eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

Trabajo Fin de Grado
Grado en Física

Diseño e implementación de algoritmos genéticos y su aplicación a problemas de física

Autora:
Miren Hayet Otero
Director:
Javier Echanove Arias

Leioa, 5 de Septiembre de 2020

Índice

1. OBJETIVOS Y DESARROLLO

En diferentes ámbitos de las ciencias y de las ingenierías surgen a menudo problemas que se caracterizan por tener gran complejidad, ausencia de un buen modelo matemático, incertidumbre en sus parámetros, etc. Resultado de todo ello es que resulta muy difícil o incluso imposible abordar su resolución mediante las técnicas convencionales. Existen sin embargo, métodos alternativos que en muchos casos son capaces de obtener una solución que, sin ser óptima, es lo suficientemente aproximada para ser del todo válida. Entre estos métodos se encuentran los llamados "bio-inspirados", dentro de los cuales los algoritmos genéticos han proporcionado excelentes resultados en muchos y diferentes problemas.

Estos algoritmos implementan los llamados operadores genéticos, los cuales imitan los diferentes procesos necesarios para que las especies naturales mejoren. Partiendo de una primera generación formada por individuos creados aleatoriamente, seleccionan los que mejor se adaptan al problema planteado y los cruzan y mutan para crear nuevos individuos. Esta nueva generación se vuelve a someter al proceso anterior, y poco a poco, como si de la selección natural se tratase, aparecen mejores individuos cada vez más próximos a la solución del problema.

Los algoritmos genéticos se han utilizado por ejemplo para optimizar el diseño de sistemas de distribución de aguas, la estrategia de recogida de basura de un territorio o la producción y distribución de la energía eléctrica, así como en campos más sofisticados como el aprendizaje automático o la bioinformática.

Como se va a poder comprobar en este trabajo también son de gran utilidad en el área de la física. El principal objetivo de este trabajo es entender y analizar el funcionamiento de los algoritmos genéticos. Para ello se han diseñado varios algoritmos, con diferentes estrategias y valores de parámetros y se han aplicado a diferentes problemas físicos para poder conocer su alcance y limitaciones.

En el capítulo 2 se explica la relación de la computación evolutiva con los sistemas naturales. Se ha profundizado en la inspiración biológica a partir de la cual se ha desarrollado y en el funcionamiento de un algoritmo evolutivo.

El capítulo 3 se centra en el planteamiento y funcionamiento teórico de los algoritmos genéticos, exponiendo las diferentes maneras que hay de representar las posibles soluciones a los problemas a resolver, así como algunas de las diferentes estrategias que existen para diseñar un algoritmo genético y las condiciones que se suelen establecer para detectar que el algoritmo ha alcanzado una solución suficientemente buena.

En el capítulo 4 se profundiza todavía más en el diseño de los algoritmos genéticos, describiendo detalladamente los operadores genéticos necesarios para crear un algoritmo genético como son la selección, el cruce o recombinación y la mutación.

En el capítulo 5 se muestran los cuatro problemas que se han resuelto en este trabajo, a través de los cuales se han probado diferentes operadores genéticos y probabilidades de aplicarlos con tal de entender cual puede ser más adecuado.

Por último, en el capítulo 6 se exponen las conclusiones obtenidas durante el desarrollo de este trabajo.

2. INTRODUCCIÓN A LA COMPUTACIÓN EVOLUTIVA

La evolución es un fenómeno predominante en la mayoría de los sistemas naturales. Las especies más complejas son resultado de una evolución continua durante mucho tiempo. Comenzaron siendo simples y durante años evolucionaron a especies nuevas, cada cual más compleja que la anterior, y con una capacidad de adaptación mayor a un entorno cambiante. Inspirados por los sistemas naturales, los sistemas artificiales están intentando también basarse en los principios evolutivos, planteando los problemas de forma que la solución evolucione conforme al tiempo. Esto hace posible una generación de soluciones de todo tipo que no dejan de mejorar conforme pasa el tiempo, consiguiendo así optimizar las soluciones de cualquier problema [1].

Si la cantidad de soluciones posibles de un problema no es muy elevada, tendría sentido evaluar todas ellas y comparar su validez, pero en muchos casos la cantidad de soluciones es tan grande que no sería posible crear todas ellas en una cantidad finita de tiempo. Por ello ha sido necesario buscar alternativas a la hora de resolverlos, y una de ellas ha sido la computación evolutiva. Los algoritmos evolutivos ofrecen una técnica universal de optimización aplicable a una amplia variedad de problemas, como optimización de parámetros, búsqueda, problemas de combinatoria o generación automática de programas de ordenador [2].

Los algoritmos evolutivos constituyen un grupo de métodos de búsqueda y optimización basados en los principios de la evolución natural. Existen diferentes técnicas dentro de la computación evolutiva como los algoritmos genéticos, las estrategias de evolución, la programación evolutiva o la programación genética. El funcionamiento de todos ellos esta basado en los mismos conceptos: una población de soluciones candidatas, combinación y alteración aleatoria de todas ellas para crear nuevas, y un mecanismo de selección para aumentar la proporción de mejores soluciones [2]. El problema que se quiere solucionar se representa mediante una función objetivo que sirve para evaluar lo adecuada que es una solución, por lo que será la función a optimizar, y el objetivo es hallar la solución que maximice o minimice su valor [1].

2.1. Inspiración biológica

Como se ha mencionado anteriormente, la computación evolutiva esta basada en la evolución natural, por lo que los algoritmos adoptan términos biológicos a la hora de describir sus elementos estructurales y diferentes operaciones. Sin embargo la utilización de términos biológicos es una simplificación, ya que los seres biológicos son mucho más complejos [3].

Concretamente, se inspira en los sistemas de reproducción y supervivencia de las especies naturales. Las especies consisten en diferentes poblaciones, las cuales no son más que grupos de individuos de una misma especie en un mismo instantes. Los diferentes individuos de una misma población interactúan sexualmente para generar nuevos individuos, que constituirán en conjunto una población mejor que la anterior. De esta manera las nuevas generaciones se adaptan mejor a un entorno cambiante, como indica la teoría de Charles Darwin sobre la supervivencia del más apto. Ésta manifiesta que solo los rasgos

de los individuos más aptos pasan de una generación a la siguiente.

En el caso de las especies, sus organismos están compuestos por células, que a su vez están formadas, entre otros elementos, por cromosomas (figura 1), las cuales son secuencias de ADN y, por lo tanto influyen en la identidad del individuo: el ADN juega un papel importante en la definición del comportamiento, las características y las propiedades de cada individuo. En los cromosomas se encuentran distribuidos los genes o secuencias de ADN, que se ocupan de codificar los productos génicos que influyen en las características de los individuos. A las diferentes formas que puede tomar un gen se les llama alelo. De esta manera, combinando los genes de diferentes formas se generan individuos con características diferentes. Muchos organismos tienen varios cromosomas en cada célula, y esta colección de material genético se denomina genoma.

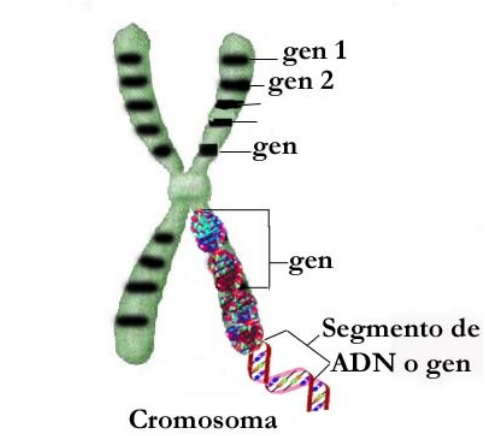

Figura 1: Parte de la composición de un cromosoma.

Para hacer referencia a las características de un individuo hay que diferenciar los términos genotipo y fenotipo. El primero se refiere al particular grupo de genes o información genética que contiene el genoma, y define en gran parte todas las características del individuo. Dos individuos con el mismo genotipo se consideran como idénticos. Por su parte, el fenotipo consiste en el grupo de características de un individuo que son resultado de la interacción del genotipo y el ambiente en el que se encuentra. Aunque dos genotipos iguales resulten en un mismo organismo, las condiciones del medio pueden resultar en un fenotipo distinto, y por lo tanto, aún teniendo la misma información genética, poder distinguir los dos organismos genéticamente.

La interacción sexual citada anteriormente no es más que una recombinación de cromosomas de dos individuos (padres), llamada *crossover*, recombinación o cruce (figura 2), en la que se intercambian los genes para formar uno nuevo (hijo). A veces hay un cambio en la organización del ADN de un padre al hijo, y este fenómeno que hace variar un gen se denomina mutación (figura 3).

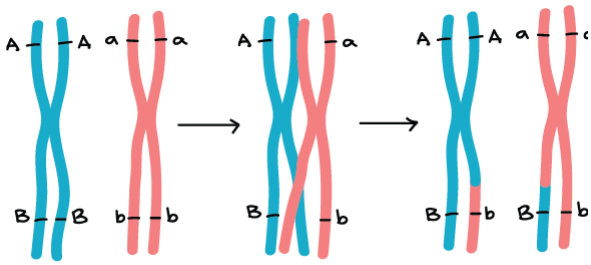

Figura 2: Recombinación de dos cromosomas.

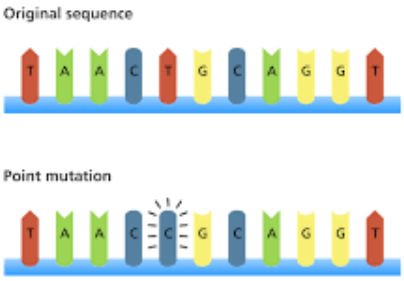

Figura 3: Mutación de un gen.

Por último, otro concepto en los sistemas naturales muy útil para evaluar la capacidad de un individuo a la hora de sobrevivir es el *fitness*. Un *fitness* alto significa una alta probabilidad de sobrevivir, o entendido de otra manera, mayor poder reproductivo o fertilidad [4].

Por su parte, los algoritmos evolutivos aprovechan los términos cromosoma y genotipo para describir un grupo de parámetros que codifican una solución candidata al problema a optimizar, mientras que utilizan el término gen para referirse a una entidad funcional de la solución, como por ejemplo un parámetro especifico en un problema multidimensional. El *fitness* de cada solución se consigue evaluándola mediante la función objetivo o *fitness F*, que juega el papel del entorno. Así, las mejores tendrán más probabilidad de reproducirse recombinándose y sometiéndose a posibles mutaciones [3].

2.2. Estructura y funcionamiento general de un algoritmo evolutivo

En la figura 4 se puede ver el esquema de un algoritmo evolutivo. Al instante de tiempo t=0 se inicializa una población P totalmente aleatoria. Mediante la función objetivo se evalúa cada individuo y se le adjudica una probabilidad de ser elegidos para reproducirse, que será mayor cuanto mejor sea esa solución. Los elegidos como padres se cruzan de dos en dos para crear nuevos individuos, y estos últimos podrán sufrir alguna mutación en alguno de sus genes.

Tal y como ocurre en la naturaleza, como resultado del ciclo evolutivo de selección, recombinación y mutación, dentro de la población de cada generación van apareciendo individuos mejores. Este proceso se repite una y otra vez hasta que se alcanza la condición o alguna de las condiciones de terminación impuestas [5].

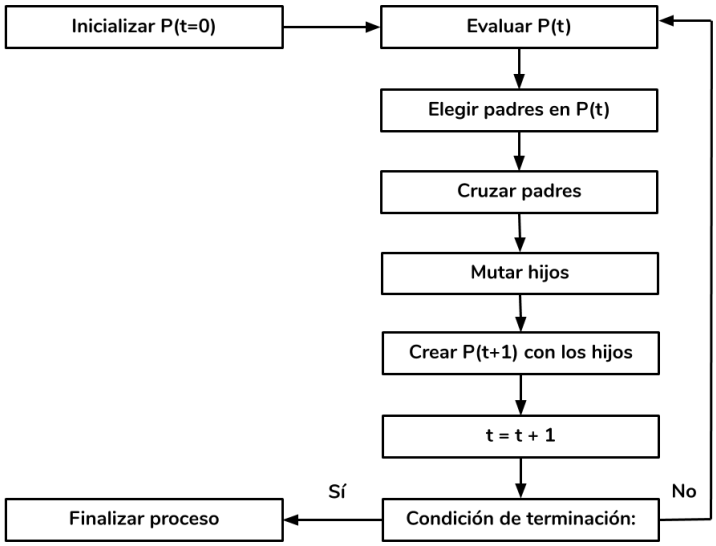

Figura 4: Estructura de un algoritmo evolutivo.

Cada variante o técnica de computación evolutiva mencionada anteriormente emplea sus particulares maneras de representar cromosomas, seleccionar individuos, cruzarlos o mutarlos. Por lo tanto existen infinidad de algoritmos para asignar una probabilidad partiendo de la función *fitness* y después hacer la selección entre individuos, de realizar el cruce entre dos individuos o mutar alguno de ellos, y también maneras de representar un cromosoma o solución.

3. ALGORITMOS GENÉTICOS

En este trabajo se van analizar el funcionamiento, ventajas y desventajas de los algoritmos genéticos (AG), ya que es la técnica mas utilizada de todas las que conforman los algoritmos evolutivos. Está comprobado que los AGs realizan una búsqueda robusta en espacios complejos, ofreciendo una aproximación valida a problemas que requieren efectividad y eficiencia [5].

Como ya se ha explicado anteriormente, los AGs operan sobre una población creada inicialmente de manera aleatoria, y la población avanza hacia mejores individuos al aplicarle los operadores genéticos de cruce y mutación. El primero es el operador principal mediante el cual se consigue crear individuos híbridos de individuos de generaciones anteriores, de manera que se explora el espacio de búsqueda existente por distintas vías. El segundo mantiene la diversidad del grupo de genes, explorando nuevas vías dentro del espacio de búsqueda total.

Para resolver un problema mediante AGs es necesario, como mínimo, fijar las siguientes cuestiones:

- Fijar de que manera se van a representar los individuos.
- Establecer cómo se va a crear la población inicial.
- Definir la función objetivo que evaluará cada individuo.
- Diseñar los diferentes operadores genéticos.
- Establecer los parámetros del AG: tamaño de la población, condiciones de finalización y probabilidades de aplicar los operadores genéticos.

3.1. Representación de los individuos e inicialización de la población

A la hora de representar los individuos, se suelen utilizar principalmente dos codificaciones: codificación binaria y codificación real.

Codificación binaria

La codificación binaria es la codificación principal y original dentro de los algoritmos genéticos. Los parámetros que conforman la solución se representan utilizando el sistema binario, por lo que el cromosoma es una cadena de n bits con forma $(s_{n-1}, ..., s_0)$, donde cada bit s_i, que puede tomar los valores 0 o 1, corresponde a un gen. Para inicializar la población, primero se decide mediante cuantos bits se va a representar cada solución, y se crea esa cantidad de bits aleatoriamente las veces que haga falta hasta llegar al tamaño de la población. Para conocer el valor decimal de ese cromosoma y poder evaluarlo después, solo hace falta aplicar la expresión 1, donde y_j es el individuo número j de la población inicial e y_{min} e y_{max} son los valores entre los que puede estar la solución.

$$y_j = y_{min} + \frac{y_{max} - y_{min}}{2^n - 1} \sum_{i=0}^{n-1} s_i \cdot 2^i \tag{1}$$

En el caso de que la solución esté compuesta por más de un parámetro, habrá que aplicar la expresión 1 a los grupos de bits correspondientes.

Codificación real

En esta variante de codificación cada parámetro que compone la solución se representa directamente mediante el sistema decimal, es decir mediante su valor real. De esta manera el cromosoma será un vector de números reales $(x_1, ..., x_n)$ donde n será la dimensión del espacio de búsqueda de soluciones, o dicho de otro modo, la cantidad de valores que conforman un individuo o solución. Cada uno de estos parámetros corresponde a un gen.

3.2. Escalado de la función *fitness*

La función objetivo asigna a cada individuo de una población un valor que representa su calidad, o, dicho de otra manera, la probabilidad de reproducirse. Sin embargo, debido a la naturaleza de la función objetivo o a la de algunos individuos, a veces, no está garantizado un valor positivo, o el valor de la función *fitness* de algún individuo es considerablemente mayor al de los demás. Para superar estas limitaciones existen varios métodos de escalado de la función *fitness*, los cuales convierten el valor de la función *fitness* de un individuo $f(C_i)$ en $\hat{f}(C_i)$, con el objetivo de hacer del proceso de selección un proceso más efectivo. Éstos son algunos de los métodos:

Escalado lineal

La transformación de la función *fitness* se realiza mediante la expresión 2 [6].

$$\hat{f}(C_i) = c_o \cdot f(C_i) + c_1 \tag{2}$$

Los parámetros c_o y c_1 se ajustan estática o dinámicamente, dependiendo de la distribución de la función *fitness* en la población. Por ejemplo, se pueden calcular estableciendo que el mejor individuo se tiene que reproducir α veces, y un individuo con el valor $\bar{f}$ una vez:

$$\hat{f}_{max} = \alpha\hat{\bar{f}} \quad \longrightarrow \quad c_0 \cdot f_{max} + c_1 = \alpha \cdot (c_0 \cdot \bar{f} + c_1)$$

Así se determina el ratio:

$$\frac{c_0}{c_1} = \frac{\alpha - 1}{f_{max} - \bar{f}}$$

Por lo tanto, eligiendo

$$c_0 = \alpha - 1$$

$$c_1 = f_{max} - \bar{f}$$

se garantiza que hasta el valor de la función *fitness* mínimo sea positivo.

Escalado de rango

Los individuos se ordenan de 1 a N de acuerdo a su valor F, siendo N el mejor individuo, y mediante la expresión 3 se vuelven a escalar dependiendo de su rango $(R(C_i))$ [7].

$$\hat{f}(C_i) = f_{min} + (f_{Max} - f_{Min})\frac{R(C_i) - 1}{N - 1} \tag{3}$$

Escalado Top

Este tipo de escalado selecciona los x mejores individuos y les adjudica el mismo valor 1/N, mientras que a todos los demás les asigna el valor 0 [8]. Es decir, solo un específico número de individuos tendrá opción a reproducirse.

3.3. Condiciones de finalización y convergencia

El objetivo principal de los algoritmos genéticos es resolver problemas que, utilizando métodos convencionales serían prácticamente imposibles de resolver en una cantidad de tiempo finito. Es necesario establecer una o varias condiciones de finalización para detener la ejecución de éste. Esta detención tiene lugar cuando se cumple cualquiera de las siguientes condiciones:

- Que se alcance una solución con un valor suficientemente bueno de la función *fitness*. Este valor dependerá del conocimiento que se tenga del problema.
- Que se alcance un número de generaciones o tiempo establecidos anteriormente.
- Convergencia de las soluciones: que la mejoría entre generaciones sucesivas sea muy pequeña.

Se dice que el algoritmo converge cuando tras varias generaciones se mantiene casi constante el valor de la función *fitness* y no aparecen nuevos individuos mejores. En algunos problemas se alcanza la convergencia con muchas menos generaciones que en otros. Sin embargo, que haya convergencia en torno a una solución no significa que sea la óptima, ya que en problemas en los que existan extremos locales puede que haya convergido en uno de estos y no en un extremo global. Por lo tanto es importante seguir insistiendo en explorar nuevos espacios de soluciones, y no tratar de conseguir una convergencia demasiado rápida, ya que esto último puede derivar en estancarse en una solución que aun siendo mejor que muchas otras, no es la mejor.

A no ser que se conozca cual es la solución del problema, es prácticamente imposible saber si la solución conseguida es la mejor, por lo tanto en muchos casos es imposible decidir si la solución en la que ha convergido el algoritmo se trata de un extremo local o global, por lo tanto conviene explorar muchas regiones del espacio.

4. OPERADORES GENÉTICOS

El proceso de pasar de una generación de individuos a otra se realiza mediante los operadores genéticos. En esta sección se van a describir los operadores genéticos principales y algunas de los diferentes estrategias existentes para llevarlos a cabo.

4.1. Selección

La selección es el primer operador genético que se aplica a la población, y su función es seleccionar los individuos que van a reproducirse para generar una nueva población. Varias estrategias han sido desarrolladas con el objetivo de seleccionar los individuos cuyo cruce resulte en individuos mejores. Todas las estrategias se basan en establecer una probabilidad de selección que esté relacionada con el valor de la función *fitness* de los individuos, por lo que los mejores individuos tienen más opciones de ser elegidos.

Selección *Roulette Wheel*

Es la estrategia principal de selección, y la que se utilizará a lo largo de este trabajo. A cada individuo se le asigna una probabilidad de selección proporcional a $f(C_i)$, expresada mediante la expresión 4:

$$P(C_i) = \frac{f(C_i)}{\sum_{j=1}^{N} f(C_j)} \tag{4}$$

La población se representa en una ruleta como la que se puede observar en la figura 5, en la que el espacio correspondiente a cada individuo es proporcional a su probabilidad de selección.

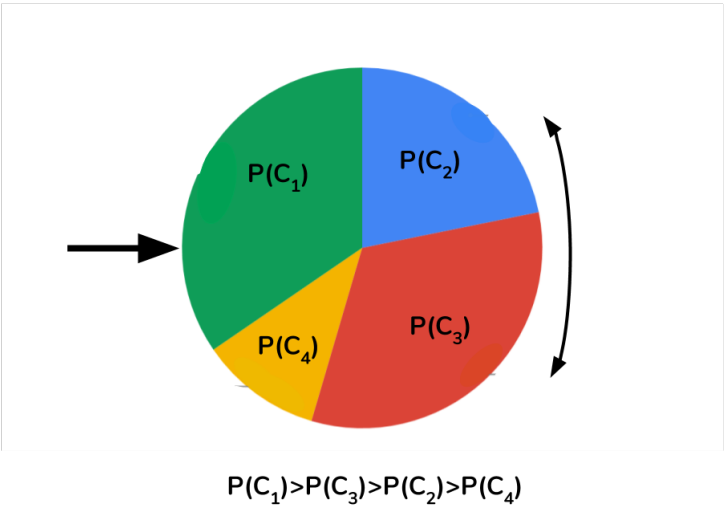

Figura 5: Esquema de la ruleta.

La ruleta se gira N veces, y en cada tirada se escoge el individuo que pasará a formar parte de una "generación intermedia", de la cual saldrá la siguiente generación.

Existen varias estrategias más de selección, enfocadas a evitar que un individuo con un valor de la función *fitness* considerablemente más alto al de los demás sea elegido

una y otra vez, ya que así unas pocas soluciones se repiten una y otra vez, eliminando la diversidad de la población. Por ejemplo el *Muestreo Estocástico Universal* consiste en una implementación de la ruleta en la que en vez de haber un seleccionador, hay N y la ruleta se gira una sola vez, seleccionando los N individuos en una tirada. Por otro lado también está la *Selección mediante torneo*, en la cual se repite N veces el proceso de elegir aleatoriamente dos individuos y someterlos a un torneo en el que el mejor tendrá más probabilidad de ganar [9].

4.2. Recombinación o cruce

Una vez que se han seleccionado los individuos que se van a reproducir, se cruzan entre ellos para crear nuevos individuos o hijos que contienen las características de sus padres. No todos los individuos seleccionados se reproducen, si no que una parte de ellos pasa a la siguiente generación tal cual son. Cuáles se reproducen y cuáles se mantienen idénticos se decide mediante la probabilidad de cruce, que normalmente adquiere valores típicos $P_c \in [0.6, 0.95]$ [10]. Los individuos que se van a reproducir se cruzan de dos en dos. Como en todos los operadores genéticos, existen diferentes estrategias de recombinación de individuos.

Cruce simple o de un punto

Es el método más simple de cruce, y es aplicable en el caso de los algoritmos genéticos codificados tanto de manera binaria como de manera real, siempre y cuando el individuo esté compuesto por más de un parámetro o valor. Tal y como se muestra en la figura 6, se elige aleatoriamente un punto entre 0 y N, siendo N la longitud o cantidad de parámetros del cromosoma, y se divide en dos desde ese punto. Los genes de la parte izquierda a ese punto se cogen del primer padre, y los de la derecha del segundo, formando un nuevo individuo con esos genes. En este trabajo se ha decidido crear dos hijos a partir de dos padres, uno con la configuración recientemente explicada, y otro con la configuración contraria.

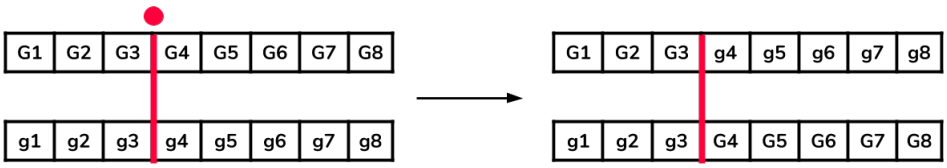

Figura 6: Ejemplo de cruce simple entre dos cromosomas de 8 genes, a partir del tercer gen.

Este procedimiento se puede generalizar eligiendo más de un punto de cruce, hasta llegar al llamado *Cruce Uniforme* (figura 7) en el que se van alternando todos los genes de los dos padres, o el *Cruce Disperso*, en el que cada gen se elige aleatoriamente entre los dos padres.

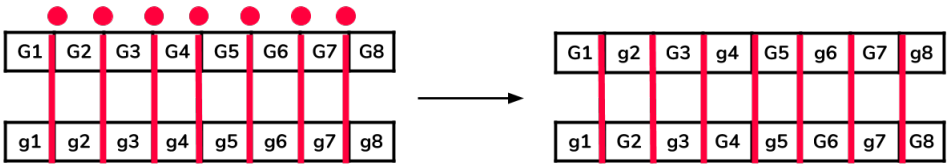

Figura 7: Ejemplo de cruce uniforme entre dos cromosomas de 8 genes.

Flat **crossover**

Esta estrategia se utiliza para cruzar cromosomas codificados mediante números reales. Siendo

$$C_1 = (x_1^1, x_2^1, ..., x_n^1)$$
$$C_2 = (x_1^2, x_2^2, ..., x_n^2)$$

los dos cromosomas que se van a cruzar, cada gen que compone el cromosoma $C' = (x'_1, x'_2, ..., x'_n)$ de su hijo se calcula utilizando la expresión 5.

$$x'_i = \lambda_i \cdot x_i^1 + (1 - \lambda_i) \cdot x_i^2 \tag{5}$$

donde λ_i es un numero aleatorio entre [0,1]. Repitiendo este proceso dos veces se pueden conseguir dos hijos de dos padres, como en el anterior caso.

4.3. Mutación

En los algoritmos genéticos se entiende como mutación el cambio que ocurre de manera aleatoria en un gen. Cuando uno de estos cambios ocurre, se añaden nuevas características a la población, ayudando a descubrir nuevas áreas en el espacio de búsqueda del algoritmo genético. Si esas características son nuevas, sobrevivirán durante generaciones siguientes, en cambio, si no lo son, desaparecerán. La mutación se lleva a cabo en unos pocos genes de unos pocos individuos, ya que de ocurrir en la mayoría de ellos el algoritmo se comportaría de una manera totalmente aleatoria. Por lo tanto, la probabilidad de mutación de un gen suele adquirir un valor $P_m \in [0.001,\ 0.02]$ [11]. Existen varias estrategias a la hora de aplicar el operador de mutación a los individuos.

Mutación aleatoria o uniforme

En los algoritmos genéticos la estrategia de mutación más común es la conocida como aleatoria o uniforme. En el caso de los algoritmos genéticos representados mediante números binarios, si el valor de un gen es 0 se convierte en 1, y viceversa, como se puede ver en la figura 8.

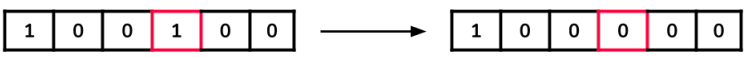

Figura 8: Mutación aleatoria del cuarto gen para un individuo codificado mediante números binarios.

Si el gen es un número real, se sustituye por un número aleatorio seleccionado en el rango de valores posibles establecido para el problema.

En el caso de los algoritmos codificados mediante números reales, existen otras estrategias a la hora de mutar un gen, como la mutación no uniforme o la mutación gaussiana, en las que el gen no tiene las mismas probabilidades de mutar a cualquiera de los valores dentro del rango de posibles soluciones, es decir, la probabilidad de mutar en un valor u otro no es uniforme.

Otra estrategia existente a la hora de aplicar el operador de mutación es no mantener constante la probabilidad de mutación. Una probabilidad alta de mutación puede resultar en un comportamiento aleatorio del algoritmo, y esto puede ser positivo al principio para explorar nuevas posibilidades, pero conforme va avanzando puede resultar en no conseguir nunca que el algoritmo converja, por eso a veces puede resultar interesante tratar de adaptar esa probabilidad al desarrollo del algoritmo. Por otro lado, tampoco es interesante que esa probabilidad sea demasiado pequeña, ya que hay que explorar nuevas soluciones con tal de evitar que la solución se estanque en un extremo local.

4.4. Otros operadores genéticos

Además de los operadores genéticos principales, existen otros adicionales desarrollados para objetivos más específicos y que pueden resultar muy útiles a la hora de solucionar ciertos problemas.

Uno de ellos es el elitismo, que consiste en trasladar directamente de una generación a otra una cantidad X de individuos con mayor valor de la función *fitness*. De esta manera los mejores individuos no tienen peligro de desaparecer o ser transformados mediante los operadores genéticos principales en la transición de una generación a otra. Si no se aplica el elitismo es muy posible que el mejor individuo de una generación tenga un valor de la función *fitness* menor que el del anterior.

5. APLICACIONES

Las aplicaciones de los algoritmos están dirigidas a problemas de optimización y de búsqueda. Concretamente, se suelen aplicar a los problemas que no tienen una solución analítica sencilla y fácil de calcular, es decir, a aquellos de mucha complejidad computacional que, de intentar solucionarlos utilizando métodos de fuerza bruta, el coste de tiempo sería inasumible.

En este capítulo se explica como se han resuelto diferentes problemas físicos mediante algoritmos genéticos. Estos algoritmos genéticos han sido diseñados e implementados en la plataforma Matlab [12]. Aunque esta plataforma contiene una herramienta para resolver problemas mediante algoritmos genéticos, no se ha hecho uso de ella ni de librerías previamente diseñadas, sino que se han programado desde inicio, siendo también parte del trabajo realizado. Se han desarrollado algoritmos y estrategias para los dos tipos de codificaciones, y se han hecho varias pruebas con ellos para comparar su funcionamiento. Finalmente se ha optado por utilizar los codificados mediante números reales y las estrategias descritas en el capítulo 4. Los algoritmos genéticos y estrategias utilizados para resolver los problemas planteados y los códigos necesarios para algunos cálculos se encuentran en el siguiente enlace: `https://github.com/mhayeto/GrAL-AG`.

5.1. Problema de la curva Braquistócrona

El objetivo del problema de la curva Braquistócrona es encontrar una curva planar óptima entre un punto inicial (x_0, y_0) y un punto final (x_k, y_k) que minimice el tiempo que necesita una masa puntual para recorrerla bajo la acción de una fuerza de gravedad constante, suponiendo que no existe fricción y con una velocidad inicial nula, siendo $y_0 > y_k$. La solución a este problema es conocida y consiste en una cicloide invertida. Aun así, es un problema muy interesante a resolver mediante algoritmos genéticos, sobre todo para entender y analizar su comportamiento, por lo que se ha aprovechado este problema para analizar el efecto de distintas estrategias, así como del valor de las probabilidades de mutación y cruce.

Para ello se aproxima la curva mediante un conjunto de $k+1$ puntos (x_i, y_i). La distancia Δx entre estos puntos se asume constante, quedando pendiente de optimizar la posición y_i de todos los puntos entre $(x_1, y_1), ..., (x_{k-1}, y_{k-1})$. La función a optimizar en este caso es el tiempo T, que se puede calcular como se indica en la ecuación 6

$$T = \sum_{i=0}^{k-1} t_i \tag{6}$$

que tarda la masa puntual en recorrer la curva, la cual es la suma de todos los tiempos t_i, que corresponden al tiempo que necesita la masa para ir del punto (x_i, y_i) al punto (x_{i+1}, y_{i+1}). La velocidad inicial v_i en cada punto se puede calcular teniendo en cuenta la conservación de la energía y que la velocidad inicial v_0 es nula:

$$\frac{m \cdot v_i^2}{2} = m \cdot g \cdot (y_0 - y_i) \rightarrow v_i = \sqrt{2 \cdot g \cdot (y_0 - y_i)} \tag{7}$$

La aceleración viene dada por

$$a_i = \frac{g \cdot dy_i}{\sqrt{dx_i^2 + dy_i^2}} \tag{8}$$

donde $dx_i = x_{i+1} - x_i$ y $dy_i = y_{i+1} - y_i$, y la distancia recorrida se consigue a través de las ecuaciones de movimiento:

$$\sqrt{dx_i^2 + dy_i^2} = \frac{a_i \cdot t_i^2}{2} + v_i \cdot t_i \tag{9}$$

Sustituyendo las ecuaciones 7 y 8 en la ecuación 9 se consigue el tiempo, en este caso expresado por la expresión 10.

$$t_i = \frac{\sqrt{dx_i^2 + dy_i^2}}{g \cdot dy_i} \cdot \left(\sqrt{2 \cdot g \cdot (y_0 - y_i) + 2 \cdot g \cdot dy_i} - \sqrt{2 \cdot g \cdot (y_0 - y_i)}\right) : dy_i \neq 0$$

$$t_i = \frac{dx_i}{\sqrt{2 \cdot g \cdot (y_0 - y_i)}} : dy_i = 0 \tag{10}$$

En este caso el objetivo es minimizar el tiempo total, es decir, cuanto menor sea el tiempo que necesita la masa para atravesar una curva, mejor será esa curva. Por lo tanto, se puede definir la función *fitness* de cada posible solución (cromosoma) C_i como indica la expresión 11.

$$f(C_i) = \frac{1}{T(C_i)} \tag{11}$$

Para analizar este problema se han definido los puntos $(x_0, y_0) = (0, 1)$ y $(x_k, y_k) = (1, 0)$, y se ha decidido discretizar la curva mediante ocho puntos: los dos mencionados anteriormente y otros seis a optimizar. Las posiciones y_i de estos seis puntos entre $(x_1, y_1), ..., (x_{k-1}, y_{k-1})$ forman los individuos candidatos a ser la solución de este problema. Se ha limitado el posible valor de estas posiciones y_i al intervalo [-1, 1]. En todas las ejecuciones se han utilizado el cruce simple y la mutación uniforme, y cada generación está formada por 20 individuos [1]. En las figuras 9 y 10 se pueden observar las curvas para la primera generación y la número 200.

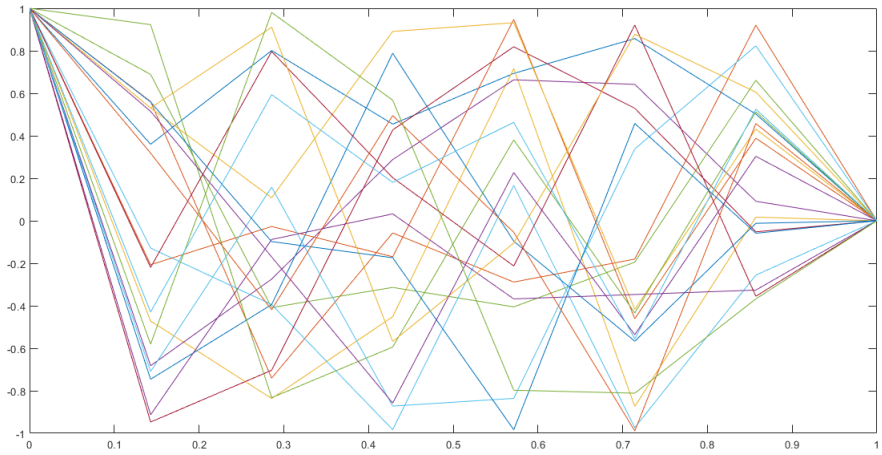

Figura 9: Curvas correspondientes a la primera generación.

[1] Después de varias pruebas iniciales estos parámetros han resultado ser los más adecuados.

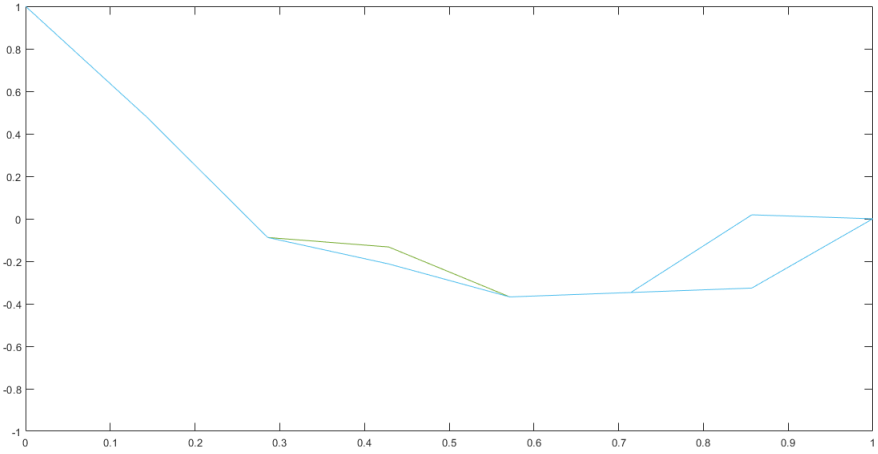

Figura 10: Curvas correspondientes a la generación 200.

Como se puede observar, las generaciones futuras van perdiendo diversidad y se acercan todas a una misma solución. Para que converjan hacia la solución óptima es muy importante, como ya se ha explicado anteriormente, incitar al algoritmo a la diversidad y a la exploración de nuevas soluciones, pero al mismo tiempo, no perder las buenas soluciones conseguidas hasta el momento. Para esto último, es muy útil aplicar la estrategia de elitismo. En las figuras 11 y 12 se puede comprobar que, al introducir la estrategia de elitismo, el valor F del mejor individuo de cada generación solo mejora, mientras que, de no introducirlo, el mejor individuo de una generación se puede perder en la transición hacia la siguiente.

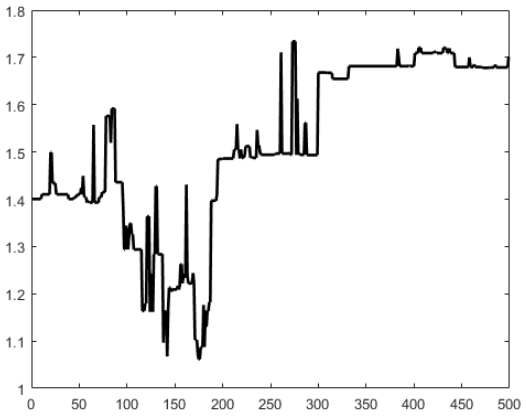

Figura 11: Mejor F de las primeras 500 generaciones sin aplicar el elitismo.

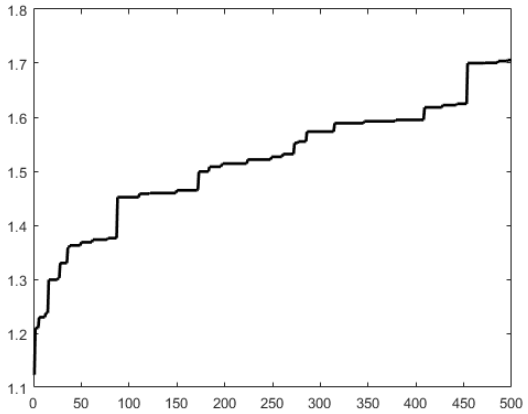

Figura 12: Mejor F de las primeras 500 generaciones aplicando el elitismo.

A continuación se han realizado diferentes ejecuciones para analizar el efecto de las probabilidades de cruce y mutación en el resultado y convergencia del algoritmo, manteniendo en todos los casos el elitismo. Para ello, primero se han aplicado tres condiciones de finalización:

1. Superar 1000000 de iteraciones.

2. Superar 30 segundos de ejecución.

3. Convergencia de la solución: como criterio de convergencia se ha tenido en cuenta la media de todos los valores F de cada generación, y se ha definido que la solución converja cuando aparezcan más de 2000 generaciones seguidas en las que esa media no varía más de un 5 %. También se podría haber utilizado como criterio la convergencia del mejor valor de F de cada generación, pero de esta manera cabe la posibilidad de que no se exploren suficientemente soluciones mejores.

Primero se ha analizado el comportamiento para diferentes valores de probabilidad de mutación P_m, realizando 10 iteraciones para cada valor, y manteniendo constante la probabilidad de cruce en P_c=0.6. En la tabla 1 se recogen las iteraciones b que ha realizado el programa hasta interrumpirse y el valor máximo de la función *fitness* que ha conseguido.

P_m=0.001		P_m=0.005		P_m=0.01	
b	F	b	F	b	F
7438	1.6991	565498	1.7801	565951	1.7801
11536	1.7647	430447	1.7801	564997	1.7801
7730	1.6672	546355	1.7801	518361	1.7801
10104	1.7103	544530	1.7801	547845	1.7801
29927	1.7798	557756	1.7801	562602	1.7801
3777	1.7698	549338	1.7801	565270	1.7801
4908	1.7083	560018	1.7801	504064	1.7801
9629	1.7791	554546	1.7801	545898	1.7801
9287	1.7754	512086	1.7801	574069	1.7801
7948	1.6718	538471	1.7801	534285	1.7801

Tabla 1: Número de iteraciones y valor del mejor F para diferentes valores P_m.

Para el valor de P_m=0.001 la interrupción ha ocurrido por convergencia, mientras que en las dos siguientes ha ocurrido por superar el tiempo establecido. Aumentando la probabilidad de mutación se llega a una solución que parece ser óptima ya que no se mejora en ningún momento, pero en cambio utiliza muchas iteraciones para ello sin llegar a converger, y como esto ha ocurrido para los valores de P_m=0.005 y P_m=0.01, de momento no se ha analizado que ocurre al aumentar todavía más la probabilidad.

Para observar todavía mejor el comportamiento del algoritmo respecto a la probabilidad de mutación, se ha escogido la solución de F=1.7801 como objetivo para implantarla como condición de convergencia. Es decir, en el momento que alguna solución alcance ese valor, la ejecución finalizará. Si no alcanza esa solución finalizará por cualquiera de las otras dos condiciones.

Se han realizado 10 iteraciones para los valores P_m 0.001, 0.005, 0.01 y 0.02. Para los cuatro valores de mutación la ejecución ha sido interrumpida porque ha llegado a la

solución entendida como óptima. En la tabla 2 se recogen el número de iteraciones medio necesario para ello, para los diferentes valores de P_m.

P_m	0.001	0.005	0.01	0.02
b medio	245873.2	61476	33946.8	23058.4

Tabla 2: Número de iteraciones medio para conseguir la solución óptima para cada valor P_m.

Por lo tanto se puede decir que el aumentar la probabilidad de mutación acelera el proceso de analizar diferentes soluciones, y, por lo tanto, la búsqueda de la solución óptima. En la figura 13 podemos ver la curva que corresponde al valor de F =1.7801 en azul y la curva braquistócrona calculada a través de la solución analítica en naranja. A la primera le corresponde un tiempo de t =0.5618, y a la segunda de t =0.5832. La diferencia de tiempo y del recorrido en la parte inicial de la curva son notorias, probablemente debido a la discretización planteada para la curva en el algoritmo genético. El discretizar la curva en más puntos con tal de conseguir una solución más cercana a la analítica conlleva ampliar el espacio de búsqueda, lo que ralentizaba demasiado la búsqueda de la solución , y en este caso el objetivo principal era entender el funcionamiento de los algoritmos genéticos. Para la discretización escogida la solución puede considerarse válida, y por lo tanto, también el algoritmo diseñado.

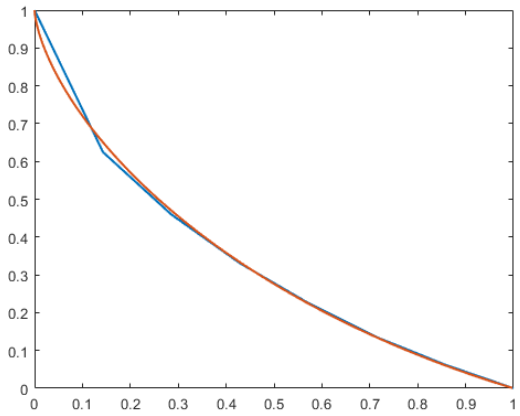

Figura 13: Curva del mejor individuo conseguido mediante el algoritmo genético diseñado (azul) y curva analítica (naranja).

Por otra parte, se puede manipular también la probabilidad de cruce P_c para cambiar el comportamiento del algoritmo. Manteniendo la probabilidad de mutación de 0.01 para así tener supuesto margen de mejora, se han hecho 10 ejecuciones con diferentes valores de P_c. Observando la tabla 3, aparentemente la probabilidad de cruce no afecta considerablemente en la reducción o aumento de iteraciones necesarias para llegar a la solución óptima.

P_c	0.6	0.7	0.8	0.95
b medio	33946.8	28425.8	33140	338938.6

Tabla 3: Número de iteraciones medio para conseguir la solución óptima para cada valor P_c.

5.2. Problema del viajante

Los problemas de optimización combinatoria son un ejemplo de problemas que se resuelven utilizando, entre otras técnicas, los algoritmos genéticos, dada su complejidad computacional, y el problema del viajante es uno de ellos.

Este problema trata de lo siguiente: dada una lista de n puntos que corresponden a diferentes lugares, ¿cuál es la ruta más corta que puede recorrer un viajante que pasa por todos los puntos una sola vez y regresa al punto de inicio? Pertenece al grupo de problemas *NP-hard*, los cuales en principio no pueden ser resueltos en tiempo polinómico. En este caso, el tiempo necesario aumenta de acorde al factorial $n!$ del número de puntos.

Como se acaba de mencionar, el objetivo es minimizar el recorrido D que el viajante debe recorrer, que corresponde a la suma las distancias geométricas entre los puntos que forman el recorrido. Por lo tanto, el valor F de cada individuo se puede definir de la siguiente manera:

$$F(C_i) = \frac{1}{D(C_i)} \tag{12}$$

El cromosoma que representa a cada individuo está formado por n números que van del 1 a n, los cuales hacen referencia a los diferentes puntos por los que tiene que pasar el viajante. Dadas las características de este problema, algunos de los algoritmos diseñados han tenido que ser modificados para asegurar la correcta evolución del problema hacia la solución correcta, y es que no todos los individuos pueden ser candidatos a ser la solución, ya que deben de incluir cada punto una sola vez.

En el caso del cruce, no tiene ningún sentido aplicar el *flat crossover*, y las estrategias de cruce simple, de varios puntos o uniforme derivan en soluciones no válidas. Se ha decidido modificar estas últimas estrategias para convertir las soluciones en válidas. La única modificación introducida comprueba los genes repetidos y los sustituye de manera que no se repitan. El proceso se explica en la figura 14. Si se compara el primer padre con el primer hijo, y el segundo padre con el segundo hijo, en azul están los genes que se han mantenido, los cuales se van a modificar. Para ello se intercambian con los genes del otro hijo, y una manera posible es la indicada por las flechas amarillas.

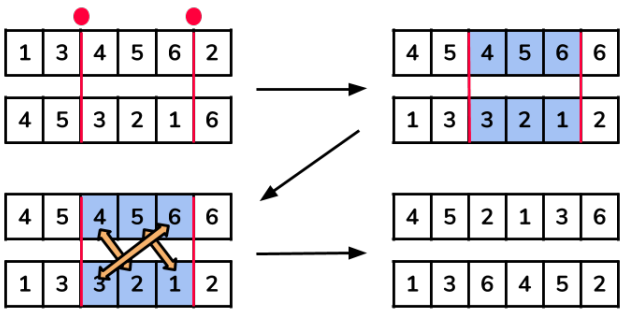

Figura 14: Ejemplo del operador de cruce en el problema del viajante para 6 puntos.

Los algoritmos principales de mutación tampoco sirven, ya que cambiar un gen del

individuo resultaría en tener dos genes repetidos, y como se ha mencionado, no tiene sentido en este caso. La alternativa utilizada en este caso ha sido aplicar la mutación sobre el individuo entero, creando un individuo totalmente nuevo.

Como ya se ha explicado anteriormente, aumentar la cantidad de puntos que debe recorrer el viajante hace que el explorar todas las posibles soluciones una a una resulte en un trabajo que se prolonga demasiado en el tiempo. Por ejemplo para 100 puntos hay 100! $\propto 10^{158}$ combinaciones posibles. Primero se ha probado el algoritmo genético diseñado para un problema de $n = 8$ puntos aleatorios, utilizando los valores de Pc=0.7 y Pm=0.02, con 20 individuos por generación [2]. En las figuras 15 y 16 se pueden ver los 20 recorridos correspondientes a dos generaciones diferentes.

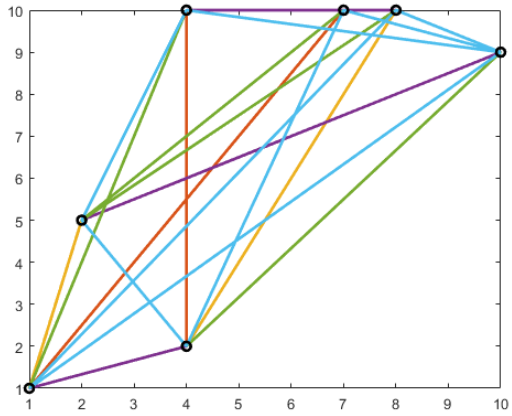

Figura 15: Recorridos correspondientes a la primera generación del problema del viajante de 8 puntos.

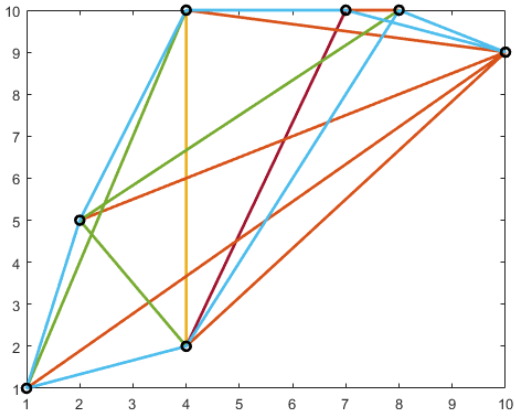

Figura 16: Recorridos correspondientes a la generación 50 del problema del viajante de 8 puntos.

En la figura 17 se muestran los valores F máximos en las primeras 500 generaciones. Concretamente, el valor F=0.0356 conseguido en la generación 157 se mantiene hasta el final, habiendo analizado un total de 773948 generaciones en total. El recorrido correspondiente a ese valor de la función *fitness* máximo se puede ver en la figura 18.

[2]Después de varias pruebas iniciales y los resultados obtenidos en el problema anterior estos parámetros han resultado ser los más adecuados.

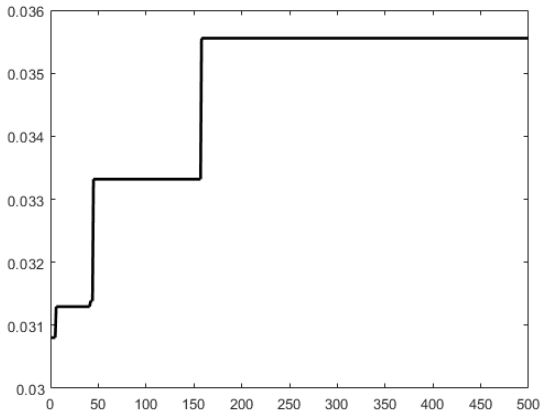

Figura 17: Valores F máximos para las primeras 500 generaciones del problema del viajante de 8 puntos.

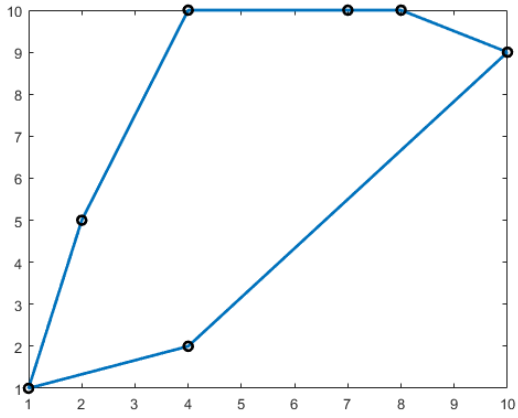

Figura 18: Recorrido óptimo para el problema del viajante de 8 puntos.

Repitiendo la ejecución del algoritmo genético 10 veces, se han necesitado una media de 688.5 ejecuciones para conseguir el resultado óptimo. En este caso, siendo 8 los diferentes puntos por los que tiene que pasar el viajante, existen 8!=40320 combinaciones de puntos, y por lo tanto, recorridos distintos. Al ser una cantidad aceptable de combinaciones existentes, se han comparado todas con tal de encontrar la mejor, y poder comparar el resultado obtenido mediante el algoritmo genético, y tal y como se puede ver en la figura 19, se consigue el mismo resultado.

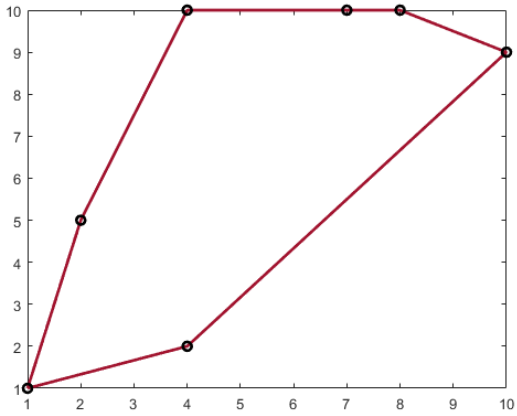

Figura 19: Recorrido que minimiza la distancia a recorrer para el problema del viajante de 8 puntos.

Después se ha probado para un problema de $n = 20$ puntos aleatorios, en el que existen $20! \propto 10^{18}$ recorridos distintos que puede recorrer el viajante, con los mismos valores para Pc y Pm que en el anterior caso. Se han establecido un tiempo máximo de 300 segundos, una cantidad máxima de 1000000 iteraciones del algoritmo y la convergencia hacia una misma solución en 2000 generaciones seguidas como condiciones de finalización de la ejecución del algoritmo para intentar resolver este problema. En la figura 20 se puede observar el mejor resultado obtenido.

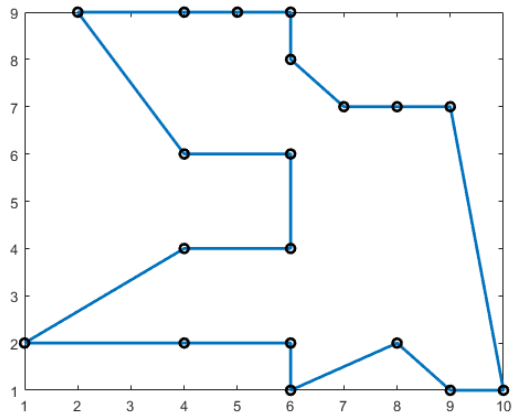

Figura 20: Mejor solución obtenida para el problema del viajante de 20 puntos.

Se han realizado 15 ejecuciones del algoritmo. En once de ellas no se ha llegado hasta este resultado antes de que se cumpliese alguna de las dos primeras condiciones de finalización, y el tiempo y las generaciones necesarias en las otras cuatro se recogen en la tabla 4. Por lo tanto, se puede decir que, haciendo uso de los algoritmos genéticos, es posible resolver este problema en una cantidad de tiempo asumible y aceptable. Aunque no se puede establecer la solución óptima obtenida como la mejor posible, se asemeja a la conseguida para el problema de 8 puntos, lo cual puede sugerir que está cerca de la solución correcta.

b	55751	44905	46876	21688
t	36.9688	24.2031	26.7344	11.2969

Tabla 4: Tiempo y generaciones necesarias para obtener el resultado óptimo en el problema del viajante en los casos en los que se ha conseguido.

5.3. Optimización de lanzamiento de cohete de dos fases

Un lanzamiento de cohete de dos fases es aquel que consta de dos etapas que posee, cada una, sus propios motores y combustible. En este trabajo se va a tratar de optimizar la cantidad de combustible necesaria en cada etapa para un lanzamiento vertical. Para ello se ha simplificado el problema, considerando un lanzamiento puramente vertical en la que la masa inicial de la aeronave está determinada.

El objetivo es conseguir la combinación de cantidad de combustible que maximiza el momento lineal final, es decir, el momento lineal alcanzado al consumir todo el combustible. Un posible objetivo de esto es desviar de su órbita diferentes objetos como misiles, basura espacial o meteoritos. Este problema ha sido planteado en [15] y [16] y en este trabajo se ha querido reproducir el experimento que ahí se describe. Así los datos que se han utilizado son los que aparecen en dichos artículos.

El momento lineal final viene dado por la expresión 13

$$p_f = v_f m_f \tag{13}$$

donde la masa final depende de las masas de los combustibles m_{c1} y m_{c2} de las dos fases, y de la masa inerte (motores, depósitos...) m_{i1} de la primera fase, como se puede ver en la expresión 14. En este caso la masa interte de la primera fase tiene un valor de 1000000 kg.

$$m_f = m_0 - m_{c1} - m_{i1} - m_{c2} \tag{14}$$

Como se ha mencionado anteriormente, la masa inicial del sistema es fija, en este caso con un valor de $m_0 = 2500000$ kg, por lo que cuanto mayor sea la cantidad de combustible, menor sera la masa final, contribuyendo menos al momento lineal. Respecto a la primera fase, cuando se termina el combustible la aeronave se desprende de la masa inerte, y su peso se reduce, conviene que esto ocurra cuanto antes. Pero, como ya se detallará más adelante, el empuje de la primera fase es mayor, por lo que puede ser positivo que la primera fase dure el tiempo máximo posible.

En general el llevar más combustible favorece a que haya mas empuje, pero a la vez añade más peso, estorbando a la aceleración y, por consiguiente a la velocidad final.

Analíticamente es muy complicado dar con la combinación de cantidad de combustible que optimice el momento lineal teniendo en cuenta todos los factores mencionados hasta ahora, por lo que utilizar un algoritmo genético puede ser útil para no tener que comprobar y comparar todas las combinaciones posibles.

Para calcular el momento lineal final, que en este caso va a representar a la función *fitness*, es necesario conocer, además de la masa final, la velocidad final. Se han utilizado métodos numéricos para integrar la ecuación 15 con tal de conseguir la velocidad final.

$$F_T = F_{empuje} - F_{aire} - F_{gravedad} \tag{15}$$

El valor del empuje viene dado por la ecuación 16 16

$$F_{empuje} = I_{sp}\dot{m} \tag{16}$$

donde I_{sp} es el impulso específico, que representa el impulso provocado por cada unidad de combustible, y $\dot{m}$ es el flujo másico, que representa la cantidad de combustible expelido por segundo. Teniendo en cuenta los datos de la tabla 5, el impulso de la primera fase es de 50 MN, y el de la segunda de 14 MN.

	1ª fase	2ª fase
I_{sp} (m/s)	5000	7000
$\dot{m}$ (kg/s)	10000	2000

Tabla 5: Impulso específico y flujo másico para cada una de las dos fases de un lanzamiento vertical de dos fases.

Por su parte, el aire causa una fuerza de rozamiento proporcional al cuadrado de la velocidad de la aeronave que viene dada por la expresión 17

$$F_{aire} = \frac{1}{2}\rho A_{ef} v^2 \tag{17}$$

donde ρ es la densidad del aire de la atmósfera, que en este caso se expresa mediante el modelo exponencial de la ecuación 18, y donde A_{ef} es el área efectiva de la aeronave a

efectos del rozamiento con el aire, que en este caso tiene un valor de 35 m^2 en la primera fase, y 10 m^2 en la segunda.

$$\rho(h) = 1.225e^{-0.00011856 \cdot h} \text{ kg/m}^3 \tag{18}$$

También hay que tener en cuenta la fuerza de atracción que ejerce la tierra sobre la aeronave, expresada mediante la ecuación 19, donde g=9.80665 m/s^2 es la aceleración de la gravedad en la superficie terrestre, m es la masa de la aeronave, h es la altura de la aeronave respecto a la superficie terrestre y R=6371000.7900 m es el radio de la Tierra [13] [14].

$$F_{gravedad}(h) = gm\left(\frac{R}{R+h}\right)^2 \tag{19}$$

Para este problema, cada individuo se ha representado mediante un vector fila que contiene las masas de combustible de cada fase, que pueden adquirir valores de entre 0 y 1000000 kg. Por su parte, como se ha explicado anteriormente, el valor de la función *fitness* viene dado por el momento lineal final de cada individuo, pero con tal de buscar de la manera más eficiente posible la solución óptima, se han establecido algunos casos en los que el individuo es directamente descartado:

- Masa final de alguna de las dos fases inferior a 1000 kg.
- Masa final de la primera fase menor a la cantidad de combustible de la segunda fase.
- Aceleración nula o negativa al comienzo de alguna de las dos fases.

Se han establecido los valores P_c=0.7 y P_m=0.02 teniendo en cuenta los resultados obtenidos anteriormente, y un máximo de 1000000 iteraciones o dos minutos de ejecución. Como en los anteriores problemas, cada generación está compuesta por 20 individuos, y en este caso cada individuo está compuesto por dos valores, por lo que puede ser interesante analizar el comportamiento del algoritmo para dos estrategias de cruce: cruce simple y *flat crossover*. Con cada estrategia se ha ejecutado el algoritmo genético 10 veces y se han recogido las iteraciones realizadas en cada ejecución y el valor F máximo obtenido en cada una de ellas en la tabla 6.

Cruce simple		*Flat crossover*	
b	F $(\cdot 10^3)$	b	F $(\cdot 10^3)$
21631	1.7868	23147	1.7869
21252	1.7861	22064	1.7868
22476	1.7868	21587	1.7869
23516	1.7867	21696	1.7869
23120	1.7861	22063	1.7869
23044	1.7864	22255	1.7869
23433	1.7859	23010	1.7869
23444	1.7867	22977	1.7869
24190	1.7864	22855	1.7869
22295	1.7868	22546	1.7868

Tabla 6: Iteraciones realizadas y F máximo alcanzado en cada ejecución para las estrategias de cruce simple y *flat crossover*.

Tras las ejecuciones se ha observado que en todos los casos se han interrumpido por haber alcanzado el tiempo máximo establecido. Así mismo, se puede observar que el utilizar la estrategia de *flat crossover* facilita la convergencia hacia mejores individuos. Los individuos con aparente mismo valor F no tienen por qué ser idénticos, ya que la cantidad de cifras significativas se ha reducido a 4.

Una vez sabiendo qué estrategia facilita la búsqueda de mejores individuos, se ha vuelto a ejecutar el algoritmo, utilizando el *flat crossover* y realizando 100000 iteraciones, con tal de conseguir una solución mejor. La combinación de masas óptima conseguida es la siguiente:

m_{c1}=338499 kg m_{c2}=855902 kg

A estas masas les corresponde un momento lineal final de p_{final}=1.7870·10^9 kg· m/s y una altura final de h=872.538 km, superior a la altura mínima para considerar la misión como razonable. En las figuras 21, 22 y 23 se han representado la altura, velocidad y aceleración de la aeronave en el tiempo que duran las dos fases, respectivamente.

La aeronave no pierde altura ni velocidad en ningún momento. Por su parte, la aceleración crece a medida que pasa el tiempo, como se puede esperar teniendo en cuenta la expresión 15. A la aeronave la gravedad terrestre le afecta cada vez menos ya que va perdiendo masa y ganando altitud. Además, esto último conlleva que la densidad del aire sea menor, y por lo tanto, también el rozamiento del aire. Por su parte, la disminución de aceleración corresponde al cambio de fase en el que el empuje se reduce.

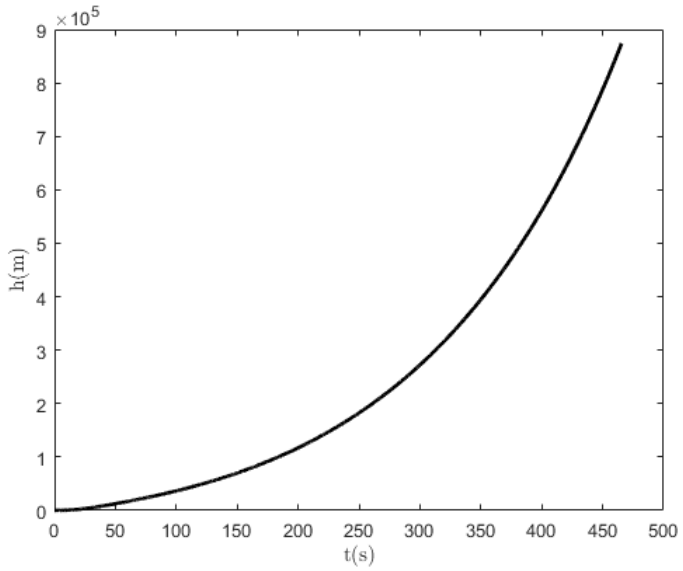

Figura 21: Altura de la aeronave respecto a la superficie terrestre durante el lanzamiento de dos fases.

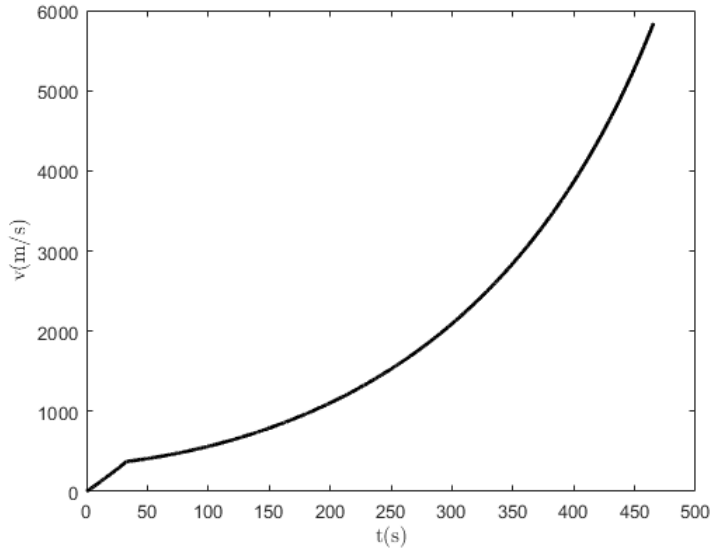

Figura 22: Velocidad de la aeronave durante el lanzamiento de dos fases.

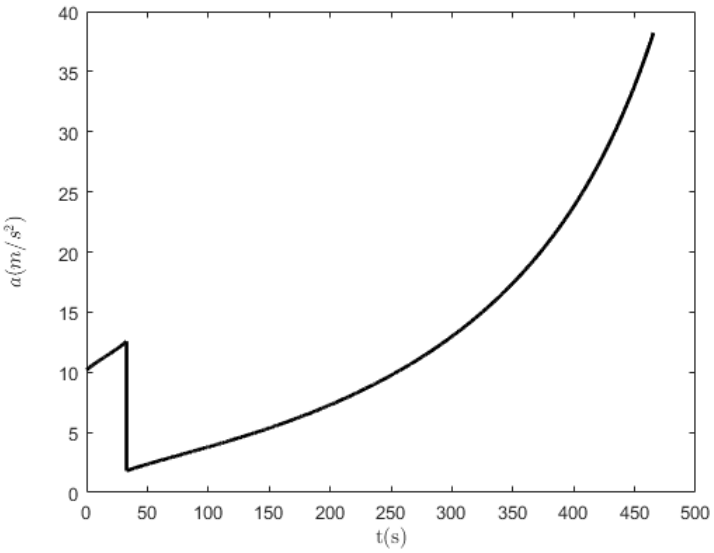

Figura 23: Aceleración de la aeronave durante el lanzamiento de dos fases.

5.4. Optimización de la altura de una órbita circular terrestre baja

En este apartado se va a tratar de optimizar la altura de una órbita circular terrestre baja, las cuales se sitúan entre 80-2400 km sobre la superficie de la Tierra, para un lanzamiento de dos fases [17]. La misión consiste en cuatro maniobras, y aunque en cada una de ellas la nave realiza correcciones en la trayectoria para minimizar los efectos de diferentes fenómenos, para este problema se han simplificado como se va a explicar próximamente.

La primera maniobra es un ascenso vertical, y el tiempo necesario, t_a, va a ser uno de los tres parámetros a ajustar en este problema. Las ecuaciones de movimiento para esta maniobra son las mismas que se han descrito anteriormente en el apartado 5.3.

La segunda maniobra es un *pitch-push over*, cuyo objetivo es dotar a la aeronave de un ángulo específico de vuelo para poder llevar a cabo la siguiente maniobra. Su duración t_{pp} y el ángulo final γ_{pp} son los otros dos parámetros a ajustar. Normalmente estos dos parámetros suelen tener valores pequeños por lo que se considera que el empuje y la velocidad del cohete tienen la misma dirección, y se asume una velocidad de rotación constante, que viene dada por la ecuación 20 [18].

$$\omega_{pp} = \frac{\gamma_{pp}}{t_{pp}} \tag{20}$$

En la tercera maniobra, denominada *gravity turn*, el empuje acelera la nave en la dirección de la velocidad al mismo tiempo que la gravedad cambia la dirección del vector de la velocidad, como se puede ver en la figura 24.

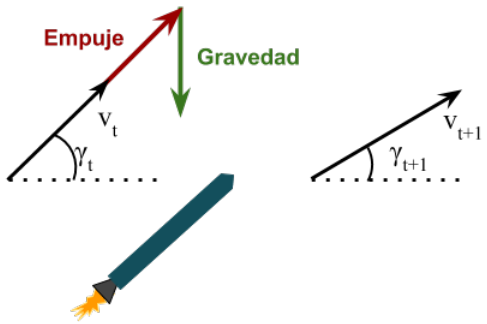

Figura 24: Esquema de la maniobra *gravity turn*.

Esta maniobra se describe mediante las ecuaciones de movimiento 21, 22, 23 y 24 [19].

$$\frac{dx}{dt} = V\cos\gamma \tag{21}$$

$$\frac{dh}{dt} = V\sin\gamma \tag{22}$$

$$m\frac{dV}{dt} = F_{empuje} - F_{aire} - \left(F_{gravedad} - \frac{m\left(\frac{dx}{dt}\right)^2}{R+h}\right)\sin\gamma \tag{23}$$

$$mV\frac{d\gamma}{dt} = -\left(F_{gravedad} - \frac{m\left(\frac{dx}{dt}\right)^2}{R+h}\right)\cos\gamma \tag{24}$$

La cuarta y última maniobra consiste en una maniobra bilineal tangencial. El valor de γ para cada instante viene dado por la ecuación 25, donde γ_0 es el ángulo al principio de la maniobra, γ_f es el ángulo final, en este caso nulo ya que el objetivo es conseguir una orbita circular, y t_f es la duración de la maniobra.

$$\tan\gamma = (\tan\gamma_f - \tan\gamma_0)\frac{t}{t_f} + \tan\gamma_0 \tag{25}$$

Se va a tratar de resolver este problema empleando un modelo de aeronave basado en las especificaciones del Ariane 5, que consta de dos fases de propulsión [20]. La primera fase proporciona el empuje para las tres primeras maniobras, y está formada por dos motores P238 y un motor H158, aunque los motores P238 no llegan a completar la tercera maniobra al completo y se desechan cuando se quedan sin carburante. Una vez finalizada la tercera maniobra al completo, se desecha también el H158. La segunda fase se realiza gracias al motor Aestus. En este caso, el área efectiva es de 50 m^2 para cuando están funcionando los dos motores P238 y el H158, de 30 m^2 para cuando esta el H158, y de 10 m^2 en la última fase, y la masa inicial de la aeronave es de 761910 kg. Las demás especificaciones técnicas de las dos fases se han resumido en la tabla 7.

	H158	P238	Aestus
Masa del carburante (kg)	155000	277000	8910
Masa de la fase vacía (kg)	15000	4000	150
Tiempo de quema (s)	589	129	1100
Flujo de masa (kg/s)	263	2147	8.1
Fuerza de empuje (N)	1118000	6050000	27540

Tabla 7: Especificaciones técnicas de los diferentes motores del Ariane 5.

Como se ha mencionado anteriormente, el objetivo es encontrar los valores de t_a, t_{pp} y γ_{pp} que maximicen la altura de una órbita circular terrestre baja. Por lo tanto, para este problema cada individuo se va a representar mediante un vector que contiene estos valores. Teniendo en cuenta las especificaciones de las diferentes fases, se ha decidido que los valores posibles para cada parámetro sean t_a, t_p (s) $\in$[1,40] y γ_p (°) $\in$[75,89.9].

Como se busca maximizar la altura de una órbita circular, a la hora de definir la función *fitness* se han tenido dos factores en cuenta: un individuo será mejor cuanto mayor altura final consiga, y cuanto más se parezca su velocidad final a la velocidad orbital correspondiente a una órbita circular de esa altura. En la ecuación 26 se define la función *fitness* para este problema, aunque se ha definido como nulo en los casos en los que la altura final tenga un valor negativo. Se han utilizado métodos numéricos para integrar las ecuaciones de movimiento de la aeronave y así poder conocer su trayectoria y altura y velocidad finales.

$$F(C_i) = \frac{h_f(C_i)}{1 + |V_f(C_i) - V_{orb}(C_i)|} \tag{26}$$

Por su parte, la velocidad orbital viene dada por la ecuación 27.

$$V_{orb} = R\sqrt{\frac{g}{R+h}} \tag{27}$$

Basándose en resultados obtenidos con los anteriores problemas, se han establecido los valores de P_c=0.7 y P_m=0.02 y se ha utilizado la estrategia de *flat crossover* a la hora de realizar el cruce entre individuos. Se han mantenido las 1000000 generaciones máximas,

cada una formada por 20 individuos, pero al ser un problema a priori más complejo, se ha aumentado el tiempo máximo de ejecución a 1800 segundos.

Después de 3843 iteraciones el algoritmo se ha interrumpido por haber alcanzado el tiempo máximo establecido, y se ha obtenido un individuo de F=$3.1803 \cdot 10^6$, con los siguientes valores con los parámetros a ajustar:

t_a=17.6937 s	t_{pp}=22.9264	γ_{pp}=81.3977°

La altura final es de h_f=3188.58 km y la velocidad final v_f=6.4486 km/s. La velocidad orbital correspondiente a una órbita a esa altura es de v_{orb}=6.4528 km/s. A primera vista las dos velocidades son parecidas y observando las figuras 25, 26, 27 y 28 en las que se describe la trayectoria de la aeronave claramente se puede ver que la órbita es estable, y por lo tanto, la solución conseguida es válida.

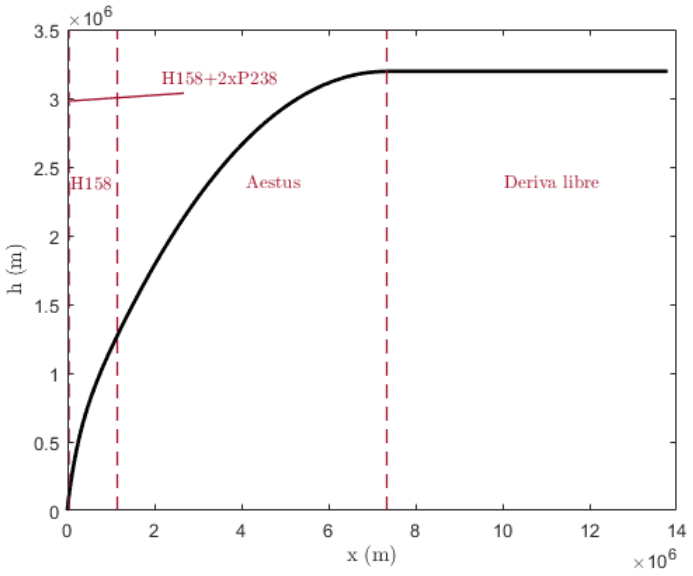

Figura 25: Perfil de altitud de la aeronave.

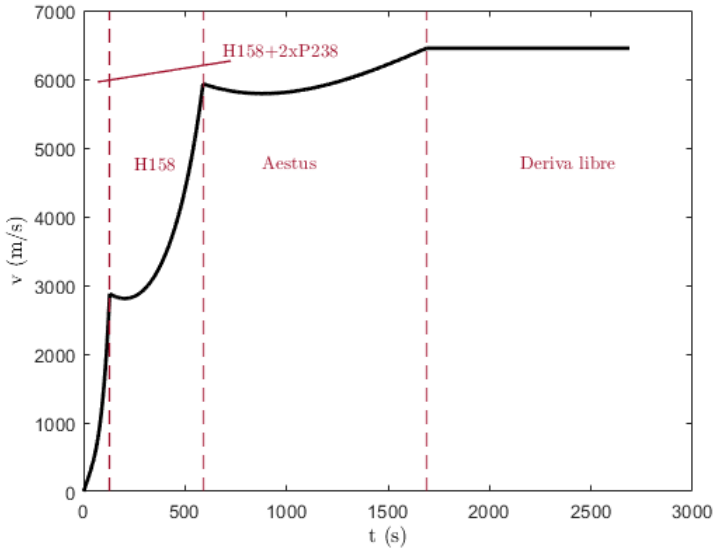

Figura 26: Velocidad de la aeronave respecto al tiempo.

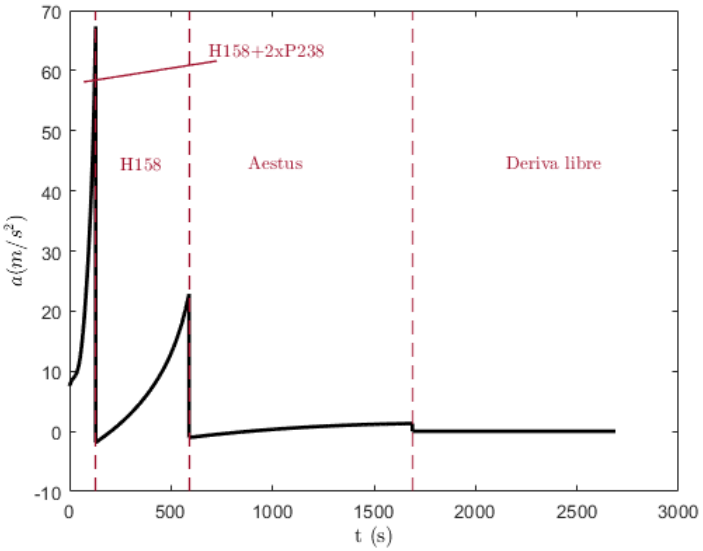

Figura 27: Aceleración de la aeronave respecto al tiempo.

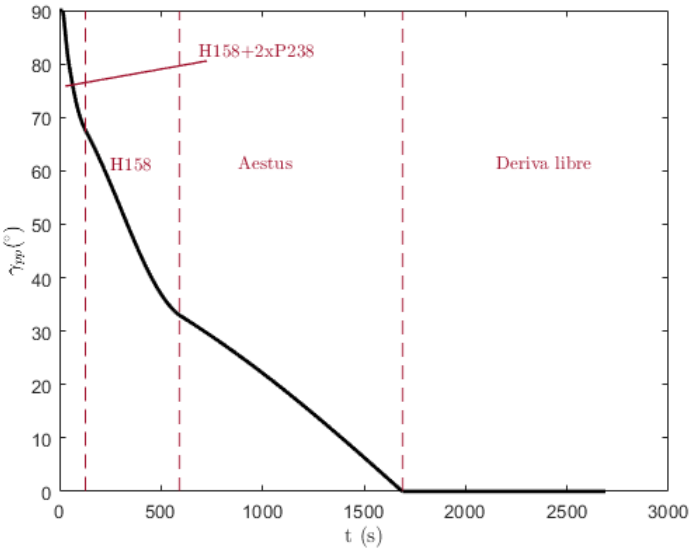

Figura 28: Ángulo γ_{pp} de la aeronave respecto al tiempo.

6. CONCLUSIONES

En este trabajo se ha llevado a cabo el estudio de los algoritmos genéticos y su aplicación a diferentes problemas físicos. En primer lugar se ha analizado con detalle el funcionamiento de dichos algoritmos así como las diferentes estrategias existentes para abordar la resolución de distintos tipos de problemas. Posteriormente, se ha implementado en Matlab un algoritmo genético junto con diferentes variantes en sus principales funciones (operadores genéticos) y se ha aplicado a la resolución con éxito de algunos problemas de física y de combinatoria.

A través del primer problema analizado, el problema de la curva braquistócrona, se ha comprobado la importancia de, por un lado, utilizar la estrategia del elitismo, y por otro lado, ajustar correctamente las probabilidades de cruce y mutación. Utilizar el elitismo asegura que el mejor individuo de cada generación nunca sea peor que el de la anterior, lo que favorece la búsqueda de la solución óptima, ya que de no utilizarlo, soluciones buenas se pueden perder por el camino. Además, se ha visto el efecto que las probabilidades de cruce y mutación realizan en el desarrollo del algoritmo. Aunque el efecto de la primera no es tan notorio, se ha comprobado que el aumentar la segunda acelera de manera considerable la búsqueda de diferentes soluciones, y por lo tanto, de la óptima.

Por su parte, el problema del viajante ha servido para constatar la validez de los algoritmos a la hora de solucionar problemas de optimización combinatoria en la que la cantidad de combinaciones existentes es muy grande. En este caso se han tenido que probar diferentes variantes de los operadores genéticos para dar con la estrategia acertada, ya que las estrategias comunes de cruce y mutación no tienen como resultado soluciones válidas para este problema. Se ha conseguido diseñar un algoritmo que ha conseguido resolver el problema para 8 puntos. Esto se ha podido comprobar gracias a que se han podido utilizar técnicas de fuerza bruta para obtener la solución del problema. Para 20 puntos, aunque es inviable comparar todas las soluciones, por el aspecto de la solución puede considerarse suficientemente buena.

El siguiente paso ha sido aplicar el algoritmo genético a dos problemas relacionados con lanzamientos de aeronaves de más de una fase. Estos dos casos han servido para reconocer la importancia que tiene calcular la función *fitness* correctamente. En estas dos ocasiones, el cálculo de esta función ha conllevado utilizar métodos numéricos para integrar las ecuaciones de movimiento de las aeronaves. Es decir, el cálculo de la función *fitness* no ha sido directo y los errores que se han cometido en un primer momento al aplicar los métodos numéricos han hecho que no haya sido inmediato conseguir una solución válida.

Concretamente, se ha aprovechado el caso de la optimización de lanzamiento de dos fases para comparar las estrategias de cruce simple y *flat crossover*, concluyendo que la última favorece la convergencia hacia mejores individuos. Por su parte, el problema de la órbita circular baja no se ha utilizado para analizar ningún aspecto del algoritmo genético, si no que simplemente, aplicando los resultados obtenidos con los anteriores problemas, se ha conseguido resolver con éxito.

Para llevar a cabo este trabajo han sido de gran utilidad los conocimientos adqui-

ridos durante la carrera. Por un lado, los conocimientos en mecánica clásica y métodos matemáticos han sido de gran ayuda a la hora de entender los aspectos relacionados con la física de cada problema. Por otro lado, ha sido totalmente necesario tener una base sólida en programación, adquirida entre otros gracias al estudio de los métodos numéricos, para ser capaz de diseñar los algoritmos genéticos y sus estrategias.

Por último, cabe recalcar que es difícil en la mayoría de los casos saber si la solución conseguida a través de un algoritmo genético es la óptima. En la mayoría de los casos siempre se puede alargar el tiempo de ejecución para poder crear más generaciones e intentar conseguir mejores soluciones, y también se puede mejorar el rendimiento de los algoritmos ajustando de manera óptima las probabilidades de cruce y mutación y sabiendo qué estrategias convienen en cada momento. Dominar y optimizar el funcionamiento de un algoritmo genético requiere muchas pruebas, por lo que puede quedar como futuro trabajo realizar este trabajo empírico para pulir el algoritmo genético diseñado.

Referencias

[1] Anuplan Shukla, Ritu Tiwari, Rahul Kala, *Real Life Application of Soft Computing*, (CRC Press, 2010), p. 147.

[2] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), p. 47.

[3] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), pp. 48-49.

[4] Anuplan Shukla, Ritu Tiwari, Rahul Kala, *Real Life Application of Soft Computing*, (CRC Press, 2010), pp. 148-149.

[5] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), pp. 49-50.

[6] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), pp. 59-60.

[7] Anuplan Shukla, Ritu Tiwari, Rahul Kala, *Real Life Application of Soft Computing*, (CRC Press, 2010), pp. 157-158.

[8] Anuplan Shukla, Ritu Tiwari, Rahul Kala, *Real Life Application of Soft Computing*, (CRC Press, 2010), p. 158.

[9] Anuplan Shukla, Ritu Tiwari, Rahul Kala, *Real Life Application of Soft Computing*, (CRC Press, 2010), pp. 159-160.

[10] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), p. 55.

[11] Oscar Cordón, Francisco Herrera, Frank Hoffmann, Luis Magdalena, *Genetic Fuzzy Systems: Evolutionary tuning and learning of fuzzy knowledge bases*, (World Scientific, 2001), p. 56.

[12] The MathWorks, «MatLab», 2020. Disponible: `https://es.mathworks.com/products/matlab.html`

[13] Bureau international des poids et mesures, The International System of Units (SI), (2006), pp. 142–143.

[14] Moritz, H., Geodetic Reference System, (1980).

[15] Fernando Alonso Zotes, Matilde Santos Peñas, Optimización de lanzamiento vertical de dos fases utilizando un algoritmo genético multicriterio.

[16] Fernando Alonso Zotes, Application of Intelligent Algorithms to Aerospace Problems, (2011), p. 87.

[17] Howard Curtis *Orbital Mechanics for Engineering Students*, (Elsevier Butterworth-Heinemann, 2005) , p.52.

[18] Fernando Alonso Zotes, Matilde Santos Peñas, Multi-criteria genetic optimisation of the manoeuvres of a two-stage launcher, (2009).

[19] D.W. Callaway, Coplanar Air Lunch With Gravity-turn, Force Institute of Technology, (2004), p. 13.

[20] E. Perez, Ariane 5 User's Manual, (2004).