Journal of Ambient Intelligence and Humanized Computing (2019) 10:1495–1507
https://doi.org/10.1007/s12652-018-0920-2

ORIGINAL RESEARCH

Comparison of Eulerian and Hamiltonian circuits for evolutionary-based path planning of an autonomous surface vehicle for monitoring Ypacarai Lake

M. Arzamendia[1] · I. Espartza[2] · D. G. Reina[3] · S. L. Toral[2] · D. Gregor[1]

Received: 11 December 2017 / Accepted: 14 June 2018 / Published online: 20 June 2018

Abstract
An evolutionary-based path planning is designed for an autonomous surface vehicle (ASV) used in environmental monitoring tasks. The main objective is that the ASV covers the maximum area of a large mass of water such as the Ypacarai Lake while taking water samples for sensing pollution conditions. Such coverage problem is transformed into a path planning optimization problem through the placement of a set of data beacons located at the shore of the lake and considering the relationship between the distance traveled by the ASV and the area of the lake covered. The optimal set of beacons to be visited by the ASV has been modeled through two different approaches such as Hamiltonian and Eulerian circuits. When Hamiltonian circuits are used, all the beacons should be visited only once. In the case of Eulerian circuits, the only limitation is that repeated routes cannot exist between two beacons. Both models have important implications on the possible trajectories of ASV throughout the lake. In this paper, we compare the application of both models for the optimization of the proposed evolutionary-based path planning. Due to the complexity of the optimization problem, a metaheuristic technique like a Genetic Algorithm (GA) is used to obtain quasi-optimal solutions in both models. The models have been compared by simulation and the results reveal that the Eulerian circuit approach can achieve an improvement of 2% when comparing to the Hamiltonian circuit approach.

Keywords Autonomous surface vehicle · Coverage path planning · Eulerian circuits · Hamiltonian circuits · Genetic algorithm

1 Introduction

Autonomous surface vehicles (ASVs) have great potential for applications in aquatic environments, such as seas, rivers, lakes, and streams (Liu et al. 2016). ASVs do not require a crew on board, which leads to the possibility of building smaller vehicles, and consequently, reducing significantly both production and operation costs. Some examples of target applications are surveillance, security, monitoring, and Search and Rescue operations (SAR), among others.

The Ypacarai Lake, which is located in Paraguay, is an adequate test scenario for monitoring applications of ASVs since it has recently been under observation for problems related to water pollution. The monitoring of the Ypacarai Lake requires carrying out periodic analysis of water quality. Such tasks can be easily accomplished by equipping an ASV with the appropriate sensors, such as temperature, pH, dissolved oxygen (DO), turbidity, oxidation–reduction potential (ORP), etc. Thus, the ASV can take samples of water and

✉ D. G. Reina
dgutierrez@uloyola.es

M. Arzamendia
marzamendia@ing.una.py

I. Espartza
irlaec@gmail.com

S. L. Toral
storal@us.es

D. Gregor
dgregor@ing.una.py

[1] Research Department, Faculty of Engineering, National University of Asuncion, San Lorenzo, Paraguay

[2] Electronic Engineering Department, University of Seville, Seville, Spain

[3] Engineering Department, Loyola University Andalusia, Seville, Spain

analyze them as it moves throughout the lake. An area of the Ypacarai Lake is said to be covered if the ASV goes through such area and takes a sample of water. Therefore, the monitoring task can be described as a coverage problem. A suitable coverage strategy is therefore necessary to take these samples and develop an up-to-date map of the conditions of the lake. It is worth indicating that this task can be even more challenging since the water conditions may vary periodically due to environmental factors, such as changes of temperature and rains, as well as human-made events, such as massive and uncontrolled waste disposal.

Furthermore, the mentioned coverage problem, which is also called Coverage Path Planning (CPP) (Galceran and Carreras 2013), can be transformed into a Path Planning Problem (PPP) since the distance traveled by the ASV is highly correlated with the area of the lake sensed by the ASV. Therefore, the main challenge for an efficient monitoring of the Ypacarai Lake is to find an optimal path planning for the ASV to follow so that the coverage of the Ypacarai Lake is maximized. Path planning problem in autonomous vehicles is a main research topic in autonomous systems because it allows a robot to tactically move over a specific region (Pascarella et al. 2015). Generally, this region is represented as a map, and the path planning finds intermediate waypoints between a starting and a destination point, which in turn gives intermediate routes. Therefore, a visiting tour can be defined between the starting and the destination points through several waypoints. In general, the objective of a path planning optimization algorithm is to minimize the distance traveled by the robot, since it implies lower operation costs. Conversely, for the proposed coverage application transformed into a path planning, the objective is now to maximize the distance, which implies a higher coverage of the lake as long as redundancy is avoided.

The intermediate waypoints, which are described as a set of vertices and edges that connect the vertices between them, can be used to build a graph. If the destination and starting points are the same, then it is said that the path is a cycle or circuit. In graph theory, a circuit that visits each vertex once is called a Hamiltonian Circuit (HC). Similarly, a circuit that visits all the edges of a graph once and all the vertices at least once (it can be more times), then it is called an Eulerian Circuit (EC). Hamiltonian Circuits are important in PPP and CPP problems because they are used to study the Traveling Salesman Problem (TSP), which is the problem of finding the minimum HC in a weighted graph. Both types of circuits present important implications in the movements and coverage of the resulting ASV path planning.

This paper explores, by means of a comparison, the suitability of both types of circuits for CPPs. ECs allow constructing a path that repeats regions of a map since waypoints can be visited as many times as desired. However, HCs limit the utilization of the waypoints, restricting the movements of the ASV. Therefore, it is expected that the flexibility gained by applying EC influences positively on the distance traveled by the ASV, and consequently, on the total covered area. Furthermore, by using an EC (non-repeated edges) it is guaranteed that paths are not repeated and, in this way, it achieves a higher coverage of the lake. Regarding complexity, finding a Hamiltonian circuit for a given number of waypoints is very high (NP-complete problem) since it is well-studied and known as the Traveling Salesman Problem (TSP), which is a classical combinatorial optimization problem (Hoffman et al. 2013). In contrast, finding an EC for a given number of waypoints is easier since there is only one main condition to be met by the graph of waypoints to be Eulerian; that is, all the vertex (waypoints) should have even number of links (Fleschner 2001). Notice that in this case, the links are the routes taken by the ASV. Yet, the complexity in the Eulerian-based approach stems from finding the best subset of selected waypoints and connections that form an Eulerian graph, so that at least one EC can be obtained from the graph. With the help of a wireless network of beacons located at the shore of the lake that act as waypoints and data collections points, a complete graph can be built. Furthermore, each edge of this graph provides coverage for all the regions of the lake, and therefore, the optimization challenge is to find a circuit that provides a satisfactory coverage of the lake.

Nowadays, there is no an algorithm that solves these types of combinatorial optimization problems in polynomial time (Hoffman et al. 2013). However, there are some techniques, such as metaheuristics, that can provide a quasi-optimal solution in a reasonable time (Gendreau and Porvin 2010). These techniques are, for example, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA), among others (Yang 2014, Su and Zhao 2017). In this paper, we demonstrate that GAs are suitable search engines for finding HCs and ECs in the proposed CPP for the monitoring of Ypacarai Lake.

In previous works, the proposed CPP problem of the Ypacarai Lake is modeled and evaluated as the TSP. The problem is solved by using the GA with different objective functions, such as the distance (Arzamendia et al. 2016) and coverage (Arzamendia et al. 2017). Furthermore, in the last study, the GA approach has been compared with random and greedy algorithms and performed some GA parameters tuning to obtain better results. In this paper we improve the previous work by applying the EC model, which we demonstrate that achieves better results than the HC model. Therefore, the main contributions of this paper are twofold:

- Modeling the coverage problem of an ASV as a CPP using Hamiltonian and Eulerian circuits.

- The comparison of both models using a genetic algorithm as search engine in a real-life scenario such as the Ypacarai Lake.

The structure of the paper is as follows. Section II describes some relevant related works. Section III presents the statement of the problem, including the application scenario and the Eulerian and Hamiltonian circuit definitions. Section IV contains the details of the proposed approach based on genetic algorithm. Section V contains a comparison between the performances of HCs and ECs in the target CPP. Finally, Section VI summarizes the main conclusions of this paper.

2 Related work

GA is a technique that has been used in several works related to the generation of HCs for the TSP. In (Mukherjee et al. 2017), the authors present the constrained covering solid traveling salesman problems (CCSTSP). In CCSTSP, a salesman visits a subset of cities, the remained cities are covered within an imprecise predetermined distance and the salesman comes back to the initial city within a restricted time. They solve this problem by using a combined method of random insertion deletion with a modified genetic algorithm (RID-MGA). This work is different to the one presented in a way that the cities that are not visited should be covered by the visited ones within a certain distance. In addition, there is a time restriction since this problem is intended for emergency cases. In (Wang 2014), the authors improve a GA with two local optimization strategies that are applied to find the shortest HC, such as the four vertices, the three-line inequalities, and the reversion of local Hamiltonian paths with more than two vertices.

Regarding the CPP, it has been mostly studied in agriculture applications. Several examples with TSP modeling and using metaheuristics can be found (Zhou et al. 2014; Ryerson and Zhang 2007). In Zhou et al. (2014), the authors propose a path planning method that generates a feasible area coverage plan for agricultural machines in a region that includes obstacles. They formulate the problem as a TSP and solve it using the ACO approach. In Ryerson and Zhang (2007), the authors study the use of GA to find optimal path to cover a farm field while avoiding known obstacles. They analyze different aspects of the GA, such as representations of the field, obstacles, grid, path, etc., and its adaptation towards the application in this area. In Schäfle et al. (2016), the authors use a GA to find the best coverage solution by dividing the field in small squares of the size of the vehicle, and then, calculating the sequence of movements of a robot that covers the whole field and minimizes the energy costs.

CPP research in agriculture involves not only UGV (Unmanned Ground Vehicles), but also UAV (Unmanned Aerial Vehicle), which can be used to take images of a crop from high distance. Moreover, it allows the evaluation of crop conditions, or for weed detection, or even for irrigation and application of fertilizers, among other applications. On this line, Pham et al. (2017) propose a cellular decomposition of the field for a given CPP with concave obstacles based on the generalization of the Boustrophedon variant and using Morse functions. Regarding ASV and monitoring tasks, Tokekar et al. (2013) present a coverage algorithm for finding tagged fish and a localization algorithm to effectively find them in a lake. Instead of covering the entire lake, the algorithm searches for covering specific regions inside the lake. First, they model the problem of visiting all the regions as a TSP with neighbors (TSPN), and then, inside each region, the ASV follows a π shape path.

Furthermore, EC and HC circuit models are the base for other approaches, such as the Chinese Postman Problem (CPP) (Eiselt et al. 1995a) and its extension Rural Postal Problem (RPP) (Eiselt et al. 1995b). Both models are widely applied in applications, such as logistics, where paths are chosen to deliver goods, in transportation services like garbage collection, in mail delivery, and snow plowing, among others. The CPP is related to finding ECs because it searches for a solution that visits all the edges in a graph, while minimizing the distance traveled. The RPP, on the other hand, is more practical in the sense that only a sub-set of waypoints is obligatory to be visited. Then, a path should be constructed with the rest of edges while minimizing the distance traveled as well.

The present work is a step forward in the evaluation of HCs and ECs for CPP problems. The presented related work shows that several works have employed HCs for CPP problems using GA as search engine (Mukherjee et al., 2017; Wang 2014). This metaheuristics engine is adequate to solve high complex optimization problems as the CPP. However, as far as the authors' knowledge, this is the first work that models the CPP using Eulerian circuits for Autonomous Surface Vehicles (ASV). To validate such approach, we compare the results of a target CPP problem like the monitoring of the Ypacarai Lake when using both HC and EC models. Furthermore, an evaluation of the GA as a solving tool vs other metaheuristics is also included. Finally, the flexibility of using EC in CPP is demonstrated by showing how the distance traveled, which is related to the coverage, can be adjusted according to the ASV autonomy requirements.

3 Statement of the problem

3.1 Coverage path problem transformed into path planning problem

The studied problem calculates the best path planning for an ASV to take water samples and deliver them to a network of data beacons located at the shore of the lake. Therefore, this problem can be formulated as having a set of beacons and determining how the ASV can take the largest number of samples to be delivered to the beacons. These beacons can be considered as intermediate waypoints. The sum of the routes between the first and last waypoints lead to the total path. An image of the Ypacarai Lake is shown in Fig. 1, which depicts the set of beacons. The number and positions of beacons have been chosen so that there is an equally spaced network with wireless capabilities for collecting data with an average distance of about 1 km between them. Still, these can be adapted in the future for a better integration with an existing data infrastructure. This distance is estimated so that it is feasible to deploy links in the 2.4 GHz (Zigbee technology, IEEE 802.15.4 standard). Under these conditions, the total number of beacons required is $n = 60$.

Using Fig. 1, a reference map is created. The origin of this map is selected as coordinates (25° 22′ 21″ S, 57° 22′ 57″ W) and all the beacon coordinates are translated into this new reference. In addition, each beacon is given an identification number that will be used throughout this study, from *b0* to *b59*. This convention is shown in Fig. 2. For example, in this map beacon 0 is located at (4,860; 13,500).

Regarding the coverage performed by the ASV, it is assumed that one sample taken at a given point represents the conditions of the water of an area S_{ASV} around the point. Then, the total coverage C_{tot} is calculated by multiplying this area S_{ASV} by the distance traveled by the ASV, which is denoted as PL. The distance traveled is calculated by finding the sum of Euclidean distances between the adjacent beacons in the sequence that forms the final path solution. Moreover, the route intersections are considered as duplicated sampled data. Therefore, the term $n_{routeinters}$ is used to account for the number of intersection among routes, which should be subtracted from the area covered. The final expression is:

$$C_{tot} = S_{ASV} \times PL - n_{routeinters} \times S_{ASV}^2 \quad (1)$$

These concepts are illustrated in Fig. 3, where the green rectangle is the covered area by the ASV and the red area is the duplicated sensing.

3.2 Hamiltonian Circuits vs Eulerian Circuits

The set of beacons and their possible connections, which are determined by the routes taken by the ASV, can be modeled as a complete and connected graph $G\{V,E\}$. Where $V = \{v$

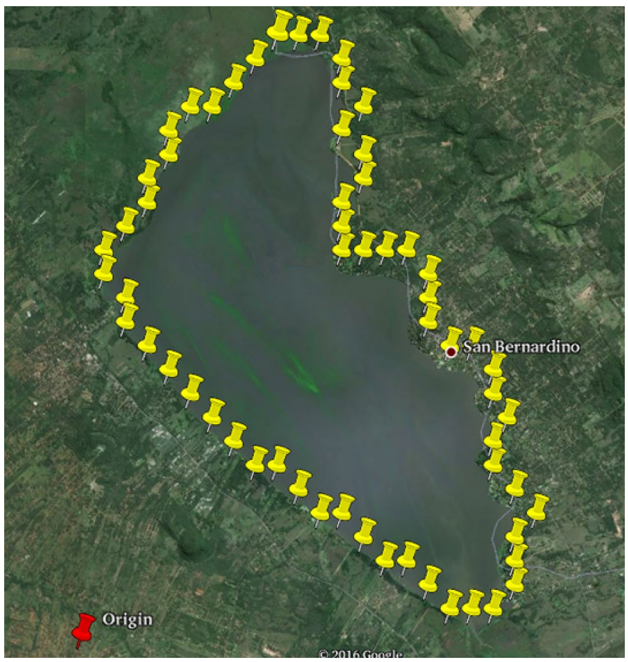

Fig. 1 Ypacarai Lake and the data wireless beacons (Arzamendia et al. 2017)

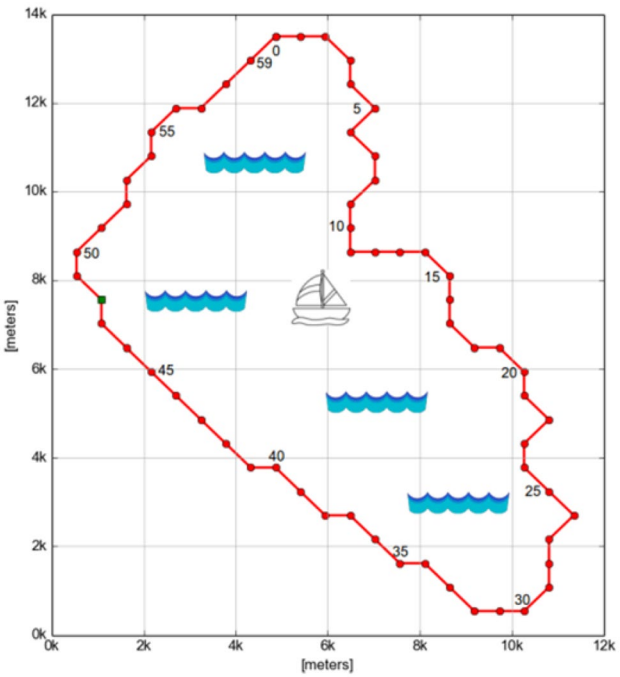

Fig. 2 Reference map (Arzamendia et al. 2017)

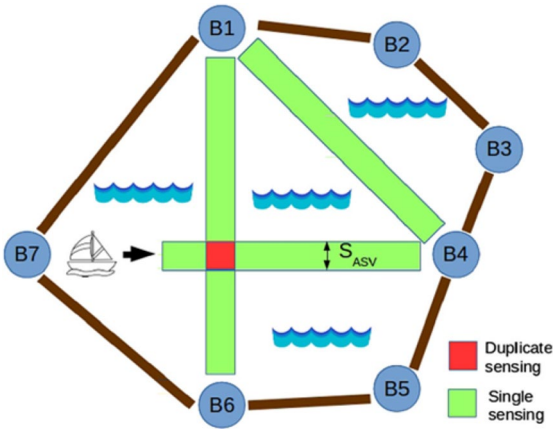

Fig. 3 Illustration of ASV coverage

$_1$,v_2,...,v_n} is the set of vertices, E={e_1, e_2,..., e_n} is the set of edges in which e_i connects v_i and v_j, $\forall i,j = \{1,2,\ldots,n\}$. Moreover, each edge has 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. Thus, 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, $\forall i,j = \{1,2,\ldots,n\}$. For the proposed CPP, the beacons are the vertices, the routes are the edges, and the distance between two beacons is the weight of the corresponding route. The order of a graph is $|V|$, calculated as the number of vertices and the size of a graph is $|E|$, 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.

By definition, a Hamiltonian cycle is a tour in a graph that visits all the vertices and edges of a graph once and starts and ends at the same vertex (Hoffman et al. 2013). For having this, each vertex should have only a pair of edges, one incoming and one outgoing edge, or what is the same, all the vertices of the graph should have a degree equal to two. Then, a Hamiltonian cycle (HC) is defined for a given graph $G\{V,E\}$ as:

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

An edge e is valid if that edge remains entirely inside the perimeter of the lake. If not, then the edge cannot be considered a part of the sub-graph G'. Because of this definition, the order and the size of G' is the same, i.e., $|V''| = |G|'$. In a graph with n vertices there are $(n-1)!/2$ possible HCs. The problem of finding the HC with minimum length is the widely studied TSP (Hoffman et al. 2013).

An Eulerian circuit (EC) is a closed tour that visits all the edges (Fleischner 2001). 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 Koningsberg bridge problem (Wallis 2013). One additional requirement is to verify that it is a connected graph, meaning that there is a path between every pair of vertices in the graph. Similarly to the HC, the definition of EC is:

$$\begin{aligned} EC = G''\{V'',E''\}\ s.t. \quad & V'' \subseteq V, \\ & E'' \subset E, \\ & deg(vi) \leq 2m \forall i = \{1,2,\ldots,o\} \\ & \quad and\ m \geq 1, o < n \\ & e_j\ is\ valid, \forall i = \{1,2,\ldots,p\} \\ & \quad and\ p < n^2 \end{aligned} \tag{3}$$

Please notice that in Eq. (3), the degree of the vertices of an EC is an even number and that not all the initial vertices from G might be included in the sub-graph, then 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.

The comparison of both circuits (2) and (3) shows that finding the optimal EC is much more difficult than finding the optimal HC because the order of HC is fixed, while in the EC is variable. The complexity of both approaches can be observed by calculating the number of possible solutions for each case. For the HC, the number of alternatives to choose from initially 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 two, because it is being considered an undirected graph. On the other hand, in the EC, the number of alternatives remains as n-1. Consequently, the number of possible tours is $(n-1)^{n-1}/2$, which is significantly higher than $(n-1)!$. Figure 4 presents a graph that compares the number of possible solutions for up to 50 vertices.

These two approaches can be applied to the proposed CPP of the Ypacarai Lake to find the path that best maximizes the distance traveled by the ASV. This CPP problem can be expressed as:

$$maxd_{total} = \sum d_{ij}\ \text{s.t.}\ d_{ij} = length(v_i v_j) \quad \text{and} \quad v_i v_j \in Ess \tag{4}$$

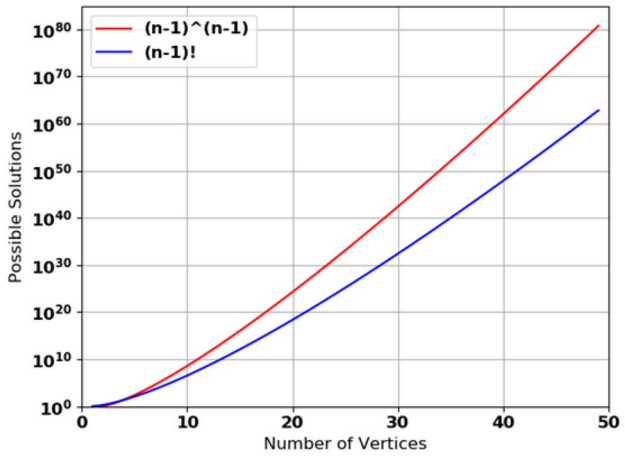

Fig. 4 Complexity of Eulerian circuit ((n – 1)^(n – 1)) vs Hamiltonian circuit (n-1)!

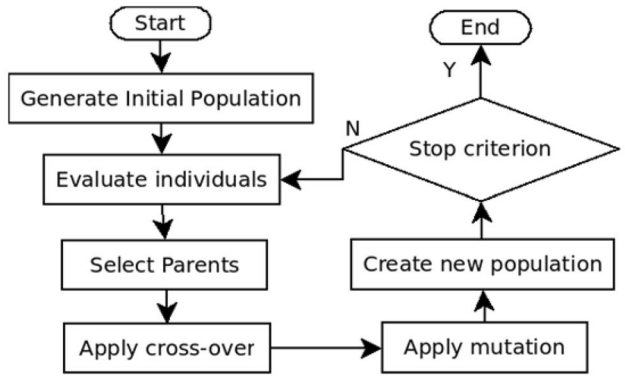

Fig. 5 Illustration of GA steps

Therefore, the problem is to find two sub-graphs $G_{HC}\{V_{HC},E_{HC}\}$ and $G_{EC}\{V_{EC},E_{EC}\}$ from the initial graph $G\{V,E\}$ that satisfy the aforementioned conditions (2) and (3). It is important to recall that for a HC, the order and size of the graph is always the same $|V|=|V_{HC}|=|E_{HC}| = n$. However, this is not necessarily true for an EC, as indicated in Eq. (3). Therefore and for the sake of making a fair comparison between these two approaches, we will force the graphs to have the same size $|E_{HC}|=|E_{EC}| = n$.

4 Evolutionary path planning

4.1 Genetic algorithm

The GA is a metaheuristic technique that mimics the evolution in nature (Sastry et al. 2014). Metaheuristic techniques are adequate for solving problems with high levels of complexity, like NP-hard problems (Dreo et al., 2006). From an initial set of possible solutions or individuals, new solutions are generated by applying genetic operators, such as selection, crossover, and mutation. Individuals with better quality have higher probability to participate in genetic operations and consequently, produce better individuals by combining the best traits of both parents. A diagram of a classical GA is shown in Fig. 5.

4.2 Individual representation

The individuals in the path planning problem are defined by their chromosomes, which are represented as arrays of the vertices of a graph. The positions inside the array determine the edges of the graph. Consequently, the definitions of the individuals for an HC and EC are shown in (5) and (6) respectively.

$$indiv_{HC} = \left[v_1 v_2 \ldots v_n\right] \text{ s.t. } vi \neq vj \quad \forall i,j = \{1,2,\ldots,n\} \tag{5}$$

$$indiv_{EC} = \left[v_1 v_2 \ldots v_n\right] \text{ s.t. } G_{EC} \text{ is Eulerian} \tag{6}$$

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 (5), each beacon can only appear once in the individual. Notice that this is a main requirement to form a HC. In (6), 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.

4.3 Generation of the initial population

The generation of the initial population is very important for the proposed evolutionary path planning based on HC and EC. In the case of HC, there are two restrictions, such as (1) no repetition of beacons in the individuals and (2) no invalid routes. It is important to recall that a route is invalid if the ASV should get out of the lake to go from one beacon to another. In the case of ECs, the procedure to generate initial populations is more complicated since there are more restrictions, such as (1) all the vertices in the graph should have an even degree, (2) no invalid routes, and (3) the resulting graph should be connected.

Two mechanisms are developed to create individuals that are HC and EC. The first one is shown in Fig. 6. Basically, this mechanism generates random routes beacon by beacon, verifying: (1) if it is already included, and (2) 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.

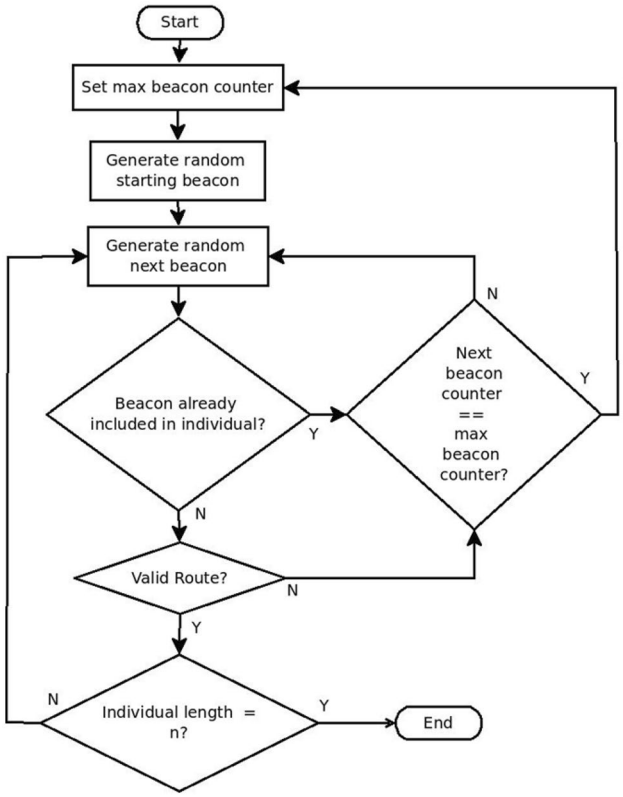

Fig. 6 Procedure to create HCs

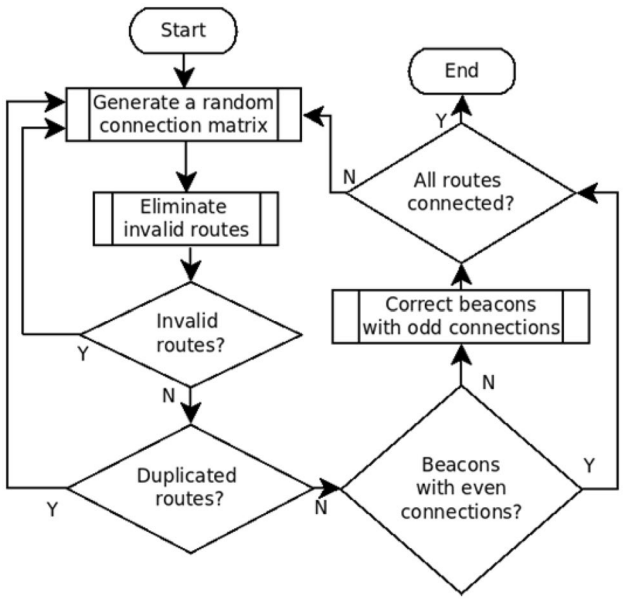

Fig. 7 Procedure to create ECs

For the Eulerian individual, since the number of possibilities is much larger, a different mechanism is designed (shown in Fig. 7). 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 total number of routes can be selected, but in this study and for the sake of comparison with the Hamiltonian based approach, n is selected equal to *60*. The number of 1's is distributed randomly across the matrix.

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 and vice-versa ($d_{ij} = d_{ji}\ \forall i,j = \{1,2,\ldots,n\}$). 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 Fig. 8. On the left (Fig. 8), there is a graph with six nodes (beacons) and beacons *b3* and *b5* have an even 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 consists of evaluating 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 two proposed procedures to generate HCs and ECs (see Figs. 6, 7) are repeated until the desired size of initial population is generated.

4.4 Fitness function

The quality of each individual is calculated with the help of a fitness function. For the proposed evolutionary-based CPP,

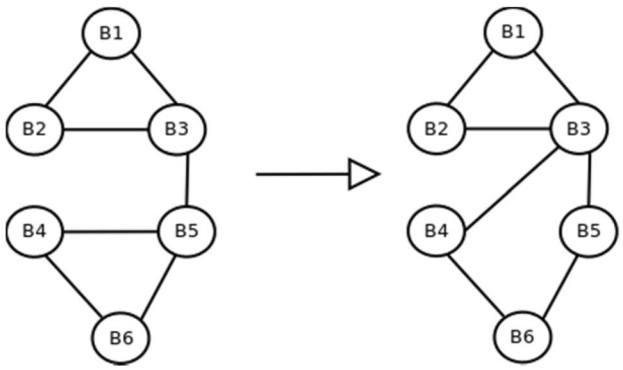

Fig. 8 Correction of nodes with odd connections

the best individuals are the paths that maximize the coverage area. The conventional fitness function is:

$$fit_{conv} = \frac{S_{ASV} \cdot PL}{A_{lake}} \tag{7}$$

Where PL is the total path length, i.e., the sum of the distances between the beacons that form the solution, S_{ASV} is the sampling area and A_{lake} is the lake area.

Two fitness functions are used in this study to improve the conventional one by considering invalid routes and the intersections between routes. These are (1) a death penalty fitness function and (2) a penalty fitness function. The first one discards the individuals with invalid routes by assigning a negative value to the fitness. The second fitness function does not discard the individual, but applies a penalty that decreases its fitness values according to the number of invalid routes. Regarding the intersection between routes, the distance traveled by the ASV is multiplied by a sampling area and the repeated areas are subtracted from the total area covered. The repeated areas are the areas where route intersections occur. A mathematical definition of the death penalty fitness function is:

$$fit_{DP} = \begin{cases} \frac{S_{ASV} \cdot PL - S_{ASV}^2 \cdot n_{intersec}}{A_{lake}}, & with\ valid\ routes \\ -1, & with\ invalid\ routes \end{cases} \tag{8}$$

where $n_{intersec}$ is the number of intersections of the path.

For the penalty factor, the definition is:

$$fit_{PF} = \left(\frac{n_{val_{rout}}}{n_{tot_{rout}}} \right) . \left(\frac{S_{ASV} \cdot PL - {S_{ASV}}^2 \cdot n_{intersec}}{A_{lake}} \right) \tag{9}$$

where the right term is the same used as in the death penalty and the left term is the percentage of the valid intermediate routes from the total included in the individual. The idea of using these two types of fitness functions is that the death penalty-based fitness function prevents individuals with invalid routes from participating in the generation of new individuals. However, when using the penalty function, there is still possibility that invalid individuals participate as parents in genetic operations. This approach considers that invalid individuals still might have valuable traits in their chromosome structure that can be used in the future to generate better individuals.

4.5 Genetic operators

The GA is basically the same shown in 5. However, a mechanism called elitism is added. This mechanism is used to avoid losing a good result found in subsequent generations. For this, the best percentages of the population are separated and the crossover and mutation operators are applied to the rest of the population. The resulting offspring replaces completely the parents and the elitist individuals are added to the offspring, maintaining the size of the population.

Pairs of individuals (parents) are selected from the rest of the population for crossing over and generating a pair of offspring (children). The main mechanisms used for parent selection are the roulette wheel and tournament (Sivanandam and Deepa 2007). Two genetic operators are applied sequentially, but with certain probability, crossover and mutation operators. In the crossover, pieces of the parents are exchanged for offspring production. The most common types are one or two-point crossover. However, special attention should be paid to the Hamiltonian case to avoid repeated vertices, which is an invalid solution. Therefore, a specific crossover mechanism should be used, like the ordered crossover. After creating the offspring, a mutation operator is applied. The mutation operator used for the problem is the shuffle index, which switches the values of two positions in the array of the individual. In this way, the repetition of beacons is avoided.

It is relevant to emphasize that the resulting offspring will always be Eulerian or Hamiltonian circuits just like their parents, since the crossover and mutation operators do not modify at any time the number of connections that each vertex of the graph has. These operators replace the edges by different edges but keeping always the same the number. This is an additional advantage of the use of the genetic algorithm in our problem. However, the resulting offspring might have invalid routes, but these will be considered and handled by the fitness function during the next parent selection.

4.6 Stop criterion

There are different approaches for stopping the GA. For example, if a number of generations have been executed or if the improvement rate of the best population individual has been stabilized or converged. The proposed approach considers the number of generations executed, but once a certain level of convergence has been achieved.

5 Performance evaluation

This section includes the details about the simulation performed. It contains the main features of the simulated environment and the results obtained under those conditions.

5.1 Simulation environment

The GA parameters used are described in Table 1. The simulator has been implemented in Python 2.7 and the library

Table 1 Simulation parameters

Number of simulations	20
Distance between beacons (km)	1 (approximately)
Lake size (km^2) A_{lake}	68.72
Sampling area (m) S_{ASV}	20
Population size	100
Number of generations	1000
Selection	Roulette Wheel
Crossover	Ordered (OX1)
Cross-over probability P_c	0.8
Mutation	Shuffle index ($indpb = 0.05$)
Mutation probability P_m	0.2
Elitism rate	0.2
Fitness	$fit_{conv} = \frac{S_{ASV} \cdot PL}{A_{lake}}$ $fit_{DP} = \begin{cases} \frac{S_{ASV} \cdot PL - S_{ASV}^2 \cdot n_{intersec}}{A_{lake}}, & \textit{with valid routes} \\ -1, & \textit{with invalid routes} \end{cases}$ $fit_{PF} = \left(\frac{n_{val_{rout}}}{n_{tot_{rout}}}\right) \cdot \left(\frac{S_{ASV} \cdot PL - {S_{ASV}}^2 \cdot n_{intersec}}{A_{lake}}\right)$

DEAP (Distributed Evolutionary Algorithms for Python) (Fortin et al. 2012) has been used for the GA implementation. This has been made public in (Arzamendia 2017). Different values of crossover and mutation probabilities are tested in Arzamendia et al. (2017), but the most suitable ones are shown in Table 1.

5.2 Simulation results

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 GA. Then, these two models are applied to other stochastic metaheuristics algorithms, such as the Iterated Local Search (ILS) and Tabu Search (TS). Second, the HC and EC are compared with conventional techniques for exploring graphs like the Depth First Search (DFS). Finally, we include an evaluation of the impact of the number of beacons visited on the achieved coverage for the EC case. Notice that this is important because it allows the final user to adjust the distance traveled by the ASV according to its energetic autonomy.

5.2.1 HC vs. EC performance

The analyses conducted to evaluate and compare the performance of the proposed approach 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 obstacles created by the shape of the lake.

The simulation results obtained from the first analysis are shown in Table 2, in which the HC and EC models are compared. For each model, three fitness functions are used: the conventional (without considering route intersections), the death penalty, and penalty factor. However, in this first set

Table 2 Simulation results of HC and EC considering invalid routes

	Coverage (%)					
	HC			EC		
	Conventional	Death penalty	Penalty factor	Conventional	Death penalty	Penalty factor
Worst	16.55	15.97	15.97	16.83	16.43	16.23
Best	16.69	16.21	16.24	19.77	18.25	18.43
Average	16.62	16.11	16.11	17.68	17.53	17.26
Standard deviation	0.04	0.06	0.07	0.71	0.47	0.54

of simulations, since the invalid routes are disregarded, both death penalty and penalty factor are the same and equal to:

$$fit_{DP} = fit_{PF} = \frac{S_{ASV} \cdot PL - {S_{ASV}}^2 \cdot n_{intersec}}{A_{lake}} \quad (10)$$

According to the results of Table 2, 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 death penalty and penalty factor, the coverage is little above 18% in the case of ECs.

Figure 9 shows the best solutions of the HC vs EC approaches of the unconstrained model. It can be observed that the routes in the EC-based approach are more spread out throughout the lake. This is possible thanks to the 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 3. In this case, only the performance of the death penalty and penalty factor fitness functions are tested because solutions with invalid routes are not considered and the conventional mechanism does not cope with these routes. It is worth recalling that the Death Penalty discards invalid solutions and the Penalty Factor penalizes the fitness function. In Table 3, the coverage values are lower than in Table 2 because many of the previously selected routes were invalid and they should be replaced by shorter but valid ones. The best overall results are achieved by the EC model with death penalty function with 16.6% coverage of the lake. The EC with penalty factor achieved 15.9% coverage of the lake, but still is about 1% better than the HC model.

Figure 10 shows the best individuals using the HC on the left and the EC on the right. It can be seen that the EC

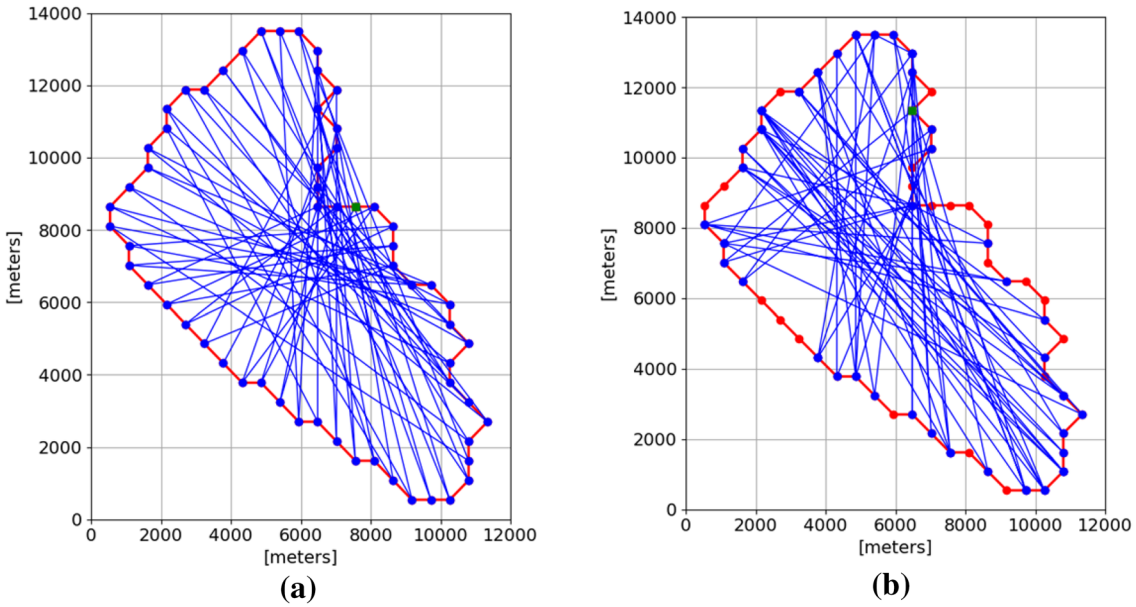

Fig. 9 ASV trajectories for the best solutions of HC (**a**) and EC (**b**) based approaches including invalid routes (unconstrained)

Table 3 Simulation results rejecting invalid routes

	Coverage (%)			
	HC		EC	
	Death penalty	Penalty factor	Death penalty	Penalty factor
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
Standard deviation	0.16	0.19	0.55	0.42

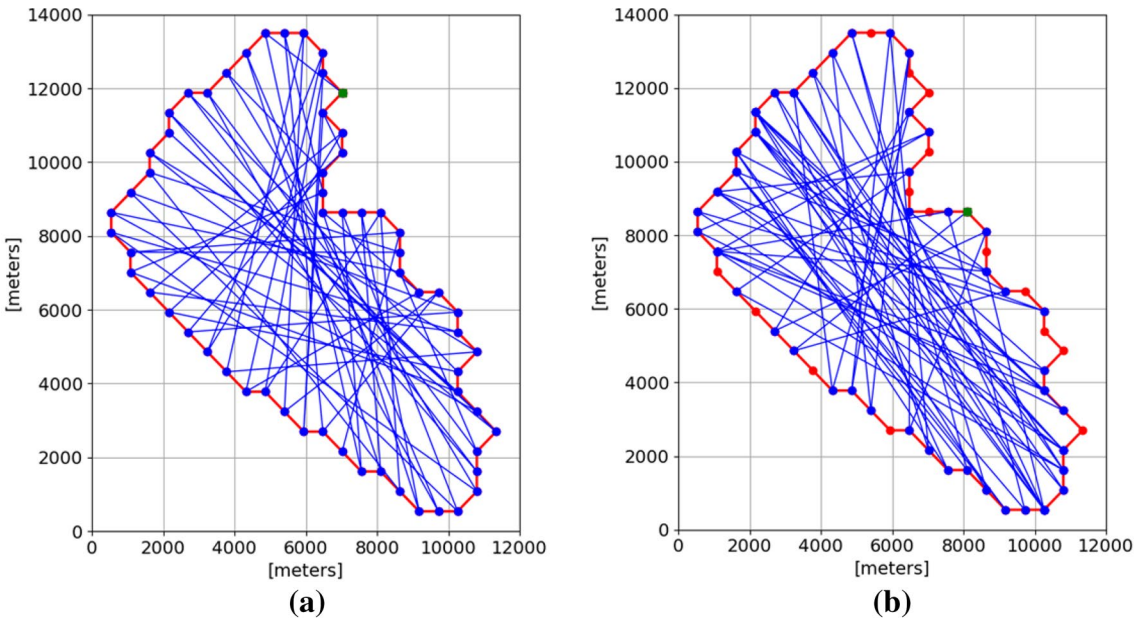

Fig. 10 ASV trajectories for the best solutions of HC (**a**) and EC (**b**) based approaches rejecting invalid routes (constrained)

approach provides better coverage of the lake. Again, the routes obtained by HC-based model are more concentrated on 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 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 Tables 2 and 3, 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.

5.2.2 Proposed method vs conventional algorithms

First, a brief description about the alternative algorithms is given. ILS and TS are algorithms based on probabilistic and stochastic processes and they are not bio-inspired like the GA (Brownlee 2011). They are predominantly global optimization algorithms and apply a neighborhood exploring (local) search procedure. In the case of ILS, 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. On the other hand, TS maintains a short-term memory of recent moves and prevents returning to recently visited areas of search space. It was originally designed to manage a hill climbing technique for the neighborhood heuristic exploration.

Regarding the implementation of ILS, it uses a double-bridge move as perturbation technique and a stochastic 2-opt as local search technique. The 2-opt is also implemented in

Table 4 HC and EC GA-based vs alternative metaheuristics

	Coverage %							
	HC				EC			
	Random	ILS	TS	GA-DP	Random	ILS	TS	GA-DP
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
Standard deviation	0.50	0.14	0.08	0.16	0.45	0.47	0.44	0.55

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 Death Penalty as fitness function in Table 4. In addition, the random solutions for each case are also included. Just like the case of the proposed approach, the stochastic algorithms were simulated 20 times and each simulation has 1000 iterations.

On the other hand, we also evaluated a DFS (Miller and Ranum 2011), which 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 below. This procedure is repeated until reaching the desired node or 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 beacon, i.e. $n=60$. The results are summarized in Table 5. These 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 a metaheuristic techniques like the GA. Then, it is seen that they are better than other stochastic metaheuristic technique like the ILS. Still, GLS and TS outperform HC but in that case, by applying the EC model it can increase the value of the coverage of the lake.

5.2.3 EC coverage vs number of beacons

By removing the restriction of the HC of visiting all the beacons, it is still possible to build a tour with a smaller distance if the requirements of the ASV demands to do so. This is required, 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 (the order of the sub-graph G") and the distance traveled. The number of beacons vary with the following values $n_{beacons} = \{10, 20,30,40,50, 60\}$, which are smaller than those of the HC. The results are shown in Table 6 and plotted in Fig. 11.

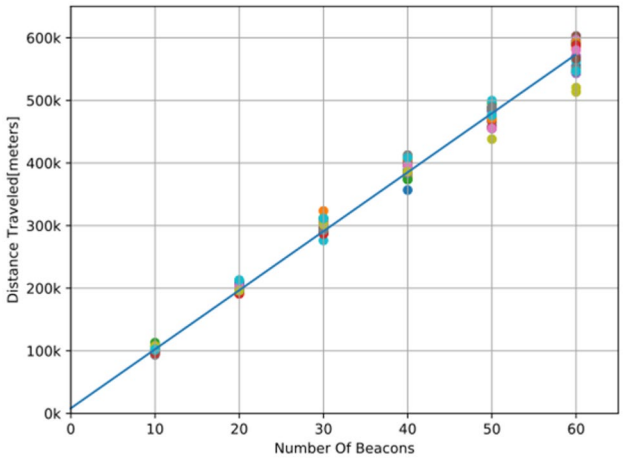

Fig. 11 EC distance traveled vs number of beacons – multiple simulations results

The simulation results show that there is almost a linear relationship between the distance traveled and the number of beacons, which can be approximated as:

$$PL(km) = 10\, x\, n_{beacons} \tag{11}$$

Notice that this information is useful for the final user because, if the ASV has an autonomy of 300 (km), then, it is known that the number of beacons should be $n_{beacons} \leq 30$. Additional information that can be extracted from Fig. 10 is that there is a greater dispersion among the simulations as the number of beacons increases, as a consequence of having a greater number of alternatives as the number of beacons increases as well.

6 Conclusion

This work presents an approach to find a suitable path planning for monitoring the Ypacarai Lake with an ASV. This CPP has been modeled using the concepts of Hamiltonian

Table 5 HC and EC GA-based vs alternative graph-exploring algorithms

	Coverage %		
	DFS	HC	EC
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

Table 6 EC distance traveled vs number of beacons–statistical summary

Distance (km)	Number of beacons					
	10	20	30	40	50	60
Best	113.05	213.20	323.50	412.73	499.71	602.87
Worse	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
Standard deviation	4.50	6.57	10.83	13.56	15.28	25.18

and Eulerian circuits, which have been taken from the graph theory. The best circuits, which maximize the coverage of the lake, are found using an evolutionary algorithm. To solve the proposed CPP, a tailored evolutionary algorithm is designed, considering the requirements for obtaining both HCs and ECs. Two fitness functions are defined to handle the invalid individuals that may appear as new individuals, such as a death penalty and penalty factor fitness function. The simulation results show that for the studied cases, the EC-based approach achieves significantly higher coverage than the HC (2% higher coverage). The main reason is the flexibility exhibited by the ECs to visit beacons more than one time. The best result is obtained for the death penalty case and it reaches almost 17% of coverage of the lake. A comparison with conventional algorithms like a random and DFS is also tested, as well as stochastic algorithms (ILS and TS), showing that the EC achieves at least 1% of coverage improvement. The last contribution shows the flexibility of the use of ECs for CPP since the path planning can be adjusted to find a suitable coverage for a given maximum distance traveled, which in turns is related to the autonomy of the ASV. The obtained results determine that there is near a linear relationship between the number of beacons used and the distance traveled by the ASV.

Acknowledgements The authors would like to thank Fundación Carolina and its PhD scholarships program. The authors would like to thank Mónica Díaz López for her help in proofreading the manuscript.

References

Arzamendia M. (2017) https://github.com/Mariuspy/ASVPathPlanningGA. Accesed 5 June 2018.

Arzamendia M, Gregor D, Reina DG, Toral SL, Gregor R (2016) Evolutionary path planning of an autonomous surface vehicle for water quality monitoring. In: IEEE 9th International Conference on Developments in eSystems Engineering (DeSE), 2016, pp 245–250.

Arzamendia M, Gregor D, Reina DG, Toral SL (2017) An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai Lake. Soft Comput. https://doi.org/10.1007/s00500-017-2895-x

Brownlee J (2011) Clever algorithms: nature-inspired programming recipes. Jason Brownlee

Dréo J, Pétrowski A, Siarry P, Taillard E (2006) Metaheuristics for hard optimization: methods and case studies. Springer, Berlin

Eiselt HA, Gendreau M, Laporte G (1995a) Arc routing problems, part I: the Chinese postman problem. Oper Res 43(2):231–242

Eiselt HA, Gendreau M, Laporte G (1995b) Arc routing problems, part II: the rural postman problem. Operat Res 43(3):399–414

Fleischner H (2001) (Some of) the many uses of Eulerian graphs in graph theory (plus some applications). Discret Math 230(1–3):23–43

Fortin FA, Rainville FMD, Gardner MA, Parizeau M, Gagné C (2012) DEAP: Evolutionary algorithms made easy. J Mach Learn Res 13:2171–2175

Galceran E, Carreras M (2013) A survey on coverage path planning for robotics. Robot Auton syst 61(12):1258–1276

Gendreau M, Potvin JY (2010) Handbook of metaheuristics, vol 2. Springer, New York

Hoffman KL, Padberg M, Rinaldi G (2013) Traveling salesman problem. Encyclopedia of operations research and management science. Springer, Berlin, pp 1573–1578

Liu Z, Zhang Y, Yu X, Yuan C (2016) Unmanned surface vehicles: an overview of developments and challenges. Ann Rev Control 41:71–93

Miller BN, Ranum DL (2011) Problem solving with algorithms and data structures using python Second Edition. Franklin, Beedle & Associates Inc, Portland

Mukherjee A, Panigrahi G, Kar S, Maiti M (2017) Constrained covering solid travelling salesman problems in uncertain environment. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-017-0620-3

Pascarella D, Venticinque S, Aversa R, Mattei M, Blasi L (2015) Parallel and distributed computing for UAVs trajectory planning. J Ambient Intell Humaniz Comput 6(6):773–782

Pham TH, Bestaoui Y, Mammar S (2017) Aerial robot coverage path planning approach with concave obstacles in precision agriculture. In: IEEE workshop on research, education and development of unmanned aerial systems (RED-UAS), 2017, pp 43–48

Ryerson AF, Zhang Q (2007) Vehicle path planning for complete field coverage using genetic algorithms. CIGR J, Agric Eng Int 9:1–11

Sastry K, Goldberg DE, Kendall G (2014) Genetic algorithms. In: Burke EK, Kendall G (eds) Search methodologies. Springer, Boston, MA

Schäfle TR, Mohamed S, Uchiyama N, Sawodny O (2016) Coverage path planning for mobile robots using genetic algorithm with energy optimization. In: Electronics Symposium (IES), 2016 International, IEEE, pp 99–104

Sivanandam SN, Deepa SN (2007) Introduction to genetic algorithms. Springer, Berlin

Su S, Zhao S (2017) A hierarchical hybrid of genetic algorithm and particle swarm optimization for distributed clustering in large-scale wireless sensor networks. J Ambient Intell Humaniz Comput:1–11

Tokekar P, Branson E, Vander Hook J, Isler V (2013) Tracking aquatic invaders: Autonomous robots for monitoring invasive fish. IEEE Robot Autom Mag 20(3):33–41

Wallis WD (2013) Mathematics in the real world. Birkhauser, Boston

Wang Y (2014) The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Comput Ind Eng 70:124–133

Yang XS (2014) Nature-inspired optimization algorithms. Elsevier, Amsterdam

Zhou K, Jensen AL, Sørensen CG, Busato P, Bothtis DD (2014) Agricultural operations planning in fields with multiple obstacle areas. Comput Electron Agric 109:12–22