 21.4 (2010), pp. 214–222.
- [194] R. E. Smith. “An investigation of diploid genetic algorithms for adaptive search of nonstationary functions”. PhD thesis. 1988.

- [195] J. A. Sousa and E. Costa. “Designing an Epigenetic Approach in Artificial Life: The EpiAL Model”. In: *Agents and Artificial Intelligence* (2011), pp. 78–90.
- [196] K. E. Stecke. “Using Mathematics to Solve Some Problems in Industry”. In: *INFORMS Transactions on Education* 5.2 (Jan. 2005), pp. 1–8.
- [197] D. H. Stolfi and E. Alba. “An Evolutionary Algorithm to Generate Real Urban Traffic Flows”. In: *Advances in Artificial Intelligence*. Ed. by J. M. Puerta, J. A. Gámez, B. Dorronsoro, E. Barrenechea, A. Troncoso, B. Baroque, and M. Galar. Vol. 9422. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 332–343.
- [198] D. H. Stolfi and E. Alba. “Computing New Optimized Routes for GPS Navigators Using Evolutionary Algorithms”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’17. Berlin, Germany: ACM, 2017, pp. 1240–1247.
- [199] D. H. Stolfi and E. Alba. “Eco-friendly Reduction of Travel Times in European Smart Cities”. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’14. Vancouver, BC, Canada: ACM, 2014, pp. 1207–1214.
- [200] D. H. Stolfi and E. Alba. “Epigenetic algorithms: A New way of building GAs based on epigenetics”. In: *Information Sciences* 424. Supplement C (2018), pp. 250–272.
- [201] D. H. Stolfi and E. Alba. “Generating realistic urban traffic flows with evolutionary techniques”. In: *Engineering Applications of Artificial Intelligence* 75 (2018), pp. 36–47.
- [202] D. H. Stolfi and E. Alba. “Green Swarm: Greener routes with bio-inspired techniques”. In: *Applied Soft Computing* 71 (2018), pp. 952–963.
- [203] D. H. Stolfi and E. Alba. “Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms”. In: *Applied Soft Computing* 24 (2014), pp. 181–195.
- [204] D. H. Stolfi and E. Alba. “Red Swarm: Smart Mobility in Cities with EAS”. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’13. Amsterdam, The Netherlands: ACM, 2013, pp. 1373–1380.
- [205] D. H. Stolfi and E. Alba. “Reducing Gas Emissions in Smart Cities by Using the Red Swarm Architecture”. In: *Advances in Artificial Intelligence*. Ed. by C. Bielza, A. Salmerón, A. Alonso-Betanzos, J. Hidalgo, L. Martínez, A. Troncoso, E. Corchado, and J. Corchado. Vol. 8109. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 289–299.
- [206] D. H. Stolfi and E. Alba. “Smart Mobility Policies with Evolutionary Algorithms: The Adapting Info Panel Case”. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’15. Madrid, Spain: ACM, 2015, pp. 1287–1294.
- [207] D. H. Stolfi and E. Alba. “Sustainable Road Traffic Using Evolutionary Algorithms”. In: *Sustainable Transportation and Smart Logistics*. Elsevier, 2018, In press.
- [208] D. H. Stolfi and E. Alba. “Un Algoritmo Evolutivo para la Reducción de Tiempos de Viaje y Emisiones Utilizando Paneles LED”. In: *X Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*. MAEB 2015. Mérida - Almendralejo, 2015, pp. 27–34.

- [209] D. H. Stolfi, E. Alba, and X. Yao. “Can I Park in the City Centre? Predicting Car Park Occupancy Rates in Smart Cities”. In: *Journal of Urban Technology* (2018), Minor review.
- [210] D. H. Stolfi, E. Alba, and X. Yao. “Predicting Car Park Occupancy Rates in Smart Cities”. In: *Smart Cities: Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings*. Ed. by E. Alba, F. Chicano, and G. Luque. Cham: Springer International Publishing, 2017, pp. 107–117.
- [211] D. H. Stolfi, R. Armas, E. Alba, H. Aguirre, and K. Tanaka. “Fine Tuning of Traffic in Our Cities with Smart Panels: The Quito City Case Study”. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. GECCO '16. Denver, Colorado, USA: ACM, 2016, pp. 1013–1019.
- [212] D. H. Stolfi, C. Cintrano, F. Chicano, and E. Alba. “Natural Evolution Tells Us How to Best Make Goods Delivery: Use Vans”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '18. Kyoto, Japan: ACM, 2018, pp. 308–309.
- [213] R. Storn and K. Price. “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces”. In: *Journal of global optimization* 11.4 (1997), pp. 341–359.
- [214] C. A. Sugar. “Techniques for clustering and classification with applications to medical problems”. PhD thesis. Stanford University, 1998.
- [215] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa. “Traffic jams without bottlenecks: experimental evidence for the physical mechanism of the formation of a jam”. In: *New Journal of Physics* 10.3 (Mar. 2008), p. 33001.
- [216] SUMO. *SUMO - Simulation of Urban MObility*. 2018. URL: <http://dlr.de/ts/sumo/> (visited on Mar. 16, 2018).
- [217] R. Suri. “An overview of evaluative models for flexible manufacturing systems”. In: *Annals of Operations Research* 3.1 (Jan. 1985), pp. 13–21.
- [218] H. Szu and R. Hartley. “Fast simulated annealing”. In: *Physics Letters A* 122.3-4 (June 1987), pp. 157–162.
- [219] I. Tanev and K. Yuta. “Epigenetic programming: Genetic programming incorporating epigenetic learning through modification of histones”. In: *Information Sciences* 178.23 (2008), pp. 4469–4481.
- [220] D. Teodorović, P. Lucic, G. Markovic, and M. Dell’Orco. “Bee Colony Optimization: Principles and Applications”. In: *8th Seminar on Neural Network Applications in Electrical Engineering* 00 (2006), pp. 151–156.
- [221] The Local. *Madrid activates anti-pollution measures as air contamination spikes*. 2016. URL: <https://www.thelocal.es/20161227/madrid-activates-anti-pollution-measures-as-air-contamination-spikes> (visited on Feb. 1, 2018).
- [222] The National Archives. *Open Government Licence*. 2018. URL: <http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/> (visited on Mar. 16, 2018).

- [223] P. Thomin, A. Gibaud, and P. Koutcherawy. "Deployment of a Fully Distributed System for Improving Urban Traffic Flows: A Simulation-based Performance Analysis". In: *Simulation Modelling Practice and Theory* 31 (Feb. 2013), pp. 22–38.
- [224] TNS Opinion & Social. *Attitudes of Europeans towards urban mobility*. Tech. rep. June. 2013.
- [225] J. Toutouh and E. Alba. "Performance analysis of optimized VANET protocols in real world tests". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. IEEE. 2011, pp. 1244–1249.
- [226] TRANSIMS. *TRANSIMS - Transportation Analysis and Simulation*. 2018. URL: <https://sourceforge.net/projects/transims/> (visited on Mar. 16, 2018).
- [227] M. Tropea and A. F. Santamaria. "Vehicular traffic optimization in VANETs: A proposal for nodes re-routing and congestion reduction". In: *Advances in Electrical and Electronic Engineering* 13.4 (2015), pp. 376–385.
- [228] L.-Y. Tseng and C. Chen. "Multiple trajectory search for large scale global optimization". In: *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE. 2008, pp. 3052–3059.
- [229] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang. "Wireless sensor networks in intelligent transportation systems". In: *Wireless Communications and Mobile Computing* 9.3 (Mar. 2009), pp. 287–302.
- [230] United Nations. *World Urbanization Prospects The 2011 Revision*. Tech. rep. 2011, p. 318.
- [231] Y. Vimont, S. Boussier, and M. Vasquez. "Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem". In: *Journal of Combinatorial Optimization* 15.2 (2008), pp. 165–178.
- [232] Vissim. *PTV Vissim*. 2018. URL: <http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/> (visited on Mar. 16, 2018).
- [233] F. Viti, M. Rinaldi, F. Corman, and C. M. J. Tampère. "Assessing partial observability in network sensor location problems". In: *Transportation Research Part B: Methodological* 70 (2014), pp. 65–89.
- [234] J. G. Wardrop. "Some Theoretical Aspects of Road Traffic Research". In: *Proc. Inst. Civil Eng.* 1952, pp. 252–378.
- [235] D. Washburn, U. Sindhu, S. Balaouras, R. A. Dines, N. M. Hayes, and L. E. Nelson. "Helping CIOs understand "Smart City" initiatives: defining the smart city, its drivers, and the role of the CIO". In: *Cambridge, MA: Forrester Research, Inc* (2010).
- [236] A. Wegener, H. Hellbrück, C. Wewetzer, and A. Lübke. "VANET Simulation Environment with Feedback Loop and its Application to Traffic Light Assistance". In: *GLOBECOM Workshops, 2008 IEEE*. IEEE. Ieee, Nov. 2008, pp. 1–7.
- [237] A. Wegener and M. Piórkowski. "TraCI: An Interface for Coupling Road Traffic and Network Simulators". In: *Proceedings of the 11th Communications and Networking Simulation Symposium*. CNS '08. New York, NY, USA: ACM, 2008, pp. 155–163.
- [238] R. Wiedemann. "Modelling of RTI-Elements on multi-lane roads". In: *Drive Conference (1991: Brussels, Belgium)*. Vol. 2. 1991.

- [239] R. Wiedemann. “Simulation des Straßenverkehrsflusses. Schriftenreihe Heft 8”. In: *Institute for Transportation Science, University of Karlsruhe, Germany* (1994).
- [240] T. Wiedmann and J. Minx. *A Definition of 'Carbon Footprint'*. Tech. rep. Durham, United Kindom: ISA UK Research & Consulting, 2007, p. 9.
- [241] E. Winsberg. “Computer simulations in science”. In: *Stanford Encyclopedia of Philosophy* (2015).
- [242] A. P. Wolffe and M. A. Matzke. “Epigenetics: Regulation Through Repression”. In: *Science* 286.5439 (1999), pp. 481–486.
- [243] D. H. Wolpert and W. G. Macready. “No Free Lunch Theorems for Optimization”. In: *Trans. Evol. Comp* 1.1 (1997), pp. 67–82.
- [244] H. Yang and J. Zhou. “Optimal traffic counting locations for origin–destination matrix estimation”. In: *Transportation Research Part B: Methodological* 32.2 (1998), pp. 109–126.
- [245] X.-s. Yang. “A New Metaheuristic Bat-Inspired Algorithm”. In: *Nicso 2010* (2010), pp. 65–74.
- [246] X.-S. Yang and S. Deb. “Cuckoo Search via Levy Flights”. In: *World Congress on Nature & Biologically Inspired Computing* (2009), pp. 210–214.
- [247] X.-S. Yang. “Firefly Algorithms for Multimodal Optimization”. In: *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications* (2009), pp. 169–178.
- [248] X. Yao. “A new simulated annealing algorithm”. In: *International Journal of Computer Mathematics* 56.3-4 (Jan. 1995), pp. 161–168.
- [249] N. A. Youngson and E. Whitelaw. “Transgenerational Epigenetic Effects”. In: *Annual Review of Genomics and Human Genetics* 9.1 (Sept. 2008), pp. 233–257.
- [250] S. K. Zaidi, D. W. Young, M. A. Montecino, J. B. Lian, A. J. Van Wijnen, J. L. Stein, and G. S. Stein. “Mitotic bookmarking of genes: a novel dimension to epigenetic control”. In: *Nature Reviews Genetics* 11.8 (2010), pp. 583–589.
- [251] S. K. Zegeye, B. De Schutter, H. Hellendoorn, and E. Breunese. “Reduction of Travel Times and Traffic Emissions Using Model Predictive Control”. In: *2009 American Control Conference* (2009), pp. 5392–5397.
- [252] Zhejun Gong. “Estimating the urban OD matrix: A neural network approach”. In: *European Journal of Operational Research* 106.1 (1998), pp. 108–115.
- [253] Y. Zheng, S. Rajasegarar, and C. Leckie. “Parking availability prediction for sensor-enabled car parks in smart cities”. In: *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. 2015, pp. 1–6.
- [254] Zong Woo Geem, Joong Hoon Kim, and G. Loganathan. “A New Heuristic Optimization Algorithm: Harmony Search”. In: *SIMULATION* 76.2 (Feb. 2001), pp. 60–68.