Universidad Complutense de Madrid

Facultad de Matemáticas

Trabajo de Fin de Grado

Estudio de los algoritmos genéticos

Guzmán Silva Álvarez

Grado en Matemáticas

Dirigido por:
Fernando Rubio Diez

Junio de 2022

Agradecimientos

En agradecimiento a mis padres, Ramón y Gema, por otorgarme una gran educación, por enseñarme unos valores y principios que me llevaré conmigo.

A mi hermano Diego y a mis amigos que me acompañaron a lo largo de este camino e hicieron del mismo uno mucho más ameno. Sé que estaréis ahí siempre.

Índice general

Resumen.

El presente trabajo tratará de desarrollar y explicar los algoritmos genéticos, que son algoritmos bio-inspirados, también conocidos como métodos heurísticos basados en la naturaleza, los cuales se engloban dentro de la computación bio-inspirada y algoritmos evolutivos. Concretamente veremos su aplicación a problemas reales de distinta índole.

El trabajo nace de un profundo interés en la aplicación de las matemáticas al mundo real, dejando a un lado su parte más abstracta y extrayendo toda la utilidad que nos brinda. Gracias a los avances tecnológicos de las últimas décadas, las matemáticas han cobrado más valor aún del que ya tenían. Desde el procesamiento de datos, algoritmos de recomendación, ciberseguridad y muchas otras, las matemáticas se encuentran presentes en el día a día de cualquier persona. Una de las nuevas ramas que se desarrolló en ciencias de la computación y las matématicas gracias a estos avances es lo que estudiaremos en este trabajo y que desarrollaremos más adelante.

A través de este trabajo se tratará de explicar y mostrar las aplicaciones que tienen los algoritmos genéticos, así como los distintos tipos de problemas a los que son capaces de dar solución, los cuales sin ellos muy difícilmente podríamos resolver. Estudiaremos las distintas metodologías que nos brindan y las probaremos para distintos problemas, estudiando qué bien funcionan para cada uno. Finalmente observaremos si existe o no alguna metodología que funcione mejor que el resto y si para problemas similares alguna de las opciones se desenvuelve mejor.

Summary.

This paper will try to develop and explain genetic algorithms, which are bio-inspired algorithms, also known as nature-based heuristic methods, which fall under the umbrella of bio-inspired computing and evolutionary algorithms. Specifically, we will look at their application to real problems of different kinds.

The work stems from a deep interest in the application of mathematics to the real world, leaving aside its more abstract side and extracting all the usefulness it offers us. Thanks to the technological advances of recent decades, mathematics has become even more valuable than it already was. From data processing, recommendation algorithms, cybersecurity and many others, mathematics is present in everyone's everyday life. One of the new branches that developed in computer science and mathematics thanks to these advances is what we will study in this paper and develop further.

In this work we will try to explain and show the applications of genetic algorithms, as well as the different types of problems they are capable of solving, which would be very difficult to solve without them. We will study the different methodologies they offer us and we will test them for different problems, studying how well they work for each one. Finally, we will observe whether or not there is any methodology that works better than the rest and if for similar problems any of the options performs better.

Capítulo 1

Contexto.

Los algoritmos genéticos son aquellos pertenecientes a lo que en ciencias de la computación se conoce como algoritmos evolutivos. Esta clase de algoritmos son bio-inspirados, es decir, se inspiran en procesos evolutivos que podemos encontrar en la naturaleza. No fue hasta mediados del siglo XX que se empezaron a desarrollar estos nuevos conceptos computacionales, los cuales nacen con la finalidad de dar solución a problemas de optimizacion global a partir de distintos procesos evolutivos. Esta nueva perspectiva otorga ciertas ventajas frente a una perspectiva informática más tradicional, ya que al estar basados en la naturaleza adquieren la capacidad de hacer frente a cambios repentinos del entorno, minimizar la capacidad energética utilizada o la resolución de problemas complejos a través de sistemas más simples.

Los algoritmos genéticos que desarrollaremos en este trabajo no son los únicos algoritmos bioinspirados existentes. Algunos ejemplos de estos algoritmos podrían ser los algoritmos de colonias de hormigas (*Ant Colony Optimization, ACO*) o de enjambre de particulas (*Particle Swarm Optimization, PSO*). Como su propio nombre indica, el *ACO* se inspira en las acciones que toma una colonia de hormigas en su busqueda por comida, y los problemas a los que da solución, simplificando, son aquellos que tratan de hallar los mejores caminos o rutas en un grafo [Wik22a]. El *PSO* nació con la intención de describir conductas sociales de organismos vivos como aves o peces, y una analogía de su funcionamiento en la naturaleza sería cómo las abejas trabajaban para buscar las mejores zonas en las que encontrar alimento, actualizando nuestras soluciones en nuestro espacio de búsqueda en función de cuán buena sea la zona para la función objetivo [Wik22b].

Como hemos mencionado, no fue hasta mediados del siglo XX cuando se empezó a desarrollar esta nueva rama de la informática. El concepto en el que se basan estos algoritmos tiene su origen incluso antes de la aparición de los ordenadores, y es que Alan Turing ya propuso un método de búsqueda

genética. Sin embargo no es considerado pionero, pues sus artículos no fueron publicados hasta después de su fallecimiento y es por ello que este título se le atribuye a Alex Fraser y a Nills Aall Barricelli, los cuales publicaron distintos artículos previamente. Serían otros investigadores los responsables de los principales avances, durante la decada de los 60, como Lawrence J. Fogel, que asentó las bases de los algoritmos evolutivos, llegando a resolver problemas de predicción y sistemas de identificación, o John Henry Holland, que introdujo los algoritmos genéticos en Estados Unidos desde un punto de vista distinto, usándolos como métodos de estudio de la adaptación en la naturaleza y tratando de hallar una forma de simularla, entre muchos otros investigadores que aportaron en su avance [Wik22e].

De las últimas aplicaciones que han tenido estos algoritmos, y para poner en valor su utilidad, podríamos destacar aquella realizada por la NASA en 2006, cuando utilizó un algoritmo genético para el desarrollo de una antena en su misión tecnológica espacial 5 (*Space Technology 5, ST5*). Como resultado se obtuvo una antena cuya forma no había sido usada previamente y con una eficacia mayor que todas las anteriores [Qar].

Desde un punto de vista más formal, los algoritmos evolutivos son un método de resolución de problemas de búsqueda y optimización, los cuales combinan características de búsqueda aleatoria con características de busqueda dirigida basadas en la adaptabilidad de los mejores individuos. Estas características les permiten abordar una gran cantidad de problemas distintos y llevarlo a cabo de manera eficiente, sin embargo estos algoritmos tienen una metodología muy general y existirán problemas particulares para los que un algoritmo específico funcione de mejor manera.

Como se ha explicado, proporcionan un esquema general y somos nosotros los encargados de especificar los distintos componentes para que se adapten al problema en cuestión, aunque es cierto que en muchos casos dichos componentes son similares para problemas distintos. De entre todos ellos cabe destacar la función de adaptabilidad o función *fitness*, que será el indicador que usaremos para saber cuan bueno es o no un individuo. Por ejemplo, en un problema en el que se busca gestionar la venta de productos de una determinada empresa, podríamos considerar el beneficio obtenido por cada producto como nuestra función *fitness*. Sobre dicho componente será que gire todo el problema y su resolución. Es importante destacar que esta clase de algoritmos no siempre garantizan la solución óptima al problema, sino que obtendremos una aproximación cuya calidad dependerá de los recursos con los que contemos, es decir, capacidad computacional, tiempo de ejecución, capacidad económica, etc; junto con la estructura de la que dotemos a los componentes [Lou09].

Capítulo 2

Algoritmos genéticos.

Comenzaremos ahora con el desarrollo de nuestro trabajo. En este capítulo se tratará de explicar y desarrollar los distintos conceptos de estos algoritmos de una manera más profunda y formal, explicaremos cómo funcionan, los distintos componentes con los que cuentan y cómo aplicarlos.

Para ello nos basaremos principalmente en [Lou09], junto con algún apunte de otras fuentes que se mencionarán cuando se precise.

2.1. Introducción.

Introducimos el esquema que comparten todos los algoritmos genéticos y las distintas fases con las que cuenta, para más tarde ahondar en su funcionamiento. Para esta sección nos basaremos también en [Wik22d] y [Wik22f].

Como hemos mencionado a la hora de contextualizar estos algoritmos y como su propio nombre indica, veremos que tienen un funcionamiento completamente análogo al desarrollo evolutivo en la naturaleza. En la naturaleza, la capacidad de supervivencia de una especie viene dada por aquella que tienen sus individuos, es decir, una especie en la que la mayoría de sus individuos tengan una mala capacidad de supervivencia probablemente acaben pereciendo, y viceversa. Para que una especie pueda prevalecer necesitan contar con buenos individuos desde un punto de vista de supervivencia, que sean capaces de adaptarse, subsistir, compitiendo con el resto de especies por alimento, refugio, un compañero, etc., con el fin de tener una descendencia y poder transmitir, a partir del material genético, el aprendizaje que han obtenido a lo largo de su vida y que será de utilidad para las próximas generaciones. Este es el proceso que cualquier especie sigue para evitar su extinción, desde los seres humanos hasta los insectos.

El hecho de que los individuos con mejor adaptación tengan mayores

probabilidades de transmitir su material genético a las siguientes generaciones, es a priori algo positivo, ya que los descendientes contarán también con grandes probabilidades de tener una alta capacidad de adaptación; pero no podemos valernos únicamente de estos individuos para que nuestra especie subsista, pues transmitiendo únicamente la información genética de los individuos mejor adaptados se crearía una generación de descendientes muy elitista, y con el paso del tiempo se acabarían perdiendo o no aparecerían ciertas características de gran valor. Es por ello que también es importante tener en cuenta a los individuos con niveles de adaptación algo menores, permitiendo así que nuestra población cuente con cierta diversidad, ayudando en la aparición de nuevas características en los individuos que puedan permitirles tener un mayor nivel de adaptación, y que a partir de un proceso elitista sería más complicado.

Este mismo desarrollo es el que siguen estos algoritmos. Una forma recurrente de explicar esta clase de algoritmos es a través de un ejemplo de aplicación a un problema sencillo y de planteamiento claro. Seguiremos este mismo esquema y según vayamos avanzando iremos detallando aquellas variantes más complejas y específicas.

Trabajaremos sobre una población en la que cada individuo será una posible solución al problema que tratemos, y cada uno contará con un valor asociado el cual servirá de indicador para saber cuál es su nivel de adaptabilidad. Será de entre todos estos individuos que obtengamos la solución óptima aproximada en caso de que cumplamos con las condiciones de finalización. El valor que asociemos a cada una de las posibles soluciones lo obtendremos a partir de la función de adaptación o función *fitness*, que hemos mencionado previamente, y es esta función la que medirá la calidad de nuestras soluciones y en torno a la cual girarán el resto de componentes. Es habitual que nuestros problemas se planteen como la optimización de una función matemática explícita y en la mayoría de estos casos la función a optimizar coincidirá con la funcion de adaptación, también es frecuente realizar algunas transformaciones a la función a optimizar y obtener una con la que trabajemos mejor o sea más eficiente para nuestro problema particular.

En los algoritmos genéticos los individuos pueden tomar distintas representaciones. La más habitual es a través de una cadena de bits, que denominaremos representación binaria, en la que el alfabeto utilizado será $\{0,1\}$, la cual aporta importantes características de eficiencia frente a otras representaciones. Usualmente emplearemos la palabra *gen* para referirnos a cada una de las posiciones de la cadena. Es importante tener en cuenta que necesitaremos contar con una forma de paso entre la representación del individuo y la solución asociada al mismo. Un ejemplo de las representaciones de los individuos y el paso entre solución y representación podría ser el que se da en el famoso problema de la mochila (*Knapsack Problem, KP*), en el que se busca llenar una mochila con n objetos, de un determinado valor

cada uno, a partir de una lista de m objetos ($n < m$). Una representación habitual para este problema es usar una cadena de m bits para cada individuo, donde la posicion i-ésima indicará si cogemos (1) o no (0) el objeto i-ésimo. He aquí la importancia de la representación en nuestro problema, pues debe dar cabida a todas las soluciones del espacio de búsqueda. También es importante que cada posición de nuestra cadena tenga un significado para la solución del problema, pues así un gen que otorgue una capacidad de adaptabilidad alta sigue dando lugar a características de adaptabilidad alta en las nuevas generaciones. Además, el hecho de que la codificación y descodificación se pueda realizar de manera eficiente cobra gran importancia, pues será un proceso recurrente en la implementación del algoritmo. Cabe destacar que si bien la representación binaria es la más habitual, no significa que sea la única y hay problemas en los que usaremos una distinta, como estudiaremos más adelante. Un ejemplo puede ser el famoso problema del Viajante de Comercio (*Traveling Salesman Problem, TSP*) el cual consiste en hallar la ruta más eficiente entre dos ciudades y formalizándolo se reduce a la búsqueda del camino hamiltoniano menos costoso en un grafo, cuya representación puede ser una permutación de números naturales la cual indica el orden en el que se deben recorrer los nodos, asignándoles a cada uno un valor. También hay otros casos donde pueden ser cadenas de números reales.

La inicialización de la población se realizará de forma aleatoria de tal forma que todos los individuos sean soluciones válidas al problema. En el caso de que la representación sea binaria, se generará una cadena donde la probabilidad de que un gen sea cero o uno es la misma. En general siempre se realizará a través de una distribución uniforme, donde la probabilidad de que un gen tome un valor u otro es la misma. En algunos problemas en los que dispongamos de información sobre los genes, como cuáles ofrecen una mayor adaptabilidad al individuo, podemos favorecer su generación otorgándoles mayor probabilidad que al resto, pero es indispensable dotar a la población de diversidad para el buen funcionamiento del algoritmo. Por otro lado, la correcta ejecución del algoritmo requiere de una serie de parámetros de funcionamiento, entre los que se encuentra el tamaño de la población, parámetros de mutación que veremos más adelante o el número de generaciones que queremos generar.

Una vez hemos decidido todos los componentes que hemos mencionado (la representación, funcion de adaptación...) y tenemos nuestra población inicial, aplicaremos una serie de operadores genéticos a partir de los cuales obtendremos las poblaciones futuras. Estos operadores genéticos son fundamentalmente la selección, el cruce, la mutación y el reemplazo. La selección será el proceso a partir del cual seleccionaremos una serie de individuos que realizarían el papel de padres en la naturaleza. Tras esto se realizará un cruce entre estos individuos seleccionados previamente y obtendremos una nueva generación de individuos, la cual no tiene por que ser del mismo tamaño que la población anterior y tras ello será necesario realizar un reemplazo entre

la generación anterior y los nuevos individuos. Tras el cruce se realiza lo que se conoce como mutación, un operador de carácter aleatorio que dota a los nuevos individuos de posibles mutaciones en sus genes. Más adelante estudiaremos los distintos operadores más profundamente y veremos las distintas herramientas que tienen cada uno.

A continuación presentaremos un esquema general del funcionamiento del algoritmo, con todo lo que acabamos de detallar, a través de pseudocódigo:

fun $algoritmoGenético(tamañoPoblación, parámetros)$:
  $población = generarPoblación(tamañoPoblación, parámetros)$;
  $fitness = evaluarPoblación(población)$;
  while $CondiciónFinalización$:
    $reproducción(población, parámetros)$;
    $fitness = evaluarPoblación(población)$;
  fwhile
ffun

Nótese que la variable *CondiciónFinalización* hace referencia a las condiciones de parada de nuestro algoritmo. Este criterio depende en gran medida del implementador o del proyecto que estemos realizando, puede estar relacionada con el hallazgo de una solución óptima adecuada, con la superación de ciertos límites de cómputo o exceder el tiempo permitido para nuestra búsqueda. Dentro del apartado de *reproducción* se llevará a cabo el proceso de selección, cruce, mutación y reemplazo, que estudiaremos de aquí en adelante.

Este fragmento de pseudocódigo es un esquema general de algoritmos genéticos, el cual está abierto a pequeñas modificaciones que afectan a su funcionamiento.

2.2. Métodos y estrategias de selección.

Comenzaremos con el desarrollo de los distintos métodos de selección. Como se ha explicado, estas técnicas son las encargadas de la selección de individuos que darán lugar a la nueva generación. Estas estrategias deben tender a favorecer la elección de individuos con alta adaptabilidad, con el fin de mantener las características positivas con las que cuentan. Para este apartado nos basaremos también en [Cap] y [Wik22g].

Selección por ruleta.

En este método de selección la probabilidad de elección de un individuo viene dada por la diferencia existente entre su adaptabilidad y la del resto, favoreciendo así a los individuos con mayor adaptabilidad. La probabilidad de seleccionar el individuo i-ésimo es proporcional a su adaptabilidad relativa:

$$p_i = \frac{adaptabilidad(i)}{adaptabilidadMedia}$$

donde la *adaptabilidad(i)* es el valor de la funcion de adaptabilidad del individuo i-ésimo, y la *adaptabilidadMedia* es la media de los valores de adaptabilidad de la población.

El procedimiento a seguir sería generar de forma uniformemente distribuida un número $a \in [0, 1]$ y se escogerá el individuo i que verifique $q_{i-1} < a < q_i$, donde q_i son las *puntuaciones acumuladas*, y se definen tal que:

$$q_0 := 0;$$

$$q_i := p_1 + ... + p_i \quad (\forall i = 1, ..., n);$$

De esta forma se habrá seleccionado un individuo y se repetirá este proceso el mismo número de veces que individuos deseemos.

Visualmente y para entender su funcionamiento, podríamos asimilarlo con una ruleta, teniendo distintos sectores, algunos mayores y otros menores. Se haría girar la ruleta y se escogería el individuo del sector en el que se pare. Tal y como se observa en la figura 2.1, en la que cada sector sería proporcional a la adaptabilidad de cada individuo.

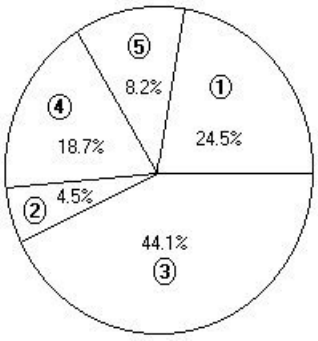


Figura 2.1: Visualización de una ruleta, [Cab].

Muestreo estocástico universal

Este método es similar a la selección por ruleta, con la diferencia que en este caso basta con generarse un único valor a tal que $a \in [0, 1]$ para seleccionar los k individuos que realizarán el papel de padres. A partir de este número generaremos los k números que se necesitan, espaciados de igual forma y que se calculan como sigue:

$$a_j := \frac{a + j - 1}{k} \quad (\forall j = 1, ..., k)$$

Una vez generados estos números el proceso de selección funciona exactamente igual que el de selección por ruleta. Este método es más eficiente que el de la ruleta.

Selección por rango.

Al igual que los dos anteriores, esta estrategia es similar, con la diferencia que funciona mejor cuando los individuos de la población tienen valores de adaptación muy dispares, pues en esos casos la diferencia de adaptabilidad entre los individuos será mayor y la selección por ruleta puede ser un tanto elitista. Además este método se puede aplicar cuando halla valores de adaptabilidad negativos.

En este caso a cada individuo se le otorgará una posición o rango en funcion del valor de adaptabilidad que posea, se asignará el 1 al peor y N al mejor (siendo N el tamaño de la población). La elección de los individuos irá acorde con su rango y no del valor explícito de adaptabilidad. La probabilidad de cada individuo se calcula como sigue:

$$p_i = \frac{r_i}{\sum_i^n r_i}$$

donde r_i es el rango asignado al individuo i-ésimo. Una vez definidas las probabilidades para cada individuo se procedería de manera análoga a la selección por ruleta.

Selección por torneo.

Una de las estrategias más utilizadas y simples de todas ellas. Esta consiste en elegir de forma aleatoria un pequeño grupo de indivuos de la población y quedarnos con aquel que tenga mejor valor de adaptación. Habitualmente este grupo suele constar de 2 o 3 individuos aunque la cantidad es variable. Deberemos repetir este proceso el mismo número de veces que individuos queramos escoger.
Este método de selección se puede realizar de forma determinista o probabilística:

- En el caso determinista se selecciona como hemos explicado, escogiendo el individuo con mayor valor de adaptabilidad dentro de su grupo.
- En el caso probabilístico, en lugar de escoger siempre el mejor individuo, generaremos un número aleatorio entre 0 y 1, si este valor es menor que cierto umbral especificado en los parámetros del algoritmo escogeremos el peor individuo, y en caso contrario, el mejor. De esta forma favorecemos la diversidad dentro de la selección.

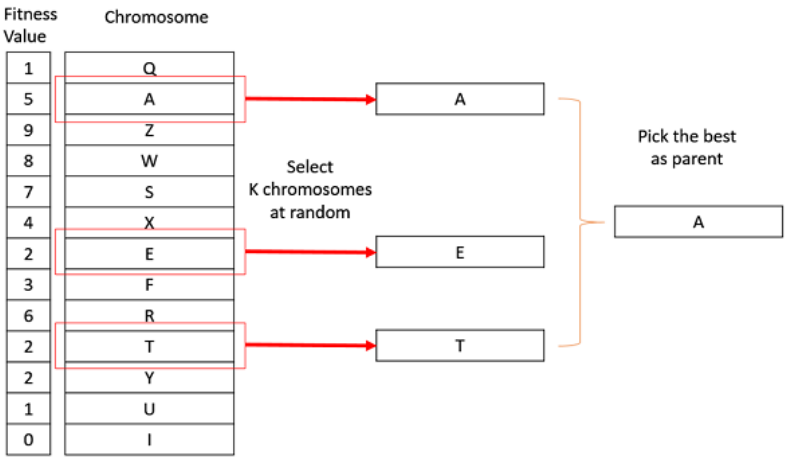


Figura 2.2: Visualización de la selección por torneo [Poia].

Selección aleatoria (RS).

Tal y como su nombre indica, en este método de selección escogeremos los individuos de manera completamente aleatoria. Este método cuenta con la desventaja de que se pierde cualquier control sobre la elección de los individuos y no se favorece de ninguna forma a aquellos individuos que cuentan con mejor adaptabilidad que el resto. Es por ello que no suele considerarse en un plano real su aplicación, pero sí se puede utilizar combinado junto con otros.

Selección elitista.

De igual manera, tal y como su nombre indica, consiste en seleccionar aquellos individuos con mejores valores de adaptabilidad. También cuenta con desventajas, y es que como hemos mencionado a lo largo del trabajo es importante favorecer la diversidad en la población y a través de este método no se favorece. Al igual que la selección aleatoria, se puede utilizar combinada con otros métodos.

Cuando se ha terminado de seleccionar los individuos que harán el papel de padres y que darán lugar a nuevos individuos, se deben agrupar de alguna forma. Habitualmente suele ser por parejas, pero también existen otras posibilidades como agruparlos en tríos, dando lugar a tres nuevos individuos. La forma en la que se elaboran estas parejas o tríos puede ser en el orden que se han ido seleccionando, de forma aleatoria, etc. No existe una única forma ni es muy relevante para la aplicación del algoritmo.

2.3. Métodos y estrategias de cruzamiento.

Una vez hemos llevado a cabo el proceso de selección y tenemos tanto a los individuos como a sus respectivas parejas, se comenzará con la fase que se asocia a la reproducción en la naturaleza. Las estrategias de cruzado serán las encargadas de dar lugar a los nuevos individuos, los cuales, a priori, no se encontrarán en la población anterior, permitiéndonos acceder a nuevas regiones en nuestro espacio de búsqueda. Los métodos que estudiaremos ahora se basan en la recombinación de los genes de los padres.

Para este apartado nos basaremos también en [Wik22c].

Cruce monopunto.

Es el método de cruzamiento más simple en los algoritmos genéticos. Consiste en seleccionar aleatoriamente una única posición en la cadena de ambos padres y recombinarlo juntando la mitad izquierda de un padre con la derecha del otro, análogamente con las mitades derecha e izquierda. Este método de cruzamiento da lugar a dos hijos que combinan propiedades de ambos padres. En la figura 2.3 se habría seleccionado la posición 5 como punto de corte, dando lugar a los dos hijos tal y como sigue.

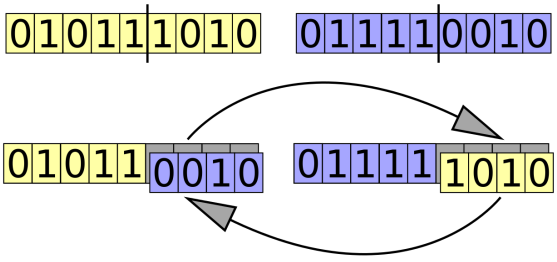


Figura 2.3: Visualización del cruce monopunto, [Wikb]

Cruce en dos puntos.

Este método es análogo al anterior, y se puede generalizar para k puntos de corte. En este caso consideraremos dos puntos de corte para los padres,

los cuales se generarán también de forma aleatoria. En este caso, entre los puntos de corte se van intercalando los genes de un padre y del otro, tal y como se muestra en la imagen.

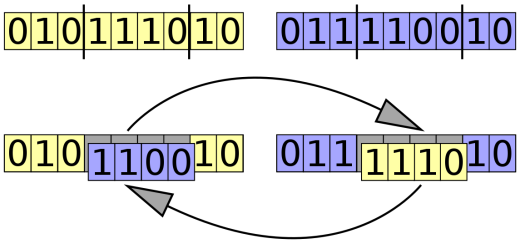


Figura 2.4: Visualización del cruce en dos puntos, [$Wika$].

Cruce uniforme.

Esta estrategia de cruce consiste en otorgar una probabilidad de intercambio de genes entre los padres, establecida generalmente en $\frac{1}{2}$ para que la probabilidad de intercambiar un gen o no sea la misma. Así, si al generar de forma uniformemente distribuida un número $a \in [0, 1]$ verifica ser menor que el umbral establecido, intercambiaremos los genes de los padres, y en caso contrario se quedarán tal y como están. Nótese que en el caso de que algún gen de los padres coincida no resulta efectivo considerarlo pues quedarían igual independientemente del valor de a.

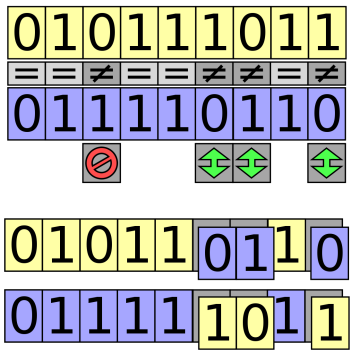


Figura 2.5: Visualización del cruce uniforme, [$Wikc$].

Cruce para cadenas ordenadas.

Hay algunos problemas en los que la recombinación de las cadenas a través de los métodos anteriores puede dar pie a soluciones no válidas para el problema, por ejemplo porque no pertenecen al espacio de búsqueda. En el problema del Viajante de Comercio (TSP), el cual hemos explicado

previamente, las posibles soluciones son los caminos hamiltonianos del grafo, el recombinarlas puede dar pie a cadenas que cuenten con alguna posición repetida y estas representan soluciones que no son caminos hamiltonianos, pues recorreríamos algún vértice más de una vez.

Algunos de los métodos de cruzamiento que resuelven estos problemas serían:

- Cruce de ciclos (CX).
- Operador de cruce basado en la posición (POS).
- Operador de cruce constructivo secuencial (SCX).
- Operador de cruce binario simulado (SBX).

Estos métodos que hemos mencionado no son los únicos y existe una gran familia de estrategias que resuelven este tipo de problemas.

2.4. Técnicas de mutación.

Este paso del algoritmo permite otorgar un carácter aleatorio al proceso de reproducción, pudiendo mutar los genes de la descendencia con independencia de las cadenas de los padres. Este paso es importante para la correcta convergencia del algoritmo. También es importante para asegurarnos la diversidad genética en la población. Este proceso es análogo a la mutación biológica. Para este apartado nos basaremos también en [Poib].

Mutación aleatoria punto a punto.

Es la forma más sencilla de mutación que se puede aplicar en un algoritmo genético y se utiliza generalmente cuando trabajamos con representación binaria. Consiste en cambiar el valor de una posición de la cadena, es decir, cambiar a cero si había un uno y cambiar a uno si había un cero. El hecho de cambiarlo o no suele realizarse partiendo de una tasa de mutación y generando un número aleatorio $a \in [0, 1]$ con una distribución uniforme, que será el que determine si cambiar o no. Si este valor es menor que la tasa de mutación se cambia el valor, y si no, no. Normalmente el valor de la tasa de mutación es pequeño y además podemos trabajar con otras distribuciones como la normal. Recorreríamos toda la cadena y para cada posición realizaríamos el proceso anterior.

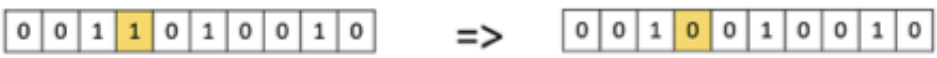


Figura 2.6: Visualización del proceso de mutación punto a punto,[Poib].

Mutación por intercambio.

Esta técnica es también muy simple. Consiste en seleccionar aleatoriamente dos posiciones de la cadena e intercambiar sus valores cuando se verifiquen las condiciones de mutación explicadas previamente.

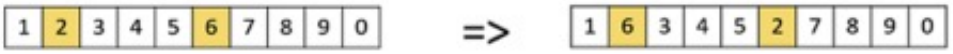


Figura 2.7: Visualización del proceso de mutación por intercambio,[Poib].

Mutación por revuelta.

Este método es utilizado comunmente cuando la representación de las soluciones es una permutación de valores, como puede ser en el caso del problema del Viajante de Comercio (TSP). Se seleccionarán aleatoriamente dos posiciones de la cadena y permutaremos los valores que se encuentren entre ambas, siempre que verifiquemos las condiciones de mutación mencionadas previamente.

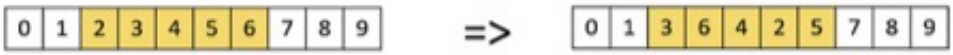


Figura 2.8: Visualización del proceso de mutación por revuelta, [Poib].

Como se puede observar, los operadores que hemos visto son de distinto carácter y cada uno de ellos funcionará de mejor forma dependiendo del problema al que nos enfrentemos. En el caso de la mutación aleatoria punto a punto no es válida para problemas como el del Viajante de Comercio (TSP) pues al aplicarlo podría repetirse algún valor en distintas posiciones y como se ha explicado daría pie a soluciones no válidas para el problema. Para este problema se podría aplicar cualquiera de las otras dos técnicas de mutación.

2.5. Métodos de reemplazo.

Esta última parte de esta sección es la correspondiente a la introducción de los nuevos individuos en la población. Como es de imaginar, hay distintas estrategias posibles para llevarlo a cabo y veremos varias de ellas. Dentro de los algoritmos genéticos, la aplicación de estos métodos se puede agrupar en dos grupos, aquellas que mantienen constante el tamaño de la población y las que la varían.

Reemplazo generacional.

En esta técnica se reemplaza toda la población por la nueva generación, por lo que el tamaño de la misma se mantiene constante. Para ello tenemos que asegurarnos que el número de hijos generados sea del tamaño de la población.

Reemplazo de los padres.

En este caso el tamaño de la población también se mantiene constante y sustituiremos la descendencia por los individuos de la población que hicieron el papel de padres, tal y como su nombre indica.

Reemplazo aleatorio.

Los individuos a eliminar de la población se seleccionarán de forma aleatoria. El número de individuos a eliminar viene definido por el tamaño de la descendencia.

Reemplazo de los peor adaptados.

Los individuos que se eliminarán serán elegidos aleatoriamente, pero los elegiremos de entre aquellos que tengan un valor de adaptación malo. Suelen considerarse aquellos que tienen valores por debajo del 10 % de la adaptación media.

Reemplazo de adaptación similar.

Cada uno de los nuevos individuos se sustituirá por uno que tenga un valor de adaptación similar. Para implementar este método se ordena la población en función del valor de adaptabilidad de los individuos y para la sustitución se escoge aleatoriamente una posición de la "vecindad" de la posición que tendría el nuevo individuo.

Como hemos mencionado al principio, existen métodos en los que el tamaño de la población se mantiene constante y otros en los que la varían. Los que hemos enumerado mantienen el tamaño de la población constante, sin embargo pueden adaptarse para que funcionen variando el tamaño, un ejemplo sencillo sería introducir en la población el 50 % de los nuevos individuos con mejor adaptabilidad, y el otro 50 % hacerlo a través de uno de los métodos anteriores. Es un método común y más adelante usaremos uno similar.

2.6. Pseudocódigo.

Para finalizar este capítulo mostraremos un esquema general, a través de pseudocódigo, del proceso de reproducción dentro de mi algoritmo. Podemos suponer que es un fragmento extraído del código de mi algoritmo:

for generacion **in range(**0,numMaxGen**)**:
    $padres = selección(población, parámetros)$;
    $hijos = cruce(padres, parámetros)$;
    $mutación(hijos, parámetros)$;
    $reemplazo(población, padres, hijos, parámetros)$;
    $fitness = evaluarPoblación(población)$;
ffor

En este caso estaríamos considerando como condición de finalización un número máximo de iteraciones, que será el número de veces que actualicemos la población. Dentro de cada una de las funciones auxiliares se realizará el método o estrategia que especifiquemos. Tras este proceso debemos actualizar los valores del *fitness*, que nos ayudarán a hacernos una idea de los cambios que se están dando en la población y cómo de bien está trabajando el algoritmo.

Capítulo 3

Enunciación de los problemas.

En este capítulo enunciaremos y explicaremos cómo afrontar los distintos problemas basados en situaciones reales. Esta serie de enunciados han sido extraídos de la competición internacional *CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems* [Swa11].

3.1. Estimación de parámetros para ondas sonoras de frecuencia modulada (FM).

La sintetización de ondas sonoras de frecuencia modulada (FM) es muy importante para las plataformas de música actuales y optimizar los parámetros de un sintetizador FM se reduce a optimizar un vector X de seis dimensiones, $X = \{a_1, \omega_1, a_2, \omega_2, a_3, \omega_3\}$. El problema consiste en generar una onda (3.1.1) que sea lo más similar posible a una dada (3.1.2).

$$y(t) = a_1 \cdot sen(\omega_1 \cdot t \cdot \theta + a_2 \cdot sen(\omega_2 \cdot t \cdot \theta + a_3 \cdot sen(\omega_3 \cdot t \cdot \theta))) \quad (3.1.1)$$

$$y_0(t) = 1 \cdot sen(5 \cdot t \cdot \theta - 1{,}5 \cdot sen(4{,}8 \cdot t \cdot \theta + 2 \cdot sen(4{,}9 \cdot t \cdot \theta))) \quad (3.1.2)$$

donde $\theta = \frac{2\pi}{100}$ y los parámetros están definidos dentro de los límites [-6.4, 6.35].

La función a optimizar, que coincide con la función de adaptabilidad, es la suma de los errores al cuadrado, comúnmente usada en los modelos de regresión:

$$f(X) = \sum_{t=0}^{100} (y(t) - y_0(t))^2$$

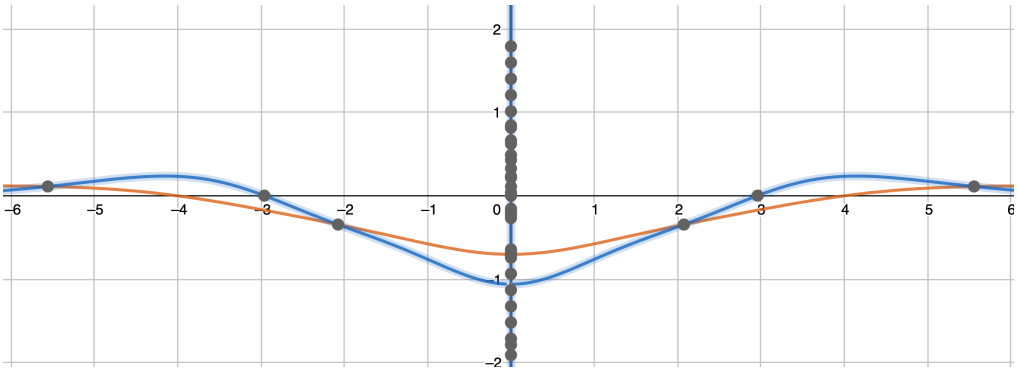


Figura 3.1: Visualización del problema.

La onda azul sería la onda dada $y_0(t)$ y la naranja sería una posible aproximación obtenida a través del algoritmo.

3.2. Problema del potencial de Lennard-Jones.

Es un problema de minimización de la energía potencial. Implica la minimización de la energía potencial molecular asociada al *cluster* de Lennard-Jones. Es un problema de optimización multimodal, compuesto por un número exponencial de mínimos locales. El algoritmo puede probarse estudiando su capacidad para conformar la estructura molecular, donde los átomos se organizan de tal forma que se minimice la energía de la molécula. Supongamos que estamos trabajando con N átomos, que representaremos a través de las coordenadas cartesianas:

$$\vec{p_i} = \{\vec{x}, \vec{y}, \vec{z}\} \tag{3.2.1}$$

El potencial de Lennard-Jones para los N átomos sería:

$$V_N(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (r_{ij}^{-12} - 2r_{ij}^{-6}) \tag{3.2.2}$$

donde $r_{ij} = \|p_i - p_j\|$ es la distancia entre los átomos i-ésimo y j-ésimo. El gradiente del potencial sería:

$$\nabla_j V_N(p) = -12 \sum_{i=1, i\neq j}^{N} (r_{ij}^{-14} - r_{ij}^{-8})(\vec{p_i} - \vec{p_j}), \qquad j = 1, 2..., N \tag{3.2.3}$$

Una forma de reducir la dimensión del problema es semi-fijar la posición de los tres primeros átomos. Podemos fijar la del primero en el origen, situar el segundo en el eje positivo de las X y el tercero sobre la parte superior del eje X. De esta manera, habiendo fijado la posición del primero y dejando

una coordenada a especificar para el segundo y dos para el tercero, tenemos un problema de minimización de tres variables para tres átomos. Por cada átomo que añadamos al problema el número de variables aumentará en tres, debido a las tres coordenadas del mismo.

Sea $\vec{x}$ el vector que contiene las tres variables para el caso de tres átomos, seis para cuatro y así sucesivamente. Definimos los posibles valores de las variables como sigue:

$x_1 \in [0, 4]$, para la coordenada asociada al segundo átomo.
$x_2 \in [0, 4]$ y $x_3 \in [0, \pi]$ para las coordenadas asociadas al tercer átomo.
Los valores que toman el resto de variables x_i se encontrarán dentro de:

$$[-4 - \frac{1}{4} \lfloor \frac{i-4}{3} \rfloor, 4 + \frac{1}{4} \lfloor \frac{i-4}{3} \rfloor]$$

donde $\lfloor r \rfloor$ es el menor entero más próximo a r.

3.3. Diseño del código polifásico de un radar de espectro ensanchado.

Cuando se diseña un sistema de radar que utiliza la compresión de impulsos hay que prestar mucha atención a la correcta elección de la forma de onda. Se conocen muchos métodos de modulación de impulsos que hacen posible su compresión. Los códigos polifásicos son una estrategia ventajosa, ya que permiten reducir los lóbulos laterales de la señal comprimida y facilitar la aplicación de las técnicas de procesamiento digital. Posteriormente, se propuso un nuevo método para la síntesis de códigos de compresión de pulsos polifásicos, que se basa en las propiedades de la función de autocorrelación aperiódica, ϕ_k, y en la suposición de un procesamiento coherente de los pulsos del radar en el receptor.

El problema considerado se modeliza como uno de optimización no lineal de minimización-maximización en variables continuas y con una gran cantidad de óptimos locales. Se puede expresar como sigue:

$$\min_{x \in X} f(x) = max\{\phi_1(x), ..., \phi_{2m}(x)\}$$

donde $X = \{(x_1, ..., x_n) \in \mathbb{R}^n : 0 \leq x_j \leq 2\pi, \ \ j = 1, ..., n\}$ y $m = 2n - 1$. Nótese que en la definición del conjunto X queda definido el rango de valores para las variables. Tanto ϕ_{2i-1} como ϕ_{2i} se definen:

$$\phi_{2i-1} = \sum_{j=i}^{n} cos(\sum_{k=|2i-j-1|+1}^{j} x_k), \ \ i = 1, ...n$$

$$\phi_{2i} = 0{,}5 + \sum_{j=i+1}^{n} cos(\sum_{k=|2i-j|+1}^{j} x_k),\ \ i = 1, ...n-1$$

y se puede demostrar que $\phi_{m+1} = -\phi_i$.

El objetivo es minimizar el módulo del máximo de los valores que toma la función de autocorrelación, que coincidirá con la función de adaptación del algoritmo.

Capítulo 4

Resultados prácticos.

En este capítulo aparecen los resultados obtenidos para los problemas descritos en el capítulo 3. El lenguaje escogido para desarrollar los algoritmos es Python y el entorno Spyder de Anaconda. Los archivos relacionados se encuentran en un repositorio de GitHub [Álv].

Antes de comenzar a estudiar los resultados obtenidos explicaremos brevemente los distintos algoritmos que hemos usado, explicando qué métodos de selección, cruzamiento, mutación y reemplazo hemos empleado. Después pasaremos a estudiar los resultados obtenidos y si existe o no relación entre ellos.

4.1. Algoritmos.

Para estudiar los resultados prácticos nos centraremos en cuatro algoritmos distintos que contarán con algunos elementos en común. Unas características que comparten es la forma de reemplazo y mutación que emplearemos. Para llevar a cabo el proceso de reproducción usaremos toda la población, obteniendo una nueva generación del mismo tamaño, supongamos N. Una vez finalizado este proceso, juntaremos esta nueva generación con el 50 % de los mejores individuos de la generación anterior y tras esto, nos quedaremos con los N mejores individuos, que serán los que conformen la nueva población. Cabe mencionar que el tamaño de la población se mantiene constante con el número de iteraciones. Otra característica común será la mutación, la cual consistirá en la sustitución de un *gen* por el valor del *gen* de uno de los 50 % mejores individuos de la generación anterior, elegido aleatoriamente.

Para los métodos de selección nos centraremos en dos métodos, la selección por ruleta tal y como la hemos explicado, y una selección por torneo en el que el tamaño del grupo se decidirá aleatoriamente. En el caso del

cruzamiento, usaremos un método un tanto distinto que no hemos explicado previamente, que dará lugar a tres hijos, y el método de cruzamiento monopunto. Para poder explicar y comparar los resultados, enumeraremos y nombraremos esquemáticamente a cada uno de los algoritmos:

Algoritmo 1.

- **Selección:** realizaremos una selección por torneo a través de grupos de 2 ó 3 individuos.
- **Cruzamiento:** usaremos el método distinto mencionado previamente, el cual involucrará un valor aleatorio *beta* con distribución gaussiana de media 0.7 y desviación típica 0.1.
- **Mutación y reemplazo:** tal y como se ha explicado.

Algoritmo 2.

- **Selección:** realizaremos una selección por ruleta.
- **Cruzamiento:** usaremos un cruce monopunto.
- **Mutación y reemplazo:** tal y como se ha explicado.

Algoritmo 3.

- **Selección:** realizaremos una selección por ruleta.
- **Cruzamiento:** usaremos el método distinto mencionado previamente, el cual involucrará un valor aleatorio *beta* con distribución gaussiana de media 0.7 y desviación típica 0.1.
- **Mutación y reemplazo:** tal y como se ha explicado.

Algoritmo 4.

- **Selección:** realizaremos una selección por torneo a través de grupos de 2 ó 3 individuos.
- **Cruzamiento:** usaremos un cruce monopunto.
- **Mutación y reemplazo:** tal y como se ha explicado.

Como podemos observar estos cuatro algoritmos nacen de jugar con las distintas combinaciones que nos brindan los métodos de selección y cruzamientos mencionados.

4.2. Resultados prácticos.

Antes de comenzar con el estudio de los resultados obtenidos expliquemos los apartados que se van a estudiar. Se resolverá cada problema de los descritos en el capítulo 3 utilizando los algoritmos mencionados. Para ello se emplearán cuatro configuraciones distintas, basadas en el tamaño de la población y el número de iteraciones. Para cada una de las configuraciones se estudiarán la media, la varianza, el valor óptimo y la distribución de las soluciones obtenidas en 20 ejecuciones de código, contemplando si existe algún hecho remarcable.

Tras esto y para complementar el estudio, se realizará un test de Kruskal-Wallis con el fin de observar si los algoritmos guardasen relación. A nivel formal, este test estadístico sirve para observar si un conjunto de datos provienen de una misma población. En este caso se tendría que la hipótesis nula y alternativa son:

$$H_0 : \text{todas las muestras provienen de la misma población.}$$

$$H_A : \text{al menos una muestra proviene de una población distinta.}$$

Mencionar que se rechazará la hipótesis nula cuando se obtenga un p-valor inferior a 0.05, por lo que se rechazará con un 95 % de seguridad.

Las configuraciones que se emplearán las representaremos a través de tuplas de la forma (*tamaño de la población*, *numero de iteraciones*) y serán las siguientes: $(90, 2000)$, $(450, 2000)$, $(90, 20000)$ y $(150, 10000)$.

4.2.1. Problema 3.1.

Este problema se corresponde con la obtención de una aproximación de una función dada, en este caso la asociada a una onda sonora. Obsérvese en un primer momento, a través del cuadro 4.1, los resultados obtenidos para cada algoritmo, teniendo en cuenta las distintas configuraciones mencionadas.

Algoritmos	Óptimo	Media	Desviación típica
1	6.502398	14.419166	3.448836
2	6.693552	15.843944	3.204954
3	4.662516	15.260061	3.885478
4	5.178939	14.390034	4.073957

Cuadro 4.1: Resultados problema 1.

Teniendo en cuenta los resultados ilustrados se observa que no difieren

en gran medida, hecho que se repetirá en diversas ocasiones. Destacar que el óptimo se obtuvo a través del tercer algoritmo, sin embargo se observa que la media es mayor que dicho valor y teniendo en cuenta que el valor de la desviación típica no es bajo, induce a pensar que los valores obtenidos son dispares entre sí. Mencionar que ocurre lo mismo con el valor obtenido a través del cuarto algoritmo, muy cercano al óptimo.

Este hecho se repite también para los otros dos algoritmos, sin embargo, para el primero y segundo se observa que la desviación típica es algo menor, por lo que en el caso de tener limitaciones en la ejecución del código podrían ser mejores opciones, pues sus resultados difieren algo menos entre sí.

Ahora se estudiarán los algoritmos por separado, teniendo en cuenta las configuraciones empleadas.

Algoritmo 1.

Como se puede observar a través del cuadro 4.2, los óptimos se encuentran muy cercanos unos a otros, encontrándose en un intervalo de longitud 0,5 aproximadamente. Observando la desviación típica se deduce que la primera de las configuraciones obtiene resultados más dispares, suceso que se podía intuir debido al menor tamaño de la población y número de iteraciones. Por otro lado, la mejor media de los valores se obtiene en la tercera configuración y a pesar de no contar con el óptimo de las configuraciones, esta es la más estable de todas, ya que cuenta con la menor desviación típica de todas y un óptimo muy cercano al obtenido en la cuarta configuración.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	6.987282	15.759360	4.193540
(450,2000)	6.618749	15.133986	2.925948
(90,20000)	6.781231	12.988277	2.486486
(150,10000)	6.502398	13.795042	3.483193

Cuadro 4.2: Resultados del algoritmo 1.

También se debe mencionar que a pesar de lo explicado en el parrafo anterior, desde un punto de vista global las distintas soluciones no difieren en exceso, hecho que se reafirma en la figura 4.1, por lo que la elección de una configuración reside en el programador. Esta elección se realizará principalmente en función de las limitaciones con las que se cuente.

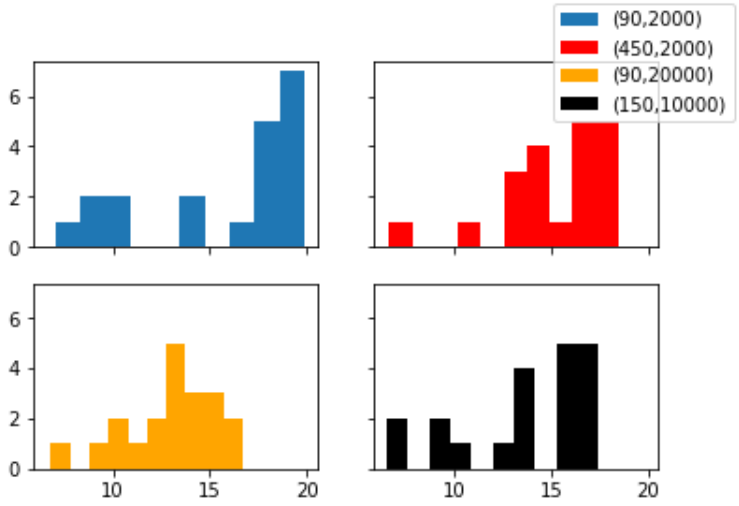


Figura 4.1: Resultados algoritmo 1.

Gracias a estos gráficos se observa lo que se ha descrito cuantitativamente y se advierte que todas las configuraciones obtienen resultados similares. Nótese que la tercera configuración cuenta con una mayor concentración de resultados, reflejando la idea de ser una opción más segura y estable.

Algoritmo 2.

A primera vista, se observa a través del cuadro 4.3 que los resultados obtenidos son peores que aquellos con el primer algoritmo, adquiriendo también peores valores en todas las configuraciones. A nivel local, el óptimo se obtiene a través de la cuarta configuración. Sin embargo es esta configuración la que cuenta con una mayor desviación típica en relación al resto, hecho que puede ser algo negativo. En cuanto a las otras tres configuraciones, los resultados son muy similares entre ellas, planteando el dilema de la eficacia de configuraciones con mayor tamaño de población y número de iteraciones.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	9.831409	17.731834	2.832765
(450,2000)	9.991845	16.408264	2.594485
(90,20000)	10.165547	14.781669	2.413165
(150,10000)	6.693552	14.454007	3.849824

Cuadro 4.3: Resultados del algoritmo 2.

A través de la figura 4.2 se deduce que la primera y cuarta configuración tienen una mayor variación de resultados en comparación a las otras, aunque

esto se debe a algunos *outliers*, como se puede observar en las barras situadas a la izquierda de sus respectivas gráficas, por lo que sin tenerlos en cuenta funcionan de forma similar y contarían con una desviación típica menor.

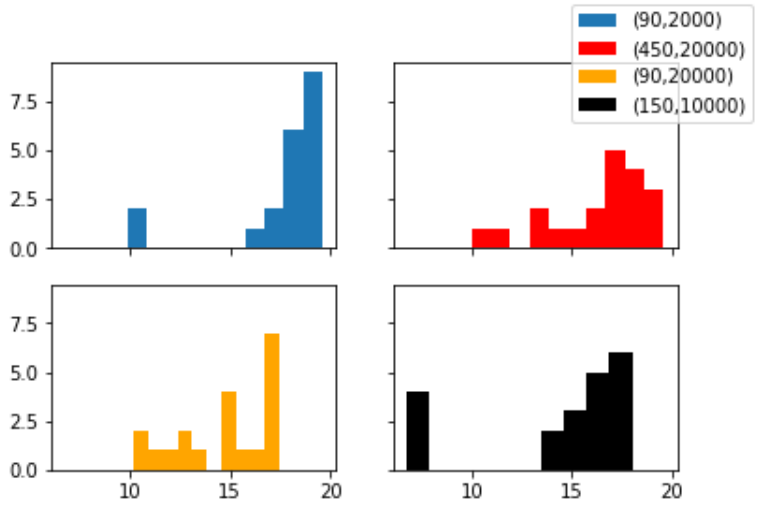


Figura 4.2: Resultados algoritmo 2.

Algoritmo 3.

De los tres algoritmos este es el que muestra unos datos más dispares entre las configuraciones. Como se observa en 4.4, la primera configuración ofrece los peores valores de todas, con un óptimo muy alejado del obtenido entre todos los algoritmos, y una desviación típica muy baja, por lo que las soluciones obtenidas serán malas. Cabe destacar que es a través de este algoritmo que se obtiene la mejor solución al problema, que es la asociada a la tercera configuración. Por otro lado, si nos fijamos, la desviación típica es la mayor de todas, suceso que puede hacernos pensar que dicha configuración es menos estable que el resto.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	12.837653	18.119292	1.994351
(450,2000)	8.734191	15.366473	3.635067
(90,20000)	4.662516	13.031505	4.412587
(150,10000)	6.672470	14.522973	3.419398

Cuadro 4.4: Resultados del algoritmo 3.

A través de la figura 4.3 se pueden observar más claramente los malos resultados obtenidos con la primera configuración, encontrándose todas las

soluciones concentradas en torno a valores lejanos al óptimo. Percíbase que a pesar de haber obtenido la mejor solución con la tercera configuración, existen dos grupos de soluciones separados entre sí, hecho que se refleja en su desviación típica y que puede implicar ser una opción menos estable. En cuanto a las otras dos son similares entre ellas y no tienen nada remarcable que destacar.

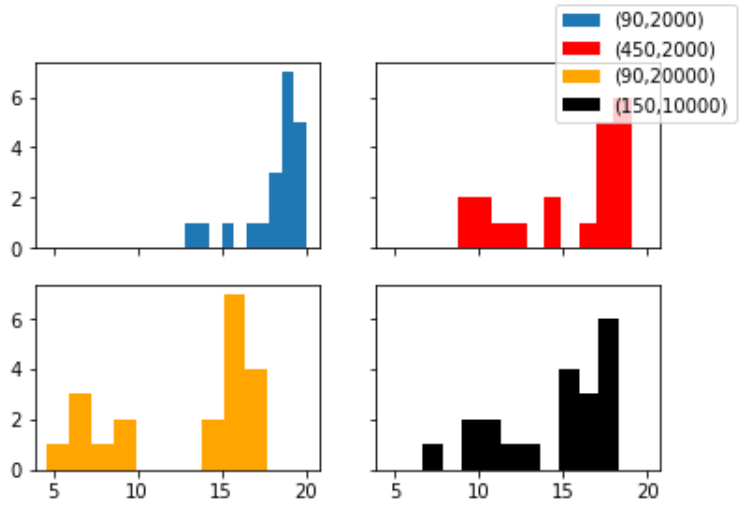


Figura 4.3: Resultados algoritmo 3.

Algoritmo 4.

Al igual que los resultados producidos por el tercer algoritmo, en el cuadro 4.5 se observan unos resultados muy dispares para cada configuración. Se repite que la primera obtiene unos peores resultados, aún más que los del tercer algoritmo, con peor óptimo y menor desviación típica, implicando una concentración de malos resultados. El mejor resultado se obtiene también a través de la tercera configuración, que aun contando con la mayor desviación típica cuenta con la menor media.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	14.688982	18.205934	1.330787
(450,2000)	8.698285	15.185932	3.431027
(90,20000)	5.178939	11.780873	3.854105
(150,10000)	6.905691	12.387398	3.637320

Cuadro 4.5: Resultados del algoritmo 4.

La figura 4.4 representa lo mencionado previamente. Se hace más evi-

dente los malos resultados obtenidos con la primera configuración. Por otro lado a pesar de obtener el mejor resultado gracias a la tercera configuración, los resultados son muy similares para las tres últimas configuraciones, sobre todo si nos fijamos en la distribución de los mismos.

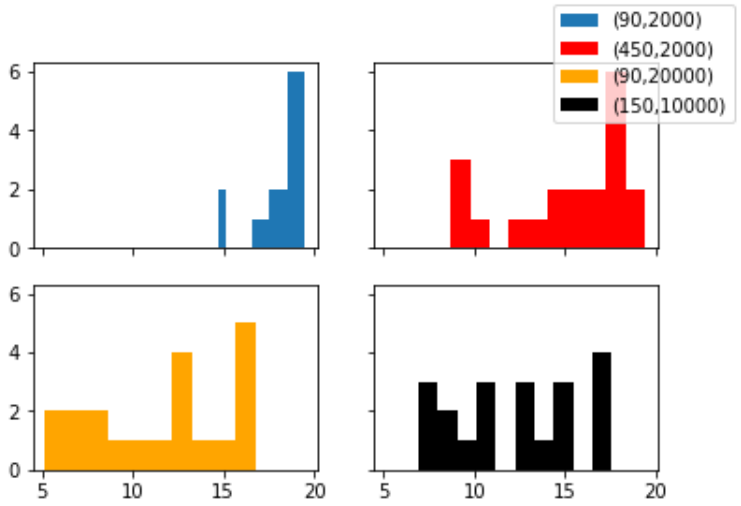


Figura 4.4: Resultados algoritmo 4.

Test de Kruskal-Wallis.

Para finalizar, realizaremos el test de Kruskal-Wallis, con el fin de examinar si entre los cuatro algoritmos existe una similitud o no a través de su distribución de resultados. Para ello se compararán cada una de las configuraciones.

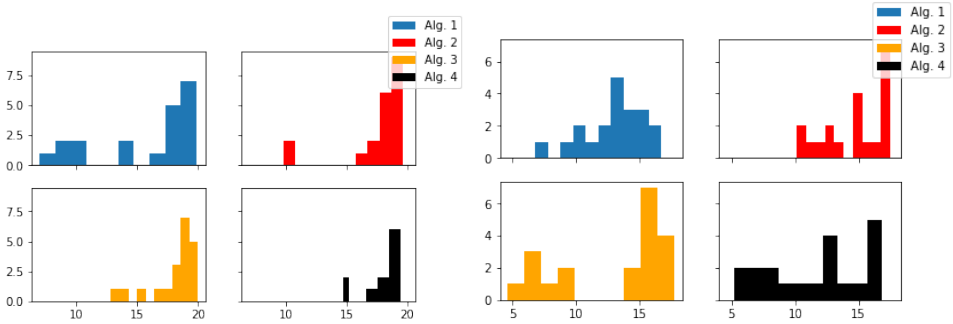
Como observamos en el cuadro 4.6, para las cuatro configuraciones obtenemos un p-valor que supera el 0.05, por lo que no rechazamos la hipótesis nula, afirmando pues que tienen una distribución similar.

Configuración	(90,2000)	(450,2000)	(90,20000)	(150,10000)
p-valor	0.085333	0.435433	0.050881	0.182111

Cuadro 4.6: Resultados del test de Kruskal-Wallis.

En el caso de la primera y tercera configuración también obtenemos un p-valor superior a 0.05, sin embargo estos son mucho más cercanos. En cuanto a la primera configuración, estudiando los cuadros y figuras expuestos previamente, puede deberse a que los resultados obtenidos gracias al primer algoritmo a través de esta configuración son mejores en todos los ámbitos que el resto de algoritmos. Dicha diferencia podemos verla reflejada en la

figura 4.5. Algo similar ocurre con la tercera configuración, cuyo valor probablemente se deba a los resultados obtenidos a través del segundo algoritmo, que como se puede observar en 4.3 son mucho peores que los del resto de algoritmos.



(a) Configuración 1. (b) Configuración 3.

Figura 4.5: Resultados con la primera y tercera configuración.

Para cerciorarse de estas deducciones, realizaremos el test por pares, con el fin de tener una mayor cantidad de datos que complementen el estudio.

Parejas	(90,2000)	(450,2000)	(90,20000)	(150,10000)
Alg.1 - Alg.2	0.065855	0.104588	0.024758	0.303995
Alg.1 - Alg.3	0.023073	0.432774	0.316897	0.330154
Alg.1 - Alg.4	0.054787	0.704909	0.372043	0.255913
Alg.2 - Alg.3	0.704909	0.569999	0.357727	0.978419
Alg.2 - Alg.4	0.626328	0.255913	0.010999	0.078703
Alg.3 - Alg.4	0.448808	0.516207	0.223507	0.069931

Cuadro 4.7: Resultados del test de Kruskal-Wallis por parejas.

Como se observa en el cuadro 4.7 hay muchos valores cercanos a 0.05 y algunos inferiores. Entre los cercanos destacar los que relacionan el primer algoritmo con el resto para la primera configuración, los cuales pueden deberse a la disparidad de resultados entre el primero y el resto, tal y como muestra el gráfico 4.5. Centrémonos en los p-valores inferiores a 0.05, que se corresponden con la pareja “Alg.1 - Alg.3” para la primera configuración y las parejas “Alg.1 - Alg.2” y “Alg.2 - Alg.4” para la tercera configuración. Respecto al de la pareja “Alg.1 - Alg.2” se deduce de las distintas tablas y gráficas, en particular de la 4.5, que se debe a la obtención de mejores resultados a través del primer algoritmo, tal y como hemos mencionado previamente. En lo referido a las parejas de la tercera configuración, probablemente se deba a los malos resultados obtenidos a través del segundo

algoritmo. Si se observa la gráfica 4.5 se ve más claramente, situados todos sus resultados más a la derecha que el resto. Además, tanto para el primero como para el cuarto se observa una mayor uniformidad de resultados, a diferencia del segundo.

En conclusión, se ha observado que en determinadas configuraciones hay algunos algoritmos que, a priori, funcionan mejor o peor que otros. Para la primera configuración el primer algoritmo obtiene mejores resultados, por lo que podría plantearse como mejor opción en caso de usar dicha configuración. Por otro lado, en cuanto a la tercera configuración, se ha observado que no hay uno que funcione mejor que el resto pero sí uno que funciona peor, en este caso el segundo, por lo que en el caso de emplear dicha configuración puede ser mejor opción utilizar cualquier otro.

4.2.2. Problema 3.2.

Este problema se corresponde con la minimización de la energía potencial de Lennard-Jones. En este problema el número de átomos que se involucran es decisión del usuario que ejecuta el código. En nuestro caso se ha decidido trabajar con 12 átomos, lo que implica que el problema conste de 30 variables.

Comencemos con el estudio generalizado teniendo en cuenta cada algoritmo a través del cuadro 4.8. Se obtiene la mejor solución a través del cuarto algoritmo, con relativa diferencia respecto al resto, y cuenta también con una desviación típica pequeña. Sin embargo este hecho no es representativo, pues tanto la desviación típica como la media de los distintos algoritmos toman valores cercanos, hecho que induce a pensar que todos funcionan de forma similar a pesar del óptimo obtenido en el cuarto.

Algoritmos	Óptimo	Media	Desviación típica
1	-6.001881	-4.736985	0.557323
2	-6.140033	-4.757488	0.501382
3	-6.301940	-4.812901	0.560694
4	-7.048872	-4.754299	0.529250

Cuadro 4.8: Resultados del problema 2.

En la figura 4.6 se advierte la similitud que hemos mencionado, sobre todo entre los tres primeros. El cuarto algoritmo, a excepción del *outlier* que se corresponde con el óptimo resultante, cuenta con sus valores concentrados también, aunque en mayor medida que los otros tres. A excepción de este detalle se observa una distribución de resultados similar.

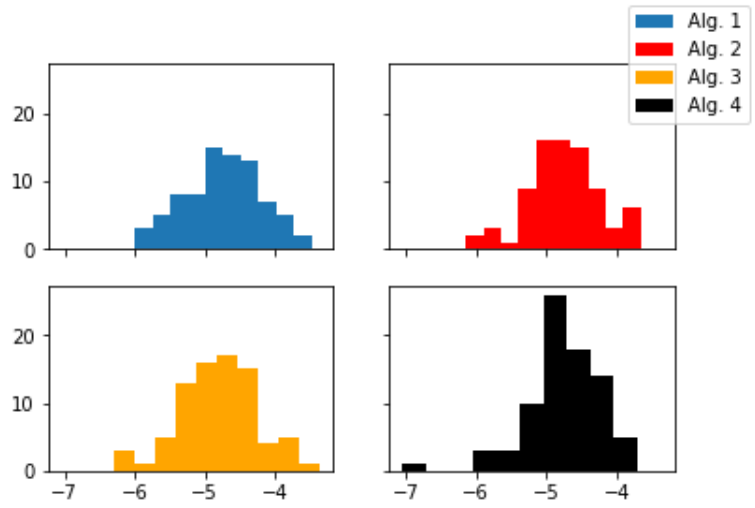


Figura 4.6: Resultados problema 2.

Algoritmo 1.

En la tabla correspondiente al cuadro 4.9 se observa, como era de esperar, unos peores valores para la primera configuración, más remarcable en el óptimo. En las otras tres destacar los resultados correspondientes a la cuarta configuración, que si bien es cierto no cuenta con el mejor óptimo de las cuatro, este es muy cercano a su valor, además de tener el valor medio más alto y una desviación típica pequeña, lo que implica a priori ser una configuración más estable.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	-4.966946	-4.155914	0.358950
(450,2000)	-6.001881	-4.809717	0.587622
(90,20000)	-5.681849	-4.951460	0.428255
(150,10000)	-5.813702	-5.030849	0.365553

Cuadro 4.9: Resultados del algoritmo 1.

Visualmente nótese claramente en la figura 4.7 las conclusiones mencionadas. Se remarca el hecho de que la primera configuración funciona peor que las otras tres, teniendo la distribución de sus valores a la derecha en su gráfica. Destacar la cuarta configuración, que cuenta con una mayor concentración de sus resultados, remarcando el hecho mencionado en el anterior párrafo.

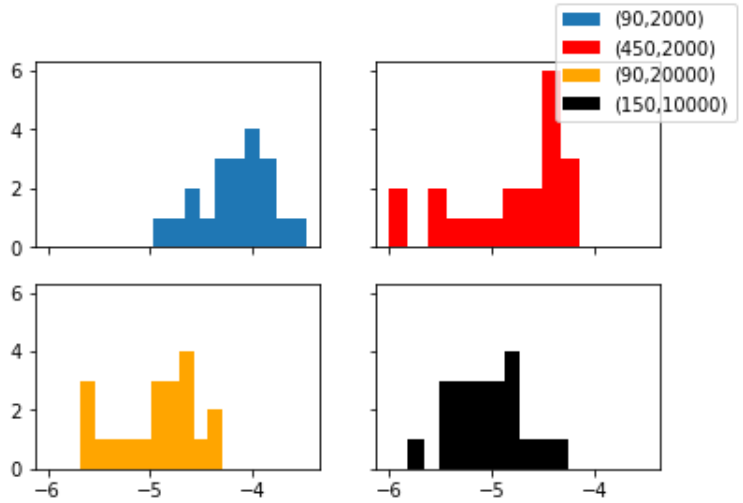


Figura 4.7: Resultados algoritmo 1.

Algoritmo 2.

Los valores obtenidos a través del segundo algoritmo son similares para las cuatro configuraciones, tal y como se representan en el cuadro 4.8. En este caso la tercera configuración obtiene unos mejores resultados, sobre todo en cuanto al óptimo y media se refiere. A pesar de contar con la mayor desviación típica de las cuatro configuraciones, en la figura 4.8 se observa que sus resultados se concentran mayoritariamente en torno al -5 y -6, hecho que se refleja en el valor de su media, por lo que aún contando con la mayor desviación típica, los resultados obtenidos son buenos. Cuantitativamente tanto la segunda como cuarta configuración son similares y la primera algo peor que estas, sin llegar a diferir mucho de ellas.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	-5.147050	-4.272694	0.441718
(450,2000)	-5.200078	-4.739746	0.284865
(90,20000)	-6.140033	-5.139021	0.493505
(150,10000)	-5.501534	-4.878490	0.330228

Cuadro 4.10: Resultados del algoritmo 2.

Un hecho llamativo de estos gráficos en relación a los estudiados previamente para este problema es la distribución de los valores obtenidos, pues se observa una mayor acumulación en determinados puntos, hecho que se refleja a través de las distintas “torres” que se generan, aunque en menor medida para la primera configuración.

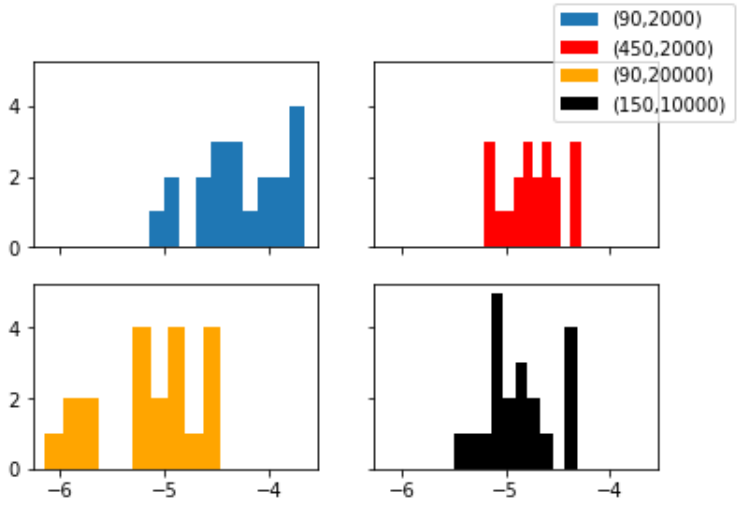


Figura 4.8: Resultados algoritmo 2.

Algoritmo 3.

Gracias al cuadro 4.11 se notifican unos peores resultados para la primera configuración. De entre las otras tres, se observa que los óptimos obtenidos son muy similares entre ellos y donde más difieren son en la media y desviación típica, hecho que se reafirmará visualmente a través de la figura 4.9. De estas tres últimas destacar que la tercera cuenta con unos resultados algo mejores que las otras, sobre todo teniendo en cuenta los puntos que se están estudiando. No cuenta con el mejor óptimo, pero este es muy cercano al mismo, y cuenta con la mejor media y desviación típica, hecho que refleja la estabilidad de sus resultados.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	-5.088567	-4.303950	0.456376
(450,2000)	-6.270376	-4.744191	0.492134
(90,20000)	-6.178416	-5.134302	0.406507
(150,10000)	-6.301940	-5.069162	0.485728

Cuadro 4.11: Resultados del algoritmo 3.

A través del cuadro 4.9 se refleja lo mencionado en el párrafo anterior. Bien es cierto que las tres últimas configuraciones obtienen sus resultados en un rango de valores similar, sobre todo entre la tercera y la cuarta, que visualmente aparentan obtener mejores resultados. Recalcar de nuevo tal y como se muestra en el gráfico la obtención de peores resultados a través de la primera configuración.

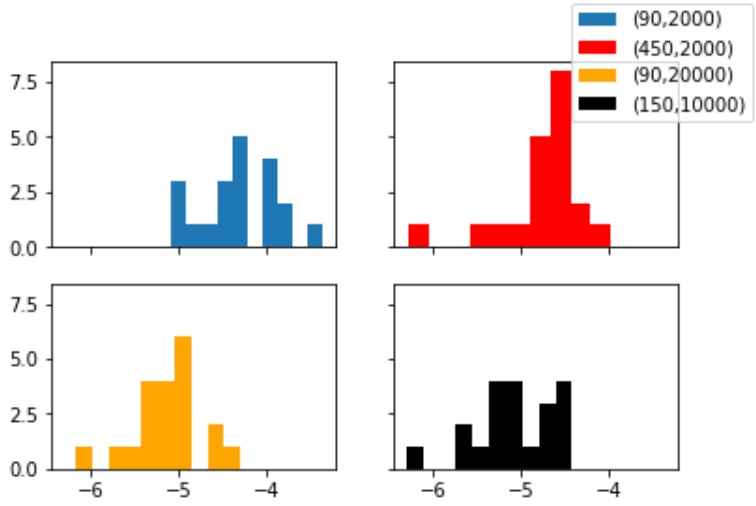


Figura 4.9: Resultados algoritmo 3.

Algoritmo 4.

Cuantitativamente, a través al cuadro 4.12, destacar el valor óptimo conseguido a través de la cuarta configuración, que se corresponde con la mejor solución obtenida para este problema. Sin tener en cuenta este hecho, las cuatro configuraciones tienen valores similares para la media, siendo algo peor para la primera configuración. Por otro lado, la cuarta configuración cuenta con una desviación típica mucho mayor que las otras tres, sin embargo y como se observa en la figura 4.10 esto se debe a algunos *outliers* entre los que se encuentra nuestro óptimo.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	-5.089655	-4.258306	0.380408
(450,2000)	-5.885457	-4.842658	0.397316
(90,20000)	-5.838074	-4.989163	0.397204
(150,10000)	-7.048872	-4.927071	0.589616

Cuadro 4.12: Resultados del algoritmo 4.

Visualmente gracias a los gráficos de 4.10, se puede observar que la distribución de valores es similar entre las tres primeras configuraciones, siendo la primera algo peor. Como se observa, la cuarta configuración otorga resultados muy concentrados si no se tienen en cuenta los *outliers* mencionados previamente, lo que puede hacernos pensar que la diversidad de soluciones es más dificil de obtener que con las otras configuraciones.

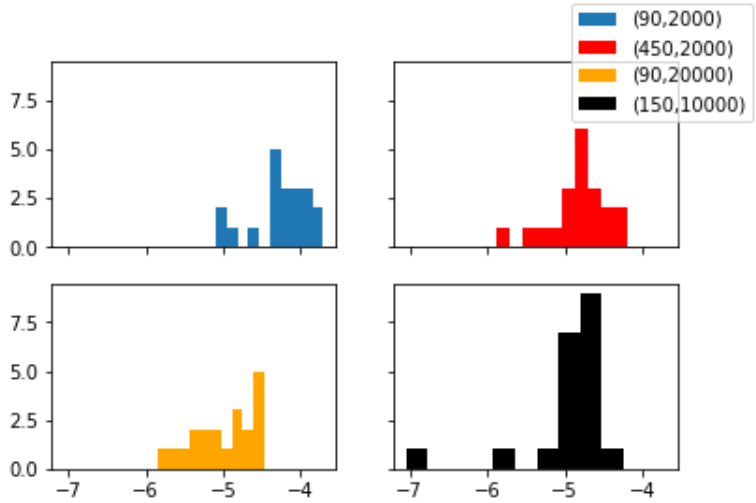


Figura 4.10: Resultados algoritmo 4.

Test de Kruskal-Wallis.

Para este segundo problema nótese que para cada configuración se obtienen p-valores muy superiores a 0.05, por lo que no se rechaza la hipótesis nula. En consecuencia se puede afirmar que cada uno de los algoritmos obtiene unos resultados similares, por lo que la elección del mismo no es relevante.

Configuración	(90,2000)	(450,2000)	(90,20000)	(150,10000)
p-valor	0.730408	0.651396	0.397515	0.20866

Cuadro 4.13: Resultados del test de Kruskal-Wallis.

Para cerciorarse completamente, si se observa el estudio por parejas del cuadro 4.14, nos damos cuenta que se reafirma lo estudiado en el cuadro anterior. Ninguno de los p-valores es inferior a 0.05 por lo que no rechazaríamos la hipótesis nula en ninguno de los casos. Así, no hay ningún algoritmo o algoritmos que contenga diferencias sustanciales en sus resultados respecto al resto.

Percíbase entonces que lo más importante a la hora de resolver este problema reside en la elección de los parámetros. Dicha elección dependerá de las limitaciones con las que se cuente, como se ha ido mencionando en distintas ocasiones.

Parejas	(90,2000)	(450,2000)	(90,20000)	(150,10000)
Alg.1 - Alg.2	0.448808	0.645622	0.255913	0.244767
Alg.1 - Alg.3	0.291446	0.849818	0.123108	0.956855
Alg.1 - Alg.4	0.448808	0.386707	0.849818	0.054787
Alg.2 - Alg.3	0.807656	0.533841	0.725099	0.194148
Alg.2 - Alg.4	0.892413	0.481867	0.386707	0.465174
Alg.3 - Alg.4	0.626328	0.223507	0.255913	0.167723

Cuadro 4.14: Resultados del test de Kruskal-Wallis por parejas.

4.2.3. Problema 3.3.

Este problema se corresponde con un problema de minimización en un conjunto acotado, tal y como se ha enunciado en el capítulo 3. En nuestro caso se ha decidido trabajar en $\mathbb{R}^{20}$, por lo que $n = 20$ y se pueden deducir el resto de parámetros a partir de este.

Observemos a través del cuadro 4.15 los resultados adquiridos. Para cada uno de los algoritmos se obtienen resultados muy similares, obteniendo la mejor solución a través del tercer algoritmo, aunque sin llegar a ser mucho mejor que las otras. Para todos los algoritmos se observa un pequeño valor en la desviación típica, lo que induce a pensar que para todos se obtienen soluciones muy parejas, siendo cada uno de ellos buenas opciones para la resolución del problema. Debido a esto último la mejor opción podría ser, a priori, aquella cuyo coste computacional sea menor.

Algoritmos	Óptimo	Media	Desviación típica
1	1.175817	1.493030	0.115026
2	1.190432	1.478203	0.131811
3	1.078052	1.485353	0.131236
4	1.120792	1.487273	0.142211

Cuadro 4.15: Resultados del problema 1.

Para notificar visualmente la similitud entre los algoritmos observemos las gráficas 4.11, que lo muestran perfectamente. Se puede ver que la distribución de los resultados obtenidos es muy similar para los cuatro algoritmos, coincidiendo en todos ellos que los valores obtenidos se encuentran muy cercanos a la media, pudiendo observar una similitud con la distribución probabilística normal o de Gauss.

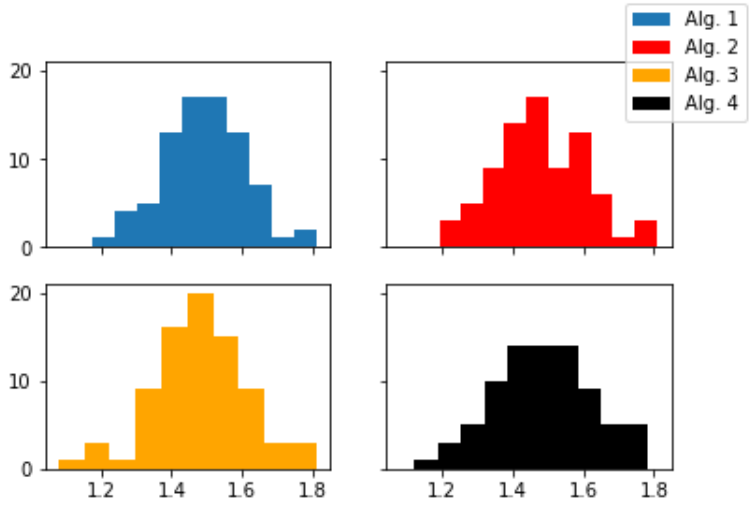


Figura 4.11: Resultados problema 3.

Algoritmo 1.

Como se notifica a través del cuadro 4.16 los resultados para las distintas configuraciones son muy similares, tanto en óptimo, media y desviación típica. El mejor resultado se obtiene a través de la cuarta configuración, contando con el mejor óptimo de los cuatro, a lo que se suma un valor de desviación típica muy bajo, lo que implica una obtención de resultados cercanos al mismo. Este detalle se observa más claramente a través de la figura 4.12. A parte de estos detalles, debido a la similitud entre las distintas configuraciones no hay ningún otro a remarcar.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	1.408215	1.596115	0.100848
(450,2000)	1.246860	1.481860	0.108367
(90,20000)	1.288570	1.462511	0.086203
(150,10000)	1.175817	1.431635	0.096969

Cuadro 4.16: Resultados del algoritmo 1.

A través del gráfico de la figura 4.12 se observa cierta similitud entre las soluciones, obteniendo en todas resultados similares. Sin embargo, se observa mayor similitud en la distribución entre las dos de los gráficos superiores y las dos de los inferiores. Salvando estos parecidos, las soluciones obtenidas son sutilmente mejores para las dos últimas, contando con una mayor concentración de valores y situados más a la izquierda de sus gráficos.

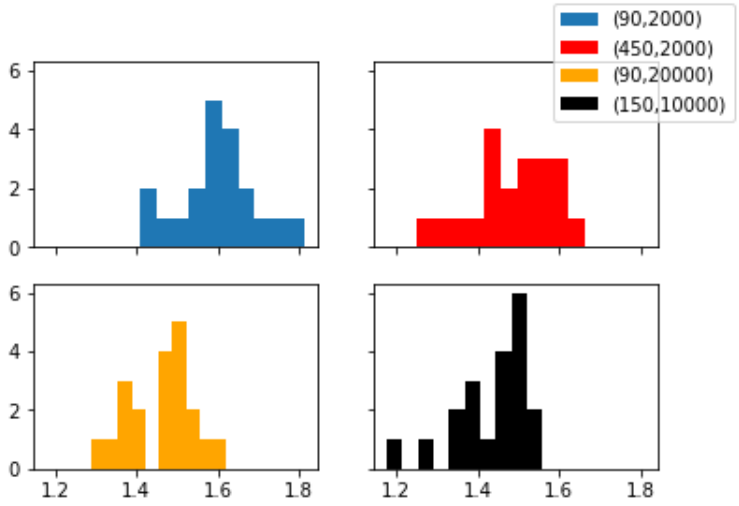


Figura 4.12: Resultados algoritmo 1.

Algoritmo 2.

Tal y como se observa en el cuadro 4.17, a pesar de obtener soluciones similares, la primera configuración otorga la peor de ellas. Cuenta con los peores valores para el óptimo y media, junto con una desviación típica con un valor bajo, lo que hace difícil que los resultados obtenidos a través de esta configuración se acerquen al óptimo que se ha obtenido entre todos los algoritmos o a través de las otras configuraciones.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	1.439961	1.620987	0.093591
(450,2000)	1.190432	1.483345	0.124388
(90,20000)	1.206060	1.382336	0.088044
(150,10000)	1.261000	1.426143	0.079002

Cuadro 4.17: Resultados del algoritmo 2.

En los gráficos 4.13 se puede observar que para las tres últimas configuraciones se obtiene una mayor cantidad de valores cercanos a nuestro óptimo, lo que induce a pensar que con estas tres configuraciones es más probable la obtencion de un mejor resultado. Por otro lado se remarca que la primera configuración obtiene peores valores, alejándose de las mejores soluciones. Aún así el hecho de que la magnitud de los valores no sea alta implica que los resultados obtenidos son cercanos entre sí y el coste computacional puede ser un hecho determinante en la elección de una configuración.

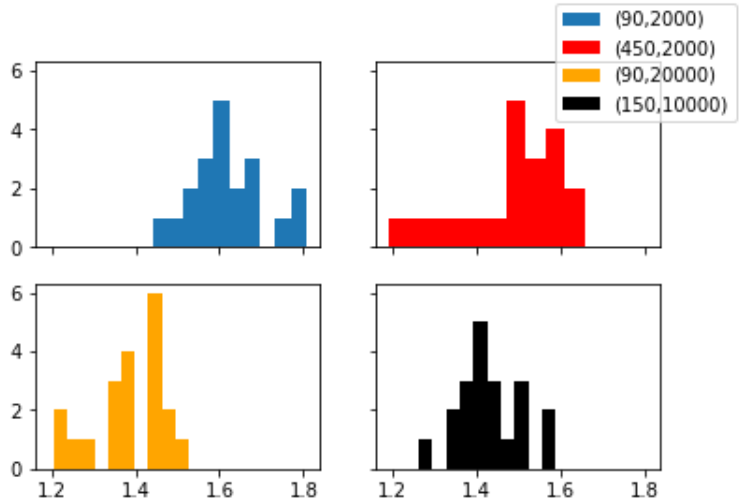


Figura 4.13: Resultados algoritmo 2.

Algoritmo 3.

En este algoritmo se repite lo que venimos estudiando para los dos primeros. Por un lado, se obtienen peores resultados a través de la primera configuración y algo mejores para las otras tres configuraciones. De las otras tres configuraciones se obtiene el mejor a través de la cuarta configuración, sin embargo, y como se observa en el gráfico 4.14, cuenta con una desviación típica algo superior, implicando que sus valores no se concentren cerca del óptimo.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	1.320878	1.621391	0.110862
(450,2000)	1.179901	1.473443	0.097310
(90,20000)	1.210180	1.424565	0.077629
(150,10000)	1.078052	1.422012	0.126817

Cuadro 4.18: Resultados del algoritmo 3.

Podemos observar, y como refleja su desviación típica en el cuadro 4.11, que la tercera configuración es la más estable, obteniendo la mayoria de sus resultados en un rango de valores pequeños y cercanos al óptimo. Sin tener en cuenta la magnitud de los valores esta sería la mejor opción de las cuatro. Salvando este detalle y teniendo en cuenta dicha magnitud, las cuatro configuraciones obtienen resultados similares, y como se ha mencionando a lo largo de este capítulo, un factor diferencial en la elección serán las limitaciones computacionales con las que se cuente.

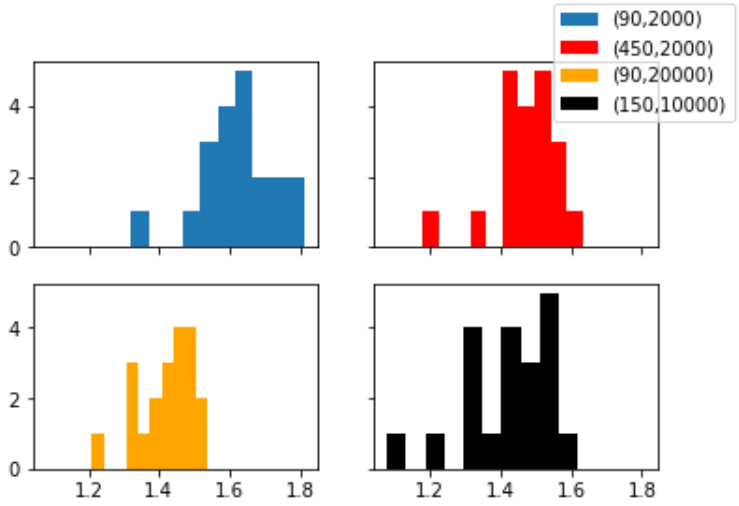


Figura 4.14: Resultados algoritmo 3.

Algoritmo 4.

A través del cuadro 4.19 se observa que se repite la misma tónica que para los otros tres algoritmos, obteniendo resultados similares en magnitud. En este caso tanto la primera como la segunda configuración cuentan con peores resultados en comparación con las otras dos, tanto en el óptimo como en las medias, tomando mayores valores. Mencionar que la mejor solución se ha obtenido gracias a la tercera configuración.

Configuración	Óptimo	Media	Desviación típica
(90,2000)	1.346598	1.643524	0.112447
(450,2000)	1.318342	1.509432	0.081806
(90,20000)	1.120792	1.401729	0.112554
(150,10000)	1.229596	1.394406	0.095326

Cuadro 4.19: Resultados del algoritmo 4.

En el gráfico 4.15 se refleja lo que hemos descrito en el párrafo anterior, las dos primeras configuraciones obtienen valores un tanto peores que las otras dos, como se puede observar encontrándose estas más a la derecha de sus gráficas. Sin embargo, las cuatro configuraciones funcionan de forma similar, obtienendo todas resultados en un rango de valores pequeño.

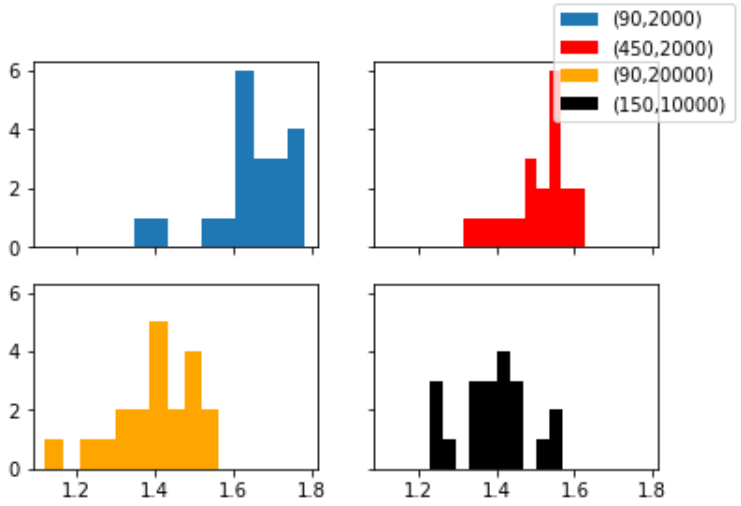


Figura 4.15: Resultados algoritmo 4.

Test de Kruskal-Wallis.

Se observa en el cuadro 4.20 que el p-valor para la tercera configuración es el primero de los tres problemas con el que rechazamos la hipótesis nula. En este caso se aceptaría la hipótesis alternativa, es decir, al menos los resultados de uno de los algoritmos proviene de una población distinta. En nuestro caso esto significaría que los resultados obtenidos por al menos uno de ellos es distinto del resto.

Configuración	(90,2000)	(450,2000)	(90,20000)	(150,10000)
p-valor	0.250962	0.634343	0.048796	0.487152

Cuadro 4.20: Resultados del test de Kruskal-Wallis.

El hecho de aceptar la hipótesis alternativa para este caso no significa que uno de ellos funcione mejor para este problema, puede ser todo lo contrario, o que haya dos de ellos que funcionen de forma distinta respecto a los otros dos, etc.

Observando la figura 4.16 no se percibe un patrón que nos indique qué algoritmo o algoritmos difieren en su distribución, trataremos de analizar a través del estudio por parejas a qué se debe dicho valor. A priori, podría deberse a cierta similitud en la distribución de valores entre los dos últimos algoritmos, sin llegar a ser algo determinante.

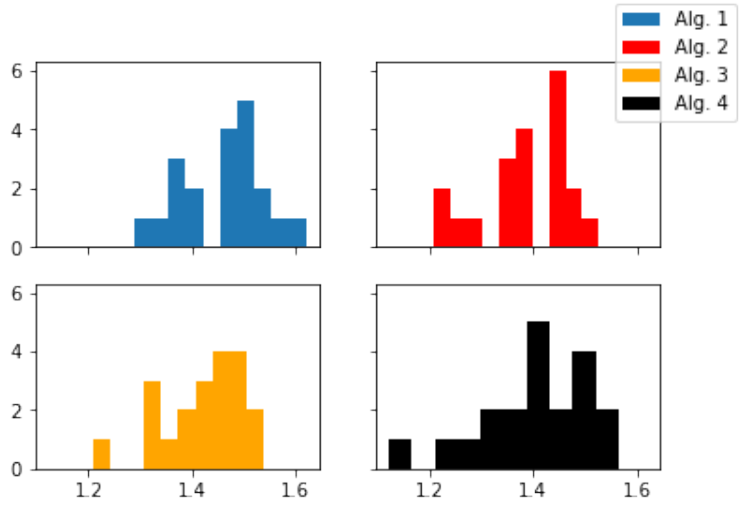


Figura 4.16: Resultados de la configuración (90,20000).

Como hemos mencionado se estudiará el test realizado por parejas. Como se observa, solo hay dos valores para los que rechazamos la hipótesis nula, que se corresponden con el de la pareja "Alg.1 - Alg.4" para la primera configuración y el del "Alg.1 - Alg.2" para la tercera. Bajo estos resultados se debe rechazar la hipótesis nula, en mayor medida para el segundo caso, afirmando entonces que no provienen de la misma población y en consecuencia obtienen resultados distintos.

Parejas	(90,2000)	(450,2000)	(90,20000)	(150,10000)
Alg.1 - Alg.2	0.481867	0.849818	0.007407	0.569999
Alg.1 - Alg.3	0.417077	0.645622	0.136815	0.892413
Alg.1 - Alg.4	0.048307	0.401720	0.093521	0.136815
Alg.2 - Alg.3	0.704909	0.533841	0.116670	0.7454830
Alg.2 - Alg.4	0.185020	0.626328	0.386707	0.330154
Alg.3 - Alg.4	0.279250	0.176213	0.645622	0.267407

Cuadro 4.21: Resultados del test de Kruskal-Wallis por parejas.

En lo que al primer valor se refiere, este es muy cercano a 0.05 y si se observa la figura 4.17 se puede intuir que la diferencia entre algoritmos no es muy relevante en cuanto a la valía de los resultados, a pesar de sí serlo en su distribución. En el estudio del cuadro 4.20 se había obtenido un p-valor inferior a 0.05 para la tercera configuración, este puede deberse a la diferencia en la distribución entre los dos primeros algoritmos o a su diferencia respecto de los otros dos. Sin embargo, a pesar de dicho valor, no se observa a través de la figura 4.16 ningún hecho relevante que afirme que

uno de los algoritmos funciona mejor que el otro.

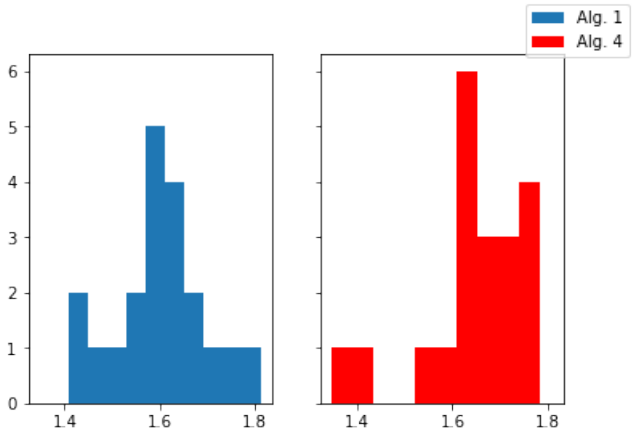


Figura 4.17: Resultados de la configuración 1.

Así, para este problema particular, no se han hallado pruebas suficientes que indiquen que un algoritmo obtenga soluciones sustancialmente mejores que el resto, por lo que no existe una mejor elección a pesar de los p-valores obtenidos. La elección de un algoritmo u otro dependerá del mismo programador.

4.3. Conclusión.

Como hemos mencionado durante este trabajo, nos gustaría saber si existe un algoritmo que funcione notablemente mejor que el resto, o si para algún problema existiese uno con mejores resultados que el resto. De igual forma con las distintas configuraciones existentes.

Es una realidad que siempre será uno de los algoritmos el asociado a la mejor solución, sin embargo no significa que dicho algoritmo funcione mejor. En nuestro caso, y como se ha ido reflejando en los distintos cuadros y figuras, los resultados que se han ido obteniendo entre los algoritmos son muy similares, sin llegar a existir uno que destacase por encima del resto. Este hecho se reafirma a través de los distintos tests de Kruskal-Wallis que se han realizado, pues a excepción de algún p-valor obtenido por parejas para determinadas configuraciones, los cuales podían deberse a un mejor o peor funcionamiento de los algoritmos a nivel local, a nivel general no era significativo. Esto puede deberse a que los cuatro algoritmos cuentan con elementos en común tal y como se ha explicado al comienzo del capítulo 4.1, sin embargo ha quedado demostrado que partiendo de los métodos de selección y cruzamiento expuestos, no hay ninguno que resalte por encima

de los demás.

Por otro lado, también debemos prestar atención a las distintas configuraciones de parámetros con las que se ha trabajado. En este caso, sí que existen casos con significantes mejorías en función de la configuración. Un hecho que se ha ido repitiendo a lo largo de la sección 4.2 es que la configuración (90,2000) obtenía peores resultados que el resto, debido a su menor tamaño de población y número de iteraciones. Sin embargo, entre las otras tres no existen sucesos remarcables que destaquen una configuración por encima del resto, pues en algunos problemas y dependiendo del algoritmo, se obtenían mejores soluciones con unas que con otras, pero sin llegar a existir un patrón visible. Sí que se debe remarcar la diferencia de soluciones entre la configuración (90,2000) y el resto de ellas, siendo notablemente mejores en las tres últimas (como se ejemplifica en la figura 4.18), sin embargo también se debe mencionar que el coste computacional de ejecutar nuestro algoritmo con ellas es mucho mayor, y en algunos casos no se llegan a obtener mejorías significativas, como en el caso del primer algoritmo para el primer problema 4.2.1.

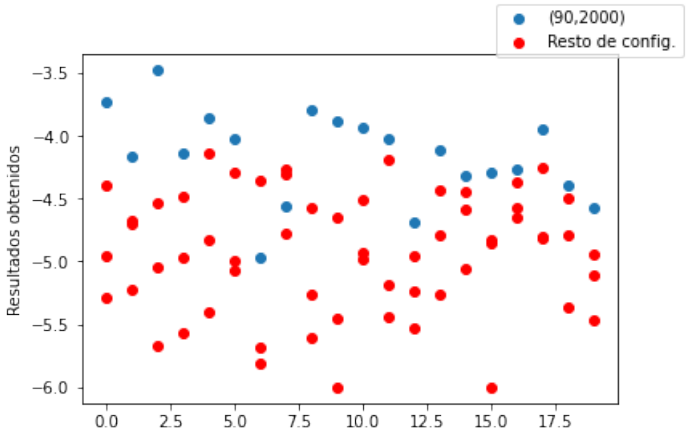


Figura 4.18: Resultados del problema 3.2 con el primer algoritmo.

Para concluir, reafirmar que no existe un algoritmo que funcione mejor que los demás, pues todos ofrecen buenas soluciones a los distintos problemas. En cambio con las configuraciones sí que existe una mejoría de las tres últimas respecto el resto. Es por ello que el uso de un algoritmo u otro no es un factor determinante a la hora de la resolución de los problemas, pero sí lo será la elección de la configuración. Como se ha ido mencionando a lo largo del capítulo, dicha elección será decisión del programador, y puede ser decidida en función de las limitaciones computacionales, los requisitos impuestos para los resultados, etc.

Bibliografía

[Álv] Guzmán Silva Álvarez. *Estudio de los algoritmos genéticos.* https://github.com/guuzsa/Estudio-de-Algoritmos-Gen-ticos.

[Cab] Elena Cabrera-Revuelta. *Selección por ruleta.* https://www.researchgate.net/figure/Figura-419-Seleccion-por-Ruleta-Ramirez-2008_fig55_319939606m.

[Cap] Fernando Sancho Caparrini. *Algoritmos genéticos.* http://www.cs.us.es/~fsancho/?e=65#:~:text=M$%$C3$%$A9todos$%$20de$%$20selecci$%$C3$%$B3n&text=(La$%$20mayor$%$C3$%$ADa$%$20de$%$20los$%$20AGs,que$%$20no$%$20surja$%$20nada$%$20mejor).

[Lou09] Lourdes Araujo, Carlos Cervigón. *Algoritmos evolutivos: un enfoque práctico.* 2.ª ed. RA-MA, 2009.

[Poia] Tutorials Point. *Genetic Algorithm Parent Selection.* https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm.

[Poib] Tutorials Point. *Genetic Algotihms.* https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm.

[Qar] Mouad Qarchli. *Genetic Algorithms.* https://qarchli.github.io/2020-11-15-genetic-algorithms/.

[Swa11] P.N.Suganthan Swagatam Das. "Competition on Testing Evolutionary Algorithms on Real World Optimization Problems". En: *IEEE Computational Intelligence Society* (2011).

[Wika] Wikipedia. *Cruzamiento en dos puntos.* https://es.m.wikipedia.org/wiki/Archivo:Computational.science.Genetic.algorithm.Crossover.Two.Point.svg.

[Wikb] Wikipedia. *Cruzamiento monopunto.* https://es.m.wikipedia.org/wiki/Archivo:Computational.science.Genetic.algorithm.Crossover.One.Point.svg.

[Wikc] Wikipedia. *Cruzamiento uniforme*. https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.Uniform.Crossover.svg.

[Wik22a] Wikipedia. *Algoritmo de la colonia de hormigas — Wikipedia, La enciclopedia libre*. [Internet; descargado 10-junio-2022]. 2022. URL: %5Curl%7Bhttps://es.wikipedia.org/w/index.php?title=Algoritmo_de_la_colonia_de_hormigas&oldid=142279759%7D.

[Wik22b] Wikipedia. *Optimización por enjambre de partículas — Wikipedia, La enciclopedia libre*. [Internet; descargado 10-junio-2022]. 2022. URL: %5Curl%7Bhttps://es.wikipedia.org/w/index.php?title=Optimizaci%C3%B3n_por_enjambre_de_part%C3%ADculas&oldid=141846308%7D.

[Wik22c] Wikipedia contributors. *Crossover (genetic algorithm) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Crossover_(genetic_algorithm)&oldid=1085992909. [Online; accessed 10-June-2022]. 2022.

[Wik22d] Wikipedia contributors. *Evolutionary algorithm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Evolutionary_algorithm&oldid=1090026011. [Online; accessed 10-June-2022]. 2022.

[Wik22e] Wikipedia contributors. *Evolutionary computation — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Evolutionary_computation&oldid=1090828814. [Online; accessed 10-June-2022]. 2022.

[Wik22f] Wikipedia contributors. *Genetic algorithm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=1090025948. [Online; accessed 10-June-2022]. 2022.

[Wik22g] Wikipedia contributors. *Selection (genetic algorithm) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Selection_(genetic_algorithm)&oldid=1069663018. [Online; accessed 10-June-2022]. 2022.