Reactive Evolutionary Path Planning For Autonomous Surface Vehicles in Lake Environments

Mario Eduardo Arzamendia López

Supervisors: Sergio Toral Marín
Daniel Gutiérrez Reina
Derlis Orlando Gregor Recalde

Department of Electronic Engineering
Higher Technical School of Engineering
University of Seville

This dissertation is submitted in partial fulfillment for the degree of
Doctor of Philosophy in Automatic, Electronic and Telecommunication Engineering

December 2018

Tesis Doctoral:	Reactive Evolutionary Path Planning For Autonomous Surface Vehicles in Lake Environments

Autor:	Mario Eduardo Arzamendia López
Directores:	Sergio Toral Marín
	Daniel Gutiérrez Reina
	Derlis Orlando Gregor

El tribunal nombrado para juzgar la Tesis arriba indicada, compuesto por los siguientes doctores:

Presidente:

Vocales:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mis padres, Mario y Ana,
por su apoyo a lo largo del tiempo en los distintos lugares que me ha tocado estar

A mi esposa Mónica,
compañera inseparable en esta aventura

A mis hijas, Itziar y Victoria,
luces de mi vida

Agradecimientos

Me gustaría comenzar haciendo unos agradecimientos institucionales, a la Fundación Carolina por haberme otorgado la beca para realizar el curso de doctorado en Sevilla; a la Facultad de la Universidad Nacional de Asunción (FIUNA), por haberme otorgado el permiso para viajar a realizarlo; y al Consejo Nacional de Ciencia y Tecnologia (CONACYT), por su esfuerzo en elevar la investigación y la calidad de las universidades de mi país mediante sus programas de incentivo al investigador (PRONII) y proyectos de investigación (PROCIENCIA).

A mis tutores, Dr. Daniel Gutiérrez Reina, Dr. Sergio Toral Marín y Dr. Derlis Orlando Gregor, por su orientación cercana y continua durante este tiempo, realmente ha sido un privilegio haber trabajado bajo la supervisión de ellos. Pero por sobretodo agradezco la exigencia para poder sacar adelante mi trabajo de la mejor manera. A mis compañeros del grupo de investigación ACETI, Dr. Federico Barrero, Jesus Sanchez García, José Manuel García, Ignacio González, Mario Bermudez y Cristina Martín, gracias por su compañerismo. También a mis compañeros de sala en la Universidad, Miguel Cabello y Raúl Roman, con quienes compartí mucho tiempo en la escuela. Tampoco quiero olvidarme de mis colegas de FIUNA, el Dr. Raul Gregor y el Dr. Jorge Rodas, quienes me recibieron en mis inicios de investigador en esta institución y fueron los primeros en hablarme e incentivarme a realizar este curso que ahora me encuentro concluyendo.

Luego, me gustaría hacer unos agradecimientos especiales a estas personas que conocí durante estos casi dos años en Sevilla. A Miguel Angel Casamitjana y su familia, por haber sido como de nuestra propia familia, por toda la ayuda recibida; a mis amigos Esteban Guerrero, José Miguel Pascual, Jaime Orejas y Don Juan José Constantino. También mis agradecimientos para Doña Rosario Bustillo y sus hijos Daniel Aviles y Simon Aviles, en cuyo piso vivimos todo este tiempo y quienes siempre nos trataron de la mejor manera. Estoy también muy agradecido al colegio Ángela Guerrero y a las Hermanas de la Cruz, por haberles permitido a mis hijas estudiar con ellas (la verdad fue una de las cosas que mas siento de haber dejado en Sevilla).

Finalmente, mis agradecimientos a mi familia, a mis padres y hermanos Juan Pablo, Rodrigo, y Arturo, por el infinito apoyo; como a mi familia política, a Don Rogelio, Doña Mercedes y todos sus hijos, por el cariño de siempre y por acompañarnos de cerca estando tan lejos.

Para terminar, a mi esposa Mónica y a mis queridas hijas Itziar y Victoria, por haber estado allí conmigo, estuvieron más que a la altura del desafío. Gracias. Las amo.

Y a Dios, que a pesar de haberlo hecho muy difícil, no lo hizo imposible.

Abstract

Autonomous Surface Vehicles (ASVs) have found a lot of promising applications in aquatic environments, i.e., sea, lakes, rivers, etc. They can be used for applications of paramount importance, such as environmental monitoring of water resources, and for bathymetry to study the characteristics of the basing of a lake/sea or for surveillance in patrol missions, among others. These vehicles can be built with smaller dimensions when compared to regular ships since they do not need an on-board crew for operation. However, they do require at least a telemetry control as well as certain intelligence for making decisions and responding to changing scenarios.

Water resources are very important in Paraguay since they provide fresh water for its inhabitants and they are crucial for the main economic activities such as agriculture and cattle raising. Furthermore, they are natural borders with the surrounding countries, and consequently the main transportation route for importing/exporting products. In fact, Paraguay is the third country in the world with the largest fleet of barges after USA and China. Thus, maintaining and monitoring the environmental conditions of these resources is key in the development of the country. This work is focused on the maintenance and monitoring of the greatest lake of the country called Ypacarai Lake. In recent years, the quality of its water has been seriously degraded due to the pollution caused by the low control of the dumping of waste thrown into the Lake. Since it is also a national icon, the government of Paraguay has put a lot of effort in recovering water quality of the Lake. As a result, it is monitored periodically but using manual procedures. Therefore, the primary objective of this work is to develop these monitoring tasks autonomously by means of an ASV with a suitable path planning strategy. Path planning is an active research area in robotics. A particular case is the Coverage Path Planning (CPP) problem, where an algorithm should find a path that achieves the best coverage of the target region to be monitored. This work mainly studies the global CPP, which returns a suitable path considering the initial conditions of the environment where the vehicle moves.

The first contribution of this thesis is the modeling of the CPP using Hamiltonian Circuits (HCs) and Eulerian Circuits (ECs). Therefore, a graph adapted to the Ypacarai Lake is created by using a network of wireless beacons located at the shore of the lake, so that they can be used as data exchange points between a control center and the ASV, and also as waypoints.

Regarding the proposed modeling, HCs and ECs are paths that begin and end at the same point. Therefore, the ASV travels across a given graph that is defined by a set of wireless beacons. The main difference between HC and EC is that a HC is a tour that visits each vertex only once while EC visits each edge only once.

Finding optimal HCs or ECs that minimize the total distance traveled by the ASV are very complex problems known as NP-complete. To solve such problems, a meta-heuristic algorithm can be a suitable approach since they provide quasi-optimal solutions in a reasonable time. In this work, a GA (Genetic Algorithm) approach is proposed and tested. First, an evaluation of the performance of the algorithm with different values of its hyper-parameters has been carried out. Second, the proposed approach has been compared to other approaches such as randomized and greedy algorithms. Third, a thorough comparison between the performance of HC and EC based approaches is presented. The simulation results show that EC-based approach outperforms the HC-based approach almost 2% which in terms of the Lake size is about 1.4 km^2 or 140 ha (hectares). Therefore, it has been demonstrated that the modeling of the problem as an Eulerian graph provides better results. Furthermore, it has been investigated the relationship between the number of beacons to be visited and the distance traveled by the ASV in the EC-based approach. Findings indicate that there is a quasi-lineal relationship between the number of beacons and the distance traveled.

The second contribution of this work is the development of an on-line learning strategy using the same model but considering dynamic contamination events in the Lake. Dynamic events mean the appearance and evolution of an algae bloom, which is a strong indicator of the degradation of the lake. The strategy is divided into two-phases, the initial exploration phase to discover the presence of the algae bloom and next the intensification phase to focus on the region where the contamination event is detected. This intensification effect is achieved by modifying the beacon-based graph, reducing the number of vertices and selecting those that are closer to the region of interest. The simulation results reveal that the proposed strategy detects two events and monitors them, keeping a high level of coverage while minimizing the distance traveled by the ASV. The proposed scheme is a reactive path planning that adapts to the environmental conditions. This scheme makes decisions in an autonomous way and it switches from the exploratory phase to the intensification phase depending on the external conditions, leading to a variable granularity in the monitoring task. Therefore, there is a balance between coverage and the energy consumed by the ASV. The main benefits obtained from the second contribution includes a better monitoring in the quality of water and control of waste dumping, and the possibility to predict the appearance of algae-bloom from the collected environmental data.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AB	Algae Bloom
ADP	Adaptive Path Planning
ASC	Autonomous Surface Craft
ASV	Autonomous Surface Vehicle
BCD	Boustrophedon Cellular Decomposition
CD	Cell Decomposition
CI	Collected Information
COLREG	Collision Avoidance Regulation
CPP	Coverage Path Planning
CRW	Cooperative Robotic Watercraft
CPP	Degree of Freedom
DP	Death Penalty
DR	Desired Region
EC	Eulerian Circuit
FMM	Fast Marching Method
FSM	Fast Square Marching Method
FR	Forbidden Region

GNC	Guidance, Navigation and Control
GP	Gaussian Process
GA	Genetic Algorithm
HAB	Harmful Algal Bloom
HC	Hamiltonian Circuit
ILS	Iterated Local Search
IMO	International Maritime Organization
IMU	Inertial Motion Unit
NN	Neural Network
NP	Non-deterministic Polynomial time
PF	Penalty Factor
PSO	Penalty Factor
RPP	Robot Path Planning
SAR	Search and Rescue
TS	Tabu Search
TSP	Travelling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
VD	Voronoi Diagram
VG	Visibility Graph
VRP	Vehicle Routing Problem

Chapter 1

Introduction

1.1 Backgrounds and Motivation

Ypacarai Lake is the greatest inland freshwater of Paraguay [1], located 25 km east from the Capital city, Asunción (Figure 1.1). According to the latest bathymetry studies performed by the Hydroinformatics International Center of the Itaipu Dam, the area of the lake is about 64 km^2 [2]. The lake has always been a national icon and a tourist attraction point, especially in summer time. Three towns namely San Bernardino, Aregua and Ypacarai are located nearby. Thus, the Lake is the main fresh water supply for their population. It also has an ecological value due to the wetlands that surround the lake. However, the waste from these towns is thrown out into it without an adequate treatment to minimize the negative impact on the lake. Additionally, the waste from other towns also reaches the water through streams drain. Uncontrolled dumping of waste leads to an excess of nutrients in the lake, i.e. phosphorus and nitrogen, being this a process known as eutrophication. In addition, common fertilizers used in agriculture activities are also responsible for the eutrophication.

One of the main consequences of the eutrophication is the appearance of green-blue algae-bloom. In fact, these blooms are made of colonies of cyanobacteria, which is a type of bacteria that obtains their energy from photosynthesis. More specifically, there is a group of cyanobacteria called Harmful Algae Bloom (HAB) that creates low-oxygen conditions (hypoxia, anoxia) in water and also produces cyanotoxins which represent a risk for animals and human beings [3]. The Microcystis species and Anabaena species are two types of cyanobacteria that produce the microcystin toxin, which affects negatively the human liver. There have been several severe cases of algae bloom in Lakes around the world, for example, in Lake Victoria (Africa) [4], Lake Erie (USA) [5], Taihu Lake (China) [6], among others. In the case of Lake Erie, one of the great lakes located at the north of USA, it suffered a blooming during 2011, reaching a peak of 5,000 km^2 (of the 25,662 km^2 of the lake). A satellite image of the lake is

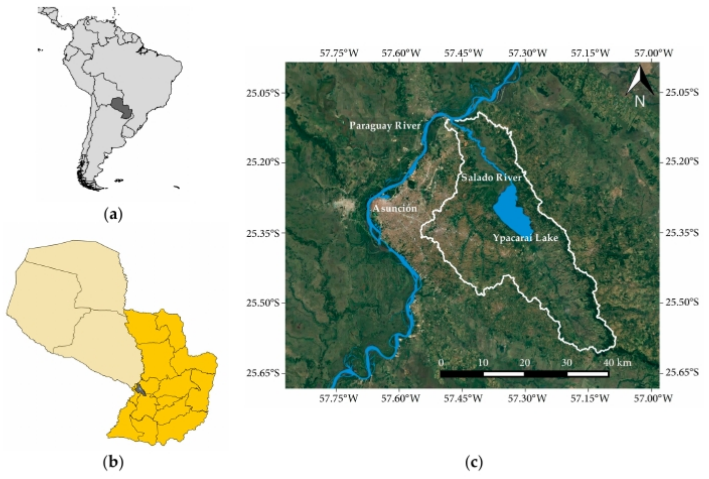


Fig. 1.1 Location of Paraguay in South America (a), the Ypacarai Lake inside Paraguay (b) and the connection of the Ypacarai Lake to the Paraguay River through the Salado River(c)

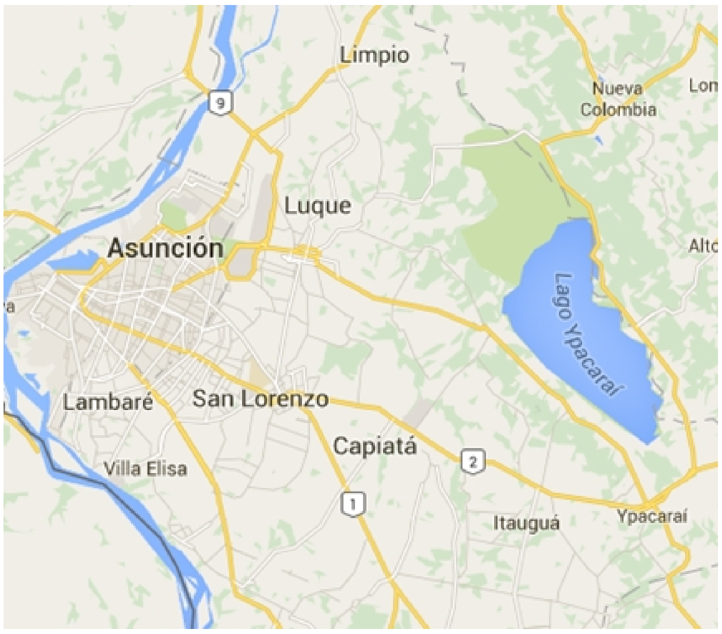


Fig. 1.2 Location of Ypakarai Lake

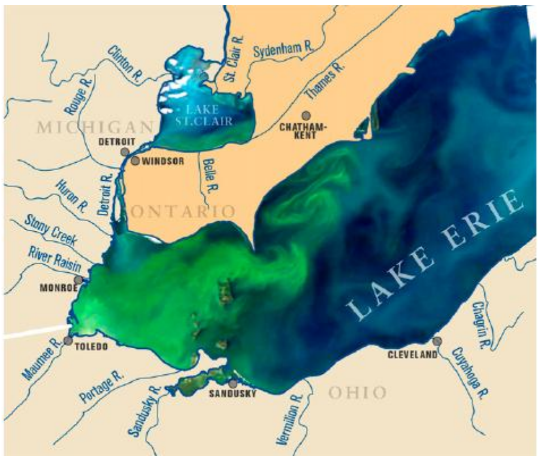


Fig. 1.3 Algal Bloom at Lake Erie (2011)

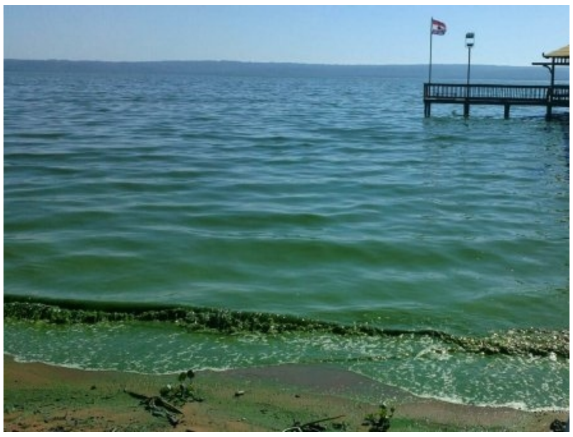


Fig. 1.4 Algal Bloom at Ypakarai Lake (2013)

shown in Figure 1.3 [5]. Levels of microcystin were found between 0.1 $\mu g/L$ (micrograms per liter) and 8.7 $\mu g/L$. It is important to highlight that the World Health Organization (WHO) recommends the exposure up to 1 $\mu g/L$. In [5] it is concluded that trends in agricultural practices, increased precipitation, weak lake circulation, and quiescent conditions yielded to the record-breaking blooming. In Taihu Lake (2,338 km^2), the main water repository for 10 million people, the levels of microcystin reached up to 44 $\mu g/L$ in the water [6]. According to [6], a prediction system was developed with the help of 18 fixed monitoring systems and periodic manual sampling. The fixed stations contained sensors for water temperature, turbidity, PH, chlorophyll a, phycocyanin, conductivity, dissolved oxygen, and other meteorological variables. From all these sensors, the phycocyanin sensor is the main indicator of the presence of cyanobacteria in the lake. The predicted model included algae biomass (i.e. chlorophyll a concentration), wind velocity, and precipitation. The effectiveness of the model was compared to satellite image data, resulting in an accuracy of 82.2±11.7%. Regarding Ypacarai Lake, the algae bloom has appeared periodically in an aggressive form as shown in Figure 1.4 [7], giving not only the characteristic stains of green color but also a fetid smell. Recent studies in 2015 [8] revealed the presence of cyanobacteria Microsystys aeroginusa. To mitigate the problem, the government of Paraguay has performed a manual monitoring campaign between 2015 and 2016 where 12 studies were carried out about the conditions of the lake. In addition, it has been installed three fixed monitoring stations. The prediction of the appearance and dynamics of the algae bloom is a rather difficult task [9]. There are two main aspects responsible for this complexity. First, there are several uncontrolled factors [10], such as temperature and eutrophicated state that contribute the bloom. Therefore, it is difficult to predict the exact moment and place when the bloom is occurring. Second, the algae bloom is a dynamic phenomenon that evolves and modifies its own characteristics (size) in time.

Under these circumstances, the use of an Autonomous Surface Vehicles (ASV)[1] can improve the sampling and monitoring of the water autonomously. In contrast to manual monitoring, it does not require people on the field, it is cheaper in fuel expenses and more ecologically friendly because it can use electrical motors powered with batteries feed by solar panels instead of fuel motors.

One important aspect of the autonomous operation is the guidance, navigation and control system of the vehicle. The guidance system is responsible for the path planning, which determines which path the ASV will follow to achieve a planned mission. On this line, this thesis evaluates the current techniques for path planning for autonomous vehicles and particularly ASV. Furthermore, it proposes a new strategy for exploring the lake, performing more detailed monitoring in case an algae bloom event is detected.

1.2 Objectives

The objectives of the thesis are showing next:

General Objective

Developing a new path planning strategy for an ASV that monitors the quality of the water of the Ypacarai Lake with an autonomous adaptive re-planning and considering the dynamic conditions of the environment.

Specific Objectives

- The proposal of a new approach for solving the path planning problem by modeling the path using Hamiltonian Circuits (HCs) and Eulerian Circuits (ECs) as well as graph theory.
- The evaluation of metaheuristics techniques and specifically GA as an alternative to solve the high complexity problem of determining the best path that maximizes the distance traveled by the ASV.
- The analysis of the performance of a reactive path planning strategy that explores the conditions of the lake and adjusts its granularity in case of algae-bloom events.

[1]Other names for these vehicles are Autonomous Surface Crafts (ASCs) or Unmanned Surface Vehicles (USVs), however, in this work, the acronym ASV will be adopted for the entire document

- The development and implementation of a simulator to compare the proposed reactive HC and EC based strategies with other approaches.

1.3 Thesis Contributions

From the scientific point of view, the main contributions of the thesis are:

1. The modeling of the Coverage Path Planning (CPP) for an ASV using HC and EC approaches for maximizing the coverage value using a metaheuristic technique like the Genetic Algorithm.

2. The development and validation of a reactive path planning algorithm that adapts to the dynamic conditions of the environment.

1.4 Document Organization

The background and motivations together with the thesis objectives have been presented in Chapter 1. Chapter 2 details the state of the art about ASV, its applications and the previously proposed path planning algorithms. Additionally, it provides the details of the project for the development of an ASV between the Faculty of Engineering National University of Asuncion and the Schools of Engineers of the University of Seville. Chapter 3 formulates the problem of the Ypacarai Lake and describes the environment under study. Also, it provides a formal mathematical definition of the model and gives the details of the GA that is used to solve the coverage path planning. Chapter 4 describes in detail the HC and EC GA-based approaches for solving the CPP in static conditions as well as a comparison of the performance with other conventional approaches and between them. Chapter 5 presents the on-line learning strategy to detect and follow environmental events in the lake. Finally, in Chapter 6 the conclusions and some discussion about future research in this line are included. Furthermore, the summary of the publication list achieved during this thesis is shown.

Chapter 2

State of the Art

2.1 Autonomous Surface Vehicles and Their Applications

This section focuses on the description of ASVs and their environmental applications, which is the target application in this study. First, it starts with describing the development of the technology during the last 25 years, in which many prototypes of ASVs have been built to test the feasibility of using autonomous vehicles in aquatic environments for different applications. Additionally, an overview of the structure of a typical ASV is also presented. Later, several use cases of ASVs in environmental monitoring are described. Finally, the ASV called Cormoran-II, which is an on-going joint project between the Faculty of Engineering, National University of Asunción (FIUNA) and the University of Seville (US), is detailed. Further information about ASVs and other types of autonomous vehicles, can be found in [11].

2.1.1 Types and General Description of ASVs

ASVs are intended to be used in a variety of applications in the aquatic environment. Typical applications include exploration, surveillance, environmental monitoring, Search and Rescue (SaR), and bathymetry, among others. ASVs have similarities and differences with other autonomous vehicles like Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs), or Unmanned Underwater Vehicles (UUVs). For example, they are not as popular as UAVs, but they are less complex because they move in 2D and not in a 3D plane. UUVs also work in aquatic environments but they move also in a 3D plane; they have another additional issue, which is the remote communication due to the limitations of electromagnetic propagation in water, so acoustic communication is used instead [12]. Inside the group of ASVs, two categories can be distinguished in terms of research, such as autonomous sail-boats and motor-boats. Sail-boats have some advantages over motor-boats; for example, they use the wind for

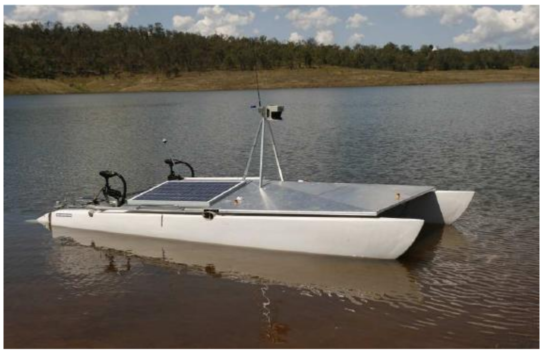


Fig. 2.1 Wivenhoe ASV

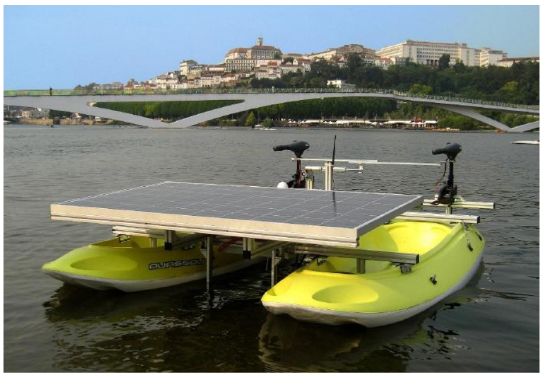


Fig. 2.2 Squirtle ASV

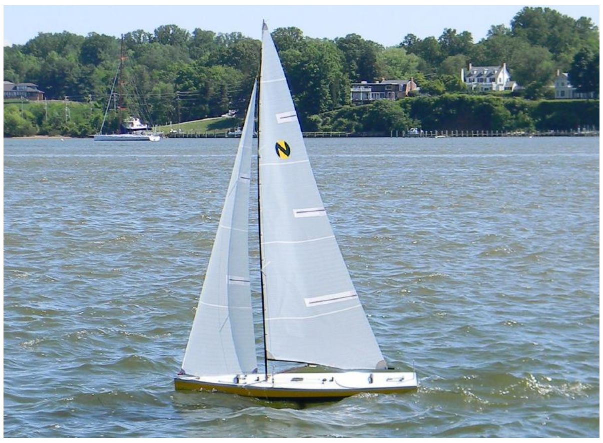


Fig. 2.3 Sea Quester ASV

their propulsion, which is more environmentally friendly and they do not produce any noise that can affect acoustic underwater communications. However, they depend on the weather conditions. In contrast, the motor-based ASV guarantees all-time operation thanks to its electric or fuel motor. This work will focus on the motor-based ASV case and all the descriptions included in this thesis will be referring to this type of boat.

Figure 2.1 and Figure 2.2 show a pair of examples of motor-based ASV, the first example is the Wivenhoe ASV, a catamaran type ASV, which takes its name from the Lake where it has been used, Lake Wivenhoe (Australia) [13], and the second is the Squirtle ASV, which is also a catamaran, but each hull was made from a Kayak. The Squirtle is an ASV developed in Coimbra (Portugal) [14]. Regarding sail-boats, Figure 2.3 presents below a sail-based ASV called Sea Quester constructed in USA for the 7th World Robotics Sailing Championship (WRSC) [15]. This is an annual event where the advances of this type of boats are presented and discussed.

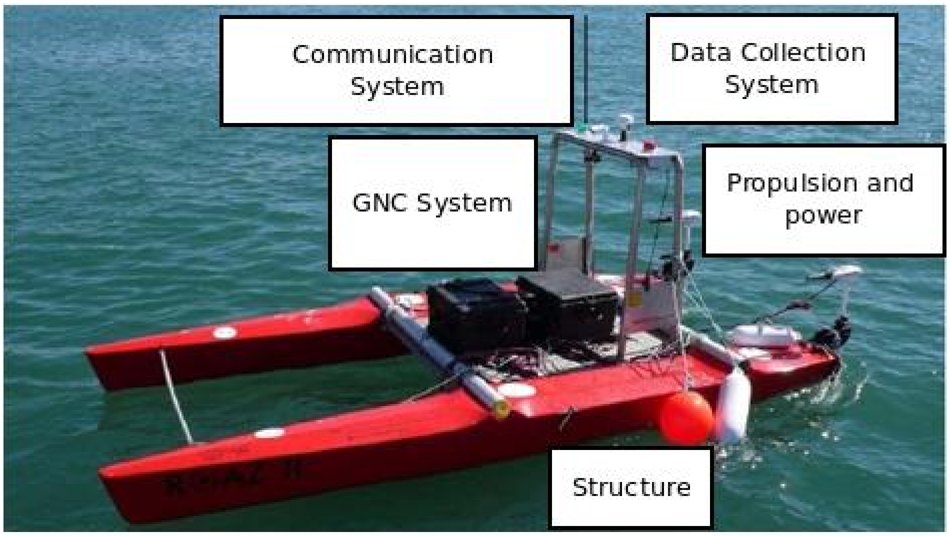


Fig. 2.4 ASV Diagram Systems

In general, an ASV can be divided into different components: the structure, the propulsion and power system, the Guiding, Navigation and Control (GNC) system, the data collection system and the communication system [16] . The structure of an ASV is composed of a hull, which can be a kayak, catamaran (twin hull), and trimaran (triple hull). The propulsion and power systems are responsible for moving the vehicle from one point to another. It includes a propeller or water jet and a rudder to control the direction of the vehicle. The GNC system is the component that makes decisions and gives the orders to all other components in the ASV. It consists of the hardware and software for managing the ASV. In addition, the data collection system is the connection element between the environment and the ASV that generates the information for navigation and for the final application. It is composed of the sensors related to the GNC such as the Inertial Motion Units (IMU) and the Global Positioning System (GPS), sensors related to the external environment such as cameras, radar, sonar, and sensors related to the final application, in this case pH, temperature, Dissolved Oxygen, Oxidation-Reduction Potential (ORP), turbidity, etc. Finally, the communication system provides the link between the ASV and a remote-control center, and it is used for receiving information from the ASV and for sending commands to it. Figure 2.4 details all these components.

According to [17], the initial research involving ASV started in the 90's, with 3 prototypes built. However, this number increased considerable during the first decade of the new millennium with 25 prototypes. In the current decade, 34 prototypes have been built so far, being 4 of them under development. The largest peak has been seen in the period (2015-2017) with 19 prototypes built. In total, they have counted 60 prototypes. Liu et al. [16] have performed a similar survey a year earlier than [17] about ASV prototypes, counting 56 prototypes being built until that date, detailing also the countries where the prototypes were developed. USA followed by the UK is at the top of this list. The first ASVs found in the academic literature were developed by the Massachusetts Institute of Technology (MIT) during the Sea Grant Program, like the ARTEMIS, ACES [18], AUTOCAT [19] and SCOUT [20]. With the new millennium, other countries also developed their own ASVs at their universities or research institutes, for example, the DELFIM [21], Zarco [22] and ROAZ I [23] and II [24] (Portugal), Charlie [25] and SESAMO [26] (Italy), or Wivenhoe ASV [13] (Australia).

Initially most of these ASVs are stand-alone vehicles with a length of few meters. However, a new approach that has appeared during this decade is the use of an ASV as part of a team or fleet of autonomous vehicles. These teams of robots could be heterogeneous or homogeneous teams i.e. using a group of the same type of autonomous vehicles or a combination between different types. Heterogeneous groups are like the one presented in Pinto et al. [27], where a fleet of robots is composed of an UAV and an ASV for riverine environmental applications. The ASV is a catamaran of 4.5 m that piggybacks a small-sized UAV. The idea behind this association is that the UAV extends the perception capability of the ASV, i.e., the Field of Vision (FoV). Using the collected data of both vehicles, a cost map is built for navigation. In Machado et al. [28] two types of ASVs were tested for SaR operations in disaster scenarios under the EU-ICARUS project. It involves the ROAZ, a catamaran-type ASV, and the SWIFT ASV, a water jet-based ASV. The ROAZ with a greater size and autonomy will carry the SWIFT near the disaster point. When this point is reached, the SWIFT gets off the ROAZ and will be directed to the disaster area with high speed, carrying on a life raft for four persons. A third example of heterogeneous network is seen in Busquets et al. [29], where they proposed the operation of an ASV and an UUV to improve the communication capabilities of the UUV with the command ground center at the coast. Homogeneous teams present a different approach. The teams consisted of similar ASVs, which form a swarm. But differently from the main ASV, its members are smaller with less computation capabilities. In this way, the cost of building a swarm is balanced with the cost of deploying one single ASV. The advantage of this strategy is that the tasks assigned to the swarm can be finished faster. Regarding experimental works, Valada et al. [30] introduced a Cooperative Robotic Watercraft (CRW) intended to work in environmental monitoring of lakes and rivers. They performed tests with teams of up to five ASVs in the Taal Lake (Batangas, Philippines), Gowanus Canal (New York, USA), and Allegheny River (Pennsylvania, USA). Another swarm of ASVs is presented in Duarte et al. [31]. They proposed a neural network-based controller for robotic tasks in a swarm of ten ASVs. Similar performance of the simulated controllers was achieved in field tests, which were developed in a small water body connected to the Tagus River (Lisbon, Portugal).

2.1.2 ASV for Environmental Monitoring

As the focus of this work is environmental monitoring, some related works in this field are reviewed. In Caccia et al. [25], the ASV SESAMO was developed to take water samples for studying the sea-surface micro-layer (top 1mm of the water mass), because of its importance as a modulator for the exchange of matter and energy between atmosphere and aquatic ecosystems. A special water sampling and distribution system made of pumps and valves was developed for this purpose. The vehicle was tested in the Thyrreanean Sea as well as in the Antarctic coastal zones. Later, Sukhatme et al. [32] was one of the first works to implement a monitoring system of the microbial communities in an aquatic ecosystem. The system consisted of a mix of ten stationary buoys and one ASV. While the buoys provided high-resolution temporal sampling and low-resolution spatial sampling, the mobile boat provided high-resolution spatial sampling. The devices were equipped with temperature and chlorophyll sensors. Dunbabin et al. [13] also proposed a heterogeneous system of an autonomous catamaran integrated to a 50-node floating sensor network for water quality monitoring in a lake. The catamaran was built with an arm that can take samples from down to 20 m in the water. In addition, the arm had a multipurpose probe that can measure environmental variables such as temperature, conductivity, chlorophyll, turbidity, dissolved oxygen, and incident radiation. The wireless sensor network provided non-line-of-sight communications with the remote operators. The Hydronet is another ASV used for sea environments described in the work of Fornai et al. [33]. They proposed an ASV with a winch and a sampling probe that allows taking samples down to 50 m. These samples can be analyzed on site or stored for later analysis. Regarding the problem of algae bloom in lakes, Hitz et al. [34] presented a novel ASV called Lizbeth designed to take measurements within a range of depths with its custom-made winch. The work described the conception and also shows field results on the waypoint navigation mode. Furthermore, the first results of a sampling campaign for monitoring algae blooms in Lake Zurich are presented. Particularly the cyanobacteria observed in this Lake are the Planktothrix rubescens. Combining surface mobility with depth measurements within a single robot allows a fast deployment in remote location, which is a cost-efficient solution for lake sampling. Studies of algae blooms in oceans with ASV have been done in the work of Robero-Vivas et al. [35], in which they used videos of sea floor to evaluate the presence of Anabaena sp cyanobacteria colonies, which are normally attached to rocks, seagrass and macro algae. They developed an image stabilization mechanism to achieve this task.

More recently, Pedersen et al. [36] have developed the Planetary Lake Lander (PLL) ASV, which has two objectives, first to evaluate the physical chemical and biological processes in high-altitude lakes, and second to prepare for future exploration of Titan's Lake, the biggest satellite of Saturn. This ASV also included a winch for taking samples with a multi-parameter probe down to 45 *m*. Tests were carried out in an alpine lake at the Andes in Chile for 13 months to evaluate the reliability of the system in harsh environments. In Laut et al. [37], the Brooklin Atlantis ASV was presented. This ASV was part of a citizen science system that allows volunteers to perform on-line research and classification tasks of polluted bodies of water in a city canal named Gowanus located at Brooklyn, NY. The included sensors can measure variables such as water temperature, conductivity, dissolved oxygen and ph. Moreover, ASVs can take an active role in environmental applications as in Jung et al. [38], in which an ASV was implemented as part of an Algae Bloom Removal Robotic System (ARROS). The ASV included an Electro-Coagulation and Flotation (ECF) reactor, a technique used in wastewater treatment. It operated together with an UAV that detects the algae bloom, and then the ASV follows a path obtained from a coverage path planning algorithm.

2.1.3 The Cormoran-II ASV Prototype Project

The Project PINV15-177 "Autonomous Surface Vehicle for the study of water quality in Lakes" is an ongoing project being developed by FIUNA with the support of the ACETI Research Group from the Electronic Engineering Department of the US. This project is funded by the National Council of Science and Technology (CONACYT) of the government of Paraguay. The objective of this project is to build a prototype of an ASV with a specialized set of sensors to evaluate the quality of the water of the Ypacarai Lake and help the local government to provide an autonomous way of collecting data from the Lake. The motivation of this project has arisen from the degradation of the Ypacarai Lake due to polluters disposed in the last decades, creating the conditions for the HAB. This project is being mainly developed at the Laboratory of Distributed Systems (LSD).

First a small prototype named Cormoran-I was built. This prototype of a single hull has a length of 1 *m* and it is shown in Figure 2.5. It was implemented to be tele-operated and it has a Pixhawk board as the main controller of the ASV (Figure 2.6). Additionally, it has a Raspberry Pi 3 that it is connected to the water quality sensors. These sensors would be installed under the ASV in a custom-made multi-parameter probe with the shape of missile as it is shown in Figure 2.5. This probe was designed and built in the laboratory with a 3D printer.

Some tests were performed in-situ at Ypacarai Lake to evaluate its operation (Figure 2.7 and Figure 2.8). An on-board camera was also included for a remote view of the environment close to the ASV. This prototype was built for becoming familiar with the technology required to build an autonomous vehicle in aquatic environment and became the basis for the building of a larger ASV that was called Cormoran-II.

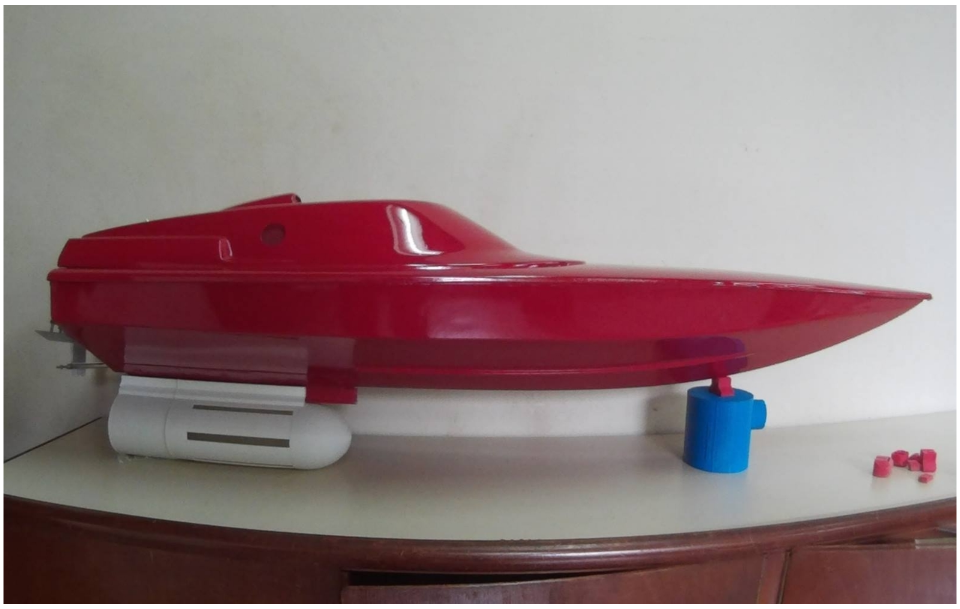


Fig. 2.5 Cormoran-I Prototype

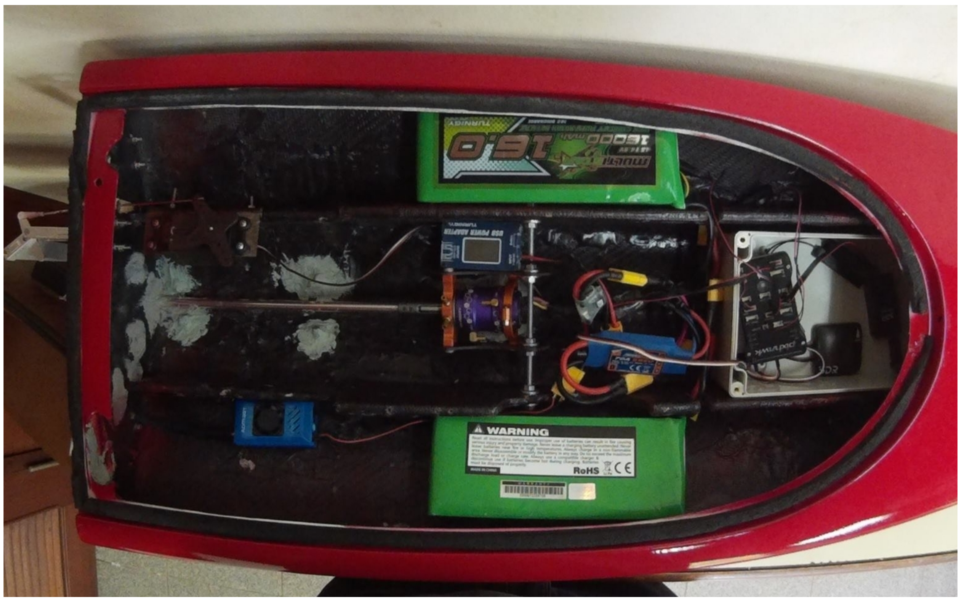


Fig. 2.6 Electronic Components of the Cormoran-I

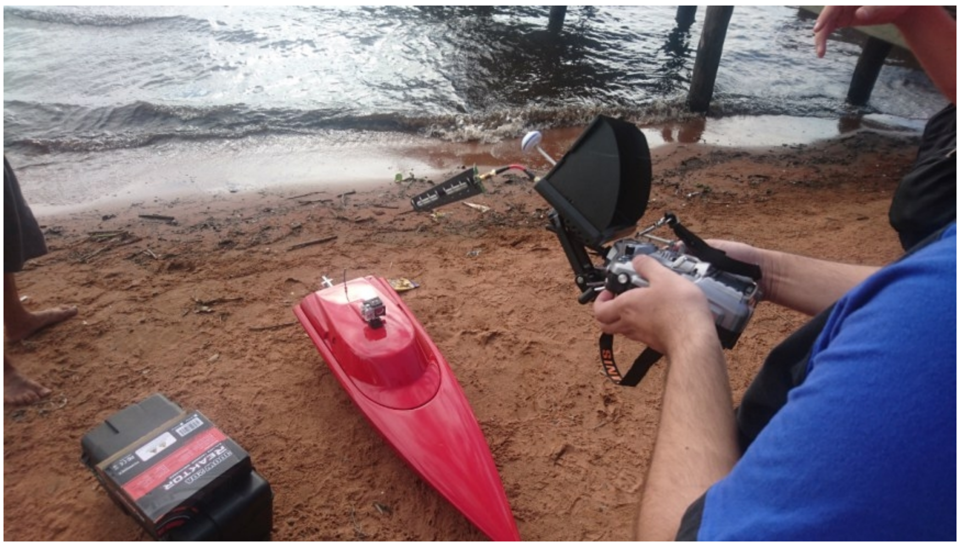


Fig. 2.7 Field Tests of Cormoran-I at Ypakarai Lake

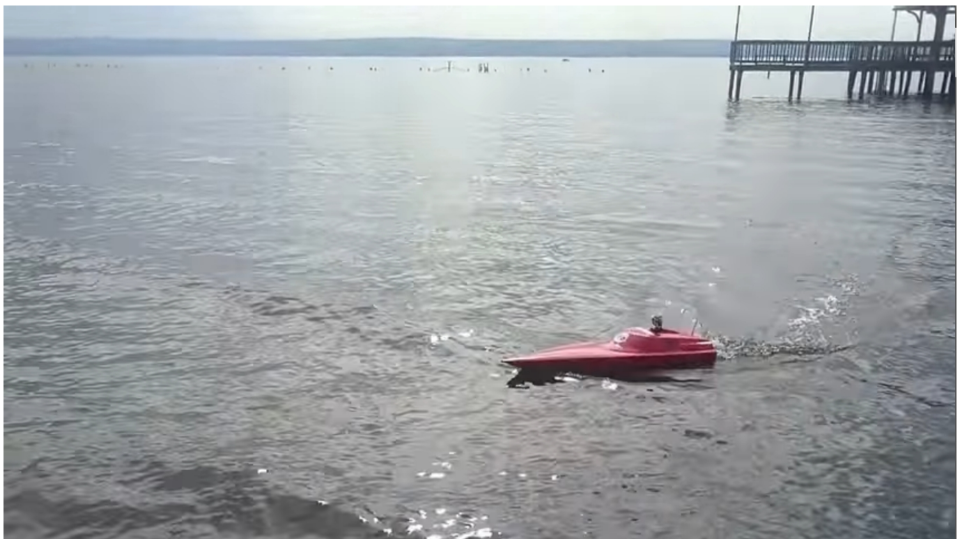


Fig. 2.8 Cormoran-I Traveling over the Ypacarai Lake

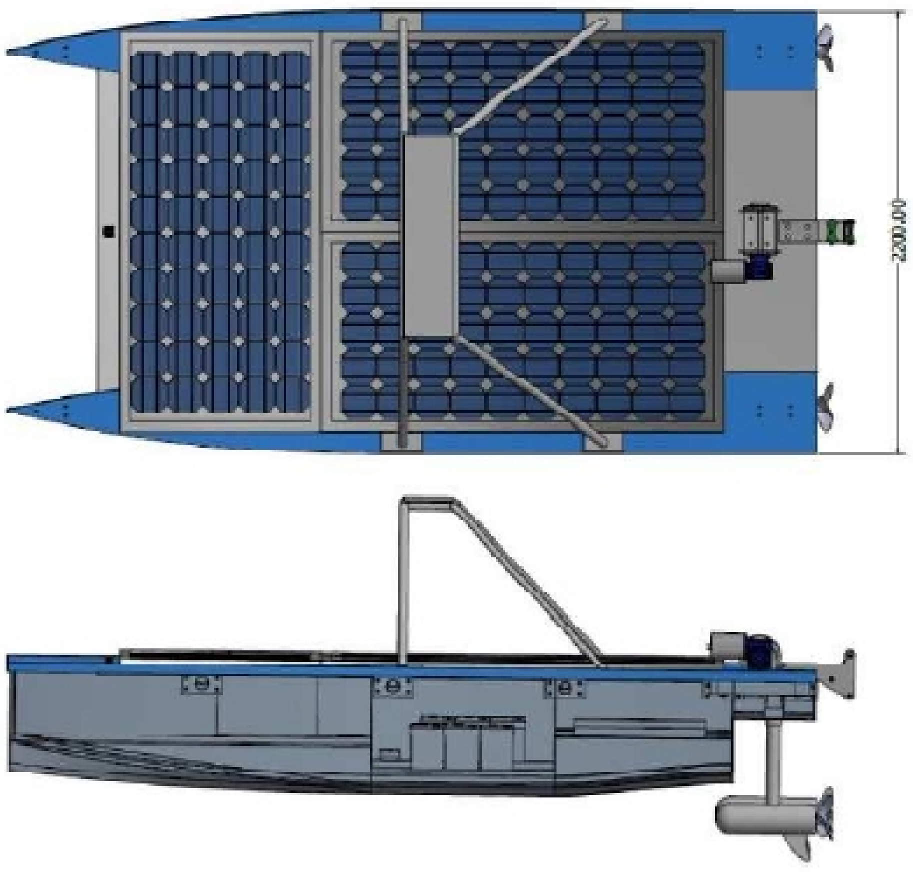


Fig. 2.9 Cormoran-II Structure Design

Currently, the initial design of the Cormoran-II has been developed. The main structure is be based on a catamaran type with an estimated length of 4 meters and a width of 2 meters. The Figure 2.9 depicts the basic structure of the ASV.

This ASV will carry three solar panels of 300 W each and four batteries of 250 mAh that will inject the necessary energy for the ASV. Additionally, the set of sensors on-board will include a camera to execute obstacle detection by image processing, a LIDAR to calculate the direct distance to a specific obstacle, an ultrasonic sensor for close range obstacle detection, a Global Navigation Satellite Systems (GNSS) receiver for retrieving its location information for navigation purposes, and the set of water quality sensors. Furthermore, a set of air quality sensors will also be included. The main processing block will consist of two boards; a Jetson Nvidia board will be the main unit and the PixHawk board that will be used to drive two off-board propulsion engines. In terms of communication, the main communication system with the control center at the shore of the Lake it will use a WiFi link in the 2.4 GHz. Additionally, there will be an auxiliary communication system in the 5.8 GHz for transmitting audio and

Table 2.1 Cormoran-II Electronic Components

Component	Description
Main CPU	1x Nvidia Jetson TX2
	1x Pixhawk
Power	3x Solar Panels 300W
	4x Batteries 250 mAh
Environmental Sensors	1x LIDAR Omnidirectional
	1x GNSS receiver
	2x Ultrasonic sensors
	1x HD Video Camera
Water Quality Sensors	1x PH
	1x Oxido-Potential Reduction (OPR)
	1x Conductivity
	1x Dissolved Oxygen (DO)
Communications	1x Wifi 2.4 GHz
	1x Auxiliary 5.8 GHz
	1x 4G LTE
Propulsion	2x Electrical Off-board 55 lb thrust

video. Full remote monitoring will be achieved with a 4G LTE module. A summary of all the elements included are listed in Table 2.1.

2.2 Algorithms for Coverage Path Planning

This section presents the algorithms that have been proposed for solving the Robot Path Planning (RPP) problem, and a derived case, which is the Coverage Path Planning (CPP). In the first case, the problem is to find a path from a starting point to a destination point, while in the second the problem is to find one that covers a given area. Each of these cases is used for different applications. Additionally, the last section reviews algorithms that adapt the path to the dynamic conditions of the environment.

Table 2.2 Classification of Motion Path Planning Algorithms

Level	Type
1	Holonomic
	Non-Holonomic
	Kinodynamics
2	Search Optimal/feasible path with Environmental Modeling
	Search Optimal/feasible path without Environmental Modeling
3	Off-line
	On-line
4	Deterministic
	Probabilistic

2.2.1 Robot Path Planning

Robot path planning is one of the main research topics in robotics. It can be defined as the problem of finding a path between two positions in a mobile space, considering that it should be collision-free (feasible) and should satisfy certain optimization criterion [39]. The criterion can be the minimization of distance, time, number of turns or energy consumption. Euclidean distance minimization is normally the first condition because it is related to other aspects such as time and energy (minimum path length). In most cases, reducing distance also represents reducing time and energy.

Path planning problems can be classified in four different levels according to [40] (see Table 2.2). The first level classifies the approaches according to the controllable degrees of freedom. If all of them are controllable, then the problem of path planning is holonomic. However, if there are differential constrains that restrict the allowable velocities at each point of the space; it is called non-holonomic. A third type is the so-called Kynodynamic problems, which are subjected to simultaneous kinematics and dynamic constraints.

In the second level, the planning problems can be classified in the search of optimal/feasible paths with or without modeling of the environment. For the group that requires environment modeling, it is performed with the help of techniques such as Cell Decomposition (CD), the Visibility Graph (VG) and the Voronoi Diagram (VD). The CD technique divides the free C-Space (Configuration space, concept introduced in [41]) environment into smaller cells, which can take different geometric shapes (triangles, square, hexagon). The cells can be equals or different between them in size or shape, forming a homogeneous or heterogeneous groups.

The VG-based approach models the topology environment by linking the obstacles vertices between themselves and with the initial and final nodes (positions) of the graph. Finally, the VD-based method calculates regions limited with lines of equal distance to the polygonal obstacles (considering a 2D environment). There is a trade-off between using the VD or the VG approaches since the VG can generate later an optimal path, while the VD has a lower computing time. If the resulting environment can be modeled as a graph, then Dijkstra or A* algorithms [42] can be applied to find the shortest path between two points. Since the algorithm was proposed in 1959 [43], the Dijkstra algorithm is one of the most well-known traversal graph algorithms. Later, the A* [44] was proposed using a heuristic function towards the goal node, orienting the search and reducing its computation time.

Techniques that do not require the environmental modeling are the Potential Fields [45] and the sampling-based models [46], such as the Rapidly exploring Random Trees (RRT) [47] and Probabilistic Roadmap (PRM) [48]. The Potential Fields techniques simulate that the robot is an electric charge, which is attracted towards the destination goal and rejected by the obstacles. In PRM and RRT, random sampling is used to create the path, but the PRM has two stages, an initial learning phase and then a query phase. This technique is referred as a multi-query planner. Conversely, the RRT is a single query planner, where a tree is incrementally grown from the start configuration towards the goal.

A third level of classification is the global or local planning. Global planning is referred when there is complete knowledge of the space in which the autonomous vehicle will move, i.e., all the feasible and unfeasible regions in the area are known a-priori. If there is only partial knowledge of the environment or if it is a dynamic environment, then a local planning is performed. Global and local path planning techniques are also referred as off-line and on-line planning, because in the first case, all the planning is calculated before the robots start moving with all the available information. In contrast, the on-line planning is calculated as the autonomous vehicles travel through the scenario, which means that on-line planning requires a faster response than off-line planning. In other words, on-line planning has a reactive behavior. Potential field technique is adequate for on-line path planning considering its easy implementation, however, it can be trapped in a local minima. Usually a combination of both techniques is required; an initial path will be defined by the global planning, but an on-line planning is also conducted when unexpected situations occur, like an unforeseen obstacle.

Finally, in the last level of classification two categories can be distinguished, the deterministic and probabilistic methods. In the deterministic approach, the same result is always achieved at each execution. Examples of this type are the already mentioned Dijkstra and A* algorithms, because they perform an exhaustive search. This will result in a very high complexity as the problem size increases, making them unsuitable for real-time planning.

Therefore, the probabilistic methods are preferred for obtaining fast solutions that sometimes can be sub-optimal but still satisfactory for the operation of the robot. In this group, the PRM and RRT can be found again.

All these previous techniques can be considered as the classical approaches; however, except for those performing an exhaustive search, they tend to produce non-optimal solutions and to get trapped in local minima. Heuristic-based methods have been proposed as an alternative approach to overcome this limitation [49]. In fact, these techniques have been increasingly used to solve path planning problems since the 90's. It has been surveyed in the literature [50] that at that decade, less than 20% of the existing works used heuristics for path planning problem. However, this number has increased to over 50% in recent years. In [50], it has been summarized that the use of metaheuristics in PPP. Particularly, it is mentioned the interest of using of Neural Networks (NN) [51] , nature-inspired algorithms like the Ant Colony Optimization (ACO) [52] or Particle Swarm Optimization (PSO) [53], and Genetic Algorithm (GA) [108]. Each technique has its advantages and disadvantages, leading the use of hybrid methods that combine the strengths of each technique and attenuates the weaknesses [54], [55], [56].

Genetic algorithm is the most widespread of the Evolutionary Algorithm (EAs) group whose operation is based on biological processes [57]. There are techniques that have similar principles with GA and that were even proposed earlier, like Evolutionary Programming (EP) or Evolutionary Strategies (ES), but GA has been found to be an interesting tool for optimizing discrete problems like the RPP. For example, GA has been applied for optimizing path planning, as in Alvarez et al. [58], where they studied the path planning of an UUV in dynamic environments. They introduced two new operators in the GA algorithm. First, the initial population is formed from the best individuals obtained from running the GA a few generations. Then, the second operator is the incorporation of random immigrants into the new population at each generation. The optimal path is found so that the energy of the UUV is minimized. Tuncer and Yildirim [59] applied GA to a RPP of a mobile robot in dynamic environment, i.e., an environment with obstacles. In [59], they also proposed a mutation operator that increases the diversity of population and avoids a premature convergence. Vehicle Routing Problem (VRP) is a constrained extension of the Traveling Salesman Problem (TSP), where GA has been applied as it is shown in Savuran and Karakaya [60]. They used an evolutionary technique to find the path planning of an UAV that is deployed on a mobile platform. The UAV must visit fixed targets and it considers the constraints of depot mobility and UAV range. Karami and Hasanzadeh [61] proposed the use of a novel selection operator to overcome the local-trap problem and avoid premature convergence in the GA for robot motion planning in a 2D environment. In the work of Ergezer and Leblebicioglu [62], after solving the TSP for a Desired Region (DR), the GA was used to maximize a variable called Collected Information (CI) in the DR and at the same time avoid flying over Forbidden Regions (FR). Roberge et al. [63], compared the performance of the GA versus the PSO for the path planning of an UAV in a 3D environment. Furthermore, they reduced the execution time by using the “single-program, multiple-data” parallel programming paradigm. After simulating in 3D real-based environments, it is observed that the GA performance is better in most of the evaluated cases. Ramirez et al. [64], used a multi-objective GA for mission planning of a fleet of UAVs and a set of Ground Control Stations (GCS). The algorithm has been tested when optimizing different variables of the mission, such as the makespan, fuel consumption, and distance. Table 2.3 presents a summary of the reviewed path planning techniques based on GA.

2.2.2 Coverage Path Planning

Coverage path planning is the task of determining an optimal path that passes over all points of an area or volume of interest while avoiding obstacles [65]. There are different applications with robots that use this type of planning, such as vacuum cleaner robots, paint robots, automated harvest robot, etc. The requirements for the robot coverage are as follows:

- Robot must move through all the points in the target area covering it completely
- Robot must fill region without overlapping paths.
- Robot must avoid all obstacles
- Continuous and sequential operation without any repetition of path is required.
- Simple motion trajectories should be used (straight lines or circles).
- An optimal path is required.

Choset [66] presents a taxonomy for the classification of CPP approaches. Mainly the techniques can be separated into two main groups, the heuristic and randomized approaches or the complete coverage approach. The heuristic approach offers a solution that does not require the use of complex sensors by the robot. This type of approach does not guarantee a complete coverage though. For the complete coverage approach, a cellular decomposition of the space is required. This decomposition of the space is divided again into approximate, semi-approximate and exact cellular decomposition categories. In the approximate cellular decomposition, the free space has a fine-grid representation, and a cell is considered covered when it is visited because each cell has the size of the robot footprint. This representation is also called grid-based. Figure 2.10 shows an example of a map with two irregular obstacles.

Table 2.3 Related Work With Static Path Planning Problems Using GA

Reference	**Application**	**Technique**	**Contributions**
[58]	UUV	GA	Novel genetic operators. Evaluation based on a real 3D environment (Mediterranean Sea)
[59]	Mobile Robot	GA	New mutation operator.
[60]	UAV	GA	Variation of the Vehicle Routing Problem is introduced. Objective function that satisfies the constrain of a mobile depot. Adaptation of the GA to include these constrains while maximizing the coverage of the UAV.
[61]	Mobile Robot	GA	Novel selection operator.
[62]	UAV	GA	Novel evolutionary operators. Path implementation using the dynamical mathematical model of an UAV.
[63]	UAV	GA/PSO	Multi-objective cost function. Reduction of execution time by using single-program, multiple-data parallel programming paradigm. Implementation on commercial off-the-self multi-core CPUs.
[64]	UAV	GA	Multi-objective GA for solving complex mission using a team of UAV and a set of CGS. Hybrid fitness function.

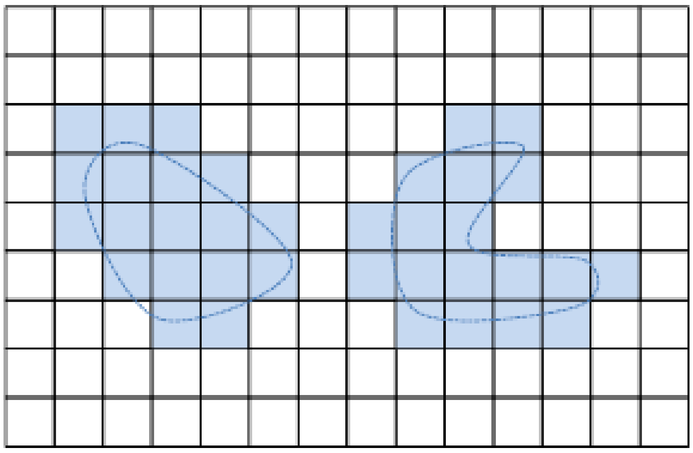


Fig. 2.10 Approximate Map

The semi-approximate approach is similar to the approximate with the space divided into cells that have fixed parallel limits but the top and bottom can have any shape. This approach avoids missing certain areas of the map because of an irregular terrain or the presence of obstacles. It is important to define this critical area, which is called inlet, and the entrance and exit points to these inlets. Figure 2.11 shows an example of a Robot R following a path in an environment with an obstacle and three inlet points ($d1$, $d2$ and $d3$) The map to be covered is divided with equally separated grid lines $L-4, L-3...L3, L4$, being $L0$ the line at the starting point. This is also the first inlet point $d1$. The robot follows the boundary of the map until detecting an inlet at $d2$. It continues along the boundary until passing two times by line $L1$, where it starts a zig-zag patter until leaving the inlet. Then, it goes along the boundary again until passing two times by line $L3$; where it moves in zig-zag. After reaching to line $L2$ moves along the boundary until detecting again line $L2$, when it returns to a zig-zag pattern. This continues until $d3$ where the entry point of the island (obstacle) is. Finally, the exact cellular decomposition is the set of non-intersection regions that fills the target environment, and at each region, the robot covers the cell with back and forth motions. An example of this type is the Boustrophedon decomposition or its derived Morse-based cellular decomposition [67]. The term Boustrophedon literally means the way of the ox, referring to the movement that an ox drags when plowing in a field. Figure 2.12 shows a map divided in different cells with a triangle and hexagon obstacles and the back and forth path in the cell below the hexagon.

The following step after dividing the space into cells or sub-regions is to define how the cells will be visited. Then, the resulting decomposed graph can be viewed as a graph with vertices and edges. The vertices are the cells of the divided space and the edges the links that connect the cells with each other. Having all the nodes defined, the problem is to select the edges. A tour that visits all the nodes once and returns to the initial point is called Hamiltonian Circuit (HC).

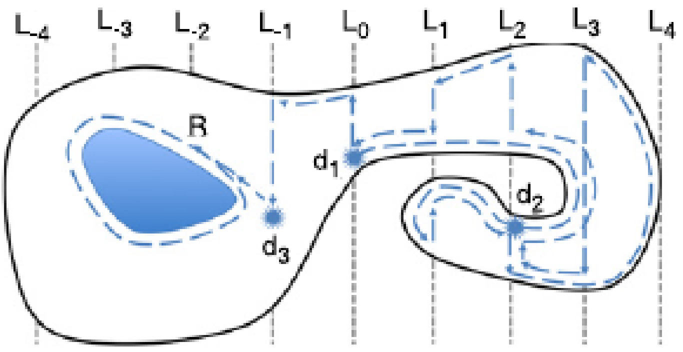


Fig. 2.11 Semi-approximate Map

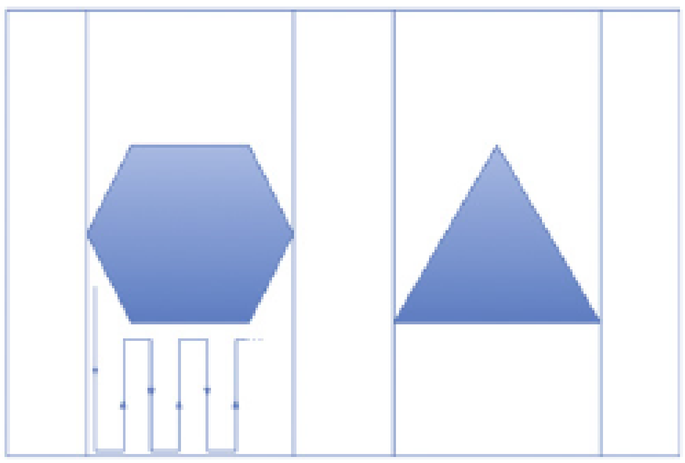


Fig. 2.12 Exact Map Decomposition

It can be concluded that in a HC every vertex has two edges, one incoming and one outgoing. Assuming that a HC exists for a given graph, finding the HC with the shortest distance is the famous mathematical problem called as TSP [68]. Even though its simple description, this problem has very powerful implications behind it, because it belongs to a class of problems with high level complexity known as NP-complete (Non-deterministic Polynomial time) [69]. Because of this, it has been used as a testbed to evaluate the performance of heuristics and exact algorithms. An alternative approach is that in order to visit all the nodes in the graph, some nodes must be passed more than once. In this case, there might be more than a pair of edges connected to the nodes. A tour cycle under these conditions is an Eulerian Circuit (EC), named after Leonhard Euler who studied the problem of the seven bridges of Konigsber problem in 1736 [70]. This problem is different from the TSP because instead of finding a circuit that visits all the nodes of the graph once, it tries to find a circuit that visits all the edges of the graph. The problem of finding the EC with the shortest distance is called the Chinese Postman Problem (ChPP) [71]. Euler stated that in order to have a circuit with the condition of visiting all the edges, the nodes must have an even number of edges connected to them. In the case that there is an odd number of edges, the ChPP tries to find an expanded graph with edges that should be traveled more than once. This problem can be solved in polynomial time; however, some variations of the ChPP are as complex as the TSP. This is the case of the Rural Postman Problem (RPP) [72], where from all the available edges, only a sub-set is required to be visited and other sub-set is optional. The same as with the TSP, this problem is also NP-complete, and it has much more practical applications than the ChPP. It can be applied in logistics, where paths are chosen to deliver goods, in transportation services like garbage collection, and in mail delivery and snow plowing, among others [73].

Precision Agriculture (PA) where technology is applied for a better crop monitoring and protection is one of the fields where CPP is used for autonomous vehicles like UAV or UGV. On this line, Xu et al. [74] presented a Boustrophedon Cellular Decomposition (BCD) based algorithm, and then solves the ChPP. The algorithm takes into account the non-holonomic vehicular dynamics. They tested the approach using a six Degrees of Freedom (DoF) high-fidelity flight simulator and a real field-test over 200 km using an UAV. Sinha [75] proposed a TSP-Greedy and a TSP-Dikjstra algorithm to explore a region, and they implemented their approach in a hexacopter that acts as a sprayer tool for weed control.

2.2.3 Adaptive path Planning (ADP)

Adaptive Path Planning is required to adapt quickly and effectively to the dynamics of the environment. Many techniques have been proposed to improve the traditional graph search techniques like the A*. For example, Trovato [76] presents the Differential A* algorithm, which adapts only the fraction of space that have been modified. This reduces the time to obtain a new optimized path. However, the technique is inefficient when the robot is near the goals. The D* overcomes the limitations by focusing on the locations likely to be used by the robot [77]. Furthermore, other techniques that have additionally introduced improvements to the A* algorithms are the Lifelong Planning A* (LPA*) [78], D* focused [79] and D* lite [80]. These algorithms exploit the change of consistency of the path to re-plan [81]. Theta* [82] is another algorithm that achieves better results because it not only evaluates the distance from one node to its neighbors but also the parent node related to the neighbors. This gives a smoother and shorter path than those generated by A*.

The conventional approaches of offline evolutionary computation for finding path planning tend to be inflexible in response to changes in the environment and the conditions of the robot. The same technique remains unmodified even if the environment has changed. On this line, Xiao et al. [83] proposed an evolutionary based planner that avoids using the same chromosome structure in the evolutionary algorithm for both off-line and on-line planning. Their work focused initially on the operators and fitness evaluation, the system performance and self-tuning, and finally on the adaptability of the navigator with respect to the operator probabilities and the on-line navigation process. An approach using GA is presented in Hu and Yang [84] in which they introduced knowledge in the model definition for usage with GA. This knowledge is included in the form of path representation that combines grids and coordinates representation. Additionally, they proposed six knowledge-based genetic operators. They claimed the suitability of this approach when an unforeseen obstacle appears in the environment. Zhow and Li [85] presented an adaptive artificial potential field method for robot obstacle avoidance path planning. One of the limitations of the potential field methods is that the solutions are trapped in local minima. For overcoming this issue, they improved the conventional approach by adjusting the weights of the obstacle potential field function.

In the aquatic environments, adaptive path planning has also been studied. Wilson and Williams [86] presented an adaptive path planning to be used in bathymetry missions. Given a target depth and a bounding polygon, the ASV will follow the intersection of the bounding polygon and the depth contour modeled with a Gaussian Process (GP). Once the perimeter is determined, a coverage algorithm based on the Boustrophedon Cellular Decomposition (BDC) is used. For an effective implementation in an ASV with a small-embedded PC, incremental algorithms have been implemented to update Cholesky decomposition of the covariance matrices required for the GP model. Moreover, a multi-thread implementation for sharing the computational load is applied. Paull et al. [87] used an information gain approach for an adaptive path planning in an AUV equipped with a side-scan sonar. First, they proposed an exact hexagonal cell decomposition and additionally, they applied the concept of branch entropy in a Directed Acyclic Graph (DAG). The entropy of each branch is calculated transforming the search from a greedy-first into a reactive A* search. Smith et al. [88] proposed an adaptive path planning for an AUV to perform sampling and tracking of an ocean front, which is a narrow band of enhanced horizontal gradients of water properties dividing broader areas into different masses or verticals. They are linked to elevated primary production and enhanced diversity of species. First, an initial path is calculated that moves in a zig-zag pattern inside the boundaries of the front. This path is adapted on-the fly after measuring and evaluating the related collected science data. The results provide new waypoints that adapts to changes in the boundaries of the front. Table 2.4 presents a summary of these works.

Table 2.4 Related Work with Adaptive Path Planning Techniques

Reference	Application	Technique	Contributions
[83]	Mobile Robot	Evolutionary Algorithm	Accommodation of different optimization criteria. Incorporation of problem-specific domain knowledge. Trade-off among near-optimality of paths, high planning efficiency and effective handling of unknown obstacles.
[84]	Mobile Robot	GA with local search	Domain knowledge incorporated in specialized operators. Unique and simple path representations. Effective evaluation method.
[85]	Mobile Robot	Potential Fields	Adjustment of the obstacle weight in potential field function.
[86]	ASV	GP with BCD	Efficient methods for on-line fitting, prediction and hyper parameter optimization within the GP. New algorithms for partition of convex polygons. Multi-thread implementation in a real ASV.
[87]	AUV	A* with exact hexagon cell decomposition	Branch entropy in a Directed Acyclic Graph (DAG) for reactive path planning.
[88]	AUV	Zig-zag pattern	Tracking an oceanfront based on predictions of other sampling assets or ocean models. Adaptation algorithm from input related parameters determine and predicts the location of the frontal boundary.

Chapter 3

Statement of the Problem

This chapter presents the details about the problem to be studied, such as the geographical characteristics of Ypacarai Lake, the wireless data beacons infrastructure, the definition used for coverage, the modeling of the problem using HC and EC approaches, and a discussion about the complexity of finding a solution to these approaches. Furthermore a brief overview of metaheuristics tools and the reasons for choosing GA as a suitable metaheuristic optimization algorithm are presented.

3.1 Coverage Problem

The Ypacarai Lake is located approximately between latitudes $25°15'$ and $25°22'$ (14 *km*) south; and between longitudes $57°17'$ and $57°23'$ west (10 *km*). The largest distance between two points at the shore is about 14 *km*. The area that should be covered is around 60 km^2. Its average depth is about 1.72 meters with a maximum value of 2.53 meters [1]. The shallow feature of the lake simplifies the task of taking samples because it will not need a special mechanism for submerging the sensor probe several meters under water, like in the case of [13] and [34].

For achieving the covered task with a single vehicle, a wireless network of n beacons around the lake will serve as waypoints that the drone can visit at least once. The final path solution consist of a sequence of visited beacons and each path segment between two consecutive beacons will be called route throughout the thesis. Furthermore, by convention the beacons have been equally distributed around the lake in such a way that each beacon is within the wireless communication range of both of its neighbors. For this particular case $n = 60$, as shown in Figure 3.1, giving an average distance between two consecutive beacons of less than 1 *km*. Notice that this is a normal transmission range for wireless devices based on Zigbee technology.

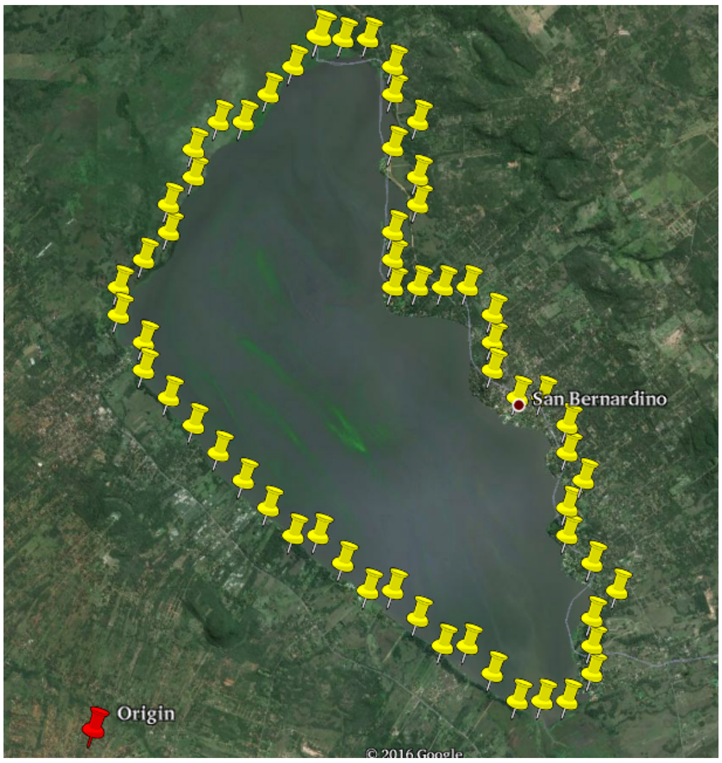


Fig. 3.1 Beacon Distribution

In order to identify each beacon, they are assigned a number between 0 and 59. Figure 3.2 shows the numbering convention that will be used throughout the thesis. In this graph, the coordinates are translated from degrees, minutes and seconds to meters, considering $1'' = 30m$. One unit in both axes of the graph is equivalent to $1m$. The origin of this graph is taken at point $25°22'21''$ S and $57°22'57''$ W, as shown as a red pin in Figure 3.1.

The main metric used in this research is the coverage of the ASV that results from following certain path. By convention, this metric of the ASV is calculated using eq. 3.1.

$$coverage = S_{ASV} * (d_{b_{n-1}b_0}) + \sum_{i=0}^{n-1} d_{b_i b_{i+1}} - {S_{ASV}}^2 * n_{intersec} \tag{3.1}$$

, where S_{ASV} is the sensing distance, $d_{b_{n-1}b_0}$ is the distance from the last beacon of the path to the first beacon, $d_{b_i b_{i+i}}$ is the distance from the beacon i to the following $i+1$ and $n_{intersec}$ is the number of intersections that the path has. The idea behind this definition is that the ASV sweeps the lake with certain beam as it travels. The aperture of the beam is given by the parameter called sensing distance S_{ASV}, which indicates the maximum distance from the measurement point that this sample represents. For example, in [30] a sensing area was defined

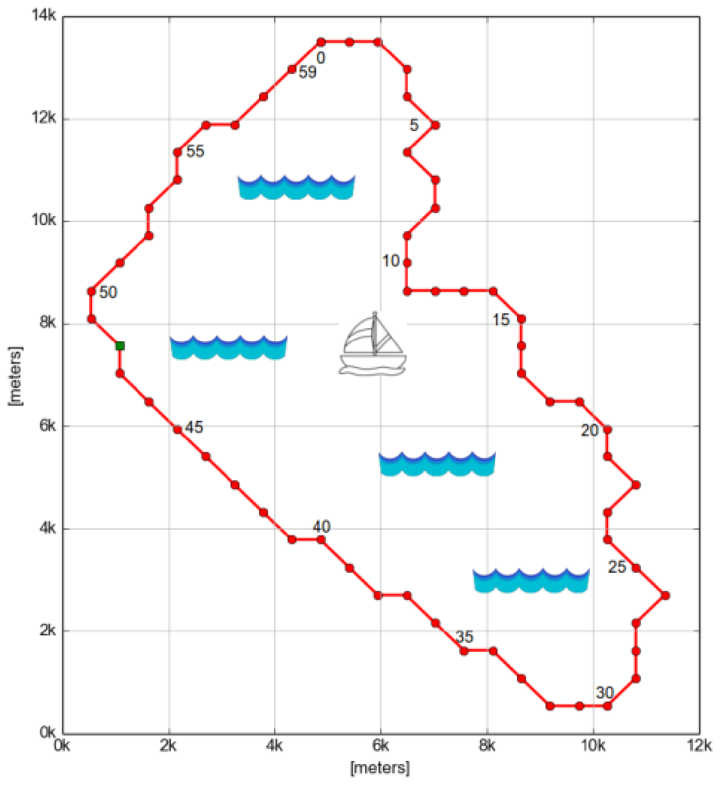


Fig. 3.2 Beacon Convention

by dividing the studied area of 6,510 m^2 is divided into a 10×10 grid cells. This gives a sensing distance of around 8 m that will be used as a reference for our simulations. Furthermore, eq. 3.1 excludes the path intersections since they can be considered as redundant data. For a better understanding Figure 3.2 shows an example where these ideas can be better illustrated. The ASV is traveling from beacon b_7 to b_4, then to b_1 and finally it arrives at b_6. The green line is the single covered area, and the red, the redundant area.

3.2 Modelling Using Graph Theory

Notice that Figure 3.3 suggests that a graph can be built with the beacons and the routes This can be modeled as a complete and connected graph $G\{V,E\}$, where $V = v_1, v_2, ..., v_n$ is the set of vertices and $E = e_1, e_2, .., e_n$ is the set of edges such that e_i connects v_i and v_j, where each edge has also a weight d_{ij} ($d_{ij} > 0$; $d_{ii} = \infty$). Furthermore, it is assumed that the path conditions between a pair of beacons is identical in either direction, then the weight is equal in both directions ($d_{ij} = d_{ji}$) and it is said that the graph is undirected. In the presented model, the weight of an edge is the Euclidean distance between v_i and v_j, $V = v_1, v_2, ..., v_n$. Also, the beacons are the vertices, the routes are the edges, and the distance between two beacons is

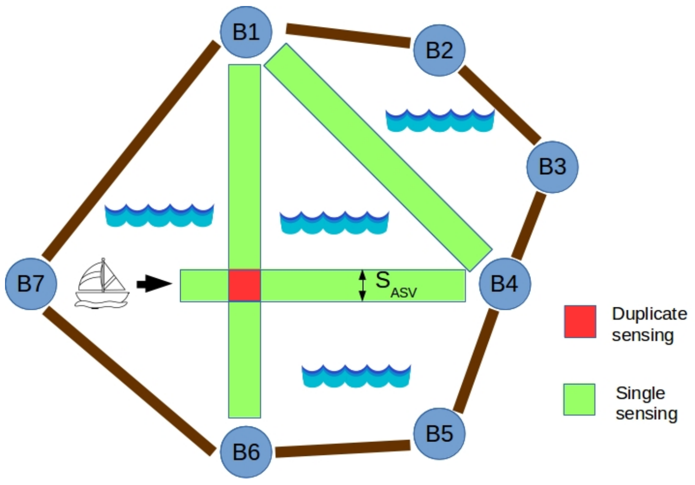


Fig. 3.3 Definition of Covered Area

the weight of the corresponding route. The order of a graph is $|V|$ and it is calculated as the number of vertices. The size of a graph is $|E|$ and it is calculated as the number of edges. The degree of a vertex is the number of edges that connect to it, where an edge that connects to the vertex at both ends (a loop) is counted twice. There are two approaches proposed in this work for finding the maximum coverage, one is the use of an HC and the other an EC.

By definition, a Hamiltonian cycle is a tour in a graph that visits all the vertices and edges once and starts and ends at the same vertex [68]. For having this, each vertex should have only a pair of edges, one incoming and one outgoing edge. In other words, all the vertices of the graph should have a degree equal to 2. Then, a Hamiltonian Circuit (HC) is defined for a given graph $G\{V,E\}$ as shown in eq. 3.2

$$\begin{aligned} HC = G'\{V',E'\} \; s.t. \; & V' = V, \\ & E' \subset E, \\ & deg(v_i) = 2, \forall i = \{1,2,\ldots,n\} \\ & e_j \; is \; valid, \forall i = \{1,2,\ldots,n\} \end{aligned} \tag{3.2}$$

Notice that an edge e is valid if that edge remains entirely inside the perimeter of the lake. If not, then the edge cannot be considered part of the sub-graph G'. As a consequence of this definition, the order and the size of G' are the same, i.e., $|V'| = |G'|$. On the other hand, an Eulerian Circuit (EC) is a closed tour that visits all the edges [89]. However, it can visit each vertex more than once. One graph has at least an EC if the degree of all the nodes is even. This condition was established by Euler in 1736 when studying the Konigsberg bridge problem [90].

One additional condition is to verify that it is a connected graph, meaning that there is a path between every pair of vertices in the graph. Similar to the HC case, the definition of an EC is:

$$\begin{aligned} EC = G''\{V'', E''\} \; s.t. \; & V'' \subseteq V, \\ & E' \subset E, \\ & deg(v_i) \leq 2m, \forall i = \{1,2,\ldots,o\} \; and \; m \geq 1, \; o < n \\ & e_j \; is \; valid, \forall i = \{1,2,\ldots,p\} \; and \; p^2 \end{aligned} \tag{3.3}$$

Please notice from eq. 3.3 that degree of the vertices of an EC is an even number and that not all the initial vertices from G are included in the sub-graph, the size of G'' (number of edges) varies and it is less than the square order of G'' (number of vertices). Furthermore, as in the HC only the valid edges are considered.

3.3 Complexity Analysis

The complexity of both approaches can be determined by calculating the number of possible solutions for each case. In this context, complexity refers to the difficulty of finding and optimal circuit. The complexity comparison of both circuits defined by eq. 3.2 and 3.3 shows that finding an EC is much more difficult than finding a HC because the order of HC is fixed while in the EC is variable. For the HC, initially the number of alternatives to choose from as the next vertex is $n-1$, and as it visits the vertices the number of alternatives is reduced by 1 after each visit. Therefore, the number of possible tours is $(n-1)!$ divided by 2 because the graph is undirected. Conversely, in the EC model, the number of alternatives remains as n-1. Then, the number of possible tours is $(n-1)^{n-1}/2$, which shows the higher complexity of EC in front of HC. In fact, finding the HC is clearly analogous to the TSP and finding the EC to the RPP, being both problems considered as NP-complete. Figure 3.4 presents a graph that compares the number of possible solutions for up to 50 vertices.

Therefore, the problem is to find from the initial graph $G\{V,E\}$ two sub-graphs $G_{HC}\{V_{HC}, E_{HC}\}$ and $G_{EC}\{V_{EC}, E_{EC}\}$ that satisfy the aforementioned conditions 3.2 and 3.3. It is important to recall that for a HC, the order and size of the graph are always the same as indicated in eq. 3.4.

$$|V| = |V_{HC}| = |E_{HC}| = n \tag{3.4}$$

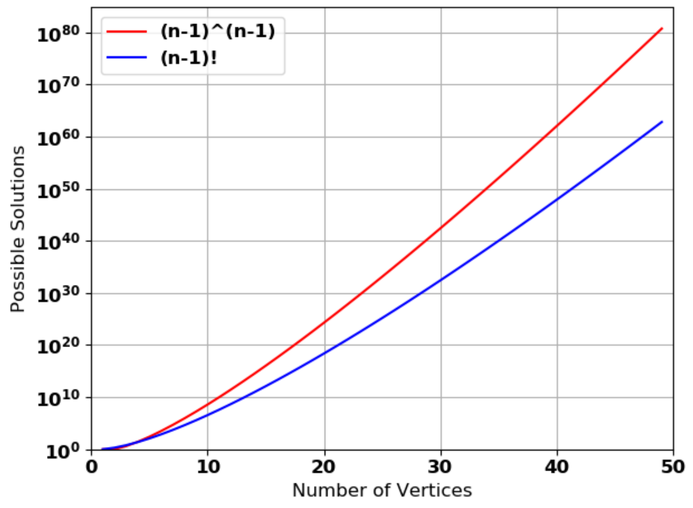


Fig. 3.4 Complexity of Eulerian Circuit ($(n-1)^{(n-1)}$) versus Hamiltonian Circuit $(n-1)!$

However, this is not necessarily true for an EC as indicated in eq. 3.3. Therefore, and for the sake of making a fair comparison between these two approaches, the graphs will be forced to have the same size as shown in the eq. 3.5.

$$|E_{HC}| = |E_{EC}| = n. \tag{3.5}$$

3.4 Metaheuristics as Optimization Tool

Problems like the TSP and its derivatives can be solved using complete or approximate algorithms. However, using the complete approach for these very high complex problems is impractical due to the immense search space. Then, the use of approximate algorithms seems more adequate. The metaheuristics is a new kind of approximate algorithms that have been used in the last 30 years, which combine basic heuristic methods in higher level framework aimed at efficiently and effectively exploring a high search space [91]. In brief, here are some characteristics of the metaheuristics:

- They are strategies that guide the search process.
- The goal is to efficiently explore the search space in order to find (near-) optimal solutions.
- They are approximate and usually non-deterministic.

- They may incorporate mechanism to avoid getting trapped in confined areas of the search space.
- They are not problem specific.

Metaheuristics algorithms were used to solve the PPP, like the Iterated Local Search (ILS) [92], Guided Local Search (GLS) [93], Tabu Search (TS) [94], Genetic Algorithm (GA) [95], etc. Special attention is given to the ILS, the TS and GA, as being popular and effective techniques to solve different types of practical problems. ILS and TS are algorithms based on probabilistic and stochastic processes and they are not bio-inspired like the GA [96]. They are predominantly global optimization algorithms and apply a neighborhood exploring (local) search procedure.

The key idea behind ILS is to focus the search not on the full space of all candidate solutions but on the solutions that are returned by some underlying algorithm, typically a local search heuristic. As its name implies, it iteratively builds a sequence of solutions generated by the embedded heuristics. It samples in a broader neighborhood of candidate solutions and uses a local search technique to refine their local optima. ILS explores a sequence of solutions created as perturbation of the current best solution. Despite being a simple concept it has led to state-of the art algorithms for many computationally hard problems.

TS maintains a short-term memory of recent moves and prevents returning to recently visited areas of search space (tabu list). It was originally designed to manage a hill climbing technique for the neighborhood heuristic exploration. It has been shown effective since its appearance in 1986 as it can be seen in a large number of papers using this technique. However, to exploit adequately its effectiveness it is important to grasp the basics of its operation.

GA is the most representative technique of the family of evolutionary algorithms. It was introduced in the book Adaptation in Natural and Artificial Systems in 1975 by John Holland [97]. It adopts the basic Darwinian principle of “survival of the fittest”. As it had been applied for several decades, it has become a very mature meta-heuristic that has been used to solve a great variety of optimization problems like the TSP [98] and others [99], [100]. Different from earlier evolutionary algorithms, which focus on the operation of the mutation operation to perform local search, GA adds the concept of recombination of information between possible solutions [95]. GA can be applied to a number of optimization problems as long as their possible solutions meet the basic requirements, such as being encoded as a combination of their building blocks and having an objective function. A diagram of a classical GA is shown in Figure 3.5. In the following chapter a thorough description of each block of Figure 3.5 is provided.

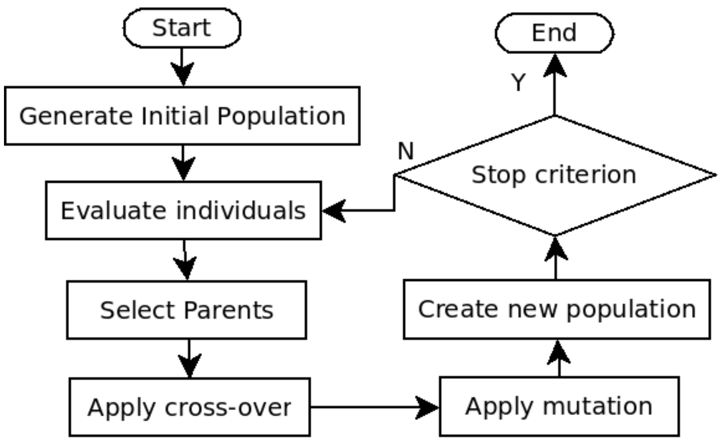


Fig. 3.5 Ilustration of GA Steps

Chapter 4

Evolutionary Path Planning for Autonomous Surface Vehicles

This chapter presents the first contribution of the thesis, the modeling of the Coverage Path Planning (CPP) using Hamiltonian Circuits (HC) and Eulerian Circuits (EC) to solve the coverage maximization problem with the help of the GA. It is divided in the following parts: first, the modeling of the HC and EC to be used with the GA is introduced. Then, the simulation environment is presented, describing the software and hardware used for simulations as well as the simulation parameters. The following part describes different results of the performance of HC and EC in a static environment, i. e., the map remains constant during the entire simulation. Different cases are evaluated, first only the case of EC, second the HC and finally a performance comparison between both.

4.1 Hamiltonian and Eulerian Circuits Modelling for Genetic Algorithm

This section describes the main concepts of a GA and its operation with the HC and EC approaches. The search space of solutions is the set with all possible solutions for the coverage path planning. In GAs, the search space is represented as a collection of individuals, which forms a population. These individuals are represented by character strings, often referred as chromosomes and each individual has a quality indicator that is calculated with the help of a fitness function. In this case, the chromosome is a permutation of the list of beacons to be visited, and the fitness function is the sum of the covered area by the ASV after visiting the beacons. This definition considers the number of samples that the ASV can take, but at the same time the redundancy of the sensed data. Therefore, we consider as the primary objective the maximization of the covered area, and as the secondary objective, the minimization of redundant data collected. Furthermore, this problem cannot be considered as a simple extension of the TSP because there are some route constrains that must be taken into account, i. e. the route of the drone cannot cross the land. A classical operation of the GA is described in the Algorithm 1, which details the pseudo-code of the GA used:

Algorithm 1: GA Procedure

```
1  Begin GA ;
2  Make initial population ;
3  Compute the fitness function of each individual;
4  while not stop do
5      Selec the best inidividuals (Elite_Group);
6      for Population_Size-Elite_Group)/2 do
7          Select parents from the population;
8          Mutate the children;
9          Replace the parents with its children in the population;
10         Compute the fitness function of each child;
11         Insert the elite group to the population;
12     end
13     if stop criteria then
14         stop = true;
15     end
16 end
17 End GA;
```

First, the initial population of the GA is created, and the quality of each individual is evaluated (line 2 and 3 in Algorithm 1). The best individuals of the group (Elite group) are separated in order to preserve them for the next generation (line 5 in Algorithm 1). New individuals are created from the leftover members of the population by applying a cross-over operator (line 8 in Algorithm 1), and a mutation operator to the new created individuals (line 9 in Algorithm 1). The new generation will consist of these individuals plus the elite group (lines 10 and 12 in Algorithm 1). New generations are created until some condition is satisfied (line 13 in Algorithm 1), for example that the best solution achieves certain level of quality, or a maximum number of generations is reached. In this thesis the later condition was used.

Each individual is represented by the path representation, which is the most natural way to express the solution in a TSP. However, there are other types of representations, like the binary or matrix representations. In the classical TSP, the path representation indicates a possible solution as a list of n cities, so that if the city i is the $i-th$ element of the list, that means that the city i is the $j-th$ city to be visited. Following this representation, Figure 4.1 shows an

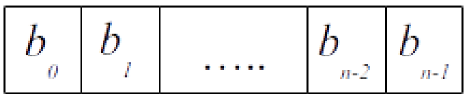


Fig. 4.1 Path Representation

example of a possible path solution for the ASV, being bi is the identification of each of the 60 beacons of the network illustrated in Figure 4.1.

Furthermore, for the HC and EC requirements, the individuals should satisfy the following conditions:

$$indiv_{HC} = [v_1, v_2, v_3, \dots, v_n] \; s.t. \; v_i \neq v_j \; \forall i, j = [1, 2, \dots, n] \tag{4.1}$$

$$indiv_{EC} = [v_1, v_2, v_3, \dots, v_n] \; s.t. \; G_{EC} \; is \; Eulerian \tag{4.2}$$

These representations take into consideration the definitions of HC and EC described in the previous section. Each element of the array has a pair of connections, one incoming from the previous beacon and one outgoing to the next beacon to be visited. According to eq. 4.1, each beacon can only appear once in the individual. Notice that this is a main requirement to form a HC. In eq. 4.2, the individual is forced to form an Eulerian graph, which is the main condition to generate an EC. Next subsection describes the proposed procedure to create random Eulerian graphs.

A mechanism for creating HC individuals for the initial population is proposed and it is presented in Figure 4.2. Basically, this mechanism generates random paths beacon by beacon, verifying: i) if it is already included, and ii) if it is a valid route until the individual with a length of n is obtained. If at some point the next beacon is not found after a user defined number of attempts, the process will restart.

The procedure to generate an initial populations of EC individuals should consider the following restrictions: i) all the vertices in the graph should have an even degree, ii) no invalid routes, and iii) the resulting graph should be connected. The mechanism is shown in Figure 4.3. Notice that this mechanism also includes some correction techniques for finding valid ECs. First, the individual is constructed from a n square matrix called the connection matrix. This matrix has values of 1 and 0 indicating if there is a route between a beacon (rows) and another beacon (columns) or not. The number of $1's$ is distributed randomly across the matrix.

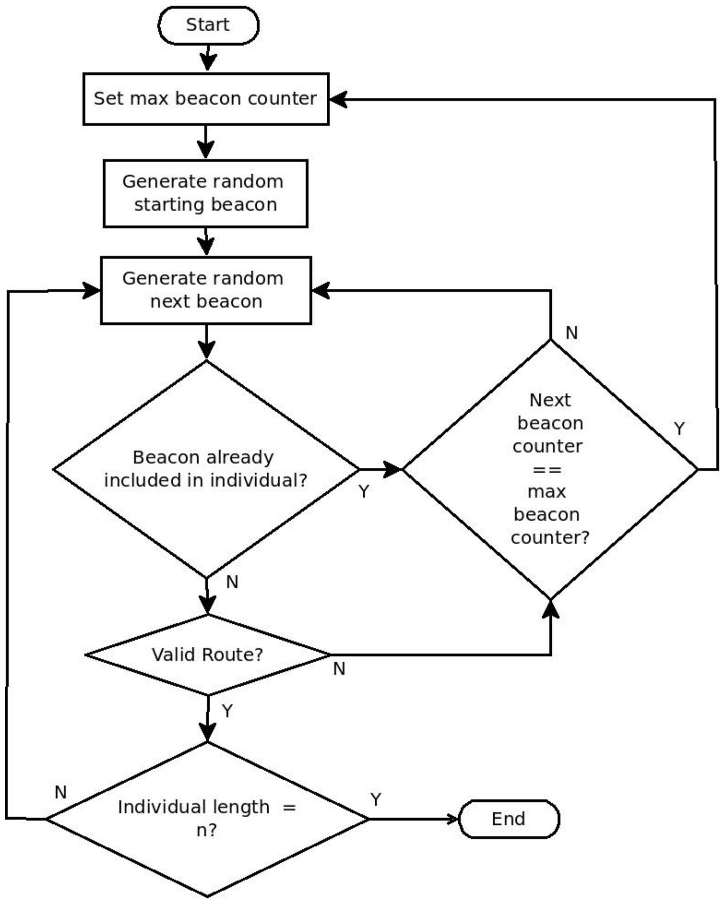

Start
Set max beacon counter
Generate random starting beacon
Generate random next beacon
Beacon already included in individual?
Y
N
Next beacon counter == max beacon counter?
N
Y
Valid Route?
N
Y
Individual length = n?
N
Y
End

Fig. 4.2 Procedure to Create HC Individuals

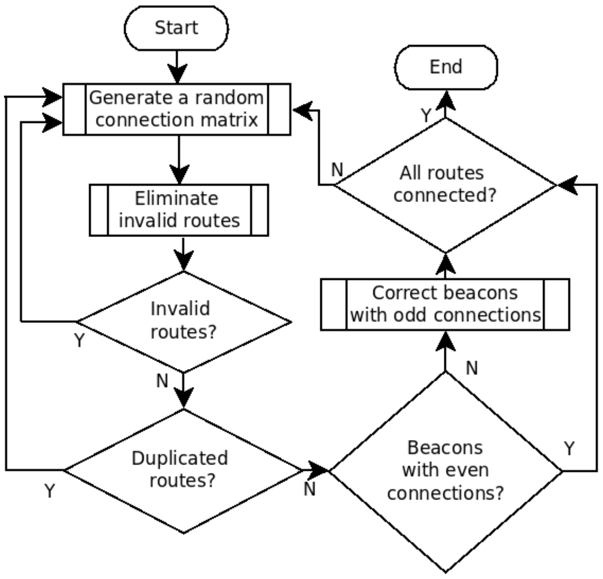

Start
Generate a random connection matrix
Eliminate invalid routes
Invalid routes?
Y
N
Duplicated routes?
Y
N
Beacons with even connections?
Y
N
Correct beacons with odd connections
All routes connected?
N
Y
End

Fig. 4.3 Procedure to Create EC Individuals

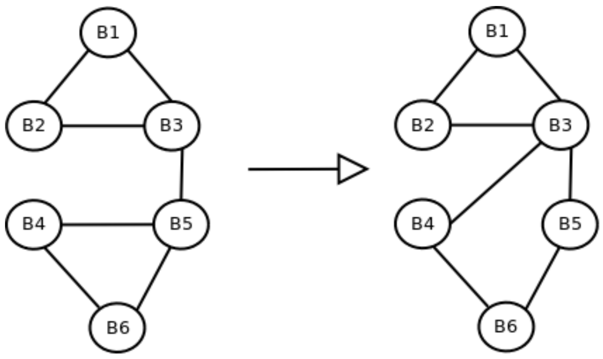


Fig. 4.4 Correction of Nodes with Odd Connections

Next step is to evaluate if there is an invalid or duplicated route. If there are invalid routes, then a correction algorithm moves the 1 at the invalid position to its left side and checks again if this is a valid route. The procedure is repeated until finding a valid route for all the invalid routes of the matrix. Then, there is a verification of repeated routes. Since it is a symmetric graph, the cost or distance is the same going from one beacon to other or vice-versa ($d_{ij} = d_{ji}$). If there are repeated routes, the connection matrix will be created again. Next step is to verify if the Eulerian conditions are satisfied, i.e., if all beacons have an even number of connections. If not, the beacons with odd connections are paired between themselves. An example of this idea is shown in Figure 4.4. On the left, there is a graph with six nodes (beacons) and beacons $b3$ and $b5$ have an odd number of routes. To correct this, the route $b4b5$ is replaced by $b4b3$. Now all the nodes in the graph have an even number of connections, and therefore, the Eulerian condition is satisfied.

The final check is to evaluate whether the resulting graph is connected, meaning that the full path is connected without any sub tour. If this condition is satisfied, then the output is an Eulerian circuit. Finally, the proposed procedure to generate ECs is repeated until the desired size of initial population is generated.

The restriction for invalid routes is implemented in the proposed GA layout by penalizing the fitness function. Two types of penalization are studied, the Death Penalty (DP) and Penalty Factor (PF). The name of the first one derives from the fact that it avoids an individual with invalid routes to create new individuals by assigning a value of -1 to its fitness value. The second type of penalty consists of multiplying the covered area by a factor that is inversely proportional to the number of invalid routes in the solution. The two fitness functions are defined in this manner in order to evaluate if considering only valid individuals for reproduction will conduct to better results and a faster convergence than considering also invalid individuals for reproduction. The mathematical representations are shown next for both fitness functions. First, the DP is defined as eq. 4.3:

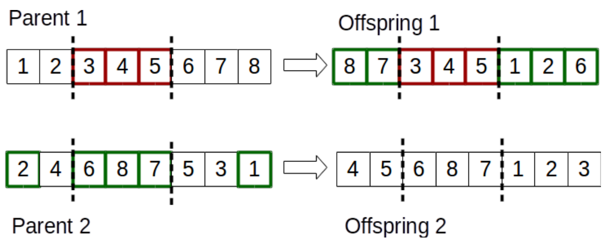


Fig. 4.5 Cross-over Operation

$$fitness_{DP} = \begin{cases} S_{ASV} * ((d_{b_{n-1}b_0}) + \sum_{i=0}^{n-2} d_{b_ib_{i+1}}) - {S_{ASV}}^2 * n_{intersec},\ if\ route = valid \\ -1,\ otherwise \end{cases} \tag{4.3}$$

In the PF fitness function, the individuals are penalized with a factor that multiplies the expression of the valid route in eq. 4.3. This factor is inversely proportional to the number of invalid routes (n_{inv_routes}) in the path of the individual. The fitness function is expressed in eq. 4.4:

$$fitness_{PF} = (1 - \frac{n_{inv\_routes}}{n_{beacons} - 1}) * (S_{ASV} * ((d_{b_{n-1}b_0}) + \sum_{i=0}^{n-2} d_{b_ib_{i+1}}) - {S_{ASV}}^2 * n_{intersec}) \tag{4.4}$$

Regarding the cross-over operator, since it is a permutation problem not all the types of GA cross-overs are allowed. Simply extracting one part of a parent and combining with another part of the second parent may create an individual with repeated elements in its chromosome structure, which is not a valid solution in a permutation problem. The mutation operator is straightforward, because it swaps the values of two positions in the solution. This will not create any issue in the resulting individual.

For a better understanding of the cross-over operation, an example is presented in Figure 4.5. More specifically, it presents a case of an ordered cross-over (line 8 in Algorithm 1) with two individuals represented by a list of 8 elements. Two crossing points are selected at each parent to create the new offspring, for example between the third and fifth elements of the list. This is indicated in Parent 1, in red squares. This part must be copied exactly and at the same position in Offspring 1. The rest of the missing elements will be added starting from the second crossing point in the same order as they appear in the other parent (Parent 2). The missing elements are indicated in green squares, and they are added to Offspring 1, following the order 1, 2, 6, 8 and 7. Offspring 2 is created in the same way, but it is based on Parent 2 and it adds the missing elements as they appeared in Parent 1.

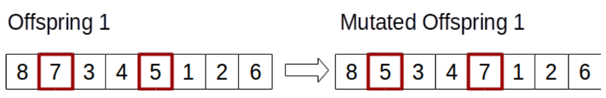


Fig. 4.6 Mutation Operation

Figure 4.6 shows the application of the mutation operator to the Offspring 1 obtained in Figure 4.5 (line 9 in Algorithm 1). The mutation operation selects two positions randomly and inter-exchanges their values between them.

Both cross-over and mutation operators have certain probability to occur; P_c and P_m respectively. While the cross-over produce individuals with the information of their parents, the mutation is used to explore the vicinity of potential solutions obtained from the cross-over.

4.2 Simulation Environment for Static Conditions

The problem was simulated in a Debian 7.9 OS server, with the following specifications, an Intel Xeon v3 E5-2603 1.6 GHz CPU and 64 GB RAM memory. The model was implemented in Python version 2.7.6 [101], where the GA was implemented with the module Distributed Evolutionary Algorithm for Python (DEAP) [102] version 1.0.2. The list of simulation parameters used is shown in Table 4.1 along with a brief description of the meaning of each one. The ordered cross-over exploits a property of the path representation, which is that the order of cities (not their positions) is important. As indicated in [103], the ordered cross-over was found to be one of the best cross-over operators that combines finding a good individual with low convergence time. As it was mentioned earlier, the ordered cross-over constructs a sub-tour of one parent while preserving the relative order of cities of the other parent. In the roulette-wheel selection, each individual in the population is assigned a roulette-wheel slot sized in proportion to its fitness. That is, in the biased roulette wheel, good solutions have larger slot sizes than badly fitted solutions [104]. The roulette wheel is perhaps the most traditional selection operator and provides a moderately strong selection operator that represents a better convergence rate [99]. Regarding the mutation operator, the shuffle index operator was selected due to its easiness implementation. This operator shuffles the attributes (genes) of an individual. Each element is exchanged with another element randomly according to an independent probability inpb (0.05) [105]. The elitism rate is the amount of the best population that will be preserved for the new generation. This mechanism is used to guarantee that the best individuals are not lost during the evolution procedure due to the probabilistic selection. The 20% of best individuals passes directly to the next generation. The population size selected is big enough to ensure a high initial exploration. The number of generations used is sufficient

Table 4.1 Simulation Parameters for HC Modelling

Parameter	**Value**
Number of simulations	100
Population size	100
Number of generations	5,000
Cross-ver type	Ordered (OX1)
Selection	Roulette Wheel
Mutation	Shuffle index ($indpb = 0.05$)
Cross-over probability P_c	0.6, 0.8, 0.95
Mutation probability P_m	0.05, 0.1, 0.2, 0.3, 0.4
Elistism rate	0.2

for the convergence of the genetic algorithm as it will be shown in the next subsection. Table 4.1 collects all the simulation parameters.

4.3 Simulation Results for Static Conditions

This section is divided into three parts. The first part studies the HC approach and includes a tuning of the GA operation parameters, the evaluation of the performance of different types of fitness functions and the comparison of the GA optimization with other simple algorithms.

The second part is focused on the EC approach, and similarly to the HC approach case, it includes a GA parameters tuning and a comparison of the EC approach versus conventional approaches.

Finally, the last part evaluates the performance of the EC approach with the HC approach.

4.3.1 Simulation Results for the HC Model

In this part, the different aspects of the performance of the HC approach are included. First, an adjustment of the GA parameters for obtaining the best results is studied. Particularly, the GA parameters adjusted were the cross-over and mutation probabilities. Next, the performance of the GA is evaluated using the PF and DP fitness functions. Finally, this approach is compared to a random algorithm and the greedy algorithm to evaluate the improvement of the proposed HC approach. The results presented here can be found in [106].

Table 4.2 GA Parameters Tuning with DP Fitness Function ($P_c = 0.6$)

		Mutation rate P_m				
		0.05	**0.10**	**0.20**	**0.30**	**0.40**
Coverage ($x10^6 m$)	Best [m^2]	10.29	10.36	10.48	10.50	10.54
	Aver. [m^2]	10.13	10.22	10.30	10.30	10.30
	Std. Dev.	0.10	0.08	0.10	0.10	0,11

Table 4.3 GA Parameters Tuning with DP Fitness Function ($P_c = 0.8$)

		Mutation rate P_m				
		0.05	**0.10**	**0.20**	**0.30**	**0.40**
Coverage ($x10^6 m$)	Best [m^2]	10.24	10.40	10.46	10.53	10.51
	Aver. [m^2]	10.11	10.22	10.32	10.30	10.31
	Std. Dev.	0.07	0.11	0.09	0.09	0.13

GA Parameters Tuning for the HC Model

Regarding the values of the parameters of the GA used such as the cross-over probability and mutation probability, they were chosen after performing a grid search. Tables 4.2, 4.3 and 4.4 show a summary of the tests performed with a DP fitness function from eq. 4.3 and the roulette wheel selector type. The cross-over operator is used to create new individuals with the hope of obtaining a better individual with the best traits of both parents. The percentage of the population obtained using cross-over is selected for an initial faster evolution of the solution. Particularly the values used were higher than 0.50 (0.60, 0.80 and 0.95). But this operator has limitations to explore new regions in the search space. As a difference, the mutation operator allows the exploration of such new regions. However, the percentage of the population to be mutated should remain low in order to avoid that the algorithm to turn into a pure random walk algorithm. This is why the values selected were below 0.50 (0.05, 0.10, 0.20, 0.30 and 0.40). From Tables 4.2, 4.3 and 4.4; it is seen that the most balanced performance between best solution and average values was obtained from combining $P_c = 0.80$ and $P_m = 0.20$. Larger probability values will result in higher computation cost without obtaining a better impact on the performance.

Table 4.4 GA Parameters Tuning with DP Fitness Function ($P_c = 0.95$)

		Mutation rate P_m				
		0.05	**0.10**	**0.20**	**0.30**	**0.40**
Coverage ($x10^6m$)	Best [m^2]	10.34	10.39	10.47	10.42	10.39
	Aver. [m^2]	10.17	10.23	10.28	10.28	10.31
	Std. Dev.	0.08	0.08	0.09	0.07	0.09

Table 4.5 Comparison of Different Fitness Functions

Case	**Convergence [gens]**	**Best solution [m^2]**	**Aver. [m^2]**	**St. Dev [m^2]**
1	5,000	11,264,927	11,211,800	34,868
2	2,000	10,508,775	10,284,874	92,065
3	3,500	10,546,443	10,357,846	80,369

Comparison of Different Types of Fitness Functions

This section presents the solutions for the cases defined in eq. 4.3 and 4.4. The first case is a fitness function where the restrictions are not included; the second is the DP approach and the third is the PF approach. The numeric values found in the simulations are summarized in Table 4.5. Figure 4.7 and 4.8 show the case where there are no restrictions (unconstrained). The fitness function for this case is eq. 3.1. Figure 4.7 presents the best solution found by the GA. The first element of the solution is the first beacon to be visited from the base station. In this case, the starting beacon is $b22$ (The numbering convention is shown in Figure 3.2). As it can be seen, since no restriction has been applied, the solution includes some invalid routes crossing the land. Figure 4.8 shows the evolution of the solution through the generations. This curve shows two clearly different parts, where initially the improvement rate is fast until the $300th$ generation approximately. After that, the rate begins to saturate and continues improving slowly until the end of simulation ($5,000th$ generation).

In order to avoid the invalid routes, the next approach tested is the use of the DP, with the fitness function shown in eq. 4.3. The results are shown in Figures 4.9 and 4.10, and the best individual found at the end of simulations is presented. The tour starts at beacon $b25$. This beacon is different from the previous set of solutions because it is considered that any beacon could be the first in the tour, i.e., the first one to be visited. Due to the high number of possible solutions it is expected that the starting beacon will not coincide from simulation to simulation.

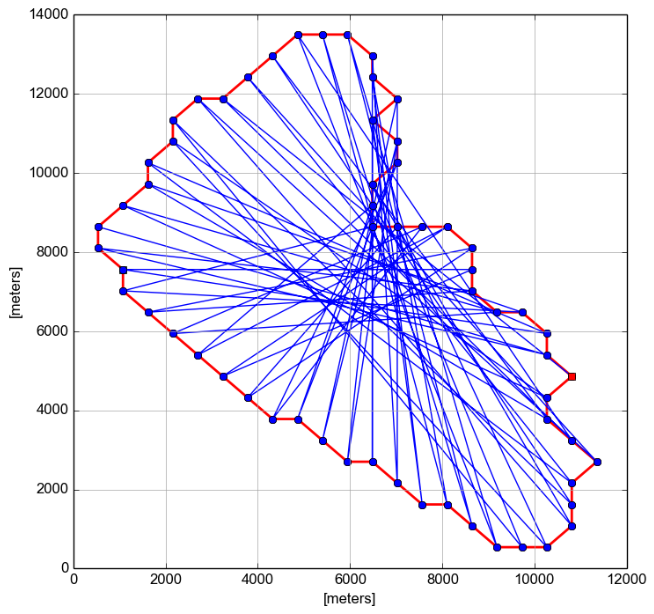


Fig. 4.7 GA with no Restriction (Unconstrained) - Best Individual

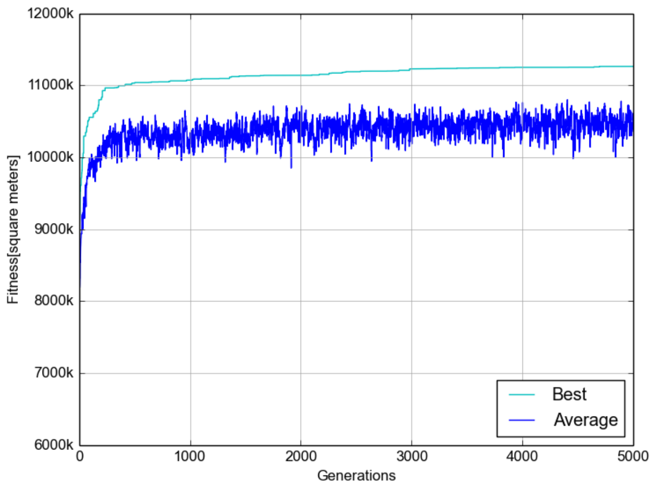


Fig. 4.8 GA with no Restriction (Unconstrained) - Evolution

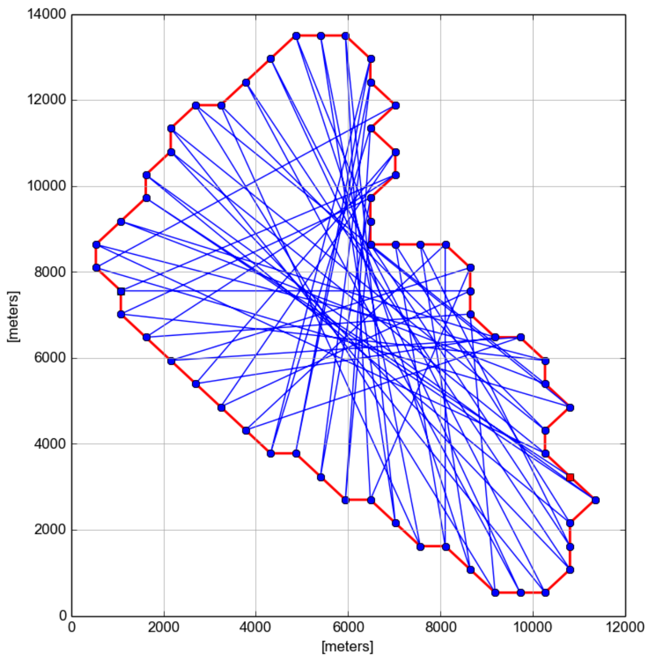


Fig. 4.9 GA with Restriction DP (Constrained) - Best Individual

It is seen now that by introducing the restriction all the routes remain inside the lake. Therefore, the area covered is less than in the previous case; about 7% lower (Table 4.5). In Figure 4.10, the evolution has a slower improvement rate than in the previous case, and around the $2,000th$ generation the solution starts to converge. The average line only considers the valid individuals, and it is seen that they are very close to the best one at each generation, practically overlapping both lines.

The third approach consists of not discarding the individuals with invalid routes but instead applying the penalizing factor. The fitness function is defined by eq. 4.4 and the results of simulations are shown in Figures 4.11 and 4.12. In 4.11, the ASV departs from beacon $b5$. The best solution is similar to the previous case, but regarding the evolution of the best solution, the improvement continues constant until the $4,000th$ and in the end a slightly better solution is found (Figure 4.12).

A closer comparison of the evolution of the three approaches is shown in Figure 4.13. It shows how the first case outperforms the other two but includes invalid routes in the solution. Considering only DP versus PF, initially the DP has higher improvement rate, but after the $2,500th$ generation the DP is surpassed by the PF, which achieves the convergence earlier, around the $3,500th$ generation. This behavior implies that invalid individuals might contain

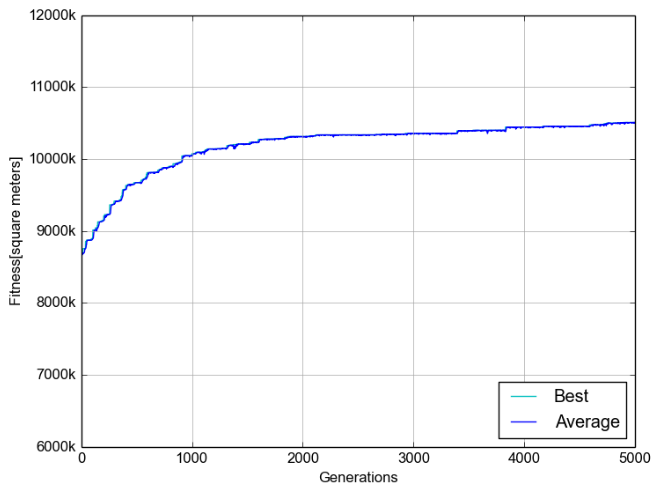


Fig. 4.10 GA with Restriction DP (Constrained)) - Evolution

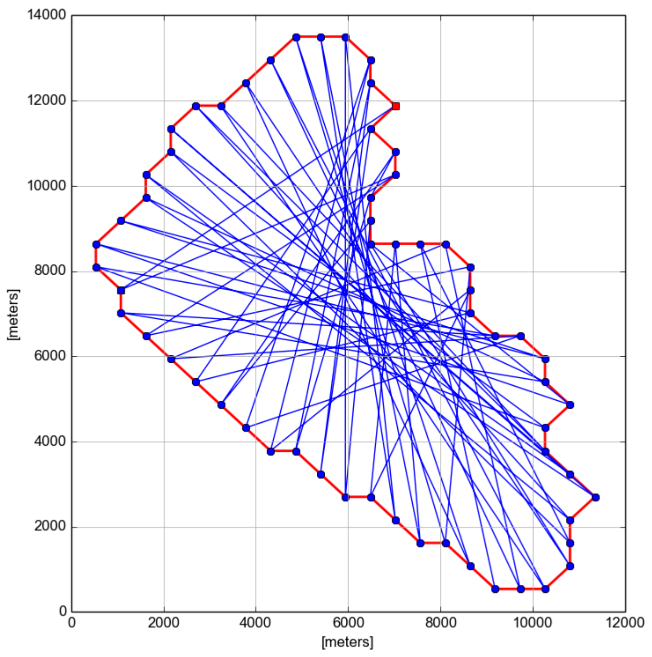


Fig. 4.11 GA with Restriction PF (Constrained) - Best Individual

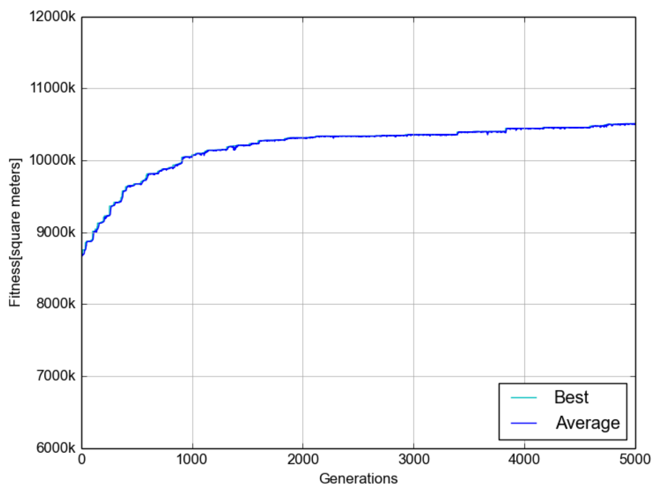


Fig. 4.12 GA with Restriction PF (Constrained) - Evolution

parts of information that can lead to the creation of better individuals faster, and for that reason they should not be discarded in the algorithm.

GA versus Other Simple Approaches

The next part of simulations compares the GA to the randomized algorithm and a greedy algorithm. In the randomized algorithm, the next buoy to visit is found randomly until a complete valid solution is determined. The randomness is given by the random function of Python, which is deterministic and based on the Mersenne Twister [107]. However, the search of the next beacon is limited to $k = 1,000$ attempts, which we called attempt factor. If this number is reached at some point of the searching, a new individual is generated again from the beginning. After testing different values (higher and lower), this value was chosen because it causes the algorithm to find a solution faster. Using a lower value, for example $attempt\_factor = 100$, it will restart the search for individuals too often. Conversely, if $attempt\_factor = 10,000$ is used; it spends too much time on a single individual so that in the end it might not have a complete valid solution (with all beacons). The simulation was run 100 times for finding a population of 100 individuals. On average, it takes about 80 attempts to find an individual under the conditions of the lake (valid routes between beacons).

In the greedy algorithm, the next buoy to visit is the farthest and feasible one to the current position. This algorithm provides only one solution for each beacon that acts as starting point. Because of this, the statistical values about the behavior of the algorithm are taken from comparing the solutions at different starting points, and not from different simulations as the randomized algorithm or the GA do.

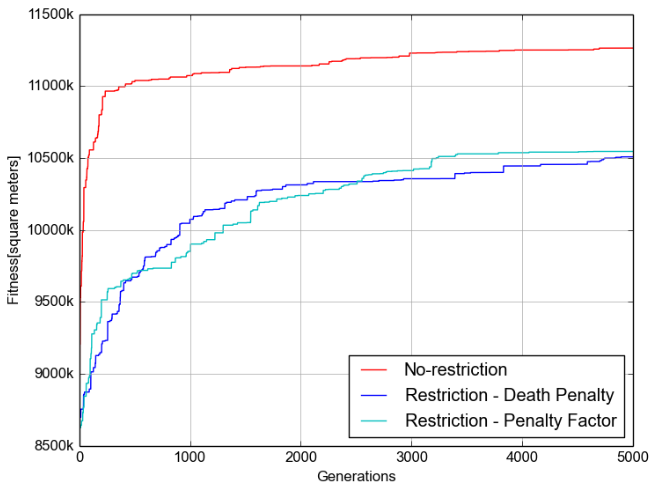


Fig. 4.13 Evolution of Best Individual for Each Case

Figures 4.14 and 4.15 show the solutions found using these techniques. In the randomized algorithm (Figure 4.14) a valid solution was found, i.e. all beacons are visited; and starting at beacon $b28$. However, in the greedy algorithm (Figure 4.15), the solutions found are incomplete because not all beacons are visited. Particularly, the best solution starting at $b34$ is incomplete because the maximum number of visited beacons is 57. From all the possible starting beacons, the best solution is the one starting at $b34$. This tour ends at beacon $b13$ visiting only 57 beacons. Using this starting point, the remaining unvisited beacons, $b5$ and $b8$ and $b10$ cannot be visited from $b13$.

Table 4.5 provides a comparison between the simple approaches and the GA algorithm with the best performance in the previous section, which is the one with fitness function DP. It is seen that the randomized algorithm provides a solution which is 15% worse than the GA algorithm. On the other hand, the greedy algorithm provides better values than the randomized algorithm. However, they are still lower than the GA by 3%. The average is also better in the GA and its standard deviation smaller, which means than in general the GA provides better solutions in a simulation than the greedy algorithm. Furthermore, the greedy algorithm fails to accomplish in visiting all the beacons of the network.

4.3.2 Simulation Results for the EC Model

This section is focused on the EC approach. Again, the GA parameters are adjusted to obtain the best values in terms of cross-over and mutation rates. Then, an algorithm is proposed for the creation of the initial population that satisfies the condition of being an EC individual. Next,

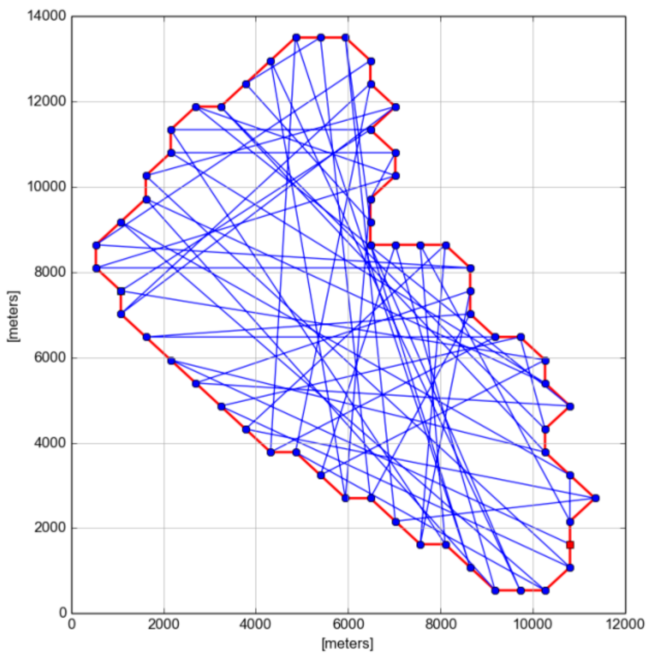


Fig. 4.14 Randomized Algorithm

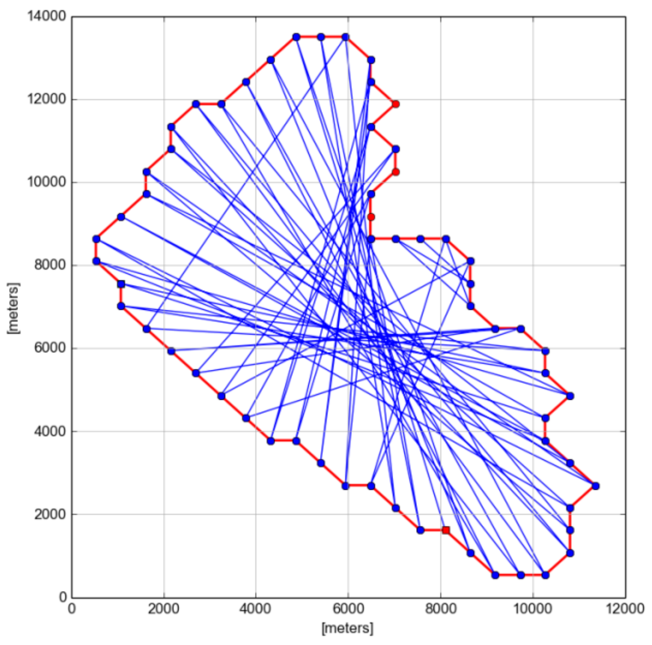


Fig. 4.15 Greedy Algorithm

Table 4.6 Comparison of GA versus Randomized and Greedy Algorithms

Algorithm	**Best solution** [m^2]	**Aver.** [m^2]	**Std. Deviation** [m^2]
Randomized	8,954,399	7,651,377	351,530
Greedy	10,273,260	10,104,920	171,253
GA with fitness 3	10,546,443	10,357,846	80,369

the comparison of the EC GA-based approach with a conventional lawn-mower pattern and random approaches is detailed. Obtained results using this approach led to two publications [108] and [109].

GA Parameters Tuning for EC Model

The first step of the analysis begins with an adjustment of the parameters of the GA, particularly two important parameters like the probabilities of the mutation (P_m) and the cross-over (P_c). The previous HC approach was modeled as a TSP and because each beacon is visited once then the final path had 60 segments. In the EC model the same number of routes is kept, however the beacons can be visited more than once. Under these conditions, the probabilities were varied between 0.05 and 0.20 for the mutation (P_m), and between 0.60 and 0.85 for the cross-over (P_c). The best solution and the average solution of the multiple simulations in terms of coverage are shown in Figure 4.16 and Figure 4.17 and Table 4.7 .

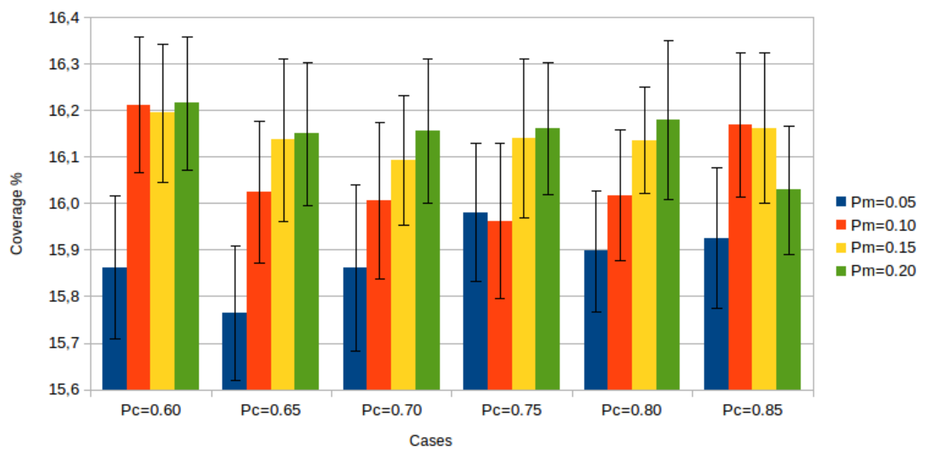


Fig. 4.16 Average Best Solutions of All Simulations

Figure 4.16 presents the average of the best individuals found in the simulations and the confidence interval of 95%. It is seen that the highest average values are obtained with the

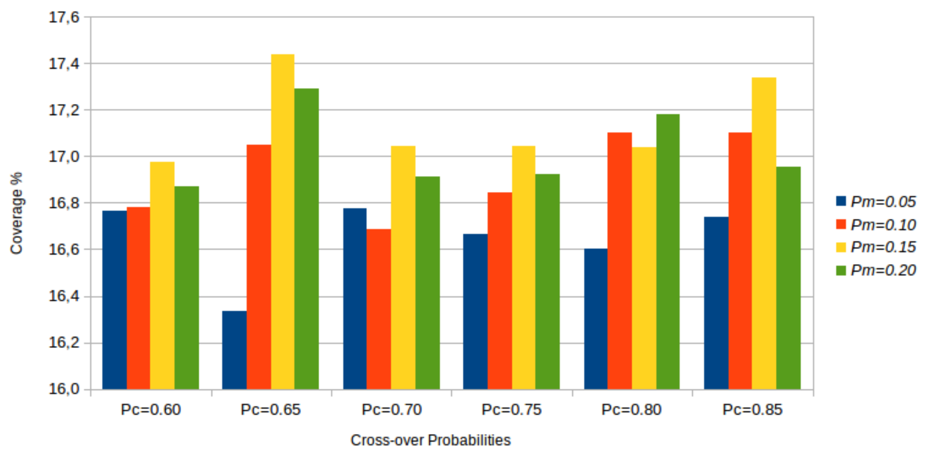


Fig. 4.17 Best Solutions of All Simulations

Table 4.7 Coverage for different Cross-over Probabilities and Mutation Probability $P_m = 0.2$

		Cross-over Rate P_c					
		0.60	0.65	0.70	0.75	0.80	0.85
Coverage [%]	Best	16.87	17.29	16.92	16.92	17.18	16.96
	Worst	15.26	15.19	15.15	15.45	15.12	15.41
	Ave.	16.22	16.15	16.16	16.16	16.18	16.03
	St. Dev.	0.400	0.429	0.433	0.399	0.478	0.389
	Conf. Int	0.143	0.154	0.155	0.143	0.171	0.139

mutation probability P_m equals to 0.20, being the best result when used together with cross-over probability equals to 0.60. But looking at Figure 4.17 , this combination obtains the worst solution in the 0.20 mutation group. Then the next best average in Figure 4.16 is selected, which is $P_c = 0.80$. Now this combination obtained the second-best solution of the 0.20 mutation group and the fourth between all the solutions in Figure 4.17. The numerical values are summarized in Table 4.7. The pair of $P_c = 0.8$ and $P_m = 0.20$ is selected for the following simulations.

The convergence of the algorithm for this pair of parameters is evaluated with the help of Figure 4.18, which show the performance of the best individual (red line) through the generations in the simulations. Also, the averages of the best individuals in the simulations (blue line) as well as the standard deviation (blue shade) around the average are shown. Different conclusions can be obtained from this graph. First, the algorithm improvement converges

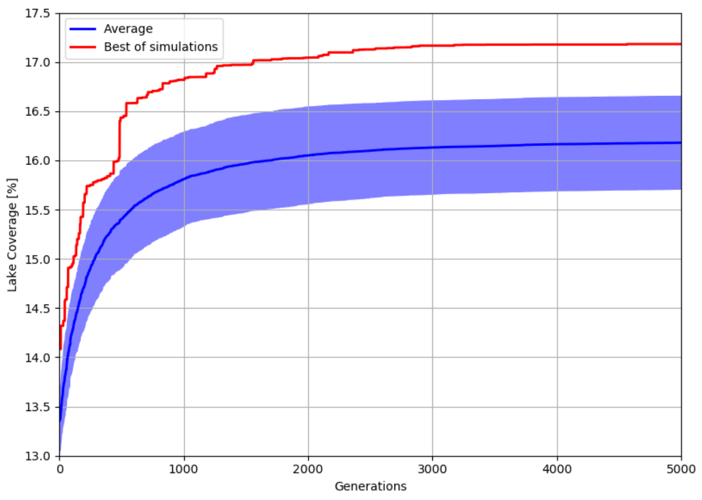


Fig. 4.18 Evolution of the Best Individual ($P_c = 0.8$, $P_m = 0.2$)

around the $3,000th$ generation. The main improvement is given up the $1,000th$, from there it starts decreasing until stabilizing around the $3,000th$ generation. Second, this behavior can be seen in average for all the best individuals. Third, the initial population is important for achieving the best individual because it is seen that the best individual is better than the average already in the beginning, above 14% of the coverage, which is more than 0.5% better than the average initial best, which is below 13.5%. This difference then increases with the generations, reaching to 1% at the end of the simulations.

EC GA-based Approach versus Lawn Mower Pattern and Random Solutions

A conventional approach for covering the lake is the lawnmower problem [110], in which a lawnmower is used to completely cut the grass of a given area with the shortest path possible. This problem is adapted in the proposed modeling considering that the coverage of the ASV at a point is much smaller than the area of the lake and finding a path that covers the whole lake would be impractical. Therefore, the following approach of lawnmower is proposed. Considering that the shape of the lake is narrow it seems adequate to select a horizontal sweep, because each route will be shorter. Starting at an initial beacon, the path is constructed gradually by selecting the next beacon at the opposite shore and located in a region strip below the current beacon. If many beacons are within this region, the one located at the farthest point is selected. This process continues until reaching one of the beacons at the other end of the lake. If at a certain point, before reaching the end, there are no valid routes to continue in the direction of the sweep, then the algorithm selects the side beacon, and attempts to find again a valid route.

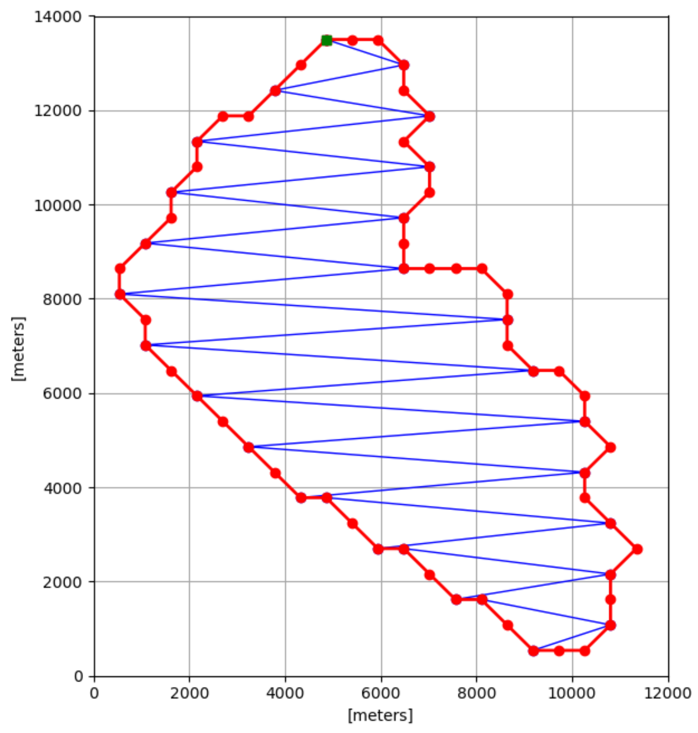


Fig. 4.19 LawnMower Pattern Solution

An implementation of this method is shown in Figure 4.19 in which the starting beacon is $b0$, indicated by a green square, and the region strip is $1,000$ meters below its current position.

The final path consists of 27 segments, with a distance of $141,916$ m and coverage of 4.13%. From these 27 segments, there are two small movements to the side beacon, from beacon $b38$ to $b37$ and $b35$ to $b34$. Now this approach will be compared to the randomized and EC approaches, in which the next beacon of the route is selected randomly, and the proposed approach. In both cases, the same number of routes will be used. The results are summarized in Table 4.8 and the best solutions of each one are shown in the Figure 4.20 and Figure 4.21. The starting points in both figures were chosen randomly and are also indicated by a green square.

It is seen that the randomized approach already improved the lawnmower one because some segments of the route are larger. This result is improved even more with the GA, almost doubling because most segments are between points located at farthest points of the lake. The improvement of the EC is in the order of 112% compared to the Lawnmower, and 35% compared to the random approach. An additional benefit of this approach when compared to the Lawnmower is that the last one performs one complete sweep without repeating any area, while in the proposed one, many regions are visited many times. It is important to emphasize that the results do not achieve the same order of the values found in the adjustment phase because in that case it was considered 60 segments ($n = 60$), and here there are only 27 segments ($n = 27$).

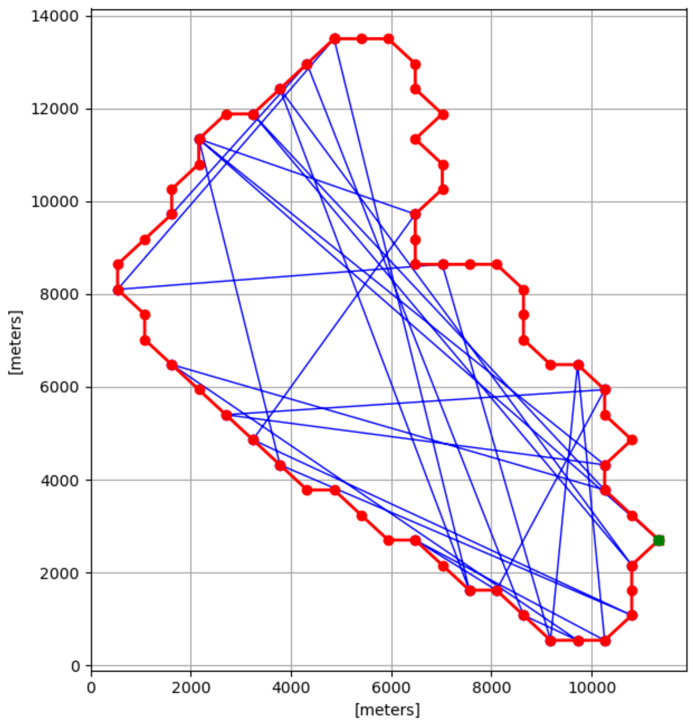


Fig. 4.20 Random Solution

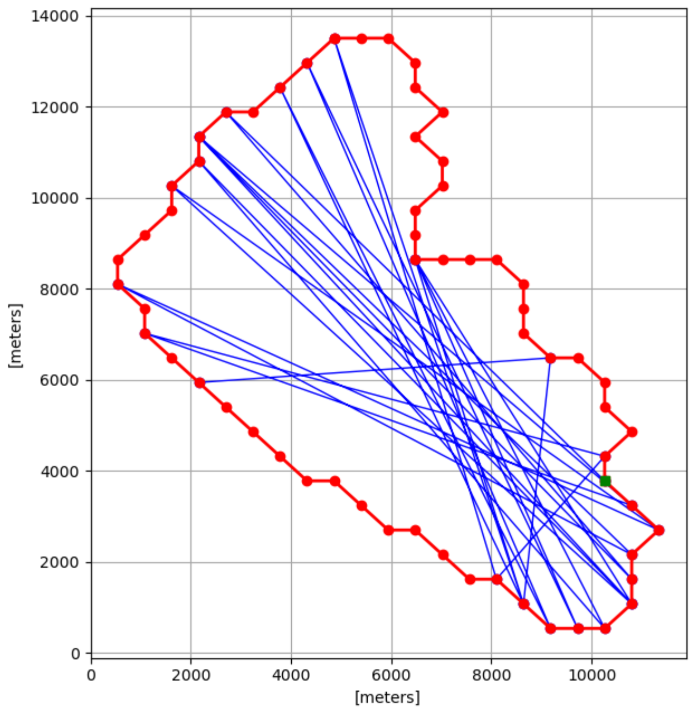


Fig. 4.21 Eulerian Solution

Table 4.8 Comparison EC versus Random and Lawnmower

		Lawnmower	**Randomized**	**EC**
Coverage [%]	Best	4.13	6.47	8.77
	Worst	N/A	4.76	7.45
	Average	N/A	5.49	8.00
	St. Dev.	N/A	0.46	0.28

4.3.3 HC and EC Comparison Analysis

This section is divided into three parts. First, an analysis is performed involving HC and EC to evaluate the capability of finding a path that has the best coverage through a GA. Second, these two models are applied with other stochastic metaheuristics algorithms, like the Iterated Local Search (ILS) and Tabu Search (TS). Then, the HC and EC are compared to a conventional technique for exploring graphs like the Depth First Search (DFS) [111]. Finally, an evaluation of the impact of the number of beacons in the achieved coverage for the EC case is detailed. This is important because it allows the final user to adjust the distance traveled by the ASV according to its energetic autonomy.

HC versus EC Performance

The analyses conducted to evaluate and compare the performance of the proposed approaches are, i) an unconstrained model, i.e., considering invalid routes as part of the solution in the trajectories of the ASV, and ii), the simulation of the constrained model, which rejects invalid routes. The unconstrained model can be seen as a graph without obstacles (invalid routes), where every beacon can be accessed from any other, while in the constrained model there are the obstacles created by the shape of the lake.

The simulation results obtained for the first analysis are shown in Table 4.9, in which the HC and EC models are compared. For each model, three fitness functions are used, the conventional (without considering route intersections), the DP and PF. However, in this first set of simulations, because the invalid routes are disregarded, the DP and PF are the same and equal to eq. 4.5.:

$$fitness_{DP} = fitness_{PF} = \frac{S_{ASV} * PL - {S_{ASV}}^{2} * n_{intersec}}{A_{lake}} \tag{4.5}$$

Table 4.9 Simulation Results of HC and EC Models Considering Invalid Routes (Unconstrained)

		HC		**EC**	
		Conventional	**DP-PF**	**Conventional**	**DP-PF**
Coverage [%]	Best	16.69	16.21	19.77	18.25
	Worst	16.55	15.97	16.83	16.43
	Average	16.62	16.11	17.68	17.53
	St. Dev.	0.04	0.06	0.71	0.47

According to the results of Table 4.9, the EC approach achieves better coverage for all fitness functions. In general, it is an improvement of at least 1% of coverage, with the best overall result achieved by the conventional fitness function with almost 20% coverage of the lake. Using the DP and PF, the coverage is a little above 18% in case of ECs. Figures 4.22 and 4.23 show the best solutions of the HC vs. EC approaches. It can be observed that the routes in the EC-based approach are more spread out throughout the lake. This is possible thanks to repetition of beacons. In the case of HCs, the ASV stays more time in the center area of the lake, which in turns implies a high level of redundancy in the collected data. Also, it can be observed that since invalid routes are not rejected from the possible solutions, they appear especially in routes that involve beacons in the interval $[0,30]$ (on the right shore of the Ypacarai Lake).

The results of the second set of simulations are shown in Table 4.10. In this case, only the performance of the DP and PF fitness functions are tested because solutions with invalid routes are not considered, and the conventional mechanism does not cope with these. In contrast the DP discards this type of solutions and the PF penalizes them. In Table 4.10, the coverage values are lower than in Table 4.9, because many of the previously selected routes were invalid and have to be replaced by shorter but valid ones. The best overall results are achieved by the EC model with DP function with 16.6% coverage of the lake. The EC with PF achieved 15.9% coverage of the lake, but still is about 1% better than the HC model. Table 4.10 includes also a comparison with the randomized algorithm obtained in the previous sub-section.

Figuress 4.24 and 4.25 show the best individuals using the HC and the EC models in a constrained environment (not including the invalid routes). It can be seen that the EC approach provides better coverage of the lake. Again, the routes obtained by HC-based model are more concentrated in the center of the lake. In contrast, in the EC-based solution the GA uses routes that involve beacons with ids ranging from $[47-59]$ and $[27-37]$. These beacons provide larger distances, and consequently, higher coverage levels of the Ypacarai Lake. It is also

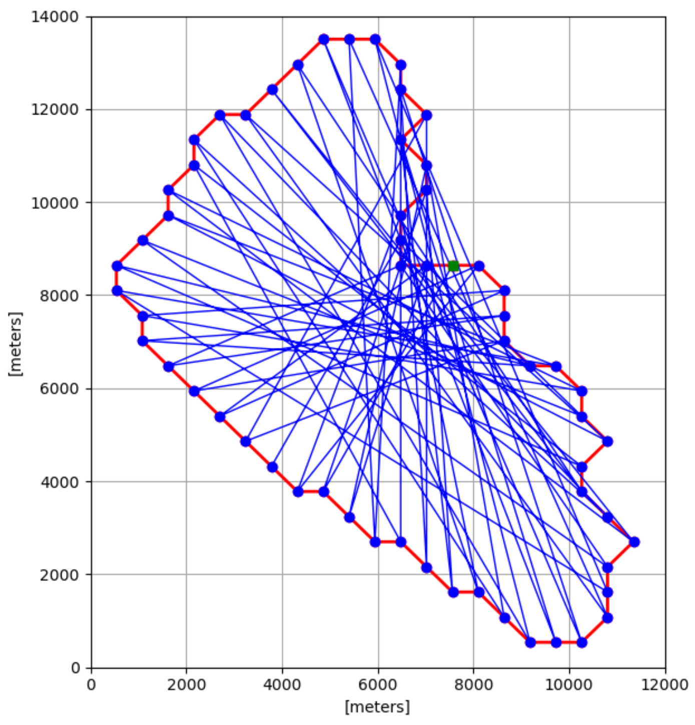


Fig. 4.22 HC Best Solution Including Invalid Path Segments (Unconstrained)

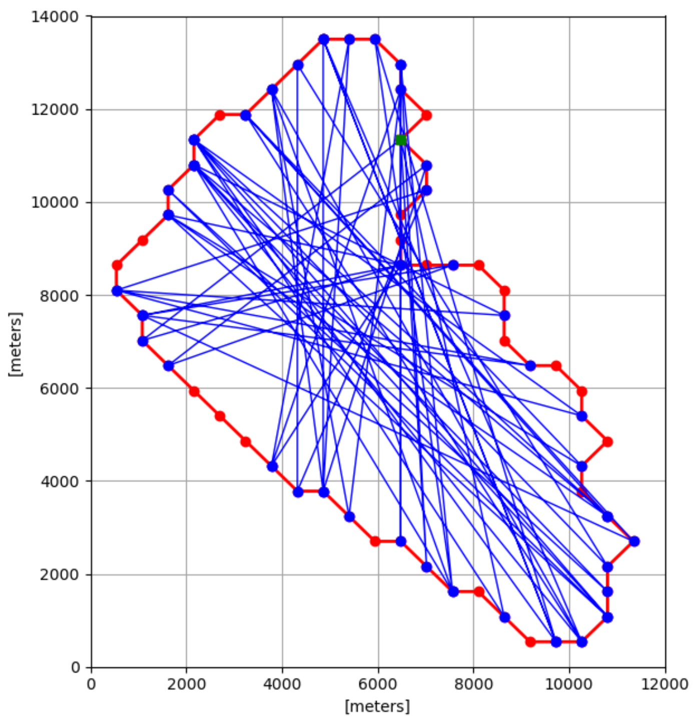


Fig. 4.23 EC Best Solution Including Invalid Path Segments (Unconstrained)

Table 4.10 Simulation Results of HC and EC Models Rejecting Invalid Routes (Constrained)

		HC		**EC**	
		Conventional	**DP-PF**	**Conventional**	**DP-PF**
Coverage [%]	Best	14.82	14.75	16.61	15.89
	Worst	14.23	14.05	14.56	14.44
	Average	14.54	14.42	15.40	15.13
	St. Dev.	0.16	0.19	0.55	0.42

noticeable that from the rest of beacons not all are chosen, and in many cases when one beacon is selected, then its neighbors are disregarded.

According to the results shown in Table 4.9 and Table 4.10, it can be concluded then that for the target CPP in the Ypacarai Lake, the EC-based approach improves the performance of the HC or TSP by almost 2% of the coverage. Considering the size of the lake, this represents a considerable amount of improvement.

HC and EC GA-based Approach Comparison to Other Conventional Metaheuristics Algorithms

The implementation for the simulations of ILS uses a double-bridge move [92] as the perturbation technique and a stochastic 2-opt [112] as the local search technique. The double-bridge move cuts 4 edges and introduce 4 new ones, while the 2-opt is a random exchange of k elements, where k is an adjustable parameter. Furthermore 2-opt is also implemented in the TS as its local search procedure. The results of these techniques for the HC and EC are summarized together with the proposed approach using DP as fitness function in Table 4.11. Additionally, the random solutions for each case are also included. Just as with the case of the proposed approach, the stochastic algorithms were simulated 20 times with $1,000$ iterations per simulation.

First, it is seen how the use of a metaheuristic algorithm improves a simply random approach by 3% for the HC case and more than 4% in the EC case. The GA-DP has slightly worst performance than ILS and TS, however, the proposed EC achieves higher coverage of the lake.

HC and EC GA-based Approach Comparison to a Conventional Exploring Algorithm

A DFS algorith [113] was also evaluated. This algorithm explores the graph conceiving it as a tree structure. It starts at a node and from there, it moves to one of its children located one level

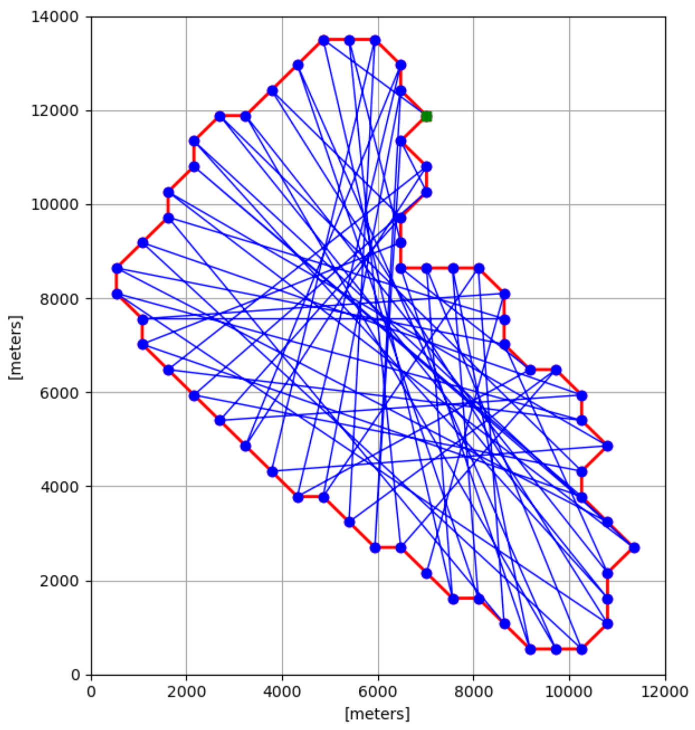


Fig. 4.24 HC Best Solution Rejecting Invalid Path Segments (Constrained)

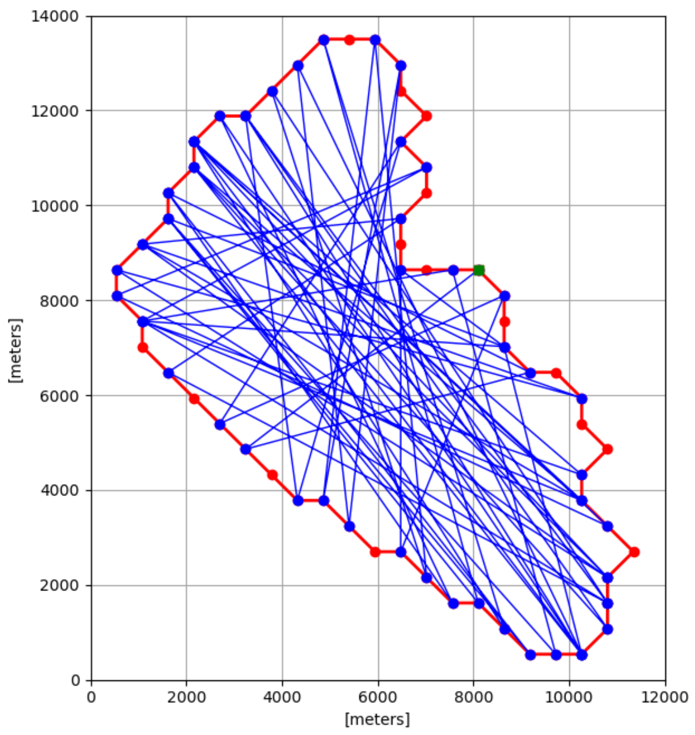


Fig. 4.25 EC Best Solution Rejecting Invalid Path Segments (Constrained)

Table 4.11 HC and EC GA-based versus Alternative Metaheuristics

		HC				EC			
		Random	**ILS**	**TS**	**GA-DP**	**Random**	**ILS**	**TS**	**GA-DP**
Cov. [%]	Best	12.16	15.00	15.48	14.82	12.01	16.21	16.33	16.61
	Worst	10.21	14.55	15.18	14.23	10.23	14.50	14.67	14.56
	Average	11.15	14.80	15.34	14.54	11.26	15.54	15.74	15.40
	St. Dev.	0.50	0.14	0.08	0.16	0.45	0.47	0.44	0.55

Table 4.12 HC and EC GA-based versus Alternative Graph-exploring Algorithm

		DFS	**HC**	**EC**
Coverage [%]	Best	12.67	14.82	16.61
	Worst	11.24	14.23	14.56
	Average	12.12	14.54	15.40
	St. Dev.	0.33	0.16	0.55

below. The criterion for selecting is left to the end user, however in this experiment the nodes with lower beacon ID will have higher priority for being chosen as next node. This procedure is repeated until reaching the desired node or the desired path length. In case that at some node it is impossible to continue descending, then the algorithm returns to its parent and searches an alternative node. In our model, we are constructing a path that has a number of segments equal to the number of beacons, i.e. $n = 60$. The results are summarized in Table 4.12. The exploration techniques only achieve slightly better performance than the random case and the proposed EC approach outperforms it by 4%.

It is seen that the proposed HC and EC models are better than the random algorithm and the conventional DFS. This justifies in the first place the use of metaheuristic techniques like the GA. Furthermore, it is seen that they are better than other stochastic metaheuristic techniques like the ILS. Still, ILS and TS outperform HC but, in that case, the application of the EC model can increase the value of the coverage of the lake.

EC Model Coverage versus Number of Beacons

Removing the restriction of the HC of visiting all the beacons, it is still possible to build a tour with smaller distance if the requirements of the ASV demands to do so. This is required

Table 4.13 Distance Traveled versus Number of Beacons

		Number of beacons					
		10	**20**	**30**	**40**	**50**	**60**
Distance [km]	Best	113.05	213.20	323.50	412.73	499.71	602.87
	Worst	93.35	190.96	276.21	356.75	438.19	513.55
	Average	102.14	200.86	300.15	390.12	474.89	568.91
	St. Dev.	4.50	6.57	10.83	13.56	15.28	25.18

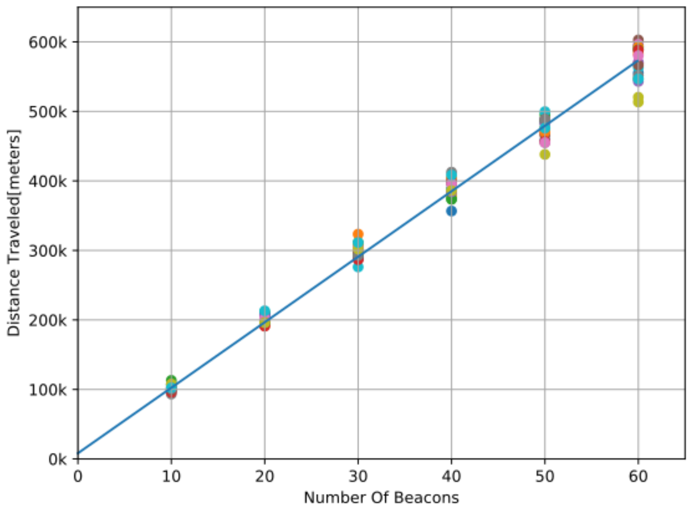


Fig. 4.26 EC Distance Traveled versus Number of Beacons – Multiple Simulations Result

for example if the ASV has an autonomy that limits the distance traveled. Simulations have been carried out to investigate the relationship between the number of beacons (order of the sub-graph G'') and the distance traveled. The number of beacons is varied with the following values $n_{beacons} = 10, 20, 30, 40, 50, 60$, which are smaller values than the ones used for the HC. The results are shown in Table 4.13 and plotted in Figure 4.26.

The results show that there is almost a linear relationship between the distance traveled and the number of beacons which can be approximated as eq. 4.6

$$PL(km) = 10 * n_{beacons} \tag{4.6}$$

This information is useful to the final user, because if the ASV has an autonomy of 300 km, then it is known that the number of beacons should be $n_{beacons} <= 30$. Additional information that can be extracted from Figure 4.26 is that there is a greater dispersion among the simulations as the number of beacons increases, as a consequence of having a higher number of alternatives as the number of beacons increases as well.

4.4 Discussion of Simulation Results for Static Conditions

The following conclusions can be stated from the results obtained in this chapter. First that the use of GA in the constrained HC and EC proposed is a valid tool to optimize the coverage of the lake with an ASV, from a set of random values of an initial population. Furthermore, while the GA is a tool that can be used in several applications, it is important to choose them adequately based on each problem. After performing some analysis in the HC and EC approaches, it is found that adequate values for the cross-over and mutation probabilities are respectively $P_c = 0.80$ and $P_m = 0.20$.

The two proposed fitness functions DP and PF of the GA have advantages and disadvantages. The solutions obtained with the DP tend to have a higher optimization rate that the ones with PF, however they converge a little earlier and in the end the solutions are slightly worse (around 4 Ha. less or 0.4%). Another difference between these two fitness functions is that the DP requires an initial population of valid individuals in order to find a valid solution. The PF does not require this, however there is not guarantee that all the solutions found after running multiple times the GA are valid ones.

The HC approach using DP outperforms other simple approaches to find the best coverage, like a random or a greedy approaches about 15% and 3% respectively. However, the greedy algorithm does not guarantee that the solution is a HC, and some beacons might not be included in the solution. The EC approach using DP was compared to a zig-zag pattern and a greedy approach for a path with 27 routes. The EC approach obtained $4,64\%$ more coverage than the first case and 2.3% higher for the second case, considering a path with 27 routes. The results showed that the EC model with DP using GA achieves better performance than other meta-heuristic tools like the ILS and TS, being these improvements of about 0.4% and 0.3%. Moreover, it has been shown that it is around 4% better than the graph search algorithm DFS.

Finally, the performance of the EC approach is better than the HC approach, and also it has the advantage that a path with any number of routes can be built using this model, which is convenient when having vehicles with different energy autonomy values. In this particular problem, the number of beacons and the distance traveled can be approximate with a value of 10 times.

Chapter 5

Intelligent Online Learning Strategy

This chapter describes the second contribution of the thesis, a reactive path planning algorithm that adapts to the detection of events of interest related to the conditions of the lake. In this case, the event is the appearance of an algae bloom. For an effective monitoring of the phenomenon the strategy consists of two phases, the exploration phase and the intensification phase. At each phase, an adequate path planning for the ASV is found to achieve the objective of the phase. More precisely, in the exploration phase, the path planning is aimed at finding one or several algae blooms in the lake. Once the bloom is located, the intensification phase focuses on a more detailed monitoring in the region.

In the first part of this chapter the strategy is described, then the simulation environment for the evaluation of the performance of the strategy is detailed, and finally the results of these simulations are discussed. Findings lead to one journal publication [114].

5.1 Coverage Path Planning for Dynamic Conditions

A diagram of the proposed strategy is shown in Figure 5.1. It consists of two phases: the exploration phase and the intensification phases. Phase 1 is the exploration phase, in which the ASV movement is chosen to maximize the coverage of the lake. Notice that at this point, no algae bloom has been detected. If an area with a presence of the algae bloom is detected, the strategy moves to the phase 2, in which the ASV movement focuses more on the region with the blooms presence. The ASV stays in this phase until it is noticed that the appearance of the algae bloom is stabilized (not growing or static). If so, the ASV returns to the phase 1, to start exploring again other areas that have been left out during the intensification phase. The basic block diagram of Figure 5.1 is expanded in Figure 5.2 with more level of detail.

An important feature of the proposed strategy is that it learns from the dynamic environment and modifies the GA objective according to the purpose of the current phase of the approach

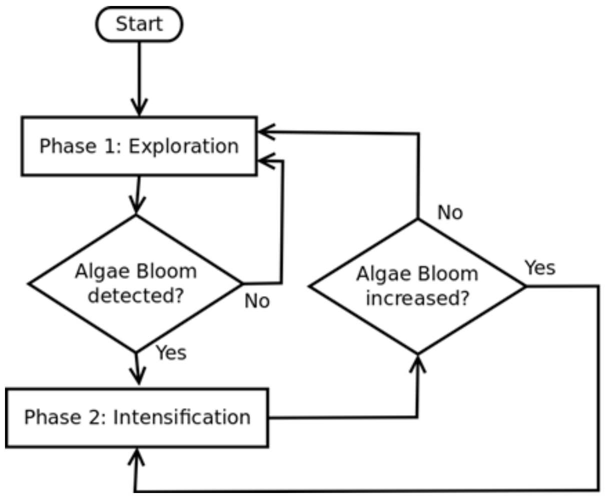


Fig. 5.1 Diagram of the Intelligent Learning Monitoring Approach

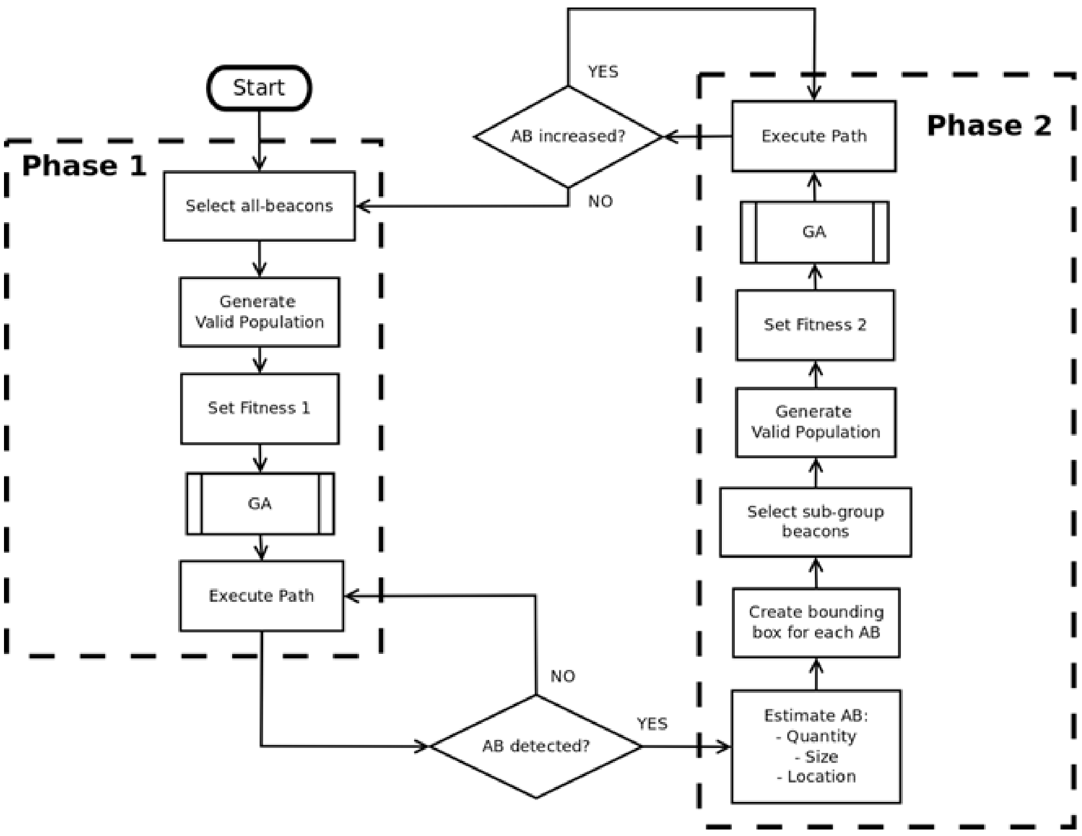


Fig. 5.2 Detailed Diagram of the Intelligent Learning Monitoring Approach

(exploration or intensification). During the first phase the ASV collects the data from the lake to learn and infer its state. Then, it returns the GA objective (fitness function) to collect more data from the polluted region and learn about the environment and its dynamic features. When the ASV detects a change in the event such as the size of the detected event (algae bloom detection indicated by the ASV onboard sensors) has decreased, the ASV adapts again its movements to gather more information about the lake. Notice that this strategy presents a continuous intelligent online learning approach.

5.1.1 Exploration Phase

The strategy begins without any knowledge about the conditions of the lake. Phase 1 will calculate a full exploratory trajectory for the ASV. Since the path planning is modeled as a TSP (HC approach) , the ASV visits the set of data collection beacons ($n_{beacons} = 60$) distributed uniformly at the shore of the lake.

To apply a GA to the problem two elements must be defined: a representation of a possible solution or individual and its quality level or fitness function.

The representation of a possible solution is the same as the one presented in Figure 4.1. As the proposed approach uses the HC model, then there are no repeated beacon numbers in the representation of the individuals. This representation is used in the exploration phase as well as the intensification phase.

Regarding the quality of the solution, this is represented by calculating the fitness function of the individual and it is represented as eq. 5.1. It can be seen that this definition is similar to the PF fitness function. This fitness function was selected as it presented the best results in the simulations of section 4.3 Coverage Path Planning in Static Environments.

$$fitness_{expl} = \frac{(1 - \frac{n_{inv_{routes}}}{n_{beacons}-1}) * (S_{ASV} * (d_{b_{n-1}b_0} + \sum_{i=0}^{n-1} d_{b_i b_{i+1}}) - {S_{ASV}}^2 * n_{intersec})}{A_{lake}} \tag{5.1}$$

,where $n_{inv_{routes}}$ term refers to the number of invalid routes, i.e., the routes that do not travel entirely in the lake; S_{ASV} is the sampling distance, i.e. the radius of the area around a sampling point that is being represented by that sampled measurement; is the distance from beacon i to beacon j; $n_{intersec}$ is the number of route intersections in the solution and A_{lake} is the lake area.

In the detailed diagram Figure 5.2, first an initial population of valid individuals is created, and the fitness function is calculated. Then, it enters to the sub-routine of the GA, which is shown in Figure 5.3. The output of this algorithm is the path with the highest fitness function.

To represent the conditions of the lake, after a path has been defined, the map of the lake is divided in a grid with n_{grid} squares of size s_{square} , Figure 5.4. When an algae bloom is

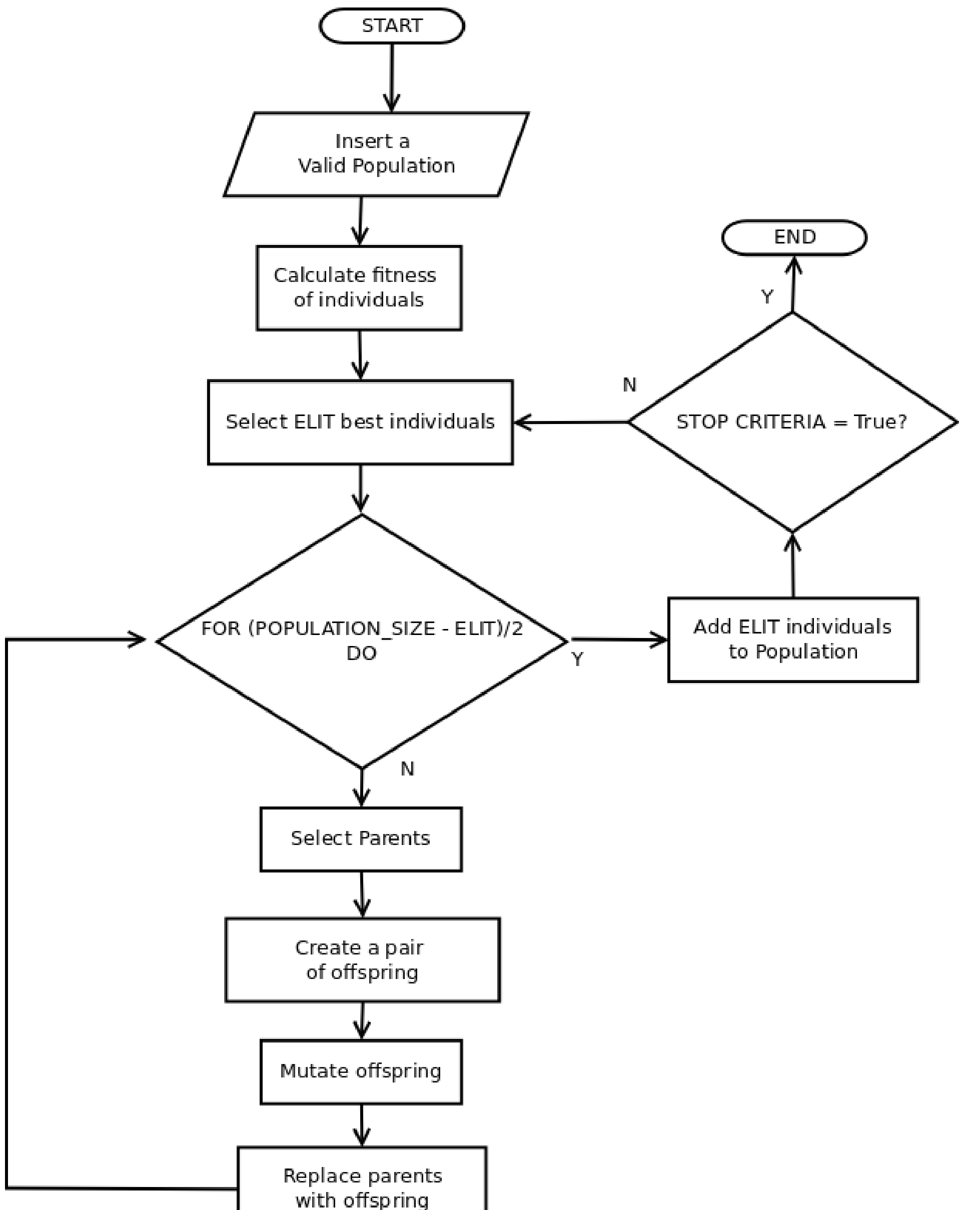

START
Insert a Valid Population
Calculate fitness of individuals
Select ELIT best individuals
FOR (POPULATION_SIZE - ELIT)/2 DO
Y
N
Select Parents
Create a pair of offspring
Mutate offspring
Replace parents with offspring
Add ELIT individuals to Population
STOP CRITERIA = True?
N
Y
END

Fig. 5.3 GA Diagram Implemented in the Intelligent Online Learning Strategy

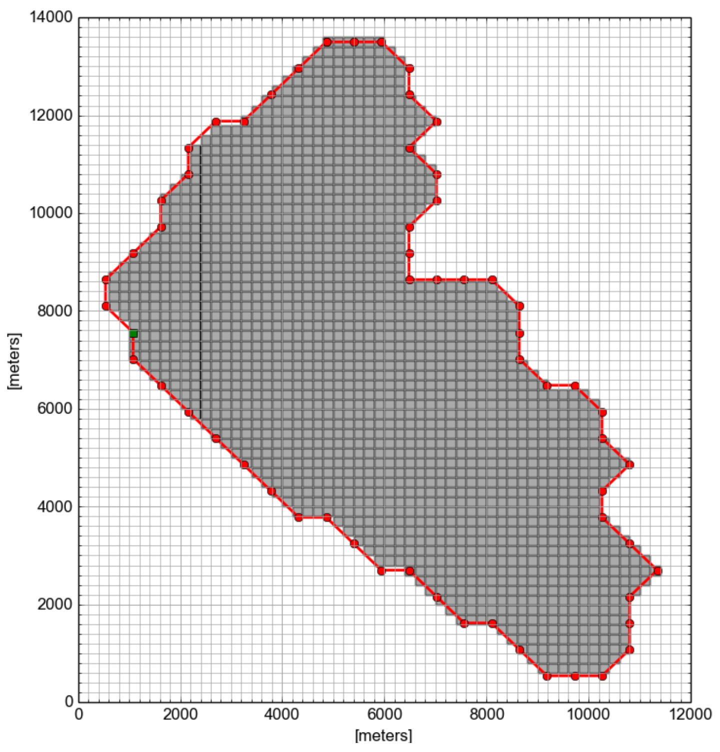


Fig. 5.4 Grid Map of Ypacarai Lake

detected then its location is recorded (square in the grid). An algae bloom is detected if the levels exceed the values defined by the end user; for example, if the levels of cells are above 20,000 or 100,000, according to the risk levels.

After visiting all the beacons, a map of the lake is obtained with all the approximate locations where the evidence of the algae bloom presence has been collected. It is important to notice that the ASV can pass through the same grid square more than once. Whenever a significant presence of the algae bloom is detected, then the ASV moves to the next phase. Otherwise, the ASV performs the same routes as before. As a result of phase 1, the lake map is obtained, which includes the grid squares where the algae bloom was detected and the number of times that this event happened.

5.1.2 Transition Between Exploration And Intensification Phases

The transition to phase 2 is performed if there is one or several areas with a significant presence of the algae bloom. For each one of these events its size and location are stored. With all this information, a bounding box is created around the event, Figure 5.5. Considering that the event is a grid square and given the characteristics of the lake, the bounding box will

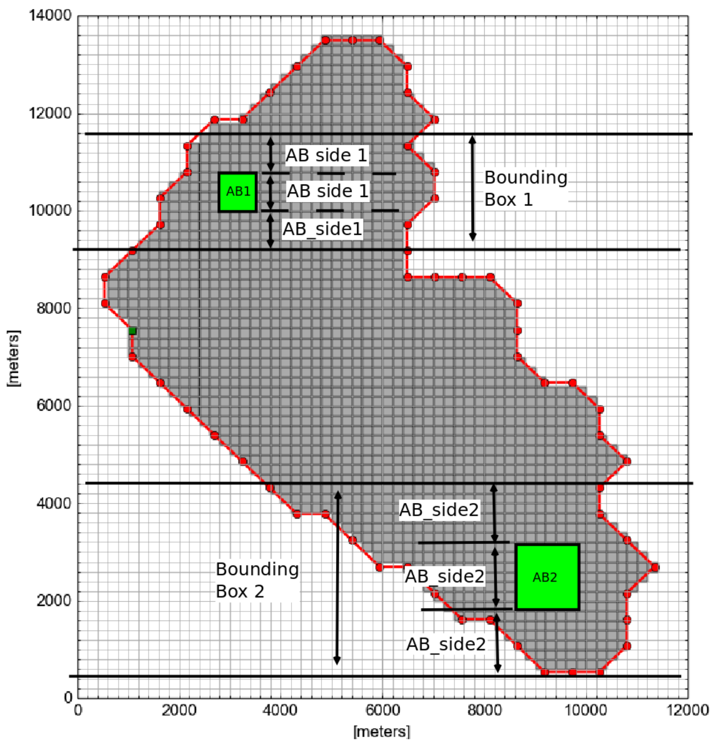


Fig. 5.5 Region for Selecting a Sub-tour Graph

be delimited by a horizontal stripe defined by the upper and lower side of the event grid square; more formally, if there is an estimated square algae blooming (AB) with side S_{AB} and center coordinates (x_{AB}, y_{AB}), the bounding box is the set of grid squares inside the lake with coordinates $(x_{grid\_sq}, y_{grid\_sq})$ that satisfies the condition of eq. 5.2

$$x_{AB} - 1.5 * S_{AB} \leq x_{grid\_sq} \leq x_{AB} + 1.5 * S_{AB} \tag{5.2}$$

As a result, the bounding box is an area that covers the width of the lake and its height is three times the size of the estimated AB size.

5.1.3 Intensification Phase

Once the bounding boxes are defined, a sub-set of beacons (B') inside the bounding boxes are selected to intensify the evaluation of the blooms. The beacons are selected from B if x_i from beacon $b_i (i = 1, 2, 3, \dots, n)$ satisfies eq. 5.3.

$$B' = [b'_0, b'_1, \dots, b'_n],\ where\ m < n - 1 \tag{5.3}$$

Therefore, the number of vertices of the new HC is reduced since a lower number of beacons is selected allowing a faster intensification of the detected areas. After selecting the sub-set of beacons, a new trajectory is calculated using the GA so that the number of times that the ASV passes over the squares that registered the presence of algae bloom is maximized. A new fitness function is defined as eq. 5.4:

$$fit_{int} = \sum_{i=0}^{n_{beacons}-2} n\_alg_{i,i+1} \tag{5.4}$$

Where $n\_alg_{i,i+1}$ is the number of squares with algae that the ASV has detected when moving from beacon i to beacon $i+1$. As a result of the phase 2, a more precise evaluation of the area with the algae bloom will be obtained. Next, the size of the algae bloom regions is recalculated again from the results obtained. If the size of any of the algae regions has increased, then the strategy remains at this phase. However, if all the regions have remained the same size or decreased, then the strategy moves again to the phase 1.

5.2 Simulation Environment in Dynamic Conditions

This section presents the parameters used for the conducted simulations. Table 5.1 contains the parameters used for modeling the lake environment and the characteristics of the events

An event refers to an occurrence of Algae Bloom (AB) area that appears in the lake because of its eutrophication state. For the simulations, it is assumed that the bloom has a square shape, and its size consists of a multiple number of grid squares defined as eq. 5.5 :

$$S_{AB}(km^2) = n_{AB} * S_{square}(km^2) \tag{5.5}$$

, where n_{AB} is the total number of grid squares that the algae bloom occupies in the simulations and S_{square} is the defined size of the square. Two algae bloom areas will be considered in the simulations, Algae Bloom 1 (AB1) and Algae Bloom 2 (AB2). Algae Bloom 2 will remain static during the simulations, while Algae Bloom 1 will vary its size. The variations will occur uniformly around the center of the bloom, i.e. it will increase or decrease

Table 5.1 Environment Parameters for Dynamic Conditions Simulation

Parameter	Value
Number of beacons	60
Distance between beacons (km)	1 (approximately)
Lake size (km^2)	68.72
Number of grids -n_{grids}	4,200
Square size - S_{square} (km^2)	0.04
Square size - S_{side} (km)	0.2
Sampling distance - S_{ASV} (km)	0.02
Algae Bloom 1 size (km^2)	1.00 - (Condition 1)
	2.96 - (Condition 2)
	0.36 - (Condition 3)
Algae Bloom 2 size (km^2)	1.96

uniformly. Figure 5.6 shows the locations of AB1 and AB2, as well as its variation through time. AB1 is located to the upper left side of the lake.

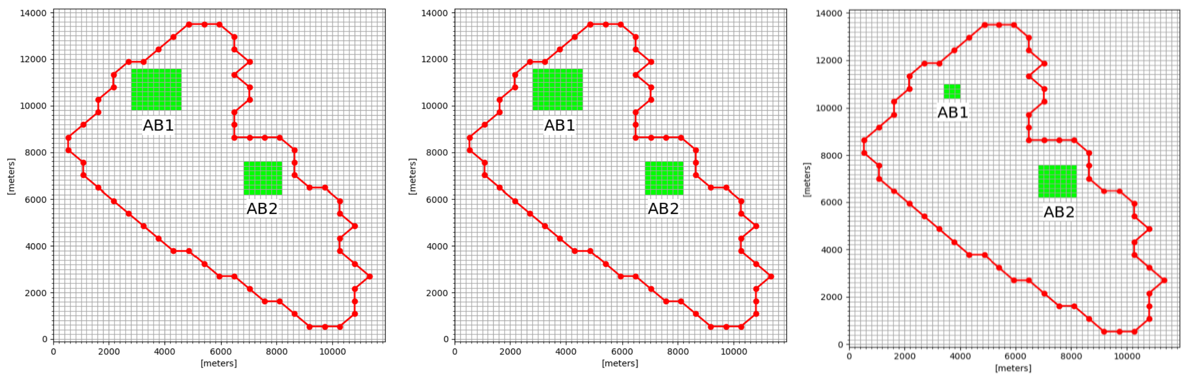


Fig. 5.6 Evolution of the Algae Bloom Regions over Time

The second group of parameters is shown in Table 5.2. and it is related to the GAs used for finding the solution of the path planning. The population size selected is big enough to ensure a high initial exploration. The number of generations used is enough for the convergence of the GA. A roulette wheel mechanism is used for the selection of the parents. The roulette wheel means that individuals with higher fitness have higher slots in the roulette than individuals with lower fitness, so that their probabilities for reproduction are increased [104]. Then the ordered

Table 5.2 GA Parameters for Dynamic Conditions Simulation

Parameter	Value
Number of simulations	10
Population size	100
Number o generations	1,000
Selection	Roulette Wheel
Cross-over type	Ordered (OX1)
Cross-over probability P_c	0.8
Mutation	Shuffle index $(indpb - 0.05)$
Mutation probability P_m	0.2
Fitness	$fit_{int} = \sum_{i=0}^{n_{beacons}-2} n\_alg_{i,i+1}$
	$fit_{expl} = \frac{(1 - \frac{n_{inv_{routes}}}{n_{beacons}-1}) * (S_{ASV} * (d_{b_{n-1}b_0} + \sum_{i=0}^{n-1} d_{b_i b_{i+1}}) - S_{ASV}{}^2 * n_{intersec})}{A_{lake}}$

cross-over exploits a property of the path representation, which is that the order of cities (not their positions) is important [103].

The problem was simulated in the same server from the previous chapter, with Debian 7.9 server and the following specifications, an Intel Xeon v3 E5-2603 1.6 GHz CPU and 64 GB RAM memory. The model was implemented in Python version 2.7.13, where the GA was implemented with the module Distributed Evolutionary Algorithm for Python (DEAP) version 1.0.2. The code of the simulator has been made online available in [115].

The whole simulation scenario was divided in six-time frames, Table 5.3. Each time frame is the period that it takes to the ASV to travel one complete path obtained from our algorithm. The periods can differ considerably because the exploration phase has a broader area to cover than the intensification phase. They were calculated using an estimated average speed of the ASV of 10 km/h, giving a total time of 170 hours to perform the six frames.

5.3 Simulation Results for Dynamic Conditions

Table 5.4 and Table 5.5 shows the behavior of the methodology as it tries to adapt to the dynamic of the bloom regions. At the end of each frame, the ASV has traveled the planned path and the sensed data results are available.

Table 5.3 Simulation Frame Duration

Time Frame	Duration (Hours)	Phase
1	55	Exploration
2	15	Intensification
3	15	Intensification
4	15	Intensification
5	15	Intensification
6	55	Exploration

First, the strategy starts in full exploratory mode and it finds a path that maximizes coverage, Frame 1 in Table 5.4. After traveling this path, the two blooms are located, and the strategy moves to the intensification phase, where a new path with a subset of selected beacons is calculated. The resulting path with its corresponding sampled results is shown in the column, Frame 2 in Table 5.4. Then, the same path is traveled again to continue monitoring these two regions, but at this point a variation is included, Frame 3 in Table 5.4. The top bloom increases its size and the resulting sample shows that there is a variation in the size of the AB. The strategy locates these new conditions and the GA is adapted to the new objective. The resulting path will provide also a new sampled set of results, Frame 4 in Table 5.5. In the next step, Frame 5 in Table 5.5, the same path is traveled again, but the top AB reduces its size. Because of this event, the strategy assumes that the AB starts to disappear, and it is possible to evaluate again other areas in the lake, switching to the exploration mode. The GA is executed at this point considering again all the beacons around the lake and the results are shown in the last step , Frame 6 in Table 5.5.

The numerical results of the simulations are shown in Table 5.6 and Table 5.7, which contains the values obtained at the end of each frame. The first row shows the conditions of the lake during the frame, according to the conditions defined in Figure 5.6. Then, the second row details the phase of the strategy used during the frame. The third row indicates if the GA was executed in this frame. Finally, the next three rows show the performance of the ASV. The fourth row is the distance, the fifth the percentage of AB that has been covered at least once, and the sixth the percentage of AB that has been covered more than once. It is important to notice that these percentages are evaluated considering the estimated size of the bloom.

Note that in Frame 1, the coverage of the bloom is already high. However, by applying the intensification phase, a better first coverage is achieved but with less distance traveled and less redundant samples. This percentage falls in Frame 3 because the size of the bloom increased.

Table 5.4 Simulation Results – Frames 1 to 3

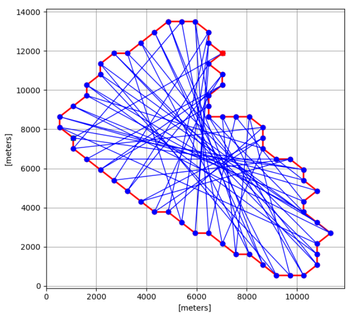
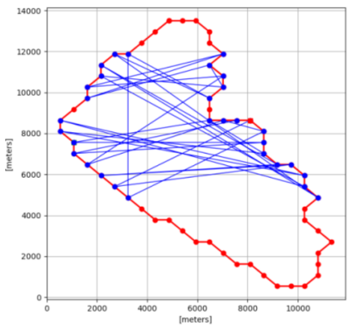
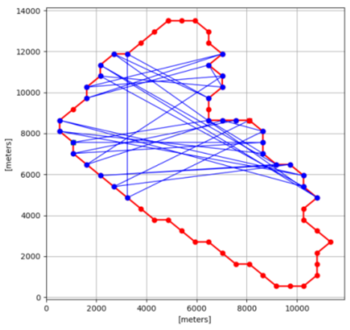
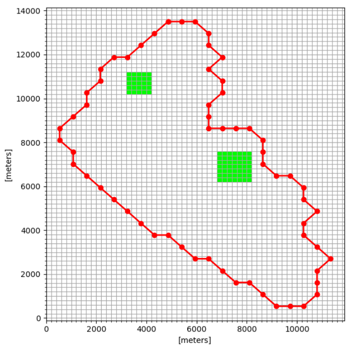
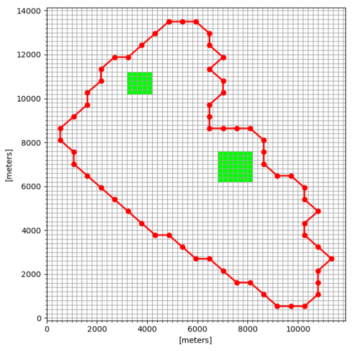
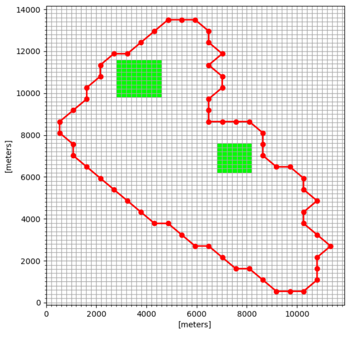
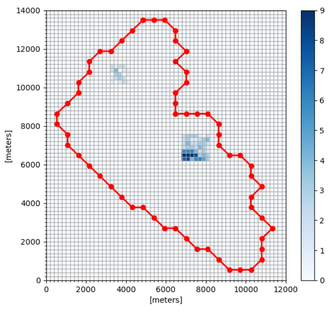
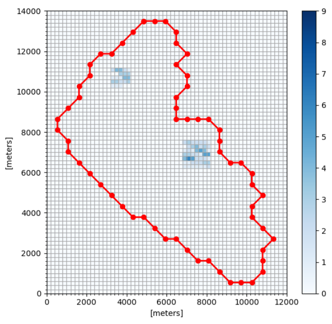
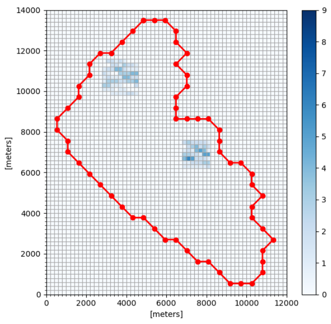
	Frame 1	**Frame 2**	**Frame 3**
Path Planning			
Real Conditions of the lake			
Sampled results			

Table 5.5 Simulation Results – Frames 4 to 6

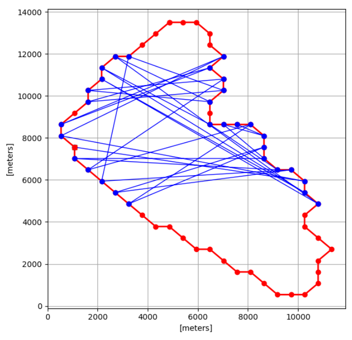
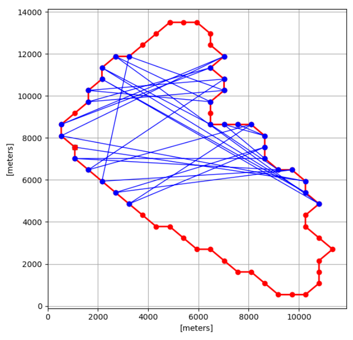
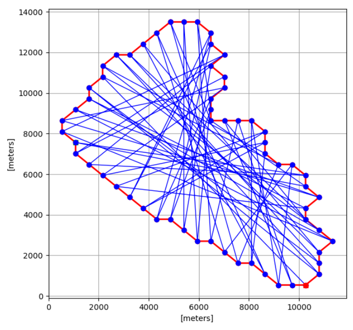
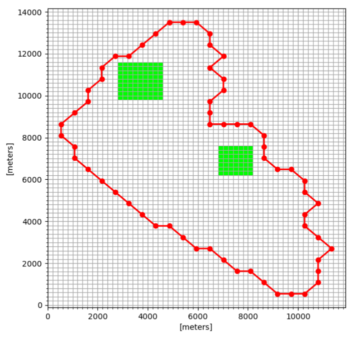
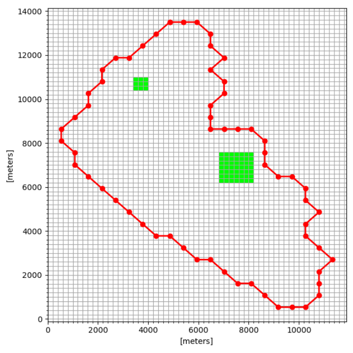
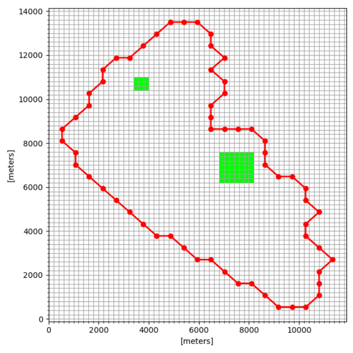
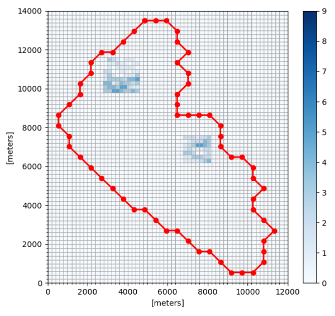
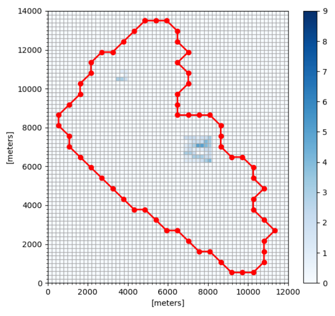
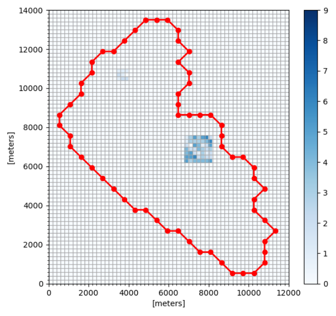
	Frame 4	**Frame 5**	**Frame 6**
Path Planning			
Real Conditions of the lake			
Sampled results			

Table 5.6 Algae Bloom Coverage Results

	Frame 1	**Frame 2**	**Frame 3**	**Frame 4**	**Frame 5**	**Frame 6**
Conditions of the lake (see Figure 5.6)	1	1	2	2	3	3
Phase (see Figure 5.1)	1	2	2	2	2	1
GA used?	Y	Y	N	Y	N	Y
Distance Traveled (km)	536.08	141.88	141.88	162.81	162.81	531.65
First Coverage (%)	90.5	93.2	86.2	87.7	89.7	96.5
Redundant Coverage (%)	200.0	125.7	97.7	110.0	101.7	196.5

The applied strategy makes the GA to increase again this coverage, Frame 4. In Frame 5 and using the same calculated path planning, the coverage increases because the size of the bloom was reduced. Finally, at Frame 6, the ASV returns to the exploratory phase, increasing the coverage, but at the expense of increasing the distance traveled. However, this requirement is accepted considering that the strategy is covering a wider area.

Table 5.7 presents the statistics related to the best individual after running the simulation multiple times. The first row is the total coverage (first + redundant coverage) of the simulation with the best solution. Then, the average and the standard deviation are shown in the following rows. Figure 5.7 represents the results of running multiple simulations in the form of boxplots.

Previous results showed the solutions of the path planning found by the proposed algorithm and how well the solutions achieve the objective of discovering and evaluating the presence of AB in the lake. Next, a closer look about the performance of the GA regarding its capability of finding solutions in the proposed problem is shown. More precisely, the evolution of the GA through the generations is shown in Figures 5.8, 5.9, 5.10 and 5.8. These correspond to frames 1, 2, 4 and 6, respectively (phase 1 or 2, depending on the frame). The yellow line represents the best solution of all simulations, the blue line is the average of best solutions of the different simulations and the blue shaded area is the standard deviation.

Table 5.7 Results of Multiple Simulations

	Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6
Best (Total Coverage %)	290.5	218.9	183.9	197.7	191,4	293,0
Average (Total Coverage %)	253.8	202.2	165.1	187.7	190.2	266.7
Standard Deviation (Total Coverage %)	37.5	9.4	9.6	7.7	15.9	26.7

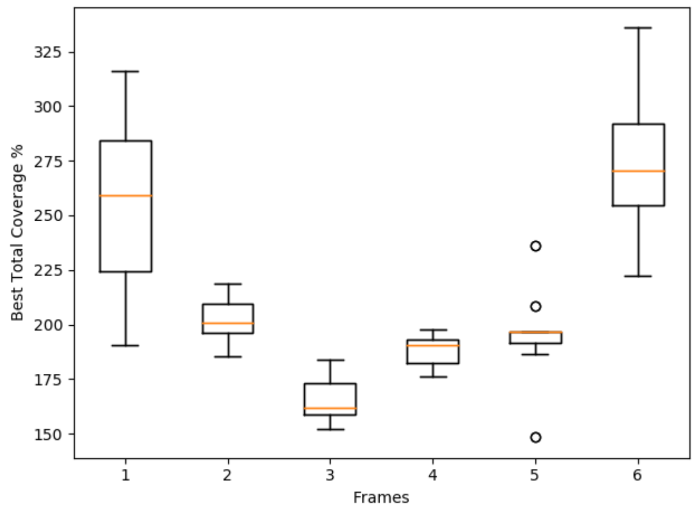
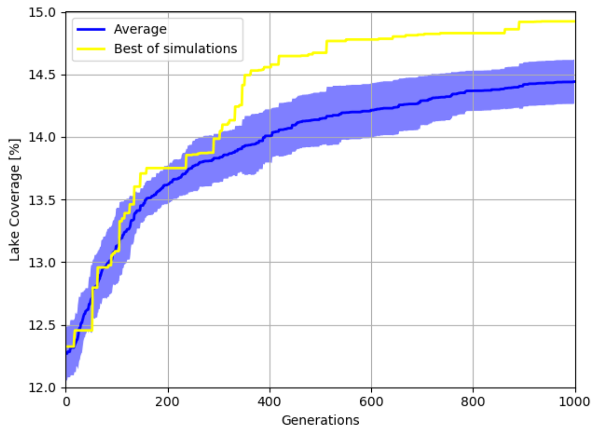


Fig. 5.7 Boxplot of Multiples Simulations



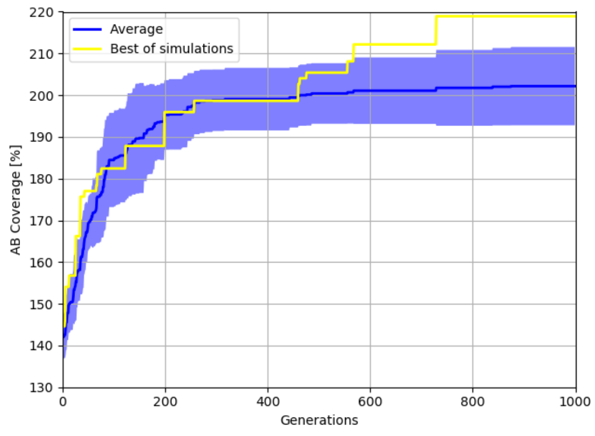


Fig. 5.8 Evolution of Solutions - Frame 1

Fig. 5.9 Evolution of Solutions - Frame 2

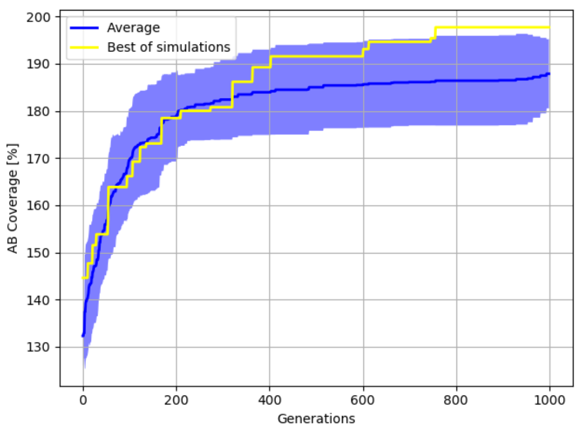


Fig. 5.10 Evolution of Solutions - Frame 4

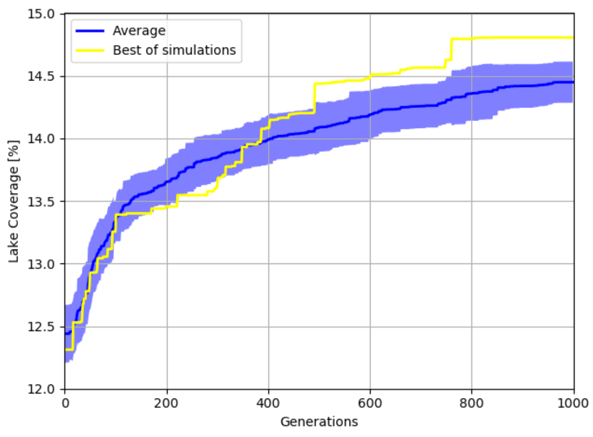


Fig. 5.11 Evolution of Solutions - Frame 6

In the previous simulations, the GA was executed four times for frames 1, 2, 4 and 6. In frames 1 and 6 the fitness functions correspond to eq. 5.1, and it gives the coverage of the lake in the exploratory phase. It is seen in Figure 5.8 and 5.11 the best coverage of the lake is near 15% and that in average the best solution starts to converge around the generation 1,000. It is important to mention that the standard deviation remains similar throughout all the generations (shaded blue region).

In frames 2 and 4 (Figure 5.9 and 5.10), the fitness function is given by the total number of squares sampled with algae bloom (eq. 5.4) divided by the estimated number of algae bloom squares. In other words, it is the percentage of the estimated AB covered. In this case, the values are the ones shown in Table 5.7 (Best total coverage). This is also the same as the sum of the first coverage plus the redundant coverage values of Table 5.6. The intensification phase results in a 219% total coverage in Frame 2, while in Frame 4 in a 198% total coverage. Compared to the exploration phase, the average converges earlier, since the number of beacons in the solution is smaller. However, the standard deviation is larger, leading to think that there is a greater variance of the best solution from simulation to simulation. In order to balance the computational effort of the algorithms in both cases, it could be run less times in the exploration phase while increasing the number of the generations. On the other hand, in the intensification phase, the number of generations could be reduced and increase the number of simulations. Another way of interpreting this result is that the initial population has greater impact in the intensification phase than in the exploration phase. This is because in the exploration phase the number of constrains is higher than in the intensification phase, meaning that the search space of this last phase is much wider.

5.4 Discussion of Simulation Results for Dynamic Conditions

The proposed adaptive mechanism includes two operation phases, one to explore the lake and the other to intensify a region where an event of interest is discovered. This idea has been implemented in the GA by adjusting the algorithm in the following ways, first by selecting a fitness function which maximizes the explored area or the coverage of an event of interest; and second by selecting the full set of beacons to perform the exploration or only a sub-set of beacons around the area of the event.

The results for the case where two regions of interest were simulated, showed that the coverage levels remain high (around 90% of the covered area by the event), when moving to the intensification phase while the distance traveled is 25% of the exploration distance. Also, there is less redundant measured data.

Other conclusion that can be observed from the multiple simulations is that in the exploration phase it is not necessary to run the simulations a large number of times, because the standard deviation is relatively small when compared to the intensification phase in which there is a greater standard deviation. But because the number of beacons of the intensification phase is smaller, then the computational effort to calculate the solution is less and perhaps it is possible to use a larger number of simulations without increasing the computational load.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The study of Autonomous Surface Vehicles (ASVs) is an active ongoing research in robotics because of its many potential applications. Among all the potential applications, environmental monitoring is one of the best suited for ASVs. There are many advantages of using autonomous vehicles for monitoring applications, e.g., the vehicles can be built smaller and remotely operated. During the last 25 years, many prototypes have been built for testing the key concepts of its operation in real scenarios.

The main system to be developed for autonomous operation is the GNC system. Inside the Guidance subsystem, a mission planner that indicates the path the ASV should follow in order to complete its assigned mission. For the particular case being studied, the vehicle must follow a path that provides a good coverage of the Ypakarai lake for monitoring its water conditions. Moreover, the planner should be prepared to re-calculate its path to adapt to the dynamic conditions of the environment, for example in order to avoid obstacles or to approximate to regions of interest.Starting from these considerations, this thesis was focused on developing a path planner for an ASV that fulfills these requirements and operates in the Ypakarai lake for environmental monitoring tasks.

In the first contribution, a planning technique was proposed to find the path, using a support infrastructure of wireless data beacons as intermediate waypoints. Because the number of possible sequences is very high, an optimization algorithm known as GA was used, that maximizes the covered area of the Ypakarai Lake by the ASV. The contributions of this work include the proposal of two graph-based models to solve the problem, the EC and the HC; a GA parameters tuning for this problem and two types of fitness function operators called the DP and the PF. The simulations demonstrate the benefits of using the GA to solve the problem in comparison with other conventional techniques such as a random algorithm or greedy algorithm. Additionally the GA has shown to be an adequate optimization tool to find the solution when compared with other conventional meta-heuristics ones like the TS and ILS (0.3% and 0.4%), and other graph search algorithm like the DFS (around 4%). Between the EC and the HC models the first one achieves near 2% higher coverage performance. Finally, it was found that when using the EC approach there is a near linear relationship between the number of beacons and the distance traveled, meaning that the number of beacons can be selected by user according to the autonomy of the vehicle.

In the second contribution, the dynamics of the environment have taken into account of the proposed model. Considering that an event of interest might be detected during the monitoring, for example an HAB; the adaptive proposed mechanism is able to re-calculate the path in order to allow the ASV to take more measurements. Particularly, the fitness function is adjusted if the ASV is in exploration mode or intensification mode. The results of the simulation showed that the proposed technique achieves similar coverage while reducing the distance and time of travel. As a consequence, less energy is required to monitor the event of interest.

6.2 Future Work

As future work, several lines can be followed. In terms of applications, the proposed approaches can be applied to other complex scenarios with unpredictable factors where the use of autonomous vehicles is suitable, such as relief actions in disaster scenarios [116], [117], [118], and vehicular ad hoc networks [119], among others.

Additionally, in terms of the metaheuristics algorithms, a future work would include the application of other nature-inspired algorithms like the Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) or a hybrid version of both, to our model. There are many Python libraries available online like NIApy [120] that can let us easily implement our problem and obtain new results. Furthermore, there are other online Python libraries, which are specific to mobile robotics like the library called Python Robotics [121] that can be used to extend the present work.

One of the difficulties found during the development of the thesis was to determine the appropriate parameters of the GA for a specific problem. However if the conditions change, the same parameters can lead to non optimal solution. Then, Hyper-heuristics have arisen to automate the design and tuning of heuristics algorithms [122]. The main characteristic of a hyper-heuristic is that it operates on a space of heuristics rather than on a space of solutions. For example, one evolutionary algorithm can be used to tune another one that is used to find the solution of the problem, or even the same type of evolutionary algorithm can be used for both tasks [123]. Another alternative for adjusting our GA is the use of another technique like the Bayesian optimization framework [?], in which a learning algorithm's generalization performance is modeled as a sample from a Gaussian process (GP). Selecting an adequate type of kernel and the treatment of its hyperparameters, can lead to a expert level performance optimizer.

Moreover, the problem can be expanded as a multi-objective problem [124], which will include conflicting variables that are tried to be optimized simultaneously. For example, it would be the case for the distance traveled and the energy consumption of the vehicle, just to mention a couple. After building the Cormoran-II, the energy characteristics of the vehicle can be obtained and included in the optimization of the multi-objective problem.

This thesis has covered the global path planning, however for real deployment local path planning must be included in the decision making. Among the recent studied algorithms, two groups can be highlighted, the group of Any-Angle algorithms, which are derived from the A* (e.g. Theta*) and find shorter paths on a continuous environment [125]. The other group is related to the Fast Marching Method (FMM), which is based on the family of potential fields-based algorithms. These algorithms improve the limitations of local minima that normally happens with the potential fields approach. Furthermore, it has been demonstrated that this approach was implemented in a real ASV like the Springer [126].

In terms of real obstacle avoidance, the Collision Regulation (COLREG) [127], can also be added to the algorithm of the ASV. These rules have been developed by the International Maritime Organization (IMO) and describe the rules that vessels at sea should follow to prevent collisions between two or more vessels at sea. In fact, a recent work shows that this is an ongoing research topic in ASVs [128].

Additionally, the proposed approach can be improved by including the restrictions that the routes would not exceed certain values, reducing the risk of reaching the limits of the autonomy of the vehicle. Another improvement is considering the frequency of visited regions.

Furthermore, the proposed strategy can be modified according to the extension of the TSP called Vehicle Routing Problem. In this case, a fleet of vehicles is used to complete the task. In addition to developing large autonomous surface vehicles, there are also other works towards the implementation of smaller vehicles with lower cost that would allow the practical implementation of the approach. This field is known as Swarm Robotics [129], and takes inspiration mainly from swarms found in nature like ants, bees, etc. Still there is plenty of room for research until reaching a decentralized,robust, versatile and scalable system.

6.3 Publication List

This subsection presents the complete list of publications derived from this thesis.

Book chapter:

1. Arzamendia, M., D. Gregor, D. G. Reina, and S. L. Toral. "A Path Planning Approach of an Autonomous Surface Vehicle for Water Quality Monitoring Using Evolutionary Computation." In Technology for Smart Futures, pp. 55-73. Springer, Cham, 2018.

Papers in peer reviewed journals:

1. Arzamendia, Mario, Derlis Gregor, Daniel Gutierrez Reina, and Sergio Luis Toral. "An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai Lake." Soft Computing (2017): 1-12.
2. Arzamendia, Mario, D. Reina, S. Toral, Derlis Gregor, Eleana Asimakopoulou, and Nik Bessis. "Intelligent Online Learning Strategy for an Autonomous Surface Vehicle in Lake Environments using Evolutionary Computation." IEEE Intelligent Transportation Systems Magazine (2018). In Press.
3. Arzamendia, M., I. Espartza, D. G. Reina, S. L. Toral, and D. Gregor. "Comparison of Eulerian and Hamiltonian circuits for evolutionary-based path planning of an autonomous surface vehicle for monitoring Ypacarai Lake." Journal of Ambient Intelligence and Humanized Computing (2018): 1-13.
4. Sánchez-García, J., J. M. García-Campos, M. Arzamendia, D. G. Reina, S. L. Toral, and D. Gregor. "A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications." Computer Communications (2018).

Papers in peer reviewed international conferences:

1. Arzamendia, M., D. Gregor, D. G. Reina, S. L. Toral, and R. Gregor. "Evolutionary path planning of an autonomous surface vehicle for water quality monitoring." In Developments in eSystems Engineering (DeSE), 2016 9th International Conference on, pp. 245-250. IEEE, 2016.
2. Arzamendia, M., Reina, D.G., Marin, S.T., Gregor, D. and Tawfik, H., Evolutionary Computation for Path Planning of Autonomous Surface Vehicles using Eulerian Graphs. In Evolutionary Computation (CEC), 2018 IEEE Congress on, pp. 546-553. IEEE, 2018.

References

[1] G. A. Lopez Moreira, L. Hinegk, A. Salvadore, G. Zolezzi, F. Holker, R. A. Monte Domecq S., M. Bocci, S. Carrer, L. De Nat, J. Escriba, C. Escriba, G. A. Benitez, C. R. Avalos, I. Peralta, M. Insaurralde, F. Mereles, J. M. Sekatcheff, A. Wehrle, J. F. Facetti-Masulli, J. F. Facetti, and M. Toffolon, "Eutrophication, Research and Management History of the Shallow Ypacaraí Lake (Paraguay)," *Sustainability*, vol. 10, pp. 1–32, July 2018.

[2] Centro International Hidroinformatica (CHI), "Bathymetry Studies of Ypacarai Lake," August 2014.

[3] H. W. Paerl, J. C. Meeks, and R. Haselkorn, "Mitigating Harmful Cyanobacterial Blooms in a Human-and Climatically-Impacted World," *Life*, vol. 4, pp. 988–1012, 2014.

[4] P. B. Ochumba and D. I. Kibaara, "Observations on blue-green algal blooms in the open waters of lake victoria, kenya," *African Journal of Ecology*, vol. 27, no. 1, pp. 23–34, 1989.

[5] A. M. Michalak, E. J. Anderson, D. Beletsky, S. Boland, N. S. Bosch, T. B. Bridgeman, J. D. Chaffin, K. Cho, R. Confesor, I. Daloglu, J. V. Depinto, M. A. Evans, G. L. Fahnenstiel, L. He, J. C. Ho, L. Jenkins, T. H. Johengen, K. C. Kuo, E. Laporte, X. Liu, M. R. McWilliams, M. R. Moore, D. J. Posselt, R. P. Richards, D. Scavia, A. L. Steiner, E. Verhamme, D. M. Wright, and M. A. Zagorski, "Record-setting algal bloom in Lake Erie caused by agricultural and meteorological trends consistent with expected future conditions.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 16, pp. 6448–6452, 2013.

[6] B. Qin, W. Li, G. Zhu, Y. Zhang, T. Wu, and G. Gao, "Cyanobacterial bloom management through integrated monitoring and forecasting in large shallow eutrophic lake taihu (china)," *Journal of Hazardous Materials*, vol. 287, pp. 356–363, 2015.

[7] ABC Digital, "Cartes promete solucionar contaminación del ypacaraí." http://www.abc.com.py/nacionales/cartes-promete-solucionar-contaminacion-del-ypacarai-538294.html, 2013. [Online; accessed April-2018].

[8] Centro Multidisciplinario Investigaciones Tecnologias (CEMIT), "Informe tecnico de la sexta campaña de muestreo," tech. rep., National University of Asuncion (UNA), 2015.

[9] L. Wang, X. Wang, X. Jin, J. Xu, H. Zhang, J. Yu, Q. Sun, C. Gao, and L. Wang, "Analysis of Algae Growth Mechanism and Water Bloom Prediction Under the Effect of Multi-affecting Factor," *Saudi Journal of Biologica Sciences*, vol. 24, no. 3, pp. 556–562, 2017.

[10] J. C. Ho and A. M. Michalak, “Challenges in Tracking harmful Algal Blooms: A Systhesis of Evidence From Lake Erie,” *Journal of Great Lakes Research*, vol. 41, no. 2, pp. 317–325, 2015.

[11] J. Sanchez-Garcia, J. Garcia-Campos, M. Arzamendia, S. L. Toral, and D. Gregor, “A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications.,” *Computer Communications*, 2018.

[12] S. Singh, S. E. Webster, L. Freitag, L. L. Whitcomb, K. Ball, J. Bailey, and C. Taylor, “Acoustic communication performance of the whoi micro-modem in sea trials of the nereus vehicle to 11,000 m depth,” in *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pp. 1–6, IEEE, 2009.

[13] M. Dunbabin, A. Grinham, and J. Udy, “An autonomous surface vehicle for water quality monitoring,” in *Australasian Conference on Robotics and Automation (ACRA)*, pp. 2–4, 2009.

[14] J. Fraga, J. Sousa, G. Cabrita, P. Coimbra, and L. Marques, “Squirtle: An asv for inland water environmental monitoring,” in *ROBOT2013: First Iberian Robotics Conference*, pp. 33–39, Springer, 2014.

[15] P. Miller, A. Beeler, B. Cayaban, M. Dalton, C. Fach, C. Link, J. MacArthur, J. Urmenita, and R. Y. Medina, “An easy-to-build, low-cost, high-performance sailbot,” in *Robotic Sailing 2014*, pp. 3–16, Springer, 2015.

[16] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, “Unmanned surface vehicles: An overview of developments and challenges,” *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[17] M. Schiaretti, L. Chen, and R. R. Negenborn, “Survey on autonomous surface vessels: Part ii-categorization of 60 prototypes and future applications,” in *International Conference on Computational Logistics*, pp. 234–252, Springer, 2017.

[18] J. E. Manley, “Development of the autonomous surface craft "aces",” in *OCEANS’97. MTS/IEEE Conference Proceedings*, vol. 2, pp. 827–832, IEEE, 1997.

[19] J. E. Manley, A. Marsh, W. Cornforth, and C. Wiseman, “Evolution of the autonomous surface craft autocat,” in *Oceans 2000 MTS/IEEE Conference and Exhibition*, vol. 1, pp. 403–408, IEEE, 2000.

[20] J. Curcio, J. Leonard, and A. Patrikalakis, “Scout-a low cost autonomous surface platform for research in cooperative autonomy,” in *OCEANS, 2005. Proceedings of MTS/IEEE*, pp. 725–729, IEEE, 2005.

[21] J. Alves, P. Oliveira, R. Oliveira, A. Pascoal, M. Rufino, L. Sebastiao, and C. Silvestre, “Vehicle and mission control of the delfim autonomous surface craft,” in *Control and Automation, 2006. MED’06. 14th Mediterranean Conference on*, pp. 1–6, IEEE, 2006.

[22] N. Cruz, A. Matos, S. Cunha, and S. O. da Silva, “Zarco-an autonomous craft for underwater surveys,” *7th Geomatic Week*, 2007.

[23] H. Ferreira, A. Martins, A. Dias, C. Almeida, J. M. Almeida, and E. P. Silva, “Roaz autonomous surface vehicle design and implementation,” *Robótica Controlo, Automação, instrumentação*, 2007.

[24] A. Martins, H. Ferreira, C. Almeida, H. Silva, J. M. Almeida, and E. Silva, “Roaz and roaz ii autonomous surface vehicle design and implementation,” in *International Lifesaving Congress 2007*, 2007.

[25] M. Caccia, M. Bibuli, R. Bono, G. Bruzzone, G. Bruzzone, and E. Spirandelli, “Unmanned surface vehicle for coastal and protected waters applications: The charlie project,” *Marine Technology Society Journal*, vol. 41, no. 2, pp. 62–71, 2007.

[26] M. Caccia, R. Bono, G. Bruzzone, E. Spirandelli, G. Veruggio, A. Stortini, and G. Capodaglio, “Sampling sea surfaces with sesamo: an autonomous craft for the study of sea-air interactions,” *IEEE robotics & automation magazine*, vol. 12, no. 3, pp. 95–105, 2005.

[27] E. Pinto, F. Marques, R. Mendonca, A. Lourenco, P. Santana, and J. Barata, “An autonomous surface-aerial marsupial robotic team for riverine environmental monitoring: Benefiting from coordinated aerial, underwater, and surface level perception,” in *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014*, pp. 443–450, 2014.

[28] D. Machado, A. Martins, J. M. Almeida, H. Ferreira, G. Amaral, B. Ferreira, A. Matos, and E. Silva, “Water jet based autonomous surface vehicle for coastal waters operations,” in *Oceans- St John’s*, pp. 1–8, 2014.

[29] J. Busquets, F. Zilic, C. Aron, and R. Manzoliz, “AUV and ASV in twinned navigation for long term multipurpose survey applications,” *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*, no. 1, 2013.

[30] A. Valada, P. Velagapudi, B. Kannan, C. Tomaszewski, G. Kantor, and P. Scerri, “Development of a Low Cost Multi-Robot Autonomous Marine Surface Platform,” *Springer Tracts in Advanced Robotics*, vol. 92, pp. 643–658, 2014.

[31] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen, “Evolution of collective behaviors for a real swarm of aquatic surface robots,” *PLoS ONE*, vol. 11, no. 3, pp. 1–25, 2016.

[32] G. S. Sukhatme, A. Dhariwal, B. Zhang, C. Oberg, B. Stauffer, and D. A. Caron, “Design and Development of a Wireless Robotic Networked Aquatic Microbial Observing System,” *Environmental Engineering Science*, vol. 24, no. 2, pp. 205–215, 2007.

[33] F. Fornai, F. Bartaloni, G. Ferri, A. Manzi, F. Ciuchi, and C. Laschi, “An autonomous water monitoring and sampling system for small-sized asv operations,” in *Oceans, 2012*, pp. 1–6, IEEE, 2012.

[34] G. Hitz, F. Pomerleau, M.-E. Garneau, C. Pradalier, T. Posch, J. Pernthaler, and R. Siegwart, “Autonomous Inland Water Monitoring: Design and Application of a Surface Vessel,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 62–72, 2012.

[35] E. Romero-Vivas, F. Von Borstel, C. Pérez-Estrada, D. Torres-Ariño, J. Villa-Medina, and J. Gutiérrez, "On-water remote monitoring robotic system for estimating the patch coverage of anabaena sp. filaments in shallow water," *Environmental Science: Processes & Impacts*, vol. 17, no. 6, pp. 1141–1149, 2015.

[36] L. Pedersen, T. Smith, S. Y. Lee, and N. Cabrol, "Planetary lakelander—a robotic sentinel to monitor remote lakes," *Journal of Field Robotics*, vol. 32, no. 6, pp. 860–879, 2015.

[37] J. Laut, E. Henry, O. Nov, and M. Porfiri, "Development of a mechatronics-based citizen science platform for aquatic environmental monitoring," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 5, pp. 1541–1551, 2014.

[38] S. Jung, H. Cho, D. Kim, K. Kim, J.-I. Han, and H. Myung, "Development of algal bloom removal system using unmanned aerial vehicle and surface vehicle," *IEEE Access*, vol. 5, pp. 22166–22176, 2017.

[39] K. Sedighi, K. Ashenayi, T. Manikas, R. Wainwright, and H.-M. Tai, "Autonomous Local Path Planning for a Mobile Robot Using a Genetic Algorithm," in *Congress on Evolutionary Computation*, pp. 1338–1345, 2004.

[40] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pp. 1–8, IEEE, 2013.

[41] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE transactions on computers*, no. 2, pp. 108–120, 1983.

[42] A. R. Soltani, H. Tawfik, J. Y. Goulermas, and T. Fernando, "Path planning in construction sites: performance evaluation of the dijkstra, a-star, and ga search algorithms," *Advanced engineering informatics*, vol. 16, no. 4, pp. 291–303, 2002.

[43] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[44] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[45] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[46] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE access*, vol. 2, pp. 56–77, 2014.

[47] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., 1998.

[48] L. Kayraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configurations space," *Proc IEEE Trans Robot Autom*, vol. 12, no. 4, pp. 566–80, 1996.

[49] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, “Metaheuristic algorithms in modeling and optimization,” in *Metaheuristic applications in structures and infrastructures*, pp. 1–24, Elsevier, 2013.

[50] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, “Heuristic approaches in robot path planning: A survey,” *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[51] H. Duan and L. Huang, “Imperialist competitive algorithm optimized artificial neural networks for ucav global path planning,” *Neurocomputing*, vol. 125, pp. 166–171, 2014.

[52] X. Chen, Y. Kong, X. Fang, and Q. Wu, “A fast two-stage aco algorithm for robotic path planning,” *Neural Computing and Applications*, vol. 22, no. 2, pp. 313–319, 2013.

[53] Y. Zhang, L. Wu, and S. Wang, “Ucav path planning by fitness-scaling adaptive chaotic particle swarm optimization,” *Mathematical Problems in Engineering*, vol. 2013, 2013.

[54] I. Chaari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and K. Al-Shalfan, “Smartpath: An efficient hybrid aco-ga algorithm for solving the global path planning problem of mobile robots,” *International Journal of Advanced Robotic Systems*, vol. 11, no. 7, p. 94, 2014.

[55] M. A. K. Jaradat, M. H. Garibeh, and E. A. Feilat, “Autonomous mobile robot dynamic motion planning using hybrid fuzzy potential field,” *Soft Computing*, vol. 16, no. 1, pp. 153–164, 2012.

[56] H. Duan, Y. Yu, X. Zhang, and S. Shao, “Three-dimension path planning for ucav using hybrid meta-heuristic aco-de algorithm,” *Simulation Modelling Practice and Theory*, vol. 18, no. 8, pp. 1104–1115, 2010.

[57] E. Alba and C. Cotta, “Evolutionary algorithms,” *Handbook of bioinspired algorithms and applications*, pp. 3–19, 2006.

[58] A. Alvarez, A. Caiti, and R. Onken, “Evolutionary path planning for autonomous underwater vehicles in a variable ocean,” *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004.

[59] A. Tuncer and M. Yildirim, “Dynamic path planning of mobile robots with improved genetic algorithm,” *Computers & Electrical Engineering*, vol. 38, no. 6, pp. 1564–1572, 2012.

[60] H. Savuran and M. Karakaya, “Efficient route planning for an unmanned air vehicle deployed on a moving carrier,” *Soft Computing*, vol. 20, no. 7, pp. 2905–2920, 2016.

[61] A. H. Karami and M. Hasanzadeh, “An adaptive genetic algorithm for robot motion planning in 2d complex environments,” *Computers & Electrical Engineering*, vol. 43, pp. 317–329, 2015.

[62] H. Ergezer and M. K. Leblebicioğlu, “3d path planning for uavs for maximum information collection,” in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pp. 79–88, IEEE, 2013.

[63] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[64] C. Ramirez-Atencia, G. Bello-Orgaz, M. D. R-Moreno, and D. Camacho, "Solving complex multi-uav mission planning problems using multi-objective genetic algorithms," *Soft Computing*, vol. 21, no. 17, pp. 4883–4900, 2017.

[65] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[66] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[67] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *The international journal of robotics research*, vol. 21, no. 4, pp. 331–344, 2002.

[68] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," in *Encyclopedia of operations research and management science*, pp. 1573–1578, Springer, 2013.

[69] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical computer science*, vol. 4, no. 3, pp. 237–244, 1977.

[70] H. Sachs, M. Stiebitz, and R. J. Wilson, "An historical note: Euler's königsberg letters," *Journal of Graph Theory*, vol. 12, no. 1, pp. 133–139, 1988.

[71] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part i: The chinese postman problem," *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.

[72] H. A. Eiselt, M. Gendreau, and G. Laporte, "Arc routing problems, part ii: The rural postman problem," *Operations research*, vol. 43, no. 3, pp. 399–414, 1995.

[73] F. Barahona, *On some applications of the chinese postman problem*. Universität Bonn. Institut für Ökonometrie und Operations Research, 1989.

[74] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Autonomous Robots*, vol. 36, no. 4, pp. 365–381, 2014.

[75] K. Sinha, *Path Planning for a UAV in an Agricultural Environment to Tour and Cover Multiple Neighborhoods*. PhD thesis, Virginia Tech, 2017.

[76] K. I. Trovato, "Differential a*: An adaptive search method illustrated with robot path planning for moving obstacles and goals, and an uncertain environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 4, no. 02, pp. 245–268, 1990.

[77] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *ICRA*, vol. 94, pp. 3310–3317, 1994.

[78] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong planning a^*,” *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[79] A. Stentz *et al.*, “The focussed d^* algorithm for real-time replanning,” in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[80] S. Koenig and M. Likhachev, “D^* lite,” *Aaai/iaai*, vol. 15, 2002.

[81] L. De Filippis, G. Guglieri, and F. Quagliotti, “Path planning strategies for uavs in 3d environments,” *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 247–264, 2012.

[82] A. Nash, K. Daniel, S. Koenig, and A. Felner, “Theta^*: any-angle path planning on grids,” in *AAAI*, vol. 7, pp. 1177–1183, 2007.

[83] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, “Adaptive evolutionary planner/navigator for mobile robots,” *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 18–28, 1997.

[84] Y. Hu and S. X. Yang, “A knowledge based genetic algorithm for path planning of a mobile robot,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 5, pp. 4350–4355, IEEE, 2004.

[85] L. Zhou and W. Li, “Adaptive artificial potential field approach for obstacle avoidance path planning,” in *Computational Intelligence and Design (ISCID), 2014 Seventh International Symposium on*, vol. 2, pp. 429–432, IEEE, 2014.

[86] T. Wilson and S. B. Williams, “Adaptive path planning for depth-constrained bathymetric mapping with an autonomous surface vessel,” *Journal of Field Robotics*, vol. 35, no. 3, pp. 345–358, 2018.

[87] L. Paull, S. Saeedi, H. Li, and V. Myers, “An information gain based adaptive path planning method for an autonomous underwater vehicle using sidescan sonar.,” in *International Conference onAutomation Science and Engineering (CASE)*, pp. 835–840, 2010.

[88] R. N. Smith, P. Cooksey, F. Py, G. S. Sukhatme, and K. Rajan, “Adaptive path planning for tracking ocean fronts with an autonomous underwater vehicle,” in *Experimental Robotics*, pp. 761–775, Springer, 2016.

[89] H. Fleischner, “(some of) the many uses of eulerian graphs in graph theory (plus some applications),” *Discrete Mathematics*, vol. 230, no. 1-3, pp. 23–43, 2001.

[90] W. D. Wallis, *Mathematics in the real world*, vol. 47. Springer, 2013.

[91] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[92] H. R. Lourenço, O. C. Martin, and T. Stützle, “Iterated local search: Framework and applications,” in *Handbook of metaheuristics*, pp. 363–397, Springer, 2010.

[93] C. Voudouris and E. P. Tsang, “Guided local search,” in *Handbook of metaheuristics*, pp. 185–218, Springer, 2003.

[94] M. Gendreau, “An introduction to tabu search,” in *Handbook of metaheuristics*, pp. 37–54, Springer, 2003.

[95] C. Reeves, “Genetic algorithms,” in *Handbook of metaheuristics*, pp. 55–82, Springer, 2003.

[96] J. Brownlee, *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011.

[97] J. Holland, “Adaptation in natural and artificial systems: an introductory analysis with application to biology,” *Control and artificial intelligence*, 1975.

[98] S. Chatterjee, C. Carrera, and L. A. Lynch, “Genetic algorithms and traveling salesman problems,” *European journal of operational research*, vol. 93, no. 3, pp. 490–510, 1996.

[99] S. Sivanandam and S. Deepa, “Genetic algorithm optimization problems,” in *Introduction to Genetic Algorithms*, pp. 165–209, Springer, 2008.

[100] D. G. Reina, P. Ruiz, R. Ciobanu, S. Toral, B. Dorronsoro, and C. Dobre, “A survey on the application of evolutionary algorithms for mobile multihop ad hoc network optimization problems,” *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, p. 2082496, 2016.

[101] G. Van Rossum and F. L. Drake, *The python language reference manual*. Network Theory Ltd., 2011.

[102] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “Deap: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, vol. 13, no. Jul, pp. 2171–2175, 2012.

[103] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, “Genetic algorithms for the travelling salesman problem: A review of representations and operators,” *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[104] K. Sastry, D. Goldberg, and G. Kendall, “Genetic algorithms,” in *Search methodologies*, pp. 97–125, Springer, 2005.

[105] DEAP Project, “DEAP documentation.” http://deap.readthedocs.org/en/1.0.x/, 2018. [Online; accessed April-2018].

[106] M. Arzamendia, D. Gregor, D. G. Reina, and S. L. Toral, “An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of ypacarai lake,” *Soft Computing*, pp. 1–12, 2017.

[107] Python Software Foundation, “Python documentation.” https://docs.python.org/2/library/, 2018. [Online; accessed April-2018].

[108] M. Arzamendia, I. Espartza, D. Reina, S. Toral, and D. Gregor, “Comparison of eulerian and hamiltonian circuits for evolutionary-based path planning of an autonomous surface vehicle for monitoring ypacarai lake,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2018.

[109] M. Arzamendia, D. G. Reina, S. T. Marin, D. Gregor, and H. Tawfik, “Evolutionary computation for solving path planning of an autonomous surface vehicle using eulerian graphs,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2018.

[110] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, “Approximation algorithms for lawn mowing and milling,” *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[111] R. Tarjan, “Depth-first search and linear graph algorithms,” *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[112] G. A. Croes, “A method for solving traveling-salesman problems,” *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[113] B. N. Miller and D. L. Ranum, *Problem solving with algorithms and data structures using python Second Edition*. Franklin, Beedle & Associates Inc., 2011.

[114] M. Arzamendia, D. G. Reina, S. L. Toral, D. Gregor, E. Asimakopoulou, and N. Bessis, “Intelligent online learning strategy for an autonomous surface vehicle in lake environments using evolutionary computation,” *IEEE Intelligent Transportation Systems Magazine*, 2018. In press.

[115] M. Arzamendia, “Intelligent Online Learning Strategy fon ASV In Lakes Environment Simulator.” https://github.com/Mariuspy/ASVPathPlanningGA, 2018. [Online; accessed April-2018].

[116] D. Reina, H. Tawfik, and S. Toral, “Multi-subpopulation evolutionary algorithms for coverage deployment of uav-networks,” *Ad Hoc Networks*, vol. 68, pp. 16–32, 2018.

[117] E. Asimakopoulou and N. Bessis, “Buildings and crowds: Forming smart cities for more effective disaster management,” in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pp. 229–234, IEEE, 2011.

[118] E. Asimakopoulou, *Advanced ICTs for Disaster Management and Threat Detection: Collaborative and Distributed Frameworks: Collaborative and Distributed Frameworks*. IGI Global, 2010.

[119] J. Sánchez-García, J. M. García-Campos, D. Reina, S. Toral, and F. Barrero, “Onsitedriverid: A secure authentication scheme based on spanish eid cards for vehicular ad hoc networks,” *future generation computer systems*, vol. 64, pp. 50–60, 2016.

[120] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr, “Niapy: Python microframework for building nature-inspired algorithms,” *J. Open Source Softw*, vol. 3, 2018.

[121] A. Sakai, “Python Robotics.” https://github.com/AtsushiSakai/PythonRobotics, 2018. [Online; accessed April-2018].

[122] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, “Hyper-heuristics: An emerging direction in modern search technology,” in *Handbook of metaheuristics*, pp. 457–474, Springer, 2003.

[123] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, “Hyper-heuristics: A survey of the state of the art,” *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[124] C. A. C. Coello, “Multi-objective optimization,” *Handbook of Heuristics*, pp. 1–28, 2018.

[125] T. Uras and S. Koenig, “An empirical comparison of any-angle path-planning algorithms,” in *Eighth Annual Symposium on Combinatorial Search*, 2015.

[126] Y. Liu, R. Bucknall, and X. Zhang, “The fast marching method based intelligent navigation of an unmanned surface vehicle,” *Ocean Engineering*, vol. 142, pp. 363–376, 2017.

[127] U. Commandant, “International regulations for prevention of collisions at sea, 1972 (72 colregs),” *US Department of Transportation, US Coast Guard, COMMANDANT INSTRUCTION M*, vol. 16672, 1999.

[128] W. Naeem, G. W. Irwin, and A. Yang, “Colregs-based collision avoidance strategies for unmanned surface vehicles,” *Mechatronics*, vol. 22, no. 6, pp. 669–678, 2012.

[129] L. Bayındır, “A review of swarm robotics tasks,” *Neurocomputing*, vol. 172, pp. 292–321, 2016.