Trabajo Fin de Máster MIT

Diseño de un algoritmo de path planning para un drone acuático de superficie basado en grafos Eulerianos

Autora: Iratxe Espartza Chueca
Tutor: Daniel Gutiérrez Reina

Dept. de Ingeniría Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Máster
MIT

Diseño de un algoritmo de path planning para un drone acuático de superficie basado en grafos Eulerianos

Autora:
Iratxe Espartza Chueca

Tutor:
Daniel Gutiérrez Reina
Doctor Ingeniero en Electrónica

Dept. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Máster: Diseño de un algoritmo de path planning para un drone acuático de superficie basado en grafos Eulerianos

Autora: Iratxe Espartza Chueca

Tutor: Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia por animarme a no parar quieta y a la oficina de Cooperación al Desarrollo de la US, así como a Dani y Sergio, por darme la oportunidad de realizar mi TFM en Paraguay.

Agradecimientos

Los estilos adoptados por nuestra Escuela y utilizada en este texto es una versión y adaptación a Word® del la versión LaTeX que el Prof. Payán realizó para un libro que desde hace tiempo viene escribiendo para su asignatura. Por ello, la Escuela le está agradecida. Por otro lado, la adaptación se hizo sobre un formato que el prof. Aguilera arregló, basándose en su tesis doctoral. Su aportación ha sido muy relevante para que este formato vea la luz. Esta adaptación la llevamos a cabo el alumno Silvio Fernández, becario del Centro de Cálculo, y yo mismo, sobre un trabajo preliminar del alumno Julián José Pérez Arias.

A esta hoja de estilos se le incluyó unos nuevos diseños de portada. El diseño gráfico de las portadas para proyectos fin de grado, carrera y máster, está basado en el que el prof. Fernando García García, de la Facultad de Bellas Artes de nuestra Universidad, hiciera para los libros, o tesis, de la sección de publicación de nuestra Escuela. Nuestra Escuela le agradece que pusiera su arte y su trabajo a nuestra disposición.

Juan José Murillo Fuentes

Subdirección de Comunicaciones y Recursos Comunes

Sevilla, 2013

Resumen

El objetivo de este trabajo es el diseño un algoritmo de path planning para un drone acuático de superficie autónomo que mida la calidad del agua en diferentes puntos del Lago Ypacaraí. Para este fin, se ha partido de un trabajo previo en el que se empleaban circuitos Hamiltonianos junto con Algoritmos Genéticos para la búsqueda del camino a realizar por el drone, adaptándose la forma en la que se genera la población inicial, de circuitos Hamiltonianos a Eulerianos. Para ello se ha creado un algoritmo con el que generar circuitos Eulerianos basándose en un grupo de balizas repartidas a lo largo del perímetro del lago. A partir de ese algoritmo se ha procedido a comparar los dos métodos planteados, y seguidamente a experimentar con diferentes parámetros para ver las diferencias de rendimiento del algoritmo. Finalmente, se ha podido comprobar que el nuevo algoritmo ha superado el rendimiento del método del que se partía.

Abstract

The aim of this master thesis is the design of a path planning algorithm for an autonomous surface vessel to measure the quality of the water on the Ypacaraí Lake. For this purpose, a previous thesis has been taken as the starting point, where Hamiltonian circuits were used along with Genetic Algorithms for the search of a path for the drone to follow, adapting the way the initial population is generated, changing from Hamilonian to Eulerian circuits. To achieve that, an algorithm has been created that generates Eulerian circuits based on a group of beacons distributed along the perimeter of the lake. From that algorithm, a comparison has ben made between the two methods, and next variations of the parameters have been tried to examine the performance of the algorithm. Finally, it could be checked the better performance of the new algorithm compared to the previous one.

Índice

Glosario

Drone Acuático de Superficie	Vehículo acuático no tripulado que se mueve por la superficie del agua con forma, por ejemplo, de catamarán.
Path Planning	Ejercicio de planear la ruta a seguir para un objeto.
Algoritmos Genéticos	Métodos inspirados en la evolución biológica que se emplean para la resolución de problemas de optimización.
Circuitos Eulerianos	Camino que pasa por cada arista una única vez.
Circuitos Hamiltonianos	Camino que pasa por cada vértice una única vez.
Python	Lenguaje de programación interpretado de alto nivel.
DEAP	Framework empleado para la implementación de técnicas de computación evolutiva.

1 OBJETIVOS Y MOTIVACIÓN DEL PROYECTO

El Lago Ypacaraí es el mayor lago de la República de Paraguay. Con una superficie aproximada de unos 60 km^2 y a solo 28 kms al este de Asunción, es un destino turístico popular entre los habitantes de la capital, además de acoger áreas de producción agrícola y ganadera en sus alrededores. El lago se nutre de varios arroyos, siendo los más importantes Yukyry y Pirayu, y más adelante estas aguas siguen su camino a través del Río Salado, que desemboca en el Río Paraguay [1]. En la Figura 1 puede verse la ubicación del Lago Ypacaraí.

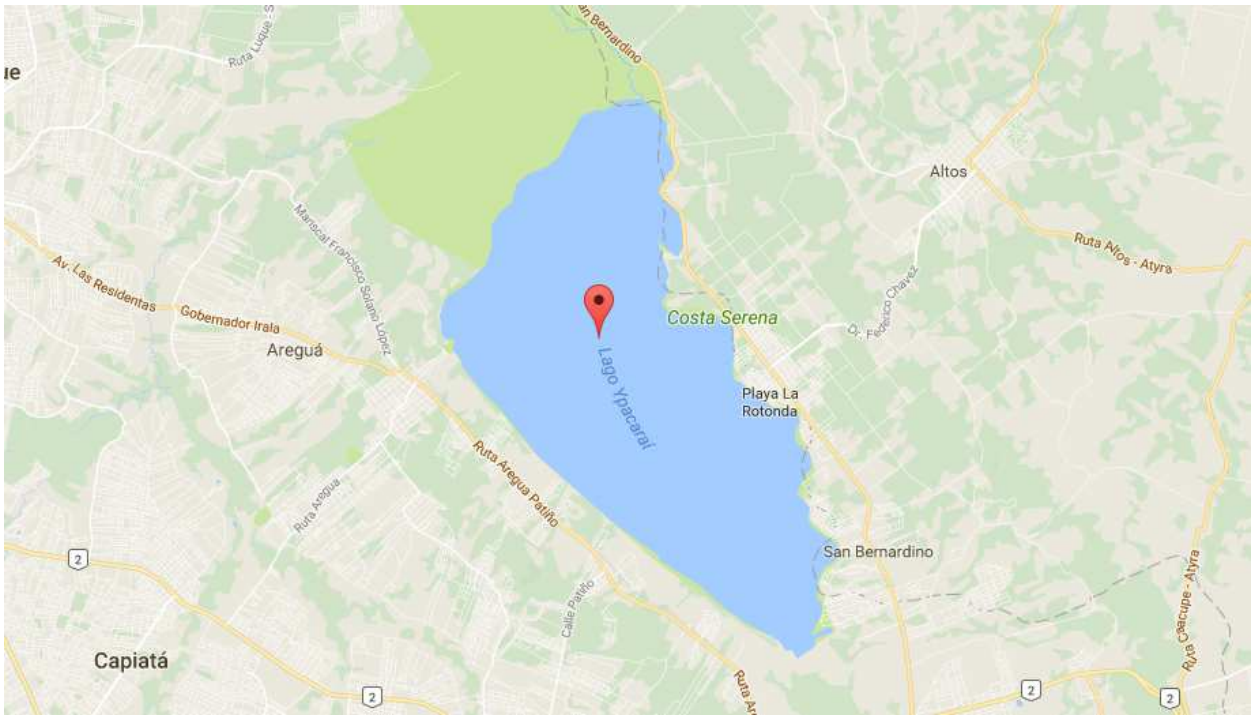


Figura 1 - Ubicación del Lago Ypacaraí

El crecimiento demográfico de los últimos años y el desarrollo económico de la región han generado un aumento de la producción industrial y consecuentemente de residuos, que han tenido un gran impacto en el lago. A causa de estos residuos, tanto domésticos como industriales, y de otros vertidos inorgánicos producidos por la industria agraria y arrastrados por los arroyos, los niveles de nitrógeno y fósforo han crecido y han favorecido la presencia de un tipo de algas llamadas cianobacterias [2]. La floración de este tipo de algas resulta alarmante ya que pueden producir toxinas que son perjudiciales tanto para los organismos en el medio acuático como para el ser humano.

Para intentar recuperar el estado natural del lago se han llevado a cabo varios proyectos en los que se han hecho mediciones de la calidad del agua. De todos modos, hasta la fecha estas mediciones se han limitado a puntos específicos del lago que no revelan más que una pequeña parte de toda la información que se puede obtener del análisis del lago. Es por eso por lo que la Universidad Nacional de Asunción (UNA), en colaboración con la Universidad de Sevilla (US), está trabajando en un prototipo de un drone acuático de superficie con el que recorrer la mayor superficie del lago posible y recabar datos de diversos puntos de la misma.

En el seno de esta cooperación hay alumnos y profesores de ambos centros trabajando desde sus respectivas universidades; y a esta colaboración se ha unido la Oficina de Cooperación al Desarrollo de la US , a través de las ayudas para la Sensibilización y Formación en Cooperación Internacional al Desarrollo, posibilitando la estancia de alumnos de la US en Paraguay, donde se tiene la oportunidad de trabajar durante 45 días en el proyecto conjuntamente con el personal del Centro de Investigación Tecnológica (CITEC) la UNA.

El objetivo final de este proyecto conjunto es construir un drone acuático de superficie, que se implementará

utilizando un catamarán de (2x4) m^2 dotado de paneles solares para su funcionamiento continuo. Este drone deberá ser totalmente autónomo, es decir, se moverá sin necesidad de ser guiado por alguien, sorteando los posibles obstáculos que pueda encontrar por el camino. Además, será el encargado de generar por sí mismo un recorrido que pase por las diferentes balizas que se hallarán a lo largo del perímetro del lago, intentando que se recorra la mayor área posible.

Para una primera fase de pruebas se ha sustituido el catamarán por una embarcación de menores dimensiones y que de momento tiene que ser guiada mediante un control remoto, con el fin de poder hacer las primeras mediciones y comprobar el correcto funcionamiento de los diferentes sensores. En paralelo se ha estado trabajando en una primera propuesta del algoritmo que creará la ruta a seguir por el drone. Esta primera propuesta se basa en el empleo de Algoritmos Genéticos (Genetic Algorithms, GAs) aplicados al Problema del Vendedor Viajero o Travelling Salesman Problem (TSP), que se encarga de encontrar la ruta más corta con la que conectar todas las balizas, pero para el caso del drone se invertiría con el fin de encontrar la ruta más larga y por lo tanto cubrir la mayor superficie del lago posible. La idea que subyace de este propuesta es que la distancia que recorre el drone está relacionada con el área que el drone será capaz de monitorizar del lago. El objetivo, pues, es intentar optimizar la distancia que recorre el drone en lago. También se tendrá en cuenta que no exista redundancia en los datos obtenidos, penalizándose para ello las intersecciones entre rutas.

Este algoritmo ha dado buenos resultados con los que poder empezar, pero para una segunda fase de optimización se ha considerado oportuno intentar buscar solución a una importante limitación que introduce, que todas las balizas tengan que ser visitadas una sola vez. Esta limitación implica que por cada baliza sólo pueden existir dos rutas, una de entrada y otra de salida. Sin embargo, existen ciertas balizas que ocupan posiciones relevantes en la orilla del lago, que de ser visitadas varias veces aumentaría notablemente el áre cubierta por el drone.

Lo que en en este trabajo se plantea es dejar de lado la limitación de visitar cada baliza una única vez para poder buscar combinaciones de balizas que resulten en un recorrido más óptimo; de este modo, algunas balizas se visitarán varias veces y otras quedarán inutilizadas. Para conseguir un recorrido de estas características, se propone el uso de recorridos Eulerianos como manera de optimizar el conocido Chinese Postman Problem (CPP).

2 INTRODUCCIÓN

El seguimiento de la calidad de las aguas del Lago Ypacaraí es de suma importancia para poder comenzar a recuperar el estado original del lago, libre de vertidos que contaminan el lago y que favorecen la aparición de cianobacterias dañinas para el ecosistema. Para ello, es necesario el empleo de varias tecnologías de manera conjunta, como son los drones acuáticos de superficie y las técnicas de path planning, que para el caso de este proyecto comprenden algoritmos para encontrar rutas óptimas y también procedimientos que ayuden a resolver los anteriores.

2.1 Drones Acuáticos de Superficie

Los Drones Acuáticos de Superficie, conocidos en inglés como Autonomous Surface Vehicles/Vessels (ASVs) o Unmanned Surface Vehicles/Vessels (USVs), son vehículos que operan sobre la superficie del agua y que no precisan de una tripulación. Esta característica hace que sean muy útiles a la hora de llevar a cabo tareas que puedan ser peligrosas para los humanos o que requieran mucho tiempo y por lo tanto tiene varias aplicaciones tanto militares o de seguridad, como comerciales o de investigación. Entre las aplicaciones del primer grupo se encuentran, entre otras, la seguridad y vigilancia de costas o el monitoreo de actividad submarinos de guerra. De las aplicaciones del segundo grupo destacan la observación de entornos marinos, los estudios geofísicos o la inspección de centrales de energía offshore [3].

Los ASVs pueden manejarse de dos maneras: con control remoto (Remote Control, RC) o autónomamente. En cualquiera de los dos casos no es necesario contar con una tripulación, pero en el caso de RC sí hace falta tener a alguien que dirija el ASV desde tierra. De la otra manera, basta con determinar un recorrido a seguir por el ASV.

Todos los ASVs comparten características similares. La forma del vehículo suele ser de dos tipos, que son los catamaranes y los vehículos de un solo casco. Los catamaranes proporcionan mayor estabilidad debido a su diseño multi-casco (con dos cascos paralelos y del mismo tamaño) y por esta causa son los más empleados para la mayoría de aplicaciones. Algunos ejemplos de ASVs de esta forma que se pueden encontrar en el mercado son C-CAT 3 de ASV Global [3] o la gama WAM-V USV de Marine Advanced Research [4]. En cuanto a los vehículos monocasco, algunos ejemplos que se comercializan son la gama C-WORKER de ASV Global o AutoNaut USV de Autonaut ltd [5].

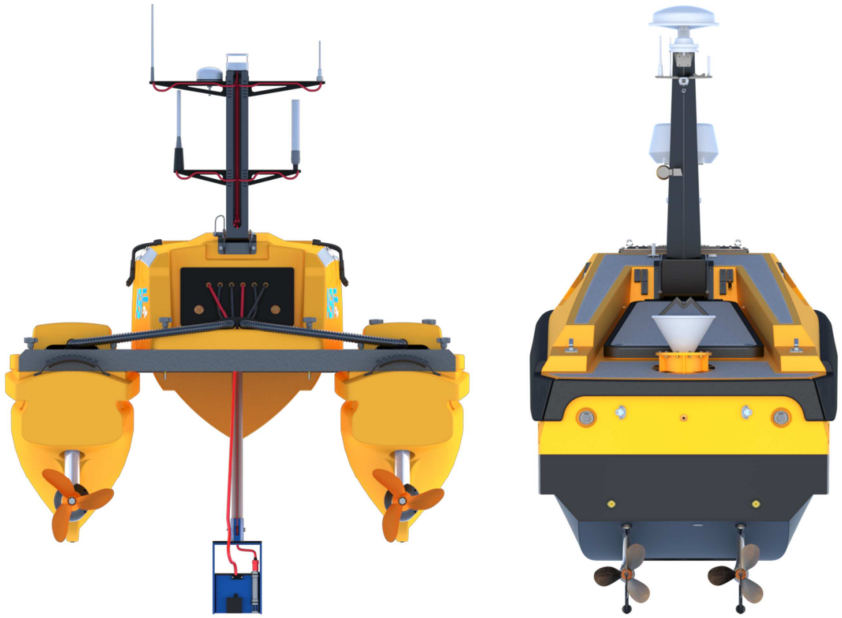


Figura 2 - Los ASVs C-CAT 3 (catamarán) y WORKER-8 (monocasco) de ASV Global [1]

Como carga de estas embarcaciones se añaden los componentes electrónicos necesarios para cada aplicación, tales como sensores (acústicos, metereológicos, batimétricos...), sistemas de comunicación, paneles solares, etc.

Anteriormente ya se ha hecho uso de ASVs para estudiar las condiciones del agua en entornos marítimos o en lagos. Algunos ejemplos en el mar son las pruebas realizadas en el puerto de Livorno en Italia [6], protegido de olas y tráfico marino, o las mediciones de floración de algas dañinas en el estrecho de Johor de Singapur [7]. En cuanto a lagos o acuíferos, también se han hecho mediciones en torno a la floración de algas en el Lago Zurich [8], o de calidad del agua en general en un estanque de acuicultura cerca de La Paz (Mexico) [9].

2.2 Path Planning

Tal y como se ha comentado anteriormente, el ASV puede funcionar de manera completamente autónoma, es decir, sin necesidad de intervención de ningún humano.

Para el funcionamiento autónomo del ASV es necesario que este tenga la capacidad de crear un recorrido a seguir, lo que se conoce como *Path Planning*. Como condiciones del recorrido se pueden establecer una serie de puntos que se han de visitar, caminos que no se pueden recorrer, puntos de comienzo o de final de la ruta, etc. Por otro lado, el recorrido se busca en función de un parámetro que se quiera optimizar, como puede ser la búsqueda de un recorrido con la menor distancia posible, el menor tiempo de recorrido, etc.

Anteriormente ya se han buscado soluciones a problemas de path planning en entornos acuáticos. Por ejemplo, en el Lago Keller en Minnesota querían controlar la invasión de carpas y optaron por un algoritmo conjunto de TSPN (Travelling Salesman Problem with Neighborhoods) que visitaría todas las áreas rectangulares propuestas, y otro algoritmo que se encargaría de hacer el barrido de cada rectángulo [10]. En el Lago Winippeg en Canadá midieron la concentración de clorofila-A valiéndose de un sistema inteligente basado en aprendizaje estadístico para clasificar las regiones y más tarde utilizando redes neuronales para cuantificar la concentración [11]. Por otro lado, también se ha planteado el problema desde el punto de vista de un recorrido entre dos puntos A y B con obstáculos, como en el caso de una simulación usando MATLAB donde se calculó el trayecto más corto posible utilizando el algoritmo de la colonia de hormigas [12], o en el del ASV *Springer* de la Universidad de Plymouth donde hicieron uso del Fast Marching Method (FMM) para intentar dar solución al mismo problema [13].

En el caso del problema del Lago Ypacaraí, el objetivo es hacer mediciones de la calidad del agua por toda la extensión del lago con tal de obtener la mayor cantidad de información posible sobre su estado en diferentes puntos. Por lo tanto, hay que planificar la ruta de modo que recorra la mayor área del lago posible, es decir, maximizar la distancia del recorrido por el que transita el ASV.

Se parte de la base de que se tiene un número fijo de balizas o nodos que el ASV puede visitar a lo largo de su recorrido, realizando en total un número de conexiones igual al número de balizas. La distancia a maximizar sería, en este caso, la resultante de la suma de las distancias equivalentes a cada tramo o conexión del recorrido. Un problema de tales características puede abordarse desde el punto de vista de la teoría de grafos, representando las balizas como vértices y las conexiones como aristas. Hay varios teoremas que se encargan de dar solución a la trazabilidad de un grafo, que se dividen entre los que crean un recorrido tocando todos los vértices y los que crean un recorrido pasando por todas las aristas.

La teoría de grafos fecha su origen en 1736 cuando el matemático Leonhard Euler resolvió el famoso *Problema de los Siete Puentes de Könisberg*. El problema trataba de dar respuesta a si existía la posibilidad de cruzar los 7 puentes de la ciudad e Könisberg (Prusia) pasando una sola vez por cada uno de ellos. Se vio que este problema podía expresarse como un problema de teoría de grafos, un área todavía inexistente en 1736, y basándose en ello surgió el concepto de grafo Euleriano. Un grafo Euleriano es aquel grafo conectado en el que cada nodo (vértice) tiene un número par de conexiones (aristas). Más adelante, y relacionado con lo anterior, se planteó el *Chinese Postman Problem*, que busca encontrar el camino más corto que pase por todas las conexiones al menos una vez, es decir, la trazabilidad del grafo pasando por todas las aristas.

Por otro lado, están los recorridos que pasan por todos los vértices. El estudio de este tipo de recorridos lo empezó el matemático irlandés William Rowan Hamilton, y de ahí se ha desarrollado el concepto de grafo Hamiltoniano, que sería el grafo que se pueda recorrer pasando por todos los vértices. Aunque no se ha podido encontrar una caracterización para grafos de este tipo, sí que se han demostrado varias condiciones suficientes para que un grafo pueda considerarse Hamiltoniano. Basado en esto, se planteó el *Travelling Salesman Problem*, que efectivamente busca el recorrido más corto que pase por todos los nodos al menos una vez.

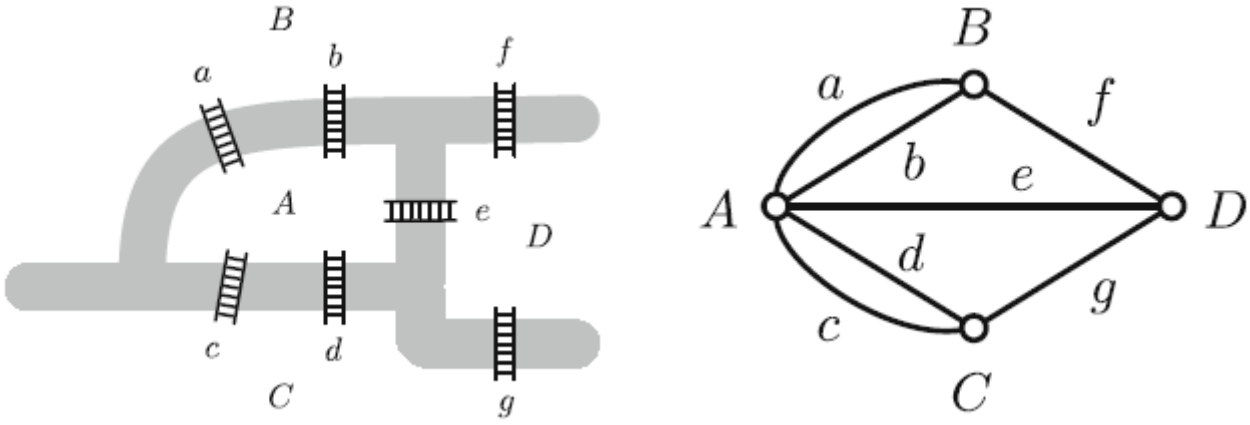


Figura 3 - Mapa de los siete puentes de Könisberg y su representación como grafo [14]

2.2.1 Los Siete Puentes de Könisberg

Euler resumió su solución al problema de los Siete Puentes de Könisberg definiendo las siguientes reglas que se podían aplicar a cualquier problema similar con un número diferente de áreas de tierra o de puentes:

> "Así, cualquiera que sea la solución propuesta, uno puede fácilmente determinar si un recorrido puede o no ser realizado, cruzando cada puente una vez y siguiendo las siguientes normas:
>
> - Si hay más de dos areas a las que llegan un número impar de puentes, el recorrido es imposible de realizar.
> - De todos modos, si el número de puentes es impar para exactamente dos áreas, el recorrido es

possible de realizer siempre que se empiece por una de estas áreas.

- Finalmente, si no hay áreas a las que lleguen un número impar de puentes, el recorrido puede realizarse sea cual sea el punto de inicio.

Con las normas presentadas, el problema propuesto puede ser resuelto [14]."

2.2.2 Chinese Postman Problem (CPP)

Este problema fue planteado por el matemático chino Meigu Guan (también conocido como Mei-Ko Kwan) en 1962 y habla de un cartero que sale de la oficina de correos y tiene que entregar cartas en casas que están en diferentes calles para luego volver a la oficina. El problema reside en encontrar el recorrido cerrado de menor distancia que cumpla con su objetivo. [14]

Si recordamos el concepto de grafo Euleriano como grafo conectado en el que cada nodo tiene un número de conexiones par, y relacionándolo con el tercer punto de la solución de Euler, se puede ver que en el caso de tener un grafo Euleriano la distancia mínima a recorrer será la suma de las distancias de todas las conexiones:

$$l = m$$

siendo *l* la distancia a recorrer y *m* la suma de las distancias de las conexiones.

Por lo contrario, si el grafo no es Euleriano, alguna de las conexiones tendrá que ser recorrida más de una vez, por lo que esa distancia será mayor a la suma de las distancias de todas las conexiones. En este caso, el número de conexiones a repetir será de:

$$\frac{(2k-1)!}{2^{k-1}(k-1)!}$$

siendo *2k* el número de nodos con conexiones impares. Para cada par de nodos con conexiones impares, se calcula la distancia que tendría una nueva arista que conectara los dos nodos y en función de esto se elige la de menor distancia. De esta manera, la distancia final a recorrer sería:

$$l = m + d$$

donde a *m* se le suma *d* que es la menor de las distancias entre los nodos impares. Puede observarse que con añadir conexiones entre los nodos impares se obtiene un grafo Euleriano, ya que todos los nodos pasan a tener un número par de conexiones.

Al recorrido que pasa por cada arista al menos una vez se le llama *camino Euleriano*, y si además el recorrido es el de menor distancia posible, como es el resultado del problema descrito, se dice que es *camino Euleriano mínimo.*

El CPP varía en complejidad dependiendo del tipo de grafo. Normalmente, si se trata de un grafo dirigido como de uno no dirigido, el problema puede resolverse en tiempo polinomial. Si en cambio el grafo es mixto, es decir, si contiene rutas tanto dirigidas como no dirigidas; o si el problema se altera de alguna manera puede llegar a ser NP-hard también [15].

2.2.3 Travelling Salesman Problem

El problema presentado es el siguiente: un comerciante quiere hacer un viaje de ida y vuelta en el que visite ciertas ciudades, pasando una vez por cada una de ellas; conociendo las distancias entre cada par de ciudades, quiere buscar el recorrido de menor distancia posible.

El problema recién descrito puede traducirse en un grafo ponderado donde los vértices corresponderían a las ciudades y las distancias entre estas serían los diferentes pesos. La solución a este problema vendría a ser la minimización de la suma los de pesos de las aristas por las que pasa el recorrido. Si el número de ciudades n es alto, el número de recorridos posibles entre ellas también lo será, dado por la siguiente expresión:

$$\frac{(n-1)!}{2}$$

por lo que se presenta como un problema muy difícil de resolver en general, categorizado como NP-hard o NP-complejo. Debido a la complejidad, hay técnicas que resultan de ayuda para resolver este tipo de

problemas, como pueden ser por ejemplo los algoritmos genéticos, dentro de los algoritmos metaheurísticos.

2.3 Algoritmos Genéticos

Tal y como se ha comentado en apartados anteriores, el TSP está categorizado como un problema NP-hard o NP-complejo. Esto significa que no puede resolverse en tiempo polinomial y por lo tanto hay que buscar alternativas para tratar de encontrar una solución, ya que intentar todas las combinaciones, lo que se llama fuerza bruta, sería inviable. El CPP por su parte sí puede ser resuelto en tiempo polinomial, si se trata de un grafo dirigido o de uno no dirigido.

Para hacer frente a problemas NP-hard, hay que buscar alternativas a la fuerza bruta. Una alternativa que se emplea frecuentemente es el uso de Algoritmos Genéticos (GAs). Los GAs están inspirados en la evolución natural: parten de una población candidata y van escogiendo las que mejor solución dan al problema, para basándose en ellas crear nuevas soluciones. Este tipo de algoritmos actúan como algoritmos heurísticos, realizando la búsqueda entre un conjunto de soluciones a un problema [16].

Volviendo a la evolución natural, todo organismo vivo tiene un conjunto de cromosomas único que lo describe. Un cromosoma es una molécula larga de ADN, que se conforma de genes, siendo cada gen representativo de una característica del individuo. En el momento en el que un organismo se reproduce, pasa copias de sus cromosomas a los siguientes individuos, ocurriendo ocasionalmente algún tipo de *mutación* que altera las características de esa copia, y por lo tanto pudiendo mejorar la descendencia. En la reproducción sexual, se combinan cromosomas de dos individuos generando nuevos individuos que son una mezcla de los anteriores, pudiendo de esta mezcla surgir mejoras también. Para los AGs se mantiene la terminología utilizada para la evolución natural, y al equivalente a la reproducción sexual se le conoce como *crossover*. Al igual que en la naturaleza los mejores individuos tienen más probabilidades de sobrevivir, lo mismo pasa con los AGs.

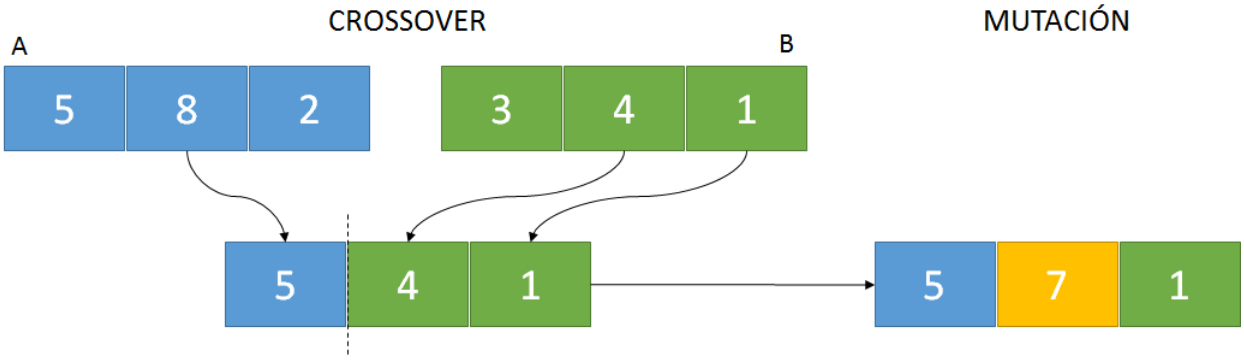


Figura 4 - Operaciones de crossover y mutación

La metodología utilizada en los AGs puede dividirse en dos pasos básicos: la selección de los mejores individuos y la reproducción de los seleccionados. La selección de los mejores individuos se hace basándose en el *fitness* de cada individuo, que depende de una función de evaluación que recibe una posible solución y devuelve un número indicando cómo de buena es esa solución. Para el segundo paso, se utilizan las operaciones de crossover y mutación y así se obtiene una nueva generación de individuos. El funcionamiento de los GAs puede verse reflejado en la Figura 5:

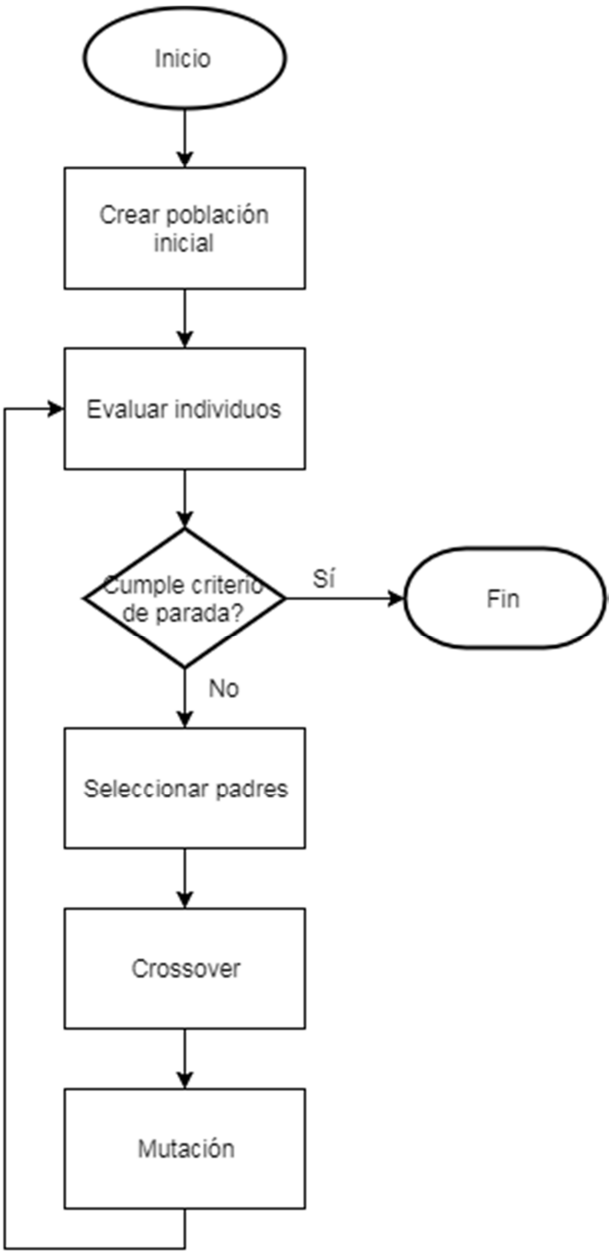


Figura 5 – Funcionamiento de los GAs

Como puede verse, el primer paso en un algoritmo genético es la creación de una población inicial. Para ello se escoje un número de individuos, las simulaciones se han realizado con 100, que se crean aleatoriamente para poder comenzar con el algoritmo. Seguidamente y si todavía no se cumple con el criterio de parada, que para este proyecto se trata de un número de generaciones, se procede a la selección de los padres. Esta selección puede realizarse con diferentes criterios, tratándose en este caso de la selección de los mejores individuos de la población. De esa selección de individuos se consigue un *offspring* o descendencia mediante los procesos de crossover y mutación anteriormente descritos y se vuelve una vez más a calcular el fitness de cada individuo, para volver a empezar con el proceso. Una vez se ha llegado al número de generaciones propuesto (el criterio de parada en este caso), se para el algoritmo. El output del algoritmo será por lo tanto una población del tamaño inicial, de donde podrá extraerse el indiviuo de mayor fitness.

Los AGs no dejan de ser métodos aproximados, así que de ellos se pueden esperar soluciones válidas pero que no tienen por qué ser óptimas. De todos modos, se pueden ajustar diferentes parámetros para conseguir resultados que se acerquen más al objetivo de la búsqueda.

3 DESCRIPCIÓN GENERAL DEL PROBLEMA

Con el objetivo de abordar el proyecto del Lago Ypacaraí, se describireron los detalles del problema a resolver y sobre ello se efectuó una primera aproximación que diera solución al problema descrito. Esta primera aproximación ya dio unos resultados válidos y buenos, pero se ha intentado replantear el modelado del problema para intentar buscar una solución más óptima y por lo tanto unos resultados mejores.

3.1. Problema de cobertura basado en putos de interés

Tal y como se ha explicado en el apartado de motivación del proyecto, el Lago Ypacaraí ha visto la calidad de sus aguas comprometida por el vertido de residuos tanto domésticos, como de procedencia agraria e industrial. Los residuos que se vierten en las aguas que desembocan en el lago han disparado los niveles de nitrógeno y fósforo, favoreciendo la floración de cianobacterias, con su consiguiente producción de toxinas que son perjudiciales para los organismos del entorno.

Para recuperar el estado natural del lago, es de suma importancia hacer un seguimiento de la calidad del agua así como de los focos de contaminación. La detección de los puntos de contaminación es el primer paso a seguir para poder acabar con la floración de cianobacterias, ya que una vez erradicados los vertidos, el lago por sí mismo será capaz de recuperarse y volver a su estado original.

En pos de dar una solución para hacer un seguimiento eficiente de la calidad del agua en el Lago Ypacaraí, se ha planteado el uso de un conjunto de balizas que estarían situadas por todo el perímetro del lago. El número de balizas a utilizar será en un principio de 60 (las cuales estarán numeradas con números entre 0 y 59), y cada baliza se encontrará a menos de 1 km de la siguiente. Las balizas se han dispuesto con una separación de 1 km porque se pretende plantear el conjunto de balizas como una red de sensores, donde el drone pueda enviar los datos recopilados. La disposición de las balizas se ve ilustrada en la Figura 6:

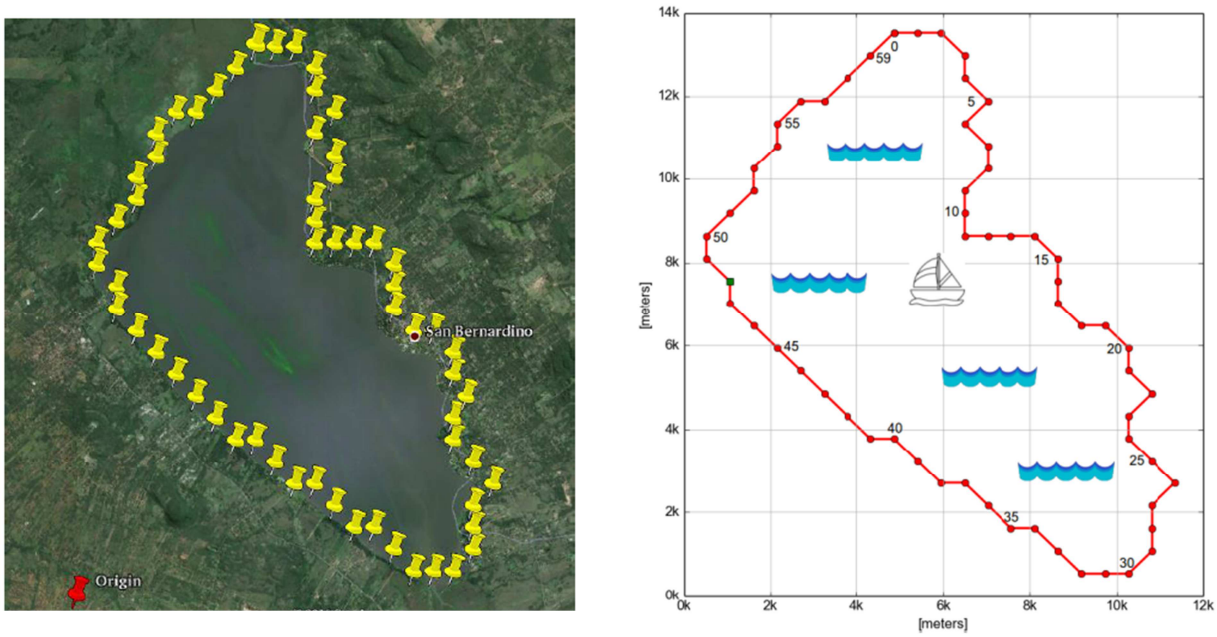


Figura 6 - Posición de las balizas sobre el Lago Ypacaraí [17]

El ASV debe viajar visitando las diferentes balizas (puntos de interés) y recogiendo muestras de agua por el camino. De esta manera, se podrá obtener un rango de muestras más completo que utilizando unos puntos fijos de muestreo.

Durante el recorrido se evaluarán los niveles de ficocianina, que es un pigmento que alerta de la presencia de cianobacterias. Este pigmento absorbe la luz natural a 620 nms y la fluorescente a 650 nm, que es lo que hace que pueda ser detectado por los sensores. A partir de los resultados obtenidos de esas mediciones se puede

categorizar la presencia de cianobacterias como perjudicial para el ser humano, a nivel moderado por encima de 20,000 cells/mL y a nivel alto si alcanza los 100,000 cells/mL [17]. Además del uso de estos sensores, el ASV también servirá para medir otras características del lago, empleando sensores de PH, de temperatura, de batimetría etc.

Para que las muestras sean representativas de todo el lago, el objetivo del ASV es el de cubrir la mayor área del lago posible, cosa que intentará conseguir recorriendo la mayor distancia posible entre las diferentes balizas. Teniendo en cuenta que las balizas más cercanas se encuentran a menos de 1 km de distancia y la distancia mayor entre balizas es de alrededor de 14 kms, el ASV tiene que ser capaz de encontrar la combinación de balizas que le permita acercarse a un recorrido óptimo desde el punto de vista de la distancia máxima. La idea principal es que la distancia que recorre el ASV está correlada con la cobertura del lago sensada por este.

3.2. Modelo del problema basado en TSP

Una primera solución dada al problema del lago se basó en el Traveling Salesman Problem (TSP), tratando de encontrar la distancia máxima que se pueda recorrer pasando por todas las balizas, una vez por cada una de ellas. El problema original trata de encontrar la distancia mínima con la que poder realizar ese recorrido, por lo que para el caso del lago se cambió la función objetivo para buscar el resultado contrario. Otra diferencia con el TSP original es que en el caso del Lago Ypacaraí hay que añadir restricciones referentes a las rutas inválidas, es decir, las que pasan por tierra.

En la introducción que se ha hecho sobre el TSP en el apartado anterior, ya se ha mencionado que el número de combinaciones de un problema de estas características es muy elevado; por un número de balizas n concretamente sería de:

$$\frac{(n-1)!}{2}$$

por lo tanto, para nuestro caso con $n = 60$ balizas el uso de la fuerza bruta para resolverlo sería impensable, ya que hay $6.93 \cdot 10^{79}$ diferentes combinaciones de balizas, sin tener en cuenta las restricciones por rutas inválidas. No por nada se cataloga al TSP como NP-hard o complejo.

Para poder abordar un problema de tal magnitud, se planteó el uso de algoritmos genéticos (GAs). De esta manera, se parte de una población inicial creada a partir de rutas válidas que pasan por todas las balizas, y utilizando los GAs se intenta llegar a una solución más óptima a efectos de distancia recorrida. Para ello, se definieron varias funciones para calcular el fitness o la calidad de cada individuo.

Las rutas creadas para la población inicial deben contener, como se ha dicho, todas las balizas y sin repetirse. Además de eso, tienen que ser creadas de manera que no incluyan rutas inválidas, es decir, que no contengan ninguna ruta que pase por tierra, ya que a efectos prácticos esto sería imposible de implementar.

En cuanto a la función de fitness, esta se encarga de valorar el área total que se cubre dependiendo de la distancia recorrida y del número de veces que se pasa por el mismo punto, para no considerarlo como un área nueva recorrida. La función de fitness también aplica penalizaciones a sus rutas inválidas, que se han definido de dos maneras diferentes: la primera, con *Death Penalty*, descarta cualquier recorrido que contenga rutas inválidas asignándole un -1 como valor de fitness; la segunda, con *Penalty Factor*, considera los individuos inválidos para la evolución, pero les aplica una penalización en el fitness [18].

Una muestra de una solución obtenida mediante el uso del TSP se puede ver en la Figura 7, a la izquierda utilizando el Penalty Factor y a la derecha con el Death Penalty:

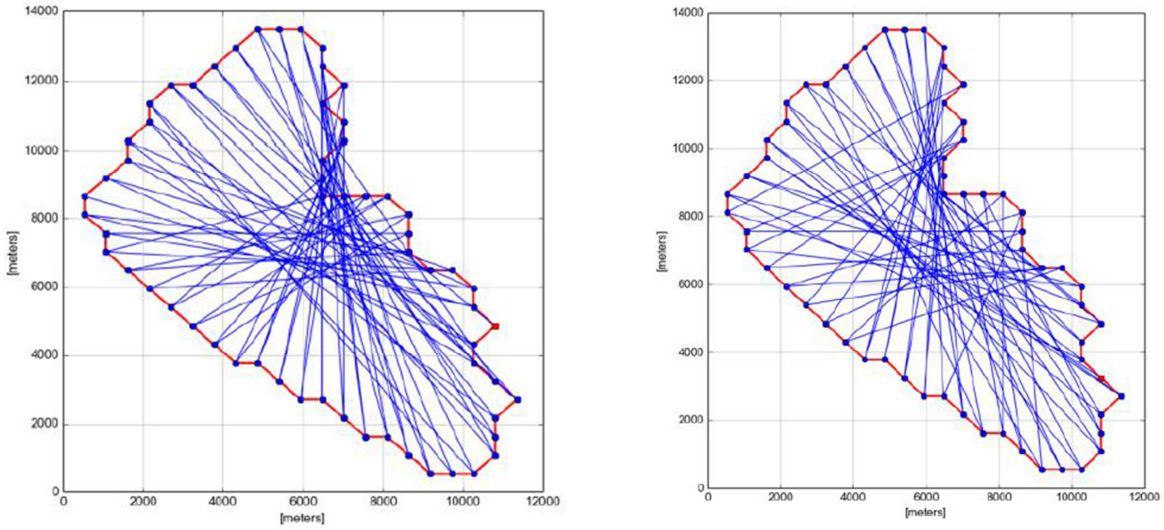


Figura 7 - Ejemplo de los resultados obtenidos mediante el uso del TSP [17]

Como puede observarse, el recorrido de la izquierda contiene varias rutas inválidas, pero el de la derecha está libre de ellas.

3.3. Modelo del problema basado en CPP

Como un intento de mejorar los resultados previamente obtenidos, se ha planteado la posibilidad de quitar una restricción que se respetaba para el caso del TSP, la que dicta que hay que pasar por todas las balizas una única vez. Se ha creído conveniente probar a hacerlo de esta manera después de ver que en las soluciones obtenidas mediante el uso del TSP, que pueden verse en el apartado anterior, el centro se transitaba varias veces en contraste con otros puntos del lago. Si se tiene en cuenta de que las intersecciones entre rutas no son útiles, ya que simplemente nos ofrecen información redundante, no conviene pasar por el mismo sitio más de una vez.

De esta manera, para este nuevo planteamiento del problema se ha abierto la posibilidad de pasar varias veces por la misma baliza y por lo tanto de dejar otras balizas sin visitar. Este nuevo planteamiento busca favorecer el uso de balizas ubicadas en sitios estratégicos, ya sea porque cuentan con rutas largas o porque no tienen muchas rutas inválidas, sobre otras balizas que pueden encontrarse peor colocadas.

También se ha decidido cambiar la manera de crear la población inicial, planteándolo desde un punto de vista de grafos. Así, en vez de ir añadiendo paradas al recorrido, se partirá de una matriz de conexiones para crear los individuos de esa población inicial.

4 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

El planeamiento con el que se ha abordado el problema del Lago Ypcaraí se basa en la teoría de grafos, que tal y como se ha comentado en la sección introductoria, fue estudiada por primera vez por el matemático suizo Leonhard Euler y su solución para el famoso *Problema de los Siete Puentes de Könisberg*. Más adelante se definió el concepto de grafo Euleriano, que hace referencia a un grafo conectado en el que cada nodo tiene un número par de conexiones.

Al contar con las 60 balizas repartidas por todo el perímetro, el problema del Lago Ypacaraí puede modelarse como un problema de grafos, tomando las balizas como nodos o vértices y las conexiones entre ellas como aristas. De esta manera, el objetivo del problema sería encontrar el recorrido que maximice la distancia utilizando un número dado de conexiones entre balizas. En un principio se quiere buscar un recorrido con tantas conexiones como nodos, es decir, un recorrido compuesto por 60 conexiones. La razón por la que se han escogido 60 conexiones es poder comparar los resultados obtenidos con los que se obtuvieron en fases previas del proyecto, con el algoritmo del TSP. En esta fase del proyecto no se ha impuesto la condición de que cada baliza tenga que ser visitada una vez, por lo que pueden crearse recorridos que pasen varias veces por la misma baliza y que dejen otras sin visitar.

4.1. Representación del invidividuo o solución

Un individuo para el algoritmo genético es, en efecto, una cadena cromosómica. Cada cadena se compone de diferentes eslavones, siendo cada eslavón la representación de una baliza. Por lo tanto, la solución obtenida sería una secuencia de balizas que en su conjunto compondrían el recorrido a seguir por el drone.

Para generar un primer conjunto de recorridos válidos con el que alimentar el algoritmo genético, se ha optado por crear una matriz de conexiones aleatorias. Esta matriz es de $n \times n$, siendo n el número de balizas que hay. La matriz es inicializada a 0 y posteriormente se añaden m unos distribuidos por la matriz de forma aleatoria; las posiciones de la matriz que contengan un 1 representarán las conexiones activas, que serán equivalentes a las conexiones entre las balizas. De esta manera, si hay un 1 en la posicion ij, significa que la ruta entre las balizas i-j está activa. El número de conexiones que habrá (m) dependerá del objetivo de cada apartado, que puede ser la comparación con métodos anteriores o la adecuación de la ruta a un catamarán con un tiempo de vida concreto.

La generación de la matriz de las conexiones se ha implementado de la manera representada en la Figura 8:

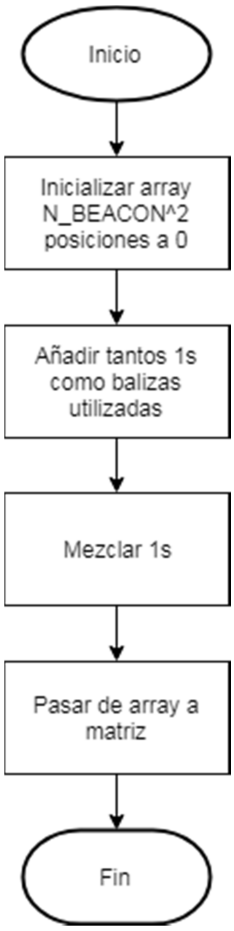


Figura 8 - Generación de matriz de conexiones

Una vez obtenida da la matriz de conexiones inicial, se comprueba que la misma no incluya rutas inválidas. Estas rutas inválidas pueden ser de dos tipos: las que unen una baliza consigo misma y las que pasan por tierra. Que pasen por tierra significa que dada la morfología del Lago Ypacaraí hay casos en los que al unir dos balizas mediante una línea recta es inevitable que esta salga del perímetro del lago, y dado que el vehículo es un catamarán esto sería imposible de implementar.

En el caso de que la matriz de conexiones incluya alguna ruta inválida, se procede a su modificación. Para ello, primero se calcula el producto Hadamard entre la matriz de conexión y la matriz de rutas inválidas. El producto Hadamard, también conocido como producto Entrywise o producto Schur, multiplica las matrices elemento por elemento, de la siguiente manera:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \circ \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} \times b_{11} & a_{12} \times b_{12} \\ a_{21} \times b_{21} & a_{22} \times b_{22} \end{pmatrix}$$

La matriz de conexión, como ya se ha dicho antes, contiene 1s solamente en las posiciones que hacen referencia a una ruta que se pretende realizar, y en el caso de la matriz de rutas inválidas los 1s se refieren a las rutas que no pueden realizarse. De esta manera, multiplicando las dos matrices mediante el producto Hadamard se encuentran las conexiones inválidas que hay en la matriz de conexiones incial. Un ejemplo de esta comprobación puede verse a continuación:

$$\underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}}_{Matriz\ de\ conexiones} \circ \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{Matriz\ de\ rutas\ inválidas} = \begin{pmatrix} 1 \times 1 & 0 \times 0 \\ 1 \times 0 & 0 \times 1 \end{pmatrix} = \begin{pmatrix} \boxed{\begin{matrix} Ruta\ inválida \\ \hat{1} \end{matrix}} & 0 \\ 0 & 0 \end{pmatrix}$$

La matriz de conexiones no será válida hasta que esté libre de rutas inválidas, por lo que cada vez que se halla una conexión inválida, se cambia por otra válida. El proceso de modificación se rige por la norma descrita en la Figura 9:

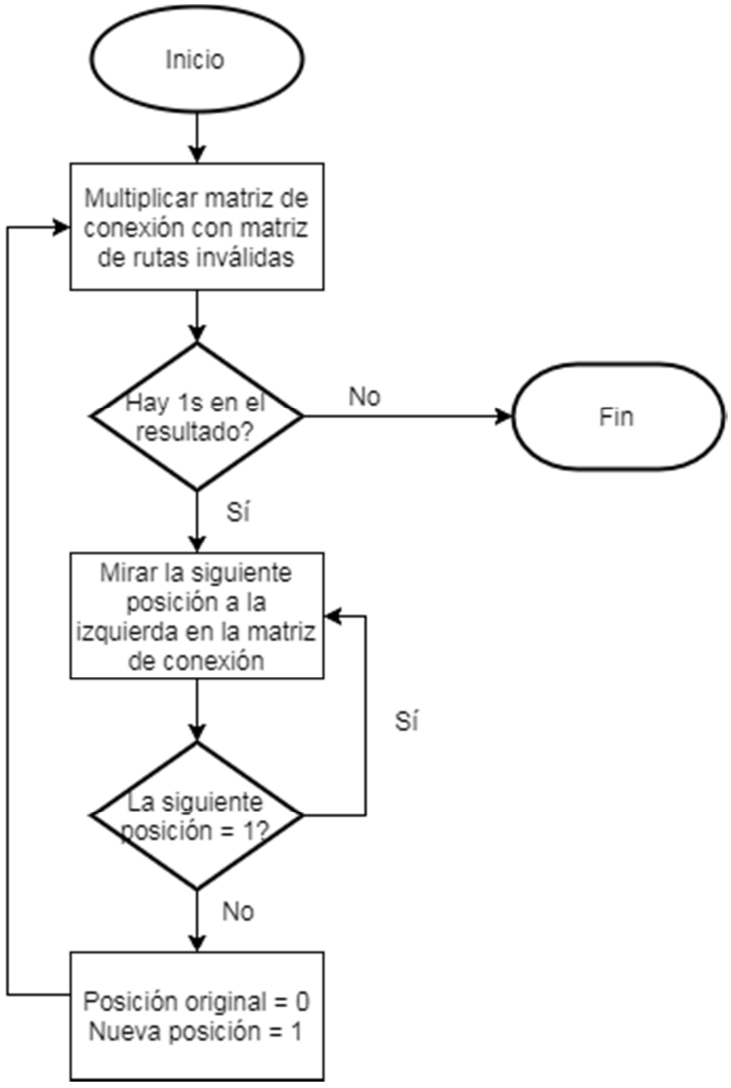


Figura 9 – Proceso de eliminar rutas inválidas

Tal y como puede verse, cada vez que el producto escalar de las dos matrices devuelve un 1, el 1 correspondiente de la matriz de conexiones se va desplazando hacia la izquierda hasta encontrar una posición referente a una conexión válida. Cuando se halla la nueva posición, la anterior se pone a 0 y la nueva a 1. Procediendo de esta manera, se modifican todas las rutas inválidas que la matriz de conexión pueda contener para finalmente obtener una matriz de conexión válida.

Una vez se obitene una matriz válida, hay un aspecto más a considerar: que no contenga rutas repetidas. Que haya una ruta repetida no hace referencia a que la conexión A-B se considere dos veces, ya que esta hace referencia a un único elemento de la matriz y por lo tanto solo puede tomar valores de 0 o 1. Lo que puede ocurrir es que se consideren las rutas A-B y B-A, que sí hacen referencia a dos posiciones diferentes de la matriz y por lo tanto se estaría ante un escenario posible. Aunque sean dos elementos diferentes de la matriz, al tratarse de un grafo no direccional A-B y B-A son iguales, y aunque fuera direccional seguiría sin ser válido en nuestro caso, ya que la distancia recorrida aumentaría pero no el área cubierta. Por lo tanto, en caso de haber conexiones duplicadas ($ruta_{ij} = ruta_{ji} = 1, para\ cualquier\ i,j \leq 60$) la matriz de conexiones generada se descarta y se inicial el proceso otra vez, creando una nueva matriz inicial y modificándola para eliminar las rutas inválidas. Este aspecto del proceso puede verse en la parte superior de la Figura 11.

Cuando se consigue una matriz de conexión sin rutas inválidas ni conexiones repetidas, el siguiente paso es obtener una ruta partiendo de la matriz de conexiones. Desde el punto de vista de la teoría de grafos, la solución a este problema lo da el Chinese Postman Problem (CPP). Tal y como ya se ha explicado en el

apartado de la introducción, el CPP busca crear un recorrido que pase por todas las aristas. Para ello, los pasos a seguir en un procedimiento estándar son los siguientes [19]:

1. Hacer una lista de todos los vértices con conexiones impares.
2. Hacer una lista de todas las combinaciones posibles entre vértices impares.
3. Para cada par de vértices, buscar las aristas que los unan con el menor peso posible.
4. Buscar las combinaciones de vértices impares que minimicen la suma de los pesos.
5. En el grafo original, añadir las aristas del paso 4.
6. La longitud del CPP óptimo es la suma de todas las aristas originales más las del paso 4.
7. Una ruta que corresponda a ese peso mínimo puede encontrarse fácilmente ahora.

Por lo descrito en los pasos de 4 a 6, se puede deducir que la solución óptima al problema general del CPP será la que minimice la suma de los pesos y esto se consigue teniendo solo la suma de los pesos de las aristas originales. Es decir, en un grafo en el que originalmente todas las aristas tengan un número par de vértices, no será necesario implementar los pasos 2 a 6, ya que desde un primer momento existirá una ruta que sea capaz de pasar por todas las aristas una única vez y que por lo tanto de solución al CPP.

Para aplicar esto al proyecto del Lago Ypacaraí, se cambia el concepto de "pesos" por la distancia de cada tramo. Siendo el objetivo cubrir la mayor área del lago posible, añadir aristas significaría pasar por el mismo lugar dos veces y por lo tanto no aportaría más que información redundante, a costa de la duración del trayecto. De esta manera, se reafirma lo anteriormente expuesto sobre que la solución óptima sea aquella que permita minimizar la suma de los pesos, o distancia en este caso, y que por lo tanto será una solución en la que no haya que añadir más vertices o tramos.

Recordando de apartados anteriores, un grafo en el que todos los vértices tienen un número de aristas pares se conoce como grafo Euleriano. Si se consigue tener un grafo Euleriano, no hay que añadir nuevas aristas para resolver el CPP y por lo tanto la solución obtenida es óptima. Para conseguir que así sea, se ha creado una función que toma una matriz de conexiones y fuerza que esta represente un grafo Euleriano. Lo primero que hace esta función es mirar cuáles son las balizas (vértices) que tienen un número impar de conexiones (aristas). En caso de que haya balizas de esa condición, se empiezan a examinar las conexiones hasta encontrar una que contenga una baliza con conexiones impares. Cuando se da con una conexión que lo cumpla, se modifica la baliza de esa conexión, reemplazándola con otra de las balizas impares. Con esto se consigue pasar de tener dos balizas impares a dos pares cada vez. El proceso se continúa hasta que al final todas las balizas son pares, tal y como puede verse en la Figura 10.

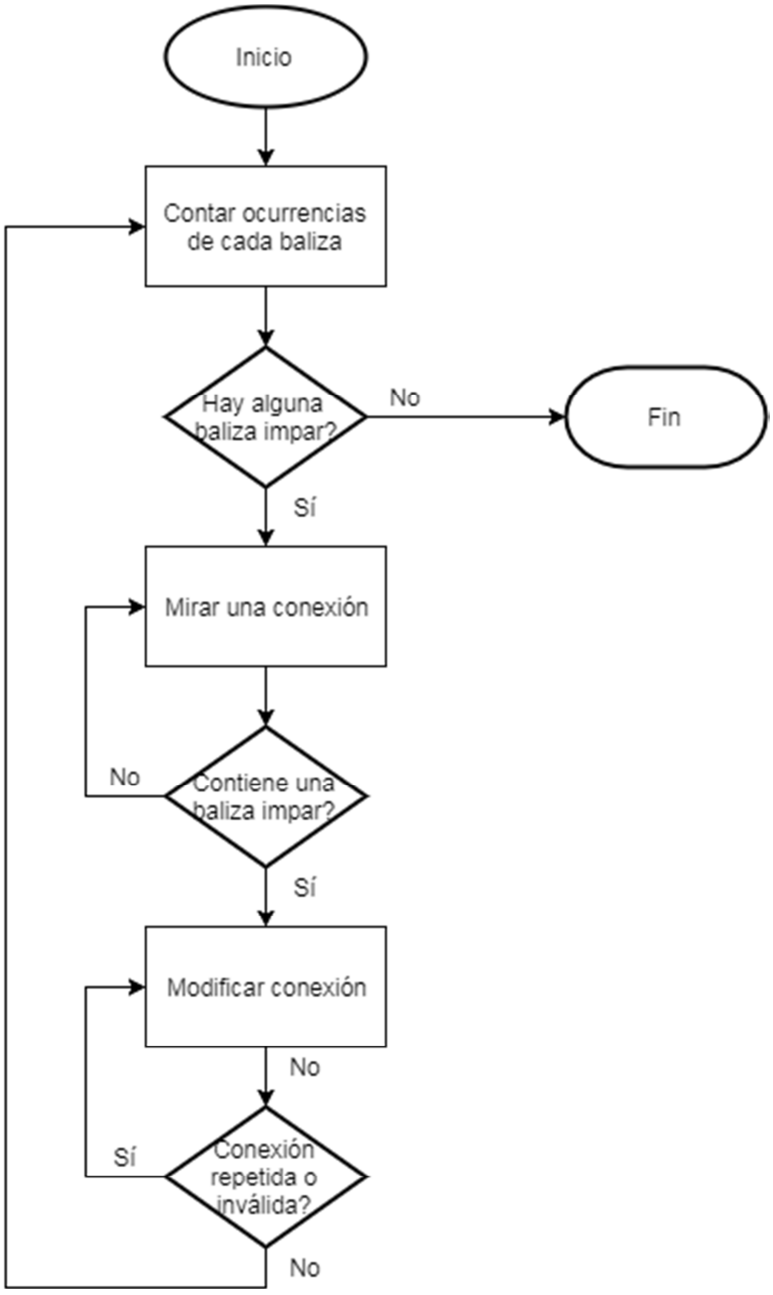


Figura 10 - Obtención de grafos Eulerianos

Teniendo un grafo Euleriano, lo único que queda por hacer es conseguir un circuito Euleriano que lo recorra. Un circuito Euleriano, al ser un circuito, tiene como punto de inicio y punto final el mismo nodo, y este nodo puede ser cualquiera de los utilizados en grafo. Por lo tanto, se puede escoger cualquiera de los nodos y partiendo de ese ir creando el circuito. Teniendo la matriz de conexiones lo único que hay que hacer es ir encadenando las conexiones en orden, es decir, si nuestra conexión inicial es A-B la próxima tiene que ser de la forma A-C o B-C, para finalmente obtener B-A-C o A-B-C en cada caso. Ya que el grafo es no direccionado no afecta cuál de los nodos de la conexión se pone primero. Una de las ventajas de tener un circuito Euleriano es que al empezar y finalizar en el mismo punto podemos escoger cualquier baliza como nodo de inicio.

Después de conseguir tener un grafo Euleriano, hay una última cosa a tener en cuenta: todos los nodos han de estar conectados entre sí. Es decir, la condición de que todos los vértices tengan un número par de aristas garantiza que todos los nodos tengan una conexión de salida por cada conexión de entrada, que es una condición indispensable para poder crear un circuito Euleriano, pero no garantiza que no puedan crearse sub-circuitos que no tengan conexión entre ellos. Por lo tanto, hay que hacer una última comprobación para mirar que esta condición se cumple.

El proceso completo de obtención de un circuito Euleriano puede verse en la Figura 11:

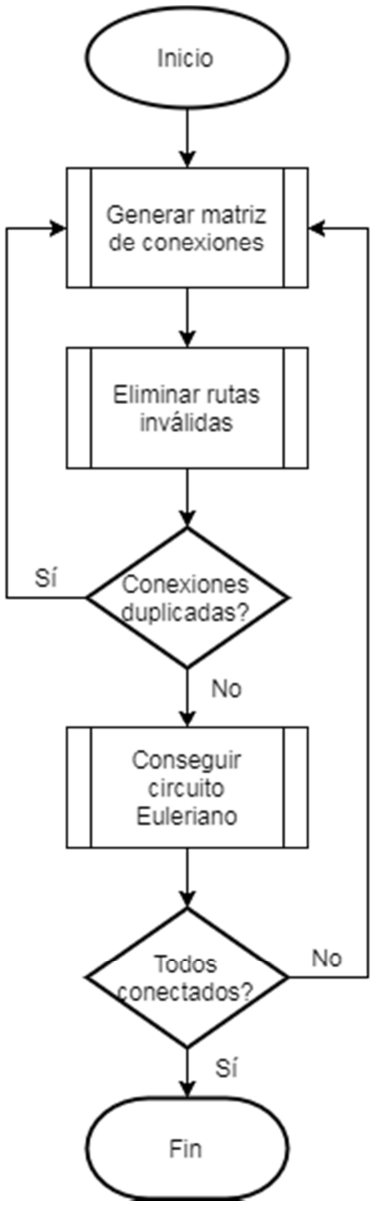


Figura 11 - Obtención de un circuito Euleriano

En resumen, el proceso de obtención de un circuito Euleriano pasa por generar una matriz de conexiones aleatorias, eliminar las rutas inválidas que pueda contener y si tampoco tiene conexiones duplicadas seguir para conseguir un circuito Euleriano. Finalmente, si todos los nodos están incluidos en este circuito, se da por finalizado el proceso y se alimenta el algoritmo genético con el recorrido conseguido.

4.2 Función de fitness

Tal y como se ha introducido en el apartado de Algoritmos Genéticos, todos los individuos se evalúan mediante una función de fitness, que representa la calidad de un individuo respecto a la función objetivo que se persigue.

Para un problema clásico de tipo TSP, la función de fitness viene dada por la distancia total recorrida, que en ese caso se busca que sea la menor posible. En la fase anterior de proyecto, donde se tomaba como base el TSP, la distancia recorrida se utilizaba para calcular el área total que se había cubierto por el drone, pero con la diferencia de que en este caso el objetivo era maximizar dicha área y por consiguiente la distancia recorrida. Como adición, también se tenía en cuenta si el drone pasaba por el mismo sitio en más de una ocasión, ya que esto no aportaba ningún valor, simplemente se trataría de información redundante. De este modo, el área

cubierta con duplicidad se restaba del valor del fitness calculado a partir de la distancia o área total. La función de fitness resultante cumple la siguiente relación:

$$fitness = S_{ASV} \cdot \left(d_{b_{n-1}b_0} + \sum_{i=0}^{n-2} d_{b_i b_{i+1}} \right) - {S_{ASV}}^2 \cdot n_{intersec}$$

donde S_{ASV} se refiere al área que cubre mientras avanza, $n_{intersec}$ es el número de intersecciones que se producen durante el recorrido (i.e., número de veces que el drone pasa por un lugar anteriormente visitado), y $d_{b_x b_y}$ es la distancia entre dos balizas consecutivas.

Ya se ha comentado anteriormente que debido al perfil del Lago Ypacaraí hay varias conexiones entre balizas que no pueden realizarse, ya que para ello necesitarían cruzar por tierra. Por lo tanto, es necesario que el fitness penalice dichas rutas inválidas en caso de que las haya. Se plantearon dos maneras diferentes de incorporar la penalización por rutas inválidas: Death Penalty y Penalty Factor.

En el caso del Death Penalty, si el recorrido contiene una ruta inválida el valor del fitness pasa a ser -1 directamente, como se muestra a continuación:

$$fitness_{DP} = \begin{cases} S_{ASV} \cdot \left(d_{b_{n-1}b_0} + \sum_{i=0}^{n-2} d_{b_i b_{i+1}} \right) - {S_{ASV}}^2 \cdot n_{intersec}, & con\ rutas\ inválidas \\ -1, & sin\ rutas\ inválidas \end{cases}$$

Por lo tanto, los recorridos que contengan rutas inválidas quedarán descartados inmediatamente, y no se tendrán en cuenta para futuras generaciones.

En el Penalty Factor, en cambio, la penalización que se le da a las rutas inválidas no es tan drástica, regiéndose por la siguiente expresión:

$$fitness_{PF} = \left(1 - \frac{n_{rinv}}{n_{beacon} - 1}\right) \cdot \left[S_{ASV} \cdot \left(d_{b_{n-1}b_0} + \sum_{i=0}^{n-2} d_{b_i b_{i+1}} \right) - {S_{ASV}}^2 \cdot n_{intersec} \right]$$

donde n_{rinv} es el número de rutas inválidas contenidas en la solución y n_{beacon} es el número de balizas que existen. En este caso el valor del fitness irá a peor por cada ruta inválida que se incluya en el recorrido, pero eso no descartará necesariamente la solución, por lo que se tendrá en cuenta para futuras generaciones.

4.3 Selección de individuos

Cada vez que se obtiene una población, y si se quiere seguir con el algoritmo genético durante más generaciones, es necesario seleccionar un número de individuos entre los que realizar los procedimientos de crossover y mutación. Esta selección se lleva a cabo mediante dos técnicas en paralelo: por un lado el elitismo y por el otro el método de la ruleta. La proporción de la población que se obtiene mediante cada uno de los dos procedimientos se representa mediante una probabilidad de elitismo que puede ajustarse para cada simulación.

La técnica del elitismo trata simplemente de pasar los individuos con mejor valor de fitness a la siguiente generación, sin que estos sufran ninguna alteración debida al crossover o mutación. De esta manera, se garantiza que los mejores individuos no se pierdan de una generación a otra.

En el método de la ruleta se le asigna a cada individuo un espacio proporcional al valor de fitness que tiene. Es decir, cuanto mejor sea el fitness de un individuo, mayor es la representación que obtiene en la ruleta. Un ejemplo de este método puede verse en la Figura 12:

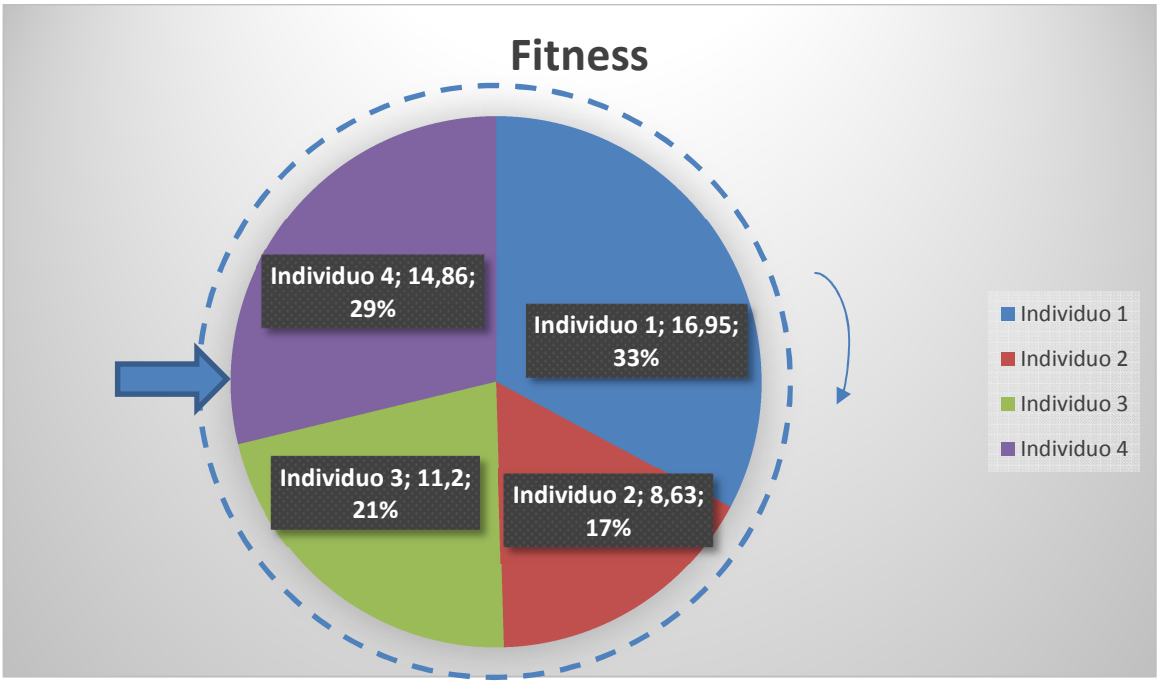


Figura 12 – Ejemplo de selección por ruleta

En este ejemplo, el Individuo 1 es el que mayor fitness tiene y por lo tanto es también el que mayor representación tiene en la ruleta, ocupando un 33% de la misma. En el otro extremo está el Individuo 2, que es el de menor fitness y consecuentemente el de menor representación. En el momento de hacer girar la ruleta, el individuo 1 tendrá casi el doble de probabilidad de ser elegido como "padre" para la siguiente generación que el individuo 2, pero ambos tienen esa posibilidad.

4.4 Operaciones genéticas

Una vez se ha hecho la selección de los individuos que actuaran como padres, se procede al crossover y la mutación de las que se obtendrán los nuevos individuos de la generación. Para ello, cada vez se escogerán dos individuos "padres" que mediante crossover producirán dos individuos "hijos", a los que dependiendo de una probabilidad también se les aplicará un procedimiento de mutación.

Hay diferentes maneras de realizar el crossover y para este problema se ha escogido el crossover ordenado (Ordered Crossover, OX). Este crossover funciona de la siguiente manera: primero, se coge un segmento de uno de los padres y se mantiene intacto y en la misma posición; luego partiendo desde la primera posición libre que ha quedado en el hijo, se va rellenando con los datos del segundo padre. La misma operación se repite cambiando el orden de los padres, para obtener un segundo hijo. Un ejemplo del procedimiento puede verse en la Figura 13:

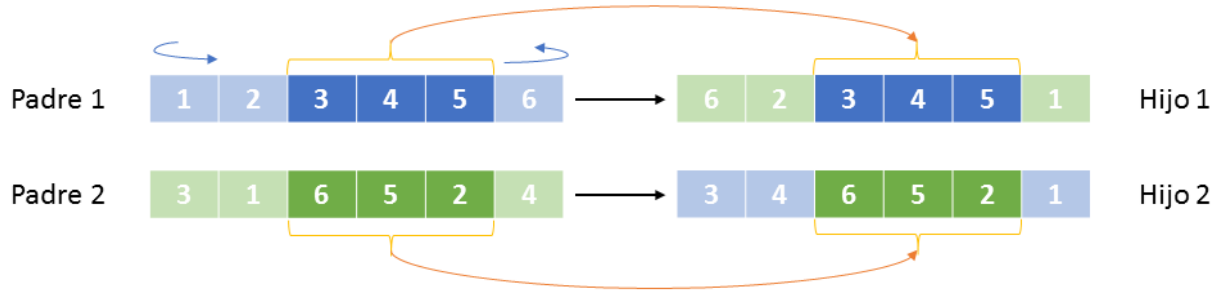


Figura 13 - Ejemplo de crossover

Como puede verse en el ejemplo, para crear al Hijo 1 primero se copia la secuencia [3, 4, 5] directamente del Padre 1 y en la misma posición. Luego, empezando desde la primera posición libre, se va rellenando con los datos del Padre 2 empezando también desde la misma posición. Es decir, para rellenar la última posición del Hijo 1 se mira la última posición del Padre 2, pero como en este caso el número 4 ya va incluido por la

herencia del Padre 1 se pasa al siguiente (que sería el correspondiente a la primera posición); como el número 3 también está ya en el hijo, se mira el siguiente que sería el 1 y este es el que se incluye en la última posición del Hijo 1. Se procede de esta manera hasta completar todas las posiciones del Hijo 1, y lo mismo se hace para la generación del Hijo 2.

Una vez se han obtenido los dos individuos "hijos", se procede con la mutación. La mutación, como ya se ha dicho, depende de una probabilidad (de mutación): ese porcentaje de individuos será mutado y el resto se mantendrá tal y como se ha obtenido en el crossover. Cuando un individuo es mutado, cada gen o atributo puede intercambiar posiciones con otro de los genes dentro del mismo individuo, estando este cambio sujeto a otra probabilidad. Por lo tanto, posición por posición el gen se intercambiará con otro o se mantendrá en su lugar original dependiendo de esta última probabilidad. Este tipo de mutación es habitual en secuencias de índices, por lo que es aplicable en las secuencias de balizas que componen el recorrido de cada solución.

Para finalizar, se calcula el fitness de cada uno de los individuos de la nueva generación y se comprueba que el mejor de ellos no contenga rutas inválidas. En el caso de que así sea, ya se tiene una generación válida con la que seguir el algoritmo; pero si por lo contrario el individuo contiene rutas inválidas se comienza con la generación desde el principio.

4.5 Criterio de parada de la evolución

El algoritmo genético se repite tantas veces como generaciones se quieran realizar. A mayor complejidad del problema, mayor será el número de generaciones necesario para que el algoritmo llegue a convergir. Para este problema se ha empleado una variable con la que poder cambiar el número de generaciones a realizar y de esta manera poder adecuarlo a cada fase.

Es importante ajustar adecuadamente el número de generaciones para cada caso, ya que hay que garantizar que el algoritmo converja para poder encontrar una solución válida y estable, pero por otro lado cada generación alarga el tiempo necesario para llegar a esa solución y por lo tanto hay que intentar no excederse.

.

5 RESULTADOS

La primera parte del apartado de resultados se divide en tres diferentes fases. En una primera fase de pruebas se compara el algoritmo del TSP con el planteado con circuitos Eulerianos y CPP. Para esta etapa no se tiene en cuenta la restricción de las rutas inválidas, en un simplemente intento de tener una primera impresión de la posible mejora o no de los resultados obtenidos. Para el segundo conjunto de resultados, se ha añadido la condición de las rutas inválidas con tal de adecuarlo al problema del Lago Ypacaraí y comprobar que la solución obtenida sigue siendo válida y mejor que el TSP. Por último, se han hecho simulaciones para diferentes números de balizas con el objetivo de poder medir las distancias recorridas en cada caso y encontrar una relación.

Para todas las simulaciones se utilizan probabilidades de mutación y de crossover de 0.2 y 0.8 respectivamente, a no ser que se especifique lo contrario, y el número de generaciones se fija a 1000. En la última parte de los resultados se hacen diferentes pruebas para decidir si los parámetros recién descritos resultan los más óptimos para las simulaciones que se llevan a cabo, o por si lo contrario hay que hacer algún ajuste en los mismos con tal de poder encontrar soluciones óptimas. Los valores utilizados habitualmente para los parámetros más relevantes del algoritmo genético se han recogido en la Tabla 1:

Tabla 1 - Parámetros GA

Número de balizas	**60**
Probabilidad de crossover	0.8
Probabilidad de mutación	0.2
Proporción de elitismo	0.2
Nº individuos en la población	100
Nº de generaciones	1000

5.1 Plataforma de simulación

En la fase anterior del proyecto se planteó una solución basada en el TSP, donde cada baliza se visitaba una única vez y no había ninguna que quedara en desuso. Para poder simular dicho algoritmo, se implementó un programa en Python versión 2.7. en el que se empleó el módulo Distributed Evolutionary Algorithm for Python (DEAP). Este módulo contiene las funciones necesarias para implementar los algoritmos genéticos, entre ellas *cxOrdered* para el crossover ordenado, *mutShuffleIndexes* para la mutación, *selRoulette* para la selección de ruleta y *selBest* para el elitismo.

Para poder marcar las balizas se guardaron las coordenadas reales del lugar que ocuparía cada una de ellas en el perímetro del lago, para a partir de ahí poder calcular las distancias que hay entre cada una de las balizas. Por otro lado, se recogió en una matriz la información referente a las rutas inválidas, para de este modo poder determinar si un recorrido pasaba por tierra o no. En lo que se refiere al cálculo del fitness, también se creó una matriz que contiene las intersecciones que puede haber entre las rutas, a fin de poder aplicar las penalizaciones pertinentes a los recorridos que las contengan.

Con todos estos datos, el simulador crea una primera población inicial que sea válida con la que alimentar el algoritmo genético. Esta primera población se trata de un recorrido por todas las balizas teniendo en cuenta la restricción de las rutas inválidas, y de que cada baliza no puede ser visitada en más de una ocasión. De ahí en adelante, haciendo uso de las funciones ofrecidas por el módulo DEAP, se llevan a cabo los procesos de selección, crossover y mutación para obtener una nueva generación con la que retomar el algoritmo genético

paso por paso. Finalmente, al alcanzar el número de generaciones propuesto, se escoge al mejor individuo, i.e., el que mejor valor de fitness tiene.

Para esta fase del proyecto se ha mantenido la estructura del programa, empleando el algoritmo genético previamente programado, pero con dos cambios significativos. Por un lado, se desecha la restricción de visitar cada baliza una única vez. Por otro lado, se ha creado una nueva función para obtener la población inicial, basándose en los recorridos Eulerianos.

5.2 Sin tener en cuenta las rutas inválidas

No todas las conexiones entre las balizas son posibles de implementar en el caso del Lago Ypacaraí, ya que por su morfología hay varias de estas rutas que pasan por tierra en algún punto del trayecto. De todas maneras, como una primera aproximación, se han hecho pruebas sin tener en cuenta estas restriccionesa aplicar por las rutas inválidas. De esta manera, se puede tener una primera idea de los resultados que se puedan obtener más adelante, y lo que es más importante, se va a poder comparar el algoritmo propuesto con la implementación anterior basada en el TSP, para basándose en esta comparación seguir adelante para buscar esas mejoras o desechar la idea.

En la Tabla 2 pueden verse los resultados obtenidos utilizando el algoritmo del TSP. Se han hecho tres pruebas diferentes:

1. Sin intersecciones: no se aplica ninguna penalización por tener intersecciones, es decir, no se penaliza el hecho de pasar dos veces por el mismo punto.
2. Death Penalty: se tienen en cuenta las intersecciones y se penalizan con un fitness de -1 las rutas inválidas.
3. Penalty Factor: se tienen en cuenta las intersecciones y se aplica una penalización en el fitness por las rutas inválidas.

Tabla 2 - TSP sin tener en cuenta las rutas inválidas

TRAVELLING SALESMAN PROBLEM

Sin Intersecciones		Death penalty		Factor de penalización	
Sim	Fitness	Sim	Fitness	Sim	Fitness
0	16,62	0	16,018	0	16,149
1	16,691	1	16,1	1	16,199
2	16,634	2	16,162	2	16,074
3	16,554	3	16,133	3	16,168
4	16,602	4	16,052	4	16,095
5	16,665	5	15,967	5	16,042
6	16,553	6	16,112	6	16,09
7	16,668	7	16,104	7	16,067
8	16,594	8	16,099	8	16,237
9	16,57	9	16,065	9	16,143
10	16,615	10	16,16	10	16,125
11	16,577	11	16,053	11	15,979
12	16,616	12	16,124	12	16,169
13	16,619	13	16,17	13	16,112
14	16,648	14	16,205	14	15,966
15	16,637	15	16,095	15	16,166
16	16,653	16	16,07	16	16,025
17	16,643	17	16,121	17	16,166
18	16,666	18	16,163	18	16,2
19	16,614	19	16,205	19	16,054
Media	16,621	Media	16,109	Media	16,111
Std. Dev.	0,039	Std. Dev.	0,061	Std. Dev.	0,074
Mejor fit	16,691	Mejor fit	16,205	Mejor fit	16,237

Al no tener rutas inválidas en estas simulaciones, el Death Penalty y el Penalty Factor dan el mismo resultado. Teniendo en cuenta que la diferencia entre el Death Penalty y el Penalty Factor reside en el tratamiento que se les da a las rutas inválidas, que en este caso no se tienen en cuenta, los dos medidores funcionan de la misma manera. Es decir, en ambos casos siemplemente se tienen en cuenta las intersecciones. Es lógico por lo tanto que para los apartados 2 y 3 el fitness que se obtiene sea menor que para el apartado 1, donde no hay penalizaciones de ningún tipo.

Los resultados obtenidos sin tener en cuenta las intersecciones son del orden de 16.6 de fitness, mientras que teniendo en cuenta las intersecciones esa puntuación baja hasta el orden de 16.1-16.2. La desviación estándar obtenida es bastante baja por lo que los resultados de las diferentes simulaciones no tienen una gran variación, parece que el algoritmo converge.

Los resultados obtenidos utilizando el CPP se pueden observar en la Tabla 3:

Tabla 3 – CPP sin tener en cuenta las rutas inválidas

CHINESE POSTMAN PROBLEM

Sin Intersecciones		Death penalty		Factor de penalización	
Sim	Fitness	Sim	Fitness	Sim	Fitness
0	17,507	0	18,253	0	16,881
1	16,95	1	17,75	1	17,651
2	18,062	2	17,219	2	16,928
3	17,801	3	17,445	3	17,427
4	17,665	4	17,36	4	18,432
5	18,397	5	17,509	5	17,174
6	17,609	6	17,128	6	17,017
7	18,96	7	17,842	7	16,618
8	17,663	8	17,328	8	17,675
9	17,542	9	16,942	9	17,326
10	19,767	10	17,757	10	16,878
11	17,553	11	17,281	11	18,152
12	17,15	12	18,153	12	17,395
13	17,531	13	17,664	13	16,229
14	17,191	14	18,124	14	17,367
15	16,827	15	17,014	15	16,955
16	17,786	16	17,889	16	16,761
17	16,942	17	16,429	17	17,863
18	17,614	18	18,101	18	16,928
19	16,993	19	17,339	19	17,602
Media	17,675	Media	17,526	Media	17,263
Std. Dev.	0,710	Std. Dev.	0,470	Std. Dev.	0,535
Mejor fit	19,767	Mejor fit	18,253	Mejor fit	18,432

Para este caso también las simulaciones sin tener en cuenta las intersecciones dan mejor resultado que las que sí las tienen en cuenta, como era de esperar. También puede verse que los valores de fitness en general son considerablemente mejores que en el caso del TSP, siendo el mejor resultado obtenido con el CPP (19.767) 3 puntos mayor que el obtenido con el TSP (16.691).

Otro dato que puede extraerse de los dos conjuntos de soluciones es que la desviación estándar para el caso del TSP es mucho menor que para el CPP, lo que significa que los resultados del segundo varían mucho más que para el primer caso. Esto puede significar que hacer las simulaciones con 1000 generaciones puede no ser suficiente para que el CPP converja.

Un ejemplo de una ruta obtenida en este apartado puede verse en la

Figura 14. Como puede observarse, hay varias rutas inválidas, ya que el recorrido pasa varias veces por tierra. También parece a simple vista que, en comparación con la Figura 7 obtenida de los resultados del TSP en el apartado 3.2 (página 18), las rutas están más distribuidas por el área del lago en vez de pasar mucho por la zona del centro, y que por lo tanto se cubre más superficie que con el TSP.

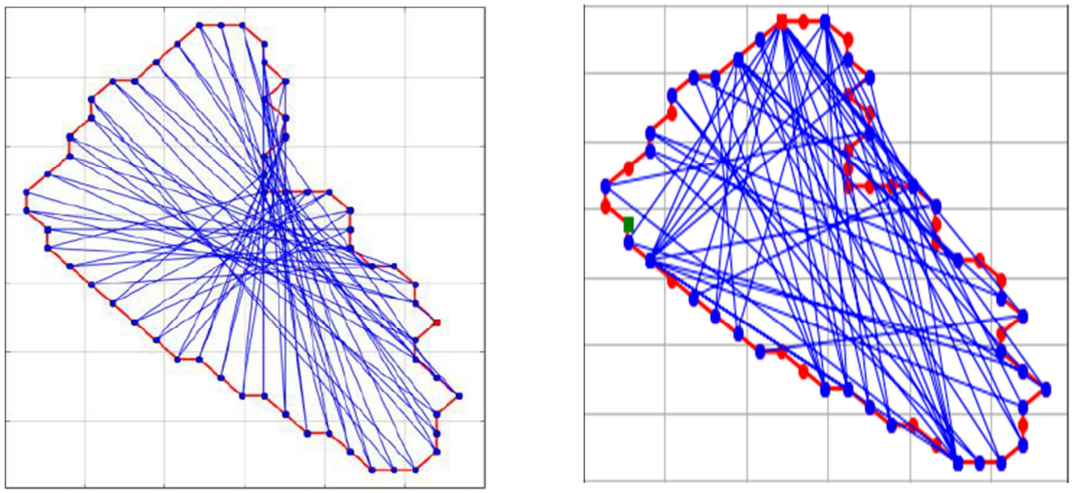


Figura 14 - Ejemplo de recorrido con rutas inválidas. TSP (izquierda) y CPP (derecha).

5.3 Teniendo en cuenta las rutas inválidas

Tal y como se ha dicho, existen varias balizas sobre el perímetro del Lago Ypacaraí que no pueden ser conectadas en línea recta a no ser que se cruce por tierra. Al tratarse del recorrido de un catamarán, las rutas tienen que ser por agua en su totalidad, por lo que hay que tener en cuenta esas conexiones inválidas a la hora de preparar el camino que el ASV seguirá. Esto se traduce en varias restricciones a tener en cuenta.

El número de estas rutas inválidas es bastante elevado, por lo que dificulta en gran medida encontrar una ruta que sea válida, además de empeorar el fitness del restulado obtenido. Volviendo a la

Figura 14 del apartado anterior, puede verse que en la zona superior derecha de la imagen hay una concentración de rutas inválidas muy significativa, que por otro lado es la que posibilita que se recorra toda el área del lago de manera balanceada en vez de concentrarse en algunas regiones concretas de la misma, como podía pasar con el algoritmo previamente utilizado, el TSP.

Los resultados que se han obtenido de las simulaciones teniendo en cuenta las rutas inválidas se han recogido en la Tabla 4.

Tal y como puede apreciarse, los resultados que da el CPP siguen siendo mejores que los obtenidos con el TSP, con una diferencia de 1.8 puntos en el mejor fitness y de 0.8 en la media de todos los fitness. En ambos casos el Death Penalty da mejor resultado que el Penalty Factor, y a eso hay que añadirle que con el Penalty Factor no es raro que el recorrido final contenga alguna ruta inválida, lo cual hace que la solución sea inservible. Por esta razón, de aquí en adelante las simulaciones se han realizado únicamente con el Death Penatly, dejando a un lado el Penalty Factor.

Tabla 4 - Comparacion teniendo en cuenta las rutas inválidas

TRAVELLING SALESMAN PROBLEM						CHINESE POSTMAN PROBLEM					
Death Penalty			Penalty Factor			Death Penalty			Penalty Factor		
Sim	R. Inv.	Fitness	Sim	R. Inv.	Fitness	Sim	R. Inv.	Fitness	Sim	R. Inv.	Fitness
0	0	14,758	0	0	14,222	0	0	15,385	0	0	15,117
1	0	14,535	1	2	14,21	1	0	14,913	1	0	14,517
2	0	14,435	2	1	14,046	2	0	15,022	2	2	14,634
3	0	14,247	3	0	14,511	3	0	15,326	3	2	14,945
4	0	14,585	4	0	14,431	4	0	15,476	4	0	15,486
5	0	14,606	5	1	14,388	5	0	15,943	5	0	15,644
6	0	14,769	6	0	14,449	6	0	15,388	6	0	14,997
7	0	14,641	7	0	14,645	7	0	15,443	7	0	14,782
8	0	14,715	8	0	14,54	8	0	15,217	8	1	14,438
9	0	14,531	9	0	14,293	9	0	15,499	9	2	14,964
10	0	14,823	10	0	14,546	10	0	15,52	10	1	15,181
11	0	14,532	11	0	14,723	11	0	14,721	11	3	15,267
12	0	14,518	12	0	14,407	12	0	14,561	12	1	15,593
13	0	14,581	13	0	14,587	13	0	16,357	13	1	15,892
14	0	14,456	14	1	14,417	14	0	16,198	14	2	15,798
15	0	14,376	15	1	14,187	15	0	15,182	15	2	14,806
16	0	14,502	16	2	14,23	16	0	15,524	16	0	15,391
17	0	14,382	17	0	14,515	17	0	16,612	17	0	14,677
18	0	14,559	18	0	14,751	18	0	14,717	18	0	15,38
19	0	14,232	19	2	14,257	19	0	14,971	19	1	15,042
Media fit	14,539		Media fit	14,418		Media fit	15,39875		Media fit	15,127	
Media r. inv.	0		Media r. inv.	0,5		Media r. inv.	0		Media r. inv.	0,9	
Std. Dev.	0,159		Std. Dev.	0,190		Std. Dev.	0,54512286		Std. Dev.	0,423	
Mejor fit	14,823		Mejor fit	14,751		Mejor fit	16,612		Mejor fit	15,892	

Un ejemplo de una ruta obtenida a partir de una simulación utilizando el Death Penalty puede verse en la Figura 15. En este caso, y a diferencia de en la Figura 14, no hay ninguna ruta inválida, por lo que todo el recorrido transcurre por el agua y no es necesario cruzar por tierra. A causa de esto, puede verse también que el área "conflictiva" que se ubica en la parte superior derecha de la imagen ya no es tan concurrida como antes. Esto se debe a que muchas veces no es rentable visitar las balizas que se ubican en esa área porque cuentan con muchas rutas inválidas y eso hace empeorar el fitness, por lo que el algoritmo decide visitar otras balizas más cómodas varias veces y dejar de visitar algunas como la 2 o la 5 por ejemplo. Pueden reconocerse las balizas que no se visitan como las de color rojo, y las que sí se visitan son las de color azul.

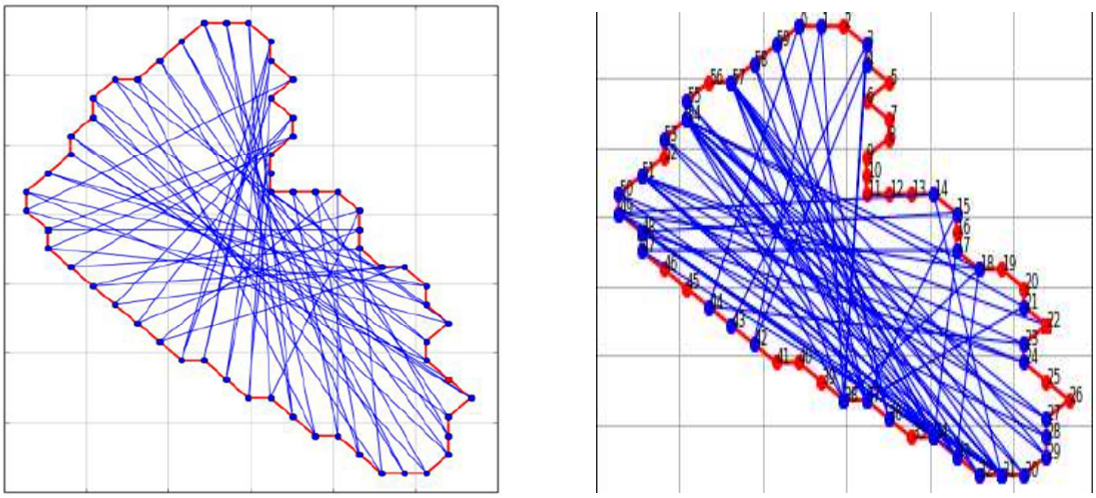


Figura 15 - Ejemplo de recorrido sin rutas inválidas. TSP (izquierda) y CPP (derecha).

5.4 Distancia recorrida por número de balizas utilizadas

En este apartado se ha buscado encontrar una relación entre balizas utilizadas y distancia total recorrida. Tener esta relación es importante ya que sabiendo la velocidad a la que viaja el catamarán y la duración de las baterías que lo impulsan, se puede fácilmente calcular la distancia que el ASV puede recorrer, que sería la dada por la siguiente expresión:

$$distancia\ (km) = v\ del\ catamarán\ \left({}^{km}/_{h}\right) \times autonomía\ de\ las\ baterías\ (h)$$

Y sabiendo la distancia que se puede recorrer, puede ajustarse el número de balizas a utilizar para que el algoritmo devuelva un recorrido realizable por el ASV.

El valor de la duración de las baterías que se emplearía en este caso sería en ausencia de sol, ya que en un día soleado las baterías se cargarían mientras el catamarán estuviera en marcha a través de los paneles solares instalados en su cubierta. Incluso en días nublados las baterías podrían cargarse ligeramente. De todos modos, para una primera aproximación y como valor mínimo es conveniente usar esta medida.

Las mediciones se han hecho para 10, 20, 30, 40, 50 y 60 balizas, siendo este último valor el que se venía utilizando hasta el momento, ya que corresponde al total de las balizas disponibles. La Tabla 5 muestra los valores obtenidos con 10, 20 y 30 balizas, mientras que en la Tabla 6 se muestran los resultados para 40, 50 y 60 balizas; todos ellos representados en metros.

Hay que indicar, por último, que la restricción del número de balizas que se puedan usar no define cuáles son las balizas que puedan usarse, por lo que es el algoritmo el que elije el conjunto de balizas que mejor resultado de en términos de valor del fitness.

Tabla 5 - Distancias recorridas con 10, 20 y 30 balizas

10 balizas		20 balizas		30 balizas	
DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000	
Simulación	Distancia	Simulación	Distancia	Simulación	Distancia
0	97385	0	205545	0	305830
1	99938	1	195957	1	306901
2	113047	2	211289	2	311449
3	103521	3	203191	3	287295
4	99456	4	201235	4	300524
5	101897	5	196531	5	305924
6	99383	6	193313	6	295544
7	93347	7	208296	7	309965
8	101007	8	203374	8	307744
9	99129	9	207606	9	276205
10	105774	10	207606	10	288931
11	100091	11	191793	11	323500
12	107824	12	195240	12	293514
13	95916	13	190961	13	286788
14	107304	14	205793	14	301590
15	99000	15	196275	15	296737
16	102147	16	202172	16	303892
17	101912	17	196336	17	292743
18	107706	18	196123	18	302069
19	102341	19	213200	19	311449
Media	101906	Media	201092	Media	300430
Mejor dist	113047	Mejor dist	213200	Mejor dist	323500

Tabla 6 - Distancias recorridas com 40, 50 y 60 balizas

40 balizas		50 balizas		60 balizas	
DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000	
Simulación	Distancia	Simulación	Distancia	Simulación	Distancia
0	397559	0	458328	0	555447
1	378602	1	473870	1	583031
2	373940	2	495377	2	547046
3	388392	3	468330	3	569198
4	382435	4	483495	4	570847
5	401433	5	465880	5	602866
6	397013	6	475296	6	596636
7	401952	7	455177	7	590411
8	381733	8	476004	8	520453
9	388524	9	499710	9	563273
10	356746	10	488392	10	593627
11	404870	11	469966	11	592852
12	373575	12	484670	12	588857
13	398248	13	456865	13	588165
14	384359	14	482972	14	543218
15	396938	15	488640	15	566702
16	396036	16	455171	16	580277
17	412726	17	488784	17	551483
18	386211	18	438186	18	513550
19	408494	19	476072	19	546821
Media	390489	Media	474059	Media	568238
Mejor dist	412726	Mejor dist	499710	Mejor dist	602866

Como puede observarse en la Tabla 5, para 10, 20 y 30 balizas las distancias recorridas son aproximadamente de 100, 200 y 300 km respectivamente, por lo que parece que sigue una distribución lineal. En la Tabla 6 en cambio puede verse que esta tendencia baja ligeramente, ya que las distancias medias recorridas son de 390, 474 y 568 km. Graficando los resultados de cada simulación para los diferentes números de balizas se obtiene lo siguiente:

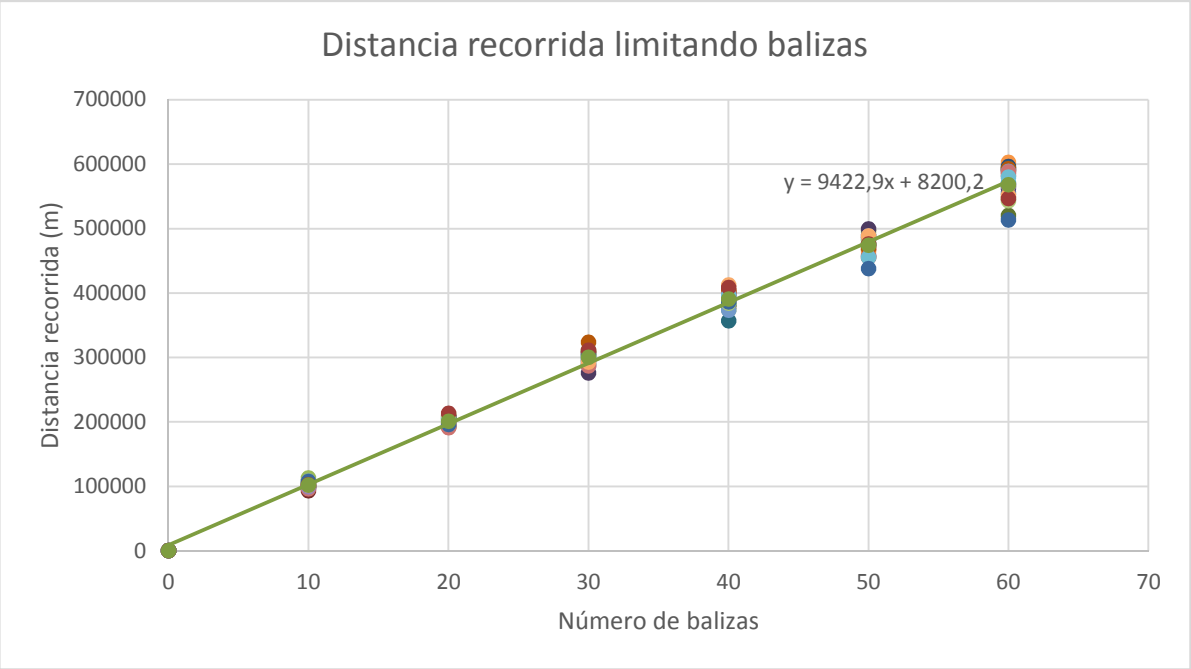


Figura 16 - Relación de la distancia recorrida por número de balizas empleado

En la Figura 16 - Relación de la distancia recorrida por número de balizas empleadoFigura 16 se puede ver que a medida que crece el número de balizas a utilizar los resultados se vuelven más dispersos en cuanto a distancias recorridas, es decir, que la varianza aumenta. Esto puede ser un síntoma de que el algoritmo no converge en las 1000 generaciones con las que se realizan las simulaciones. La convergencia para cada caso se muestra en la Figura 17, de izquierda a derecha y de arriba abajo, empezando en 10 balizas y hasta 60.

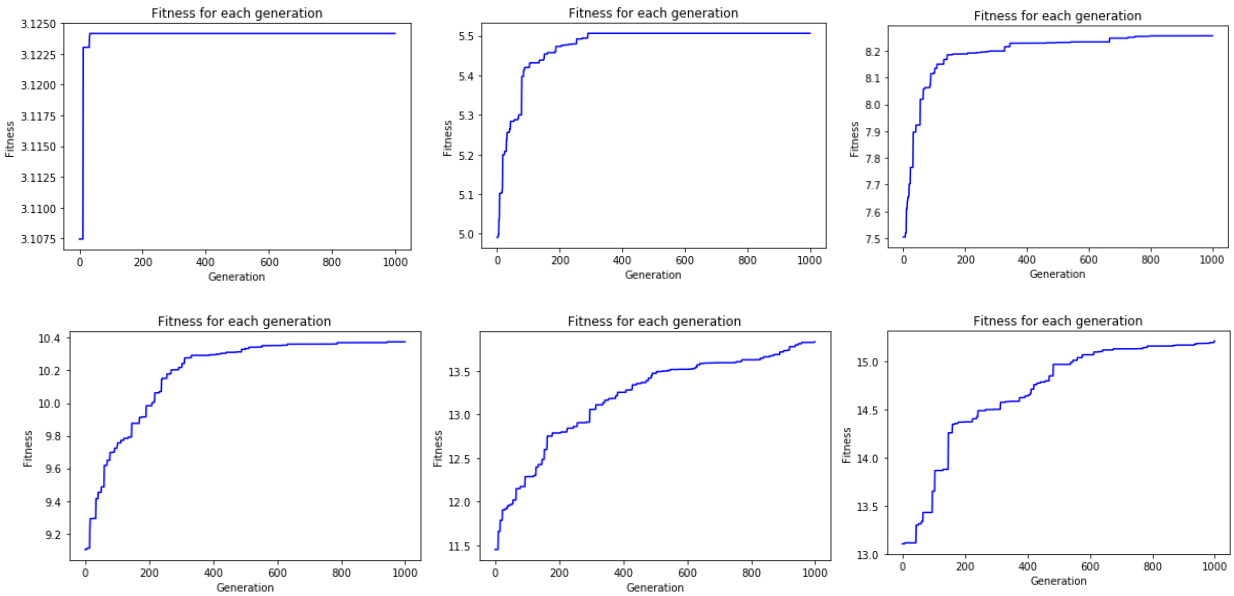


Figura 17 - Convergencias para las diferentes cantidades de balizas utilizadas

Tal y como puede verse en la Figura 17, para las simulaciones con 10 y 20 balizas el algoritmo converge claramente, incluso para el caso con 30 balizas. De ahí en adelante en cambio el algoritmo no llega a converger, por lo que para obtener un resultado óptimo habría que aumentar el número de generaciones.

Teniendo esto en cuenta, y viendo resultados anteriores en los que con 60 balizas utilizadas se conseguían recorridos de alrededor de 600 kms, se puede suponer que al aumentar las generaciones para los casos de 40, 50 y 60 balizas se mantendrá la tendencia de recorrer 100 kms por cada 10 balizas. Es decir, la relación vendría dada finalmente por:

$$Distancia\ (km) = Nº\ balizas\ \times 10$$

Esto nos da una relación muy cómoda ya que se trata de una relación lineal, y además de una proporción muy manejable para realizar los cálculos.

5.5 Comprobación de los parámetros de simulación

Todas las simulaciones llevadas a cabo hasta el momento se han hecho utilizando los siguientes parámetros:

- Probabilidad de crossover = 0.8
- Probabilidad de mutación = 0.2
- Número de generaciones = 1000

Es necesario, de todos modos, comprobar que los parámetros empleados son los adecuados y que están debidamente ajustados, ya que esto puede afectar a los resultados obtenidos, tal y como ya se ha podido ver en el caso de la convergencia del algoritmo.

Para ello, se ha procedido a hacer simulaciones variando uno de los parámetros cada vez. En el caso de la proabilidad de crossover, se han mantenido la probabilidad de mutación a 0.2 y el número de generaciones en 1000 para hacer pruebas con diferentes probabilidades de crossover, siendo estos valores 0.2, 0.4, 0.6, 0.8 (el empleado hasta ahora) y 1. No se ha tenido en cuenta el valor de 0 porque eso significaría no utilizar crossover. Los mismos valores se han utilizado para hacer simulaciones con diferentes probabilidades de mutación, una vez más manteniendo los valores de los otros dos parámetros. Por último, se han vuelto a utilizar los valores de crossover y mutación de 0.8 y 0.2 respectivamente y se ha ido aumentando el número de generaciones hasta llegar a un valor con el que el algoritmo converge. Las pruebas en este caso se han hecho para 1000, 1500, 2000, 2500, 5000, 10000 y 20000 generaciones.

5.5.1 Probabilidad de crossover

Los resultados obtenidos con diferentes valores de crossover pueden verse en la Tabla 7.

Como puede observarse, las medias obtenidas de las simulaciones para cada valor de la probabilidad del crossover parecen bastante similares y no parece haber ninguna variación que dependa del ajuste de este parámetro. De todos modos, con tal de certificar que se pueda hacer esta asunción es conveniente realizar la prueba de Anova (ANalysis Of VAriance). El análisis de la varianza sirve para determinar si las medias de diferentes subgrupos de muestras pueden tomarse como iguales, o si por lo contrario hay diferencias significativas y por lo tanto esa afirmación no es válida.

En la Tabla 8 puede verse el Anova para los resultados de diferentes probabilidades de crossover. Los datos más significativos que devuelve el análisis son el valor de F y el su valor crítico. En este caso el valor de F es menor al de F crítico, $1.57 < 2.47$, por lo que puede asumirse que todas las medias son iguales. Esto significa que puede decirse que los resultados del algoritmo no varían cambiando la probabilidad de crossover, y que por lo tanto los resultados obtenidos de las simulaciones hechas hasta el momento son los adecuados.

Tabla 7 - Resultados variando la probabilidad de crossover

CXPB = 0.2		CXPB = 0.4		CXPB = 0.6		CXPB = 0.8		CXPB = 1	
DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000	
Sim	Fitness	Sim	Fitness	Sim	Fitness	Sim	Fitness	Sim	Fitness
0	15,704	0	16,789	0	15,659	0	15,631	0	15,77
1	15,752	1	15,829	1	15,678	1	15,787	1	15,703
2	15,503	2	15,848	2	15,843	2	14,778	2	15,917
3	16,1	3	15,951	3	16,424	3	15,655	3	15,96
4	15,66	4	15,333	4	15,638	4	15,813	4	15,035
5	16,195	5	16,743	5	15,173	5	15,779	5	15,571
6	15,285	6	15,262	6	15,881	6	16,162	6	15,134
7	15,821	7	15,879	7	16,569	7	15,476	7	15,138
8	15,96	8	15,685	8	15,364	8	16,17	8	16,235
9	16,14	9	15,396	9	16,176	9	15,993	9	15,624
10	15,587	10	15,56	10	15,93	10	15,059	10	15,722
11	15,919	11	16,054	11	16,246	11	15,466	11	15,875
12	15,546	12	15,649	12	15,499	12	15,991	12	16,103
13	14,804	13	15,52	13	16,791	13	15,256	13	15,966
14	16,267	14	16,069	14	15,991	14	15,785	14	15,298
15	15,749	15	16,077	15	15,669	15	15,628	15	15,271
16	16,447	16	16,212	16	16,039	16	16,196	16	15,893
17	15,086	17	15,768	17	15,908	17	16,024	17	16,254
18	15,813	18	15,1	18	16,475	18	15,56	18	15,766
19	16,018	19	16,017	19	15,924	19	15,609	19	15,154
Media	15,768	Media	15,837	Media	15,944	Media	15,691	Media	15,669
Std dev	0,402	Std dev	0,437	Std dev	0,413	Std dev	0,369	Std dev	0,380

Tabla 8 - Anova para diferentes probabilidades de crossover

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
Columna 1	20	315,356	15,7678	0,16128069
Columna 2	20	316,741	15,83705	0,19138637
Columna 3	20	318,877	15,94385	0,17067508
Columna 4	20	313,818	15,6909	0,13636304
Columna 5	20	313,389	15,66945	0,14445026

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de cuadrados*	*F*	*Valor crítico para F*
Entre grupos	1,00788194	4	0,251970485	1,56667773	2,467493623
Dentro de los grupos	15,27895345	95	0,160831089		
Total	16,28683539	99			

5.5.2 Probabilidad de mutación

La Tabla 9 recoge los resultados de las simulaciones realizadas con diferentes valores de la probabilidad de mutación.

Tabla 9 - Resultados variando la probabilidad de mutación

MUTPB = 0.2		MUTPB = 0.4		MUTPB = 0.6		MUTPB = 0.8		MUTPB = 1	
DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000		DP, NGEN = 1000	
Sim	Fitness	Sim	Fitness	Sim	Fitness	Sim	Fitness	Sim	Fitness
0	15,353	0	15,654	0	14,919	0	16	0	16,022
1	16,387	1	15,946	1	15,837	1	15,844	1	16,366
2	15,777	2	15,377	2	16,333	2	15,982	2	16,016
3	15,619	3	15,95	3	15,644	3	16,115	3	15,595
4	15,053	4	15,676	4	15,802	4	16,056	4	16,319
5	16,166	5	16,01	5	15,495	5	15,309	5	15,957
6	15,59	6	15,753	6	15,819	6	16,221	6	16,49
7	15,112	7	15,62	7	15,837	7	16,151	7	15,988
8	15,916	8	16,179	8	15,914	8	16,264	8	15,676
9	16,083	9	16,515	9	15,571	9	16,335	9	16,038
10	15,921	10	16,112	10	15,658	10	16,039	10	16,281
11	16,287	11	16,169	11	15,813	11	16,127	11	16,641
12	16,7	12	16,567	12	16,501	12	16,165	12	15,763
13	15,531	13	16,019	13	16,197	13	16,251	13	15,544
14	15,47	14	16,568	14	15,681	14	15,52	14	15,864
15	16,365	15	16,257	15	15,864	15	16,622	15	16,894
16	16,196	16	15,583	16	16,379	16	16,307	16	16,03
17	16,234	17	15,957	17	15,519	17	16,067	17	16,028
18	15,715	18	15,719	18	16,807	18	15,959	18	17,017
19	16,262	19	16,073	19	16,131	19	16,004	19	16,163
Media	15,887	Media	15,985	Media	15,886	Media	16,067	Media	16,135
Std dev	0,451	Std dev	0,335	Std dev	0,418	Std dev	0,282	Std dev	0,399

Igual que en el caso para diferentes valores de probabilidad de crossover, aquí también las medias parecen similares, por lo que es necesario volver a comprobar que puedan tomarse como iguales utilizando el procedimiento del Anova, en la Tabla 10.

Observando los valores de F y F crítico obtenidos del análisis, se ve que el valor de F crítico es igual que para el apartado anterior, y en este caso también F sigue siendo menor que F crítico, 1.66 < 2.47. Una vez más, se puede asumir de esta manera que las medias obtenidas de las diferentes simulaciones de la Tabla 9 son iguales y que por lo tanto las simulaciones realizadas hasta el momento son válidas.

Tabla 10 - Anova para diferentes probabilidades de mutación

RESUMEN

Grupos	*Cuenta*	*Suma*	*Promedio*	*Varianza*
Columna 1	20	317,737	15,88685	0,2030474
Columna 2	20	319,704	15,9852	0,11201512
Columna 3	20	317,721	15,88605	0,17491952
Columna 4	20	321,338	16,0669	0,0793762
Columna 5	20	322,692	16,1346	0,15899331

ANÁLISIS DE VARIANZA

Origen de las variaciones	*Suma de cuadrados*	*Grados de libertad*	*Promedio de los cuadrados*	*F*	*Valor crítico para F*
Entre grupos	0,96545806	4	0,24136451	1,65692321	2,46749362
Dentro de los grupos	13,8386793	95	0,14567031		
Total	14,8041374	99			

5.5.3 Número de generaciones

Como ya se ha comentado en apartados anteriores, por los resultados obtenidos de las simulaciones con 1000 generaciones parecía que el algoritmo no estaba convergiendo. Es importante saber a partir de qué cantidad de generaciones converge el algoritmo, para conocer cuándo pueden tomarse los resultados obtenidos como óptimos. De todos modos, cabe destacar que la convergencia del algoritmo por si no nos da el mejor resultado, ya que siempre queda la probabilidad de que se caiga en un máximo local y el algoritmo converja en cuanto a ese valor.

En la Tabla 11 pueden verse los resultados para 1000, 2000 y 3000 generaciones. Como puede apreciarse, a medida que se aumenta el número de generaciones la calidad de los resultados también mejora, obteniendo cada vez mejores valores de fitness.

Otro dato que se puede extraer de los resultados es que aunque la media de los fitness obtenidos mejore, en todos los conjuntos de resultados pueden verse valores de fitness basante "malos" en comparación con el resto, como por ejemplo en la simulación 18 con 2000 generaciones, donde el fitness es de solo 14.966, significantemente peor que para la mayoría de los valores obtenidos con 1000 o 15000 generaciones. Esto puede ser debido a que el algoritmo se haya quedado en algún máximo local y por lo tanto no haya podido llegar a un resultado mejor.

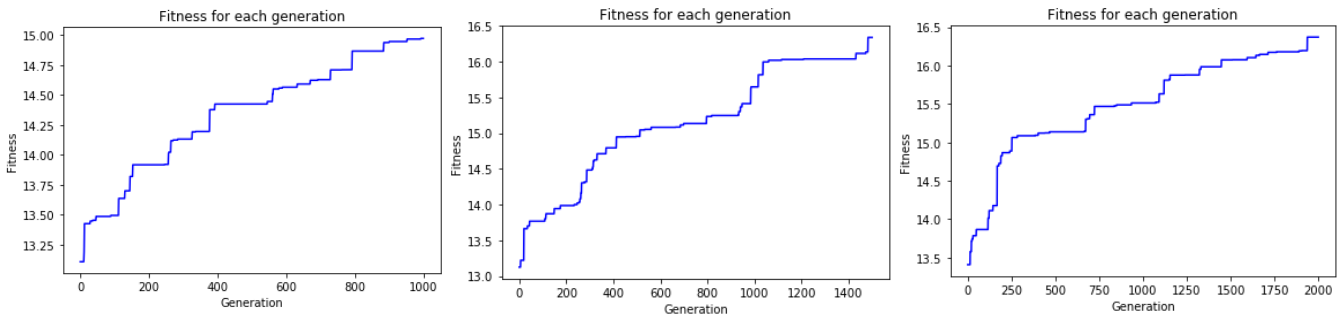
Tabla 11 - Resultados para 1000, 2000 y 3000 generaciones

NGEN = 1000	
Death Penalty	
Simulación	Fitness
0	15,385
1	14,913
2	15,022
3	15,326
4	15,476
5	15,943
6	15,388
7	15,443
8	15,217
9	15,499
10	15,52
11	14,721
12	14,561
13	16,357
14	16,198
15	15,182
16	15,524
17	16,612
18	14,717
19	14,971
Media	15,399
Std. Dev.	0,545
Mejor fit	16,612

NGEN = 1500	
Death Penalty	
Simulación	Fitness
0	15,722
1	15,208
2	15,759
3	16,204
4	15,216
5	15,084
6	15,741
7	15,886
8	14,969
9	15,179
10	16,149
11	15,372
12	15,828
13	15,536
14	15,565
15	16,171
16	16,194
17	16,184
18	16,409
19	16,34
Media	15,736
Std. Dev.	0,454
Mejor fit	16,409

NGEN = 2000	
Death Penalty	
Simulación	Fitness
0	15,19
1	16,519
2	15,442
3	16,372
4	16,072
5	15,402
6	15,872
7	15,205
8	16,048
9	16,827
10	16,184
11	16,653
12	16,586
13	16,143
14	15,399
15	15,23
16	15,497
17	16,286
18	14,966
19	16,37
Media	15,913
Std. Dev.	0,573
Mejor fit	16,827

Dibujando en una gráfica el fitness obtenido generación tras generación, para cada uno de los tres casos de la Tabla 11, se han obtenido los resultados mostrados en la Figura 18 (de izquierda a derecha los gráficos para 1000, 1500 y 2000 generaciones),. Como puede observarse el algoritmo no llega a converger en ninguno de los casos .



Por lo tanto es necesario aumentar el número de generaciones si se quiere llegar a obtener mejores resultados.

Figura 18 - Convergencia con 1000, 1500 y 2000 generaciones

En el siguiente conjunto de simulaciones se ha probado con 2500, 5000 y 10000 generaciones, y se han recogido los resultados en la Tabla 12 - Resultados para 2500, 5000 y 10000 generaciones:

Tabla 12 - Resultados para 2500, 5000 y 10000 generaciones

NGEN = 2500	
Death Penalty	
Simulación	Fitness
0	16,508
1	15,536
2	15,44
3	16,277
4	15,801
5	16,014
6	15,248
7	16,623
8	16,155
9	16,082
10	16,381
11	16,113
12	15,67
13	16,356
14	14,813
15	15,977
16	15,676
17	16,071
18	16,125
19	15,951
Media	15,941
Std. Dev.	0,444
Mejor fit	16,623

NGEN = 5000	
Death Penalty	
Simulación	Fitness
0	15,374
1	16,583
2	15,702
3	16,282
4	16,734
5	16,053
6	16,489
7	16,558
8	16,017
9	16,623
10	15,832
11	16,515
12	17,065
13	15,555
14	15,192
15	16,17
16	16,977
17	16,797
18	16,176
19	16,043
Media	16,237
Std. Dev.	0,523
Mejor fit	17,065

NGEN = 10000	
Death Penalty	
Simulación	Fitness
0	15,756
1	16,179
2	15,867
3	16,019
4	16,773
5	16,181
6	16,465
7	15,423
8	15,985
9	16,269
10	16,233
11	15,928
12	15,826
13	16,027
14	15,929
15	16,489
16	15,222
17	15,664
18	16,315
19	15,84
Media	16,019
Std. Dev.	0,365
Mejor fit	16,773

Con las nuevas simulaciones puede verse que la tendencia de mejora del fitness sigue junto con el aumento del número de generaciones hasta llegar a las 5000 generaciones, ya que los resultados obtenidos con 10000 no presentan mejoras. Esto se debe a que, aunque con más generaciones el algoritmo converge más, hay varios máximos locales y por lo tanto la solución obtenida no siempre es la referente al máximo global, de modo que dependiendo de la simulación puede obtenerse un fitness mejor o peor.

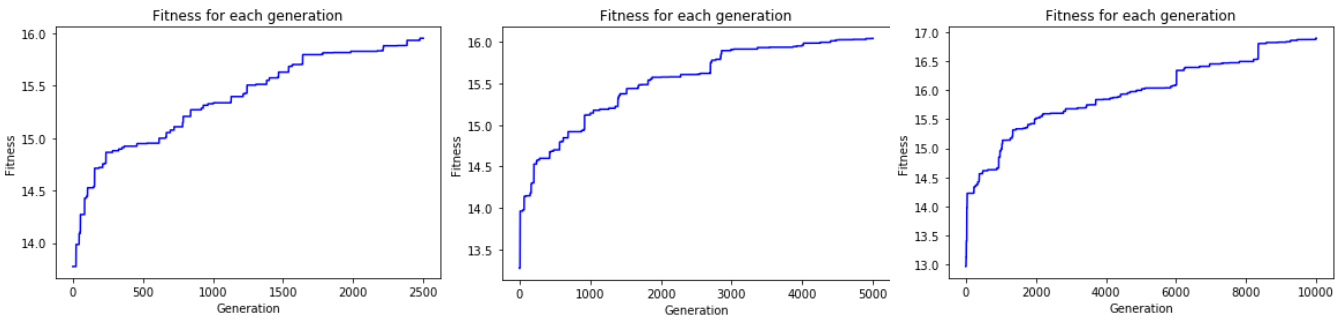


Figura 19 - Convergencia con 2500, 5000 y 10000 generaciones

Mirando en la Figura 19 las convergencias para las generaciones de la Tabla 12 (de izquierda a derecha los gráficos para 2500, 5000 y 10000 generaciones), puede verse que sigue sin conseguirse que el algoritmo converja.

Por lo que ha podido verse, es necesario aumentar el número de generaciones con tal de encontrar el punto en el que el algoritmo converja. Es así que se han realizado un nuevo conjunto de simulaciones utilizando 20000 generaciones. Los resultados para estas nuevas simulaciones pueden verse en la Tabla 13.

Tabla 13 - Resultados para 20000 generaciones

NGEN = 20000

Death Penalty	
Simulación	Fitness
0	16,368
1	17,002
2	16,099
3	15,967
4	16,14
5	16,49
6	16,556
7	15,924
8	15,026
9	16,425
10	15,983
11	16,284
12	15,945
13	16,593
14	15,526
15	15,867
16	15,667
17	15,037
18	16,226
19	16,495

Media	16,081
Std. Dev.	0,499
Mejor fit	17,002

Una vez más, puede verse que los resultados ya no se mejoran, ya que en este caso siguen siendo peores que los obtenidos con 5000 aunque algo mejores que los obtenidos con 10000. De todos modos, tal y como ya se ha comentado esto responde a que el algoritmo a veces se queda en máximos locales y por lo tanto no mejora a partir de un punto.

Lo que sí que puede verse ahora es que el algoritmo por fin converge, como puede apreciarsen en las diferentes gráficas recogidas en la Figura 20:

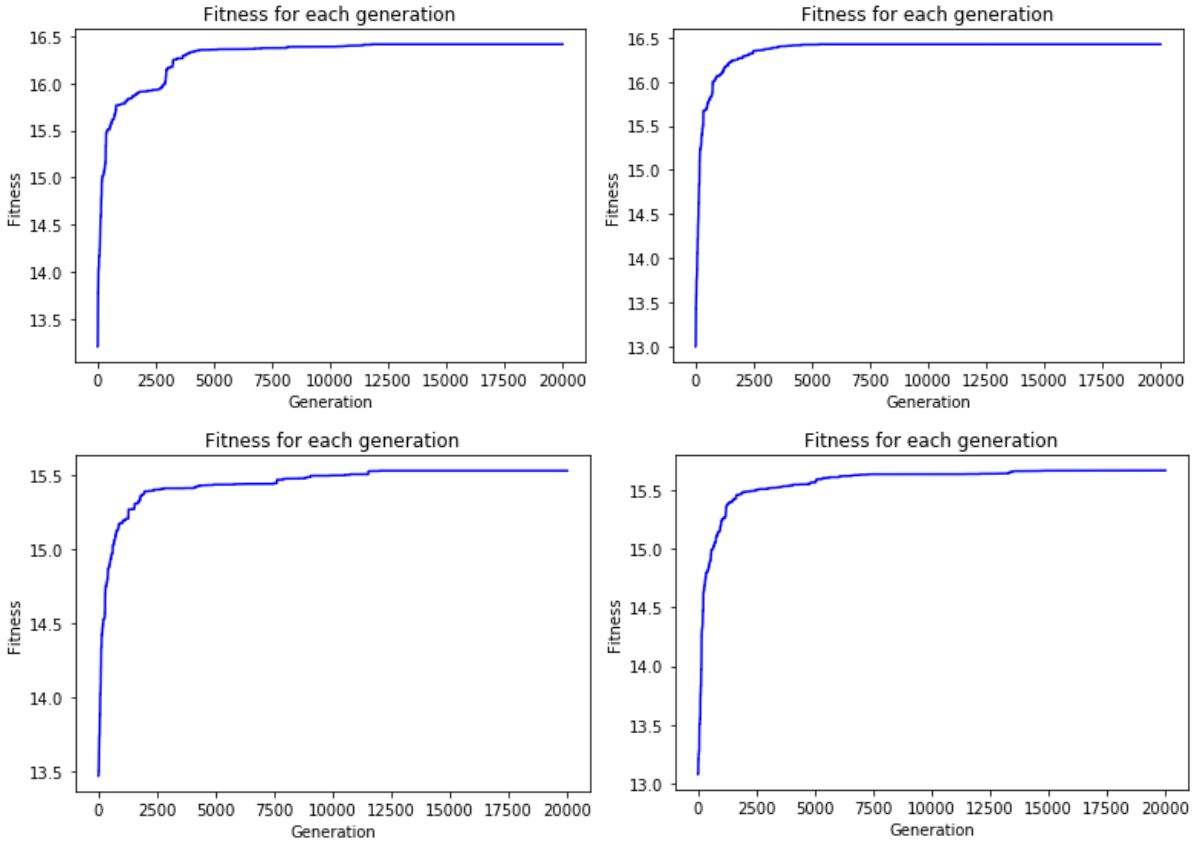


Figura 20 – Convergencia con 20000 generaciones

Como puede apreciarse, dependiendo de la simulación la generación a la que el algoritmo converge varía, pero en todos los casos llega a converger antes de alcanzar las 20000 generaciones. De hecho, con 15000 también se consigue en todos los casos, siendo en la mayoría de ellos suficiente llegar a 125000 generaciones.

6 CONCLUSIONES Y FUTUROS TRABAJOS

Se ha visto que la utilización de algoritmos genéticos para el diseño de algoritmos de path planning ofrece una interesante solución al problema propuesto para el Lago Ypacaraí. El trabajo previo realizado basándose en el problema del viajero o Traveling Salesman Problem (TSP) que tomaba las balizas repartidas a lo largo del perímetro del lago como "ciudades" a visitar, dio una primera alternativa al problema planteado, con buenos resultados. Para esta primera solución se tuvieron en cuenta varias restricciones como que cada baliza solo podía visitarse una vez (condición impuesta por el TSP) o que varias rutas no eran válidas (aquellas que necesitaban pasar por tierra). También se plantearon diferentes escenarios, con diferentes funciones de fitness para la evaluación del algotirmo, para de esta manera buscar el que mejores resultados obtenía.

Partiendo de esta primera aproximación, se ha planteado el problema basándose esta vez en el Chinese Postman Problem (CPP). De acuerdo a este problema, el objetivo es pasar por todas las rutas, no por todas las "ciudades", por lo que desaparece la restricción de visitar cada baliza una única vez aunque se mantengan las rutas inválidas. Para conseguir un escenario en el que poder aplicar el CPP se ha desarrollado una técnica para buscar circuitos Eulerianos con los que poder alimentar el algoritmo.

Como primer paso se ha comparado el rendimiento de ambas soluciones sin tener en cuenta la restricción de las rutas inválidas, con intención de comprobar a primera vista podía tratarse de una mejora de los resultados y proseguir con las pruebas. El segundo paso ha sido añadir las rutas inválidas, con lo que el resultado obtenido ha seguido siendo mejor que el que se tenía de base. Por último, se ha calculado una relación entre el número de balizas empleados y la distancia recorrida, con el objetivo de poder ajustar el funcionamiento del drone a las necesidades del momento.

El resultado obtenido mediante el uso de este método ha conseguido mejorar al anterior, pasando de un valor de fitness máximo de 16.612 a uno de 14.823 en el escenario con rutas inválidas y 60 balizas utilizando Death Penalty y 2000 generaciones; una mejora de un 12.1%. En el caso del Penalty Factor para el mismo escenario, la mejora es del 7.7%, con un fitness de 14.751 para el caso de TSP y un 15.892 para CPP. Sobre la comparativa de estos dos métodos se ha escrito un artículo que está a espera de revisión para ser publicado.

Todavía quedan muchas variaciones del problema con los que experimentar como trabajos futuros. Por un lado, se podría cambiar de técnica de optimización. Se ha visto que los algoritmos genéticos dan resultados competentes, pero también existen otros algoritmos como por ejemplo el Particle Swarn Optimization (PSO) donde un grupo de soluciones candidatas van cambiando para acercarse a máximos locales conocidos, o el Simulated Annealing (SA) donde se va disminuyendo la probabilidad de aceptar soluciones "malas" hasta llegar a una aproximación del máximo que se pueda obtener.

En relación al Algoritmo Genético (AG), también queda la opción de explorar la utilización de diferentes técnicas de crossover o de mutación. En este caso se ha empleado el Ordered Crossover (OC) junto con unos parámetros de mutación específicos, pero variando en estos procedimientos podría llegarse a soluciones diferentes, tal y como ya se ha visto con la variación de las probabilidades de ocurrencia de los dos eventos.

Otros aspectos específicos del problema del Lago Ypacaraí sobre los que profundizar pueden ser el procedimiento para la creación de individuos válidos con los que alimentar el AG, por ejemplo en lo que respecta a la eliminación de rutas inválidas en la matriz de conexiones, o las posiciones de las balizas del lago. Esto último por sí solo constituiría un problema de optimización diferente.

REFERENCIAS

[1] [En línea]. Available: http://rescatemosnuestrolago.gov.py/. [Último acceso: Julio 2017].

[2] [En línea]. Available: http://www.lagoypacarai.com/. [Último acceso: Julio 2017].

[3] [En línea]. Available: https://www.asvglobal.com/. [Último acceso: Julio 2017].

[4] [En línea]. Available: http://www.wam-v.com/. [Último acceso: Julio 2017].

[5] [En línea]. Available: http://www.autonautusv.com/. [Último acceso: 2017 Julio].

[6] G. Ferri, A. Manzi, F. Fornai, F. Ciuchi y C. Laschi, «The HydroNet ASV, a Small-Sized Autonomous Catamaran for Real-Time Monitoring of Water Quality: From Design to Missions at Sea,» *IEEE Journal of Oceanic Engineering,* vol. 40, July 2015.

[7] S. C. Y. Leong, P. Tkalich y N. M. Patrikalakis , «Monitoring harmful algal blooms in Singapore: Developing a HABs observing system,» *OCEANS,* 2012.

[8] G. Hitz, F. Pomerleau, M. Garneau, C. Pradelier, T. Posch, J. Pernthaler y R. Y. Siegwart, «Autonomous Inland Water Monitoring: Design and Application of a Surface Vessel,» *IEEE Robotics & Automation Magazine,* vol. 19, 2012.

[9] J. Gutierrez, F. Vila-Medina y M. A. Porta-Gandara, «Autonomous Surface Vehicle for Measuring Water Body Parameters,» de *IEEE Conference on Electronics, Robotics and Automotive Mechanics Conference (CERMA)*, 2010.

[10] P. Tokekar, E. Branson, J. V. Hook y V. Isler, «Tracking Aquatic Invaders: Autonomous Robots for Monitoring Invasive Fish,» *IEEE Robotics & Automation Magazine,* vol. 20, 2013.

[11] F. Halal, P. Pedrocca, T. Hirose, A. M. Cretu y M. B. Zaremba, «Remote-sensing based adaptive path planning for an aquatic platform to monitor water quality,» *Robotic and Sensors Environments (ROSE),* 2014.

[12] W. Yuan-hiu y C. Cen, «Research on optimal planning method of USV for complex obstacles,» de *IEEE International Conference on Mechatronics and Automation (ICMA)*, 2016.

[13] Y. Liu, R. Song y R. Bucknall, «A practical path planning and navigation algorithm for an unmanned surface vehicle using the fast marching algorithm,» *OCEANS,* 2015.

[14] F. Fugie y P. Zhang, Covering Walks in Graphs, Springer, 2010.

[15] H. Jiang, L. Kang, S. Zhang y F. Zhu, Genetic Algorithm for Mixed Chinese Postman Problem, 2010.

[16] L. D. Chambers, de *Practical Handbook of Genetic Algorithms: Complex Coding Systems*, CRC Press, 1998, pp. 29-33.

[17] M. Arzamendia, D. G. Reina, S. L. Toral y D. Gregor, «An Event Detection and Tracking Adaptative Strategy for an Autonomous Surface Vehicle in Lake Environments using Evolutionary Computation».

[18] M. Arzamendia, D. G. Reina, S. L. Toral y D. Gregor, «An Evolutionary Approach to Constrained Path Planning of an Autonomous Surface Vehicle for Maximizing the Covered Area of Ypacaraí Lake,» de *International Conference on Developments in eSystems Engineering (DeSE)*, Liverpool, 2016.

[19] D. Pearson y V. Bryant, de *Decision Maths 1: Advancing Maths for AQA*, Heinemann, 2004, pp. 44-64.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CONCEPTOS