

8 de mayo de 2023

Pablo Medina de la Iglesia, Javier Garrido cobo, Miguel ángel lobo bartolomé y Diego viñals lage

Inteligencia artificial 2

Laboratorio 04

Implementación de un Algoritmo Genético

Índice

[Descripción del problema a resolver 3](#_Toc134467807)

[Resultados de la Práctica 1 4](#_Toc134467808)

[Cuestión 1 4](#_Toc134467809)

[Fitness 4](#_Toc134467810)

[Probabilidad de selección (Ps) 5](#_Toc134467811)

[Replicación/Mutación 6](#_Toc134467812)

[¿Hay algún valor máximo que puede alcanzar el fitness? ¿Se puede saber cuál es? 11](#_Toc134467813)

[¿Podría ser negativo el fitness máximo? ¿Por qué? 13](#_Toc134467814)

[¿Puedes mejorar el cálculo del fitness? Explícalo y justifica tu elección final. 13](#_Toc134467815)

[¿Qué alternativas al proceso de mutación podrías plantear? Explícalo y justifica tu elección final. 14](#_Toc134467816)

[Cuestión 3 16](#_Toc134467817)

[Cuestión 4 22](#_Toc134467818)

[Cuestión 5 25](#_Toc134467819)

[Cuestión 6 28](#_Toc134467820)

[Generación 300: 33](#_Toc134467821)

[Generación 400: 34](#_Toc134467822)

[Generación 500: 34](#_Toc134467823)

[Generación 600: 34](#_Toc134467824)

[Generación 700: 35](#_Toc134467825)

[Generación 800: 35](#_Toc134467826)

[Conclusión 35](#_Toc134467827)

[Bibliografía 37](#_Toc134467828)

[Ilustración 1 - Función de fitness (Fi) 4](#_Toc134467829)

[Ilustración 2 - Función en Python que calcula el Fi 5](#_Toc134467830)

[Ilustración 3 - Función de la Probabilidad de Selección (Ps) 5](#_Toc134467831)

[Ilustración 4 - Función que calcula el Ps en Python 5](#_Toc134467832)

[Ilustración 5 - Función de la Probabilidad de Selección Acumulada (Psa) 6](#_Toc134467833)

[Ilustración 6 - Ejemplo de Intercambiar-Estación 7](#_Toc134467834)

[Ilustración 7 - Función relacionada con la replicación/mutación del cromosoma. 8](#_Toc134467835)

[Ilustración 8 - Función Ruleta 8](#_Toc134467836)

[Ilustración 9 - Función Selecciona Cromosoma 9](#_Toc134467837)

[Ilustración 10 - Tabla padres 9](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467838)

[Ilustración 11- Sumatorio poblaciones 12](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467839)

[Ilustración 12 - Fitness máximo 12](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467840)

[Ilustración 13- Fitness máximo negativo 13](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467841)

[Ilustración 14. Configuración pruebas 16](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467842)

[Ilustración 15. Grafica con E cercano a 0 16](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467843)

[Ilustración 16- Mejor Fitness E cercano a 0 17](#_Toc134467844)

[Ilustración 17- Evolución best n 18](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467845)

[Ilustración 18- Evolucion BEST% 18](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467846)

[Ilustración 19- Muestreo generación E menor 19](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467847)

[Ilustración 20- Fitness/Media E mayor 20](#_Toc134467848)

[Ilustración 21- Evolucion mayor fitness E mayor 20](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467849)

[Ilustración 22- Best% en E mayor 21](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467850)

[Ilustración 23- Bestn E mayor 21](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467851)

[Ilustración 24- Resumen generación 50 E mayor 22](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467852)

[Ilustración 25- Comparativa 3 24](file:///C:\Users\javit\Documents\UFV\IAII\LAB04-GRUPOB3\LAB04-GRUPOB3.docx#_Toc134467853)

# Descripción del problema a resolver

Se quiere construir una nueva línea de tren ligero que pueda cubrir áreas que no están siendo atendidas por la red actual de trenes. Para llevar a cabo este proyecto, se han preseleccionado un **conjunto de lugares (L)** donde se podrían construir estaciones de metro y han calculado el **número potencial de usuarios (P)** que usarían cada una de ellas en un día promedio.

Sin embargo, la línea de metro sólo puede tener un **número determinado de estaciones (E)**, que debe ser menor al número total de lugares preseleccionados (E < L).

El **objetivo** es determinar cuáles de estos lugares (E lugares) se deben seleccionar para la construcción de las estaciones y en qué orden deben ser conectadas, de manera que se logre atender al mayor número de personas posible, y al mismo tiempo, se mantenga el recorrido total de la línea lo más corto posible.

Disponemos de los datos de las distancias entre las estaciones y los datos de población en un fichero .xlsx.

Debemos crear un algoritmo genético que funcione de manera autónoma para mejorar la solución a un problema o función específicos. El algoritmo trabajará con una población de individuos generados aleatoriamente y utilizará un mecanismo evolutivo, que incluirá selección y operadores genéticos.

# Resultados de la Práctica 1

## Cuestión 1

En la primera cuestión se nos pide explicar detalladamente el código que implementa las funciones de cálculo del **Fitness (Fi)**, la **probabilidad de selección (Ps)** y **replicación/mutación** de un individuo.

### Fitness

En primer lugar, **la función de fitness** para este problema es la siguiente:

Diagrama

Descripción generada automáticamente con confianza baja

Ilustración - Función de fitness (Fi)

En esta función encontramos distintas variables que hay que definir:

* **dist** es la distancia entre dos estaciones (**E**) consecutivas según el orden que indica el cromosoma (para n = E es la distancia respecto a la estación de origen).
* **𝛼** es un valor entre 0 y 1 que permite dar más o menos importancia a cada criterio.

A partir de los datos de distancias y población creamos dos dataframes: **df\_distancias** y **df\_población** gracias a la librería pandas de Python.

La función recibe ambos dataframes junto con un cromosoma (del cual calcularemos el fitness) y el valor “**𝛼**” definido anteriormente.

En la variable **sum\_p** guardamos el sumatorio de las poblaciones de las estaciones por donde el cromosoma (chromosome en la función).

En la variable **sum\_dist** se almacena el sumatorio de las distancias entre dos estaciones contiguas:

* Si la estación no es la última, se suman las distancias de la estación actual con la estación próxima (dist[chromosome[i]][chromosome[0]]).
* Si la estación es la última, se suman las distancias de la estación actual con la estación inicial (dist[chromosome[i]][chromosome[0]]).

Texto

Descripción generada automáticamente

Ilustración - Función en Python que calcula el Fi

Multiplicamos el valor de **𝛼** por **sum\_p** y restamos el producto de **(1- 𝛼)** y **sum\_dist**, almacenamos el resultado en la variable **fitness**, que es la variable que se devuelve en la función y corresponde con el **fitness** del **cromosoma** que utilizamos.

### Probabilidad de selección (Ps)

En segundo lugar, el cálculo de la Probabilidad de Selección viene descrito por la siguiente fórmula:

Imagen que contiene Esquemático

Descripción generada automáticamente

Ilustración - Función de la Probabilidad de Selección (Ps)

Texto

Descripción generada automáticamente

Ilustración - Función que calcula el Ps en Python

En la variable **suma\_fitness\_mejores** almacenamos la suma de los fitness de todos los cromosomas de la población o grupo de cromosomas de los cuales queramos sacar el fitness.

Esa variable actuará como divisor del fitness de cada uno de los individuos de la población (dividendo). El resultado de esa división la almacenamos en la variable final **ps** la cual se añadirá a la tabla que se nos pide en el enunciado.

La variable ps nos dará entonces la representatividad del fitness de cada padre dentro del fitness total de todos los padres, esta probabilidad será usada posteriormente en el método de la ruleta.

En la variable **suma\_ps** almacenaremos la Probabilidad de Selección Acumulada, la cual viene definida por la siguiente función:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración - Función de la Probabilidad de Selección Acumulada (Psa)

La Probabilidad de Selección (Ps) la obtenemos tanto cuando queremos hallar el 20% de los padres que tienen el mejor fitness en la función: def ind\_mejores(), como cuando queremos hallar el 80% de los padres con el fitness más bajo en la función: def ind\_peores().

### Replicación/Mutación

En tercer lugar, encontramos la replicación/mutación de los cromosomas. Consiste en la construcción de un nuevo hijo a partir del padre (seleccionado por el método de la ruleta).

En el método de la ruleta, se genera un valor aleatorio entre 0 y 1. Se recorre uno a uno cada padre seleccionado, si el valor aleatorio es menor o igual al ps del cromosoma entonces el cromosoma padre entrará a formar parte del proceso de mutación.

Como ps representa la relevancia de cada padre dentro del conjunto de padres, cuanta mayor relevancia, mayor será la probabilidad de sufrir alguna mutación. Dentro del proceso de mutación podemos distinguir entre tres posibilidades:

1. Cambia-estación, asociada a **Pc** (Probabilidad de Cambiar la Estación), implica que una estación (gen) pueda ser cambiada por otra que no se encuentre actualmente en el cromosoma.
2. Intercambiar-estación, asociada a **Pi** (Probabilidad de Intercambiar la Estación), que implica que una estación (gen) se intercambie con la estación próxima:



Ilustración - Ejemplo de Intercambiar-Estación

1. Nada, asociada a **Pnada**, no hay ninguna implementación puesto que no habría modificación en el gen correspondiente.

Estas tres subfunciones las encontramos en el código en la función **intercambiar\_cambiar\_nada**:

Texto

Descripción generada automáticamente

Ilustración - Función relacionada con la replicación/mutación del cromosoma.

La función recibe **tabla\_padres**, la cual contiene los padres a partir de los cuales se construirán los hijos. De esa tabla hacemos una copia (**tabla\_copia**) para no perder los datos de la tabla original cuando estos se modifiquen. Además, la función recibe **L**, **E**, **Pc**, **Pi** y **Pnada**, que se utilizarán en la propia función.

En primer lugar, seleccionamos el padre que se va a replicar, con el método de la ruleta (ver Ilustración 9). Después recorreremos el cromosoma, gen a gen, decidiendo si este cambia, intercambia o no hace nada. Esto viene determinado por la función **ruleta(pc, pi, pnada)**, la cual, a partir del valor de Pc, Pi y Pnada del cromosoma en cuestión, decidirá si el gen entra en la opción pc (ejecuta el cambio-estación), entra en la opción pi (ejecuta el intercambio-estación) o no entra en ninguna de las tres opciones (pnada) en este caso el gen seguirá siendo el mismo.

Texto

Descripción generada automáticamente

Ilustración - Función Ruleta

Los valores que le damos a Pc y Pi determinan cuanto cambia el cromosoma respecto al padre, por lo tanto, cuanta más probabilidad les otorguemos, el nuevo cromosoma se parecerá menos al padre y viceversa.

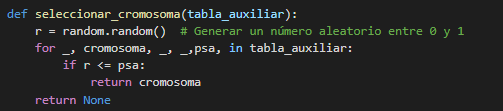


Ilustración - Función Selecciona Cromosoma

Para cada generación, generamos la siguiente tabla para los padres seleccionados:

Texto

Descripción generada automáticamente

Ilustración - Tabla padres

En la columna número se observan los cromosomas que han sido seleccionados como padre dentro de la población, en la columna Fi tenemos el fitness calculado para el cromosoma, en ps obtenemos la relevancia del fitness de cada cromosoma respecto al total. En Psai se obtienen los valores acumulados de ps, el último valor debe ser siempre uno.Cuestión 2

### ¿Hay algún valor máximo que puede alcanzar el fitness? ¿Se puede saber cuál es?

Podemos analizar las posibles situaciones según la fórmula fitness planteada en la práctica:

* ∑ 𝑃(𝐸𝑛): Esta parte de la fórmula calcula la suma de las probabilidades de cada estación en el cromosoma. Para cada estación 𝐸𝑛, se obtiene su probabilidad 𝑃(𝐸𝑛) y se suma a lo largo de todas las estaciones. En este sumatorio obtendremos la suma de las distintas poblaciones para cada una de las estaciones.
* ∑ 𝑑𝑖𝑠𝑡(𝐸𝑛, 𝐸𝑛+1): Esta parte de la fórmula calcula la suma de las distancias entre estaciones consecutivas en el cromosoma. Para cada par de estaciones consecutivas (𝐸𝑛, 𝐸𝑛+1), se obtiene la distancia entre ellas y se suma a lo largo de todas las estaciones. Esto refleja la eficiencia en términos de distancia recorrida en el cromosoma.

El parámetro 𝛼 es un valor entre 0 y 1 que permite ajustar la importancia relativa de cada componente en el cálculo del fitness. Si 𝛼 es cercano a 0, se dará más importancia a la suma de distancias y menos importancia a la suma de poblaciones. Si 𝛼 es cercano a 1, se dará más importancia a la suma de poblaciones y menos importancia a la suma de distancias.

Para obtener el valor máximo de Alpha, deberemos entonces asignar un Alpha que sea igual a 0 o 1. El mayor fitness posible podrá ser:

* La suma de distancias de todas las estaciones, al no ser un valor absoluto, el fitness máximo no tendría la necesidad de corresponderse con el total de las distancias.
* La suma de poblaciones de todas las estaciones, en este caso el fitness máximo si que se corresponderá con la suma total de las poblaciones.

Entonces para hallar el fitness máximo, deberemos ejecutar el algoritmo genético dándole máxima prioridad al sumatorio de las poblaciones, es decir con 𝛼 = 1, con el total de las estaciones, E = 20. El fitness entonces deberá corresponderse al total de la suma de poblaciones.

Realizando el cálculo en el archivo de población.xlsx obtenemos el siguiente valor:

Tabla

Descripción generada automáticamente

Ilustración - Sumatorio poblaciones

Obtenemos que el resultado final es 703.249 como total de población de las diferentes estaciones, si le damos prioridad máxima a las poblaciones dentro de la fórmula fitness obtendremos el siguiente resultado:

Ilustración - Fitness máximo

Al generar la población inicial, ya estamos obteniendo el fitness máximo ya que obtenemos todos los cromosomas como distintas combinaciones de todas las estaciones, obteniendo como resultado de fitness 703.249 de fitness.

Al tener total relevancia la población, el fitness máximo será la suma total de las poblaciones de cada estación.

### ¿Podría ser negativo el fitness máximo? ¿Por qué?

En la pregunta anterior hemos visto como el fitness máximo se obtiene otorgando una relevancia total al sumatorio de las poblaciones, debido a la configuración de la fórmula de fitness, si le otorgamos relevancia total a las distancias obtendremos un fitness negativo.

El sumatorio de las distancias se multiplica por su relevancia mediante Alpha y se le resta al sumatorio de las poblaciones. Si el sumatorio de las poblaciones tiene relevancia 0 y el sumatorio de distancias tiene relevancia 1, entonces el fitness máximo obtenido será negativo.

El valor del fitness máximo para este caso nos saldría negativo, lo que no implica que sea un resultado correcto.

Ilustración - Fitness máximo negativo

Al generar una población inicial con todas las estaciones disponibles y dándole una prioridad total a las distancias, obtenemos valores negativos de fitness, en este caso obtenemos para un cromosoma -755,75.

El valor del fitness que obtenemos dependerá entonces del orden de las estaciones y la distancia que exista entre ellas.

### ¿Puedes mejorar el cálculo del fitness? Explícalo y justifica tu elección final.

Se puede mejorar el cálculo del fitness incorporando un criterio de parada basado en la convergencia del algoritmo genético. Este criterio consiste en monitorear el valor del mejor fitness en cada generación y detener el algoritmo cuando no haya una mejora significativa durante un número determinado de generaciones consecutivas.

La justificación de esta mejora es que una vez que la población ha convergido, es poco probable que se produzcan mejoras significativas en el fitness en cada generación.

Continuar la ejecución del algoritmo más allá de este punto solo implicaría un consumo adicional de recursos computacionales sin un beneficio real en la mejora de las soluciones.

### ¿Qué alternativas al proceso de mutación podrías plantear? Explícalo y justifica tu elección final.

* En el proceso de mutación podríamos plantear una probabilidad de cambio que en vez de ir gen a gen dentro del cromosoma para ver si se intercambia por otro gen, tener una probabilidad de cambio para un solo gen dentro del cromosoma.

Esta probabilidad de cambio seleccionaría un gen aleatorio dentro del cromosoma y lo cambiaría por otro. Esto haría que los hijos creados a partir de estos padres solo tuvieran pequeñas modificaciones.

* Una alternativa al proceso de mutación en un algoritmo genético es el proceso de recombinación o entrecruzamiento. En lugar de introducir cambios aleatorios en un cromosoma individual, la recombinación implica mezclar la información genética de dos o más cromosomas para generar nuevos individuos.

Una de las formas más comunes de recombinación es el cruce desde un punto medio en el cromosoma. En este método, se selecciona un punto de corte aleatorio en los cromosomas padres, y se intercambian las porciones de cromosoma a partir de ese punto. De esta manera, se generan dos descendientes que combinan características de ambos padres.

Como justificación de elección final, el entrecruzamiento genera más diversidad a la hora de crear a los hijos, generando que una mayor parte de los hijos tenga más probabilidades de mutar consiguiendo un mejor fitness.

## Cuestión 3

Texto

Descripción generada automáticamenteEn esta cuestión probaremos con distintos valores de E, manteniendo el resto de los valores predeterminados con la siguiente configuración:

Ilustración . Configuración pruebas

* Número de estaciones bajo, E = 5. Cuando el numero de estaciones es bajo, es muy probable que el algoritmo converja rápidamente debido a que probará las distintas combinaciones posibles de todos los genes encontrando la solución óptima rápidamente.

Gráfico

Descripción generada automáticamente

Ilustración . Grafica con E cercano a 0

Media fitness tiene almacenado la media de los valores fitness del 20% mejor de la población, es decir, de los padres seleccionados en cada generación para crear nuevos hijos y BESTf contiene el mejor fitness obtenido en cada generación. Vemos que el algoritmo genético converge y ya no es capaz de encontrar mutaciones que sean mejores.

Gráfico

Descripción generada automáticamente

Ilustración - Mejor Fitness E cercano a 0

Podemos observar por separado el valor del mejor fitness encontrado en las distintas generaciones, observando que a partir de la décima generación el algoritmo ya no es capaz de obtener un fitness mayor. Como consecuencia, los hijos se van pareciendo cada vez más a los padres, generando que existan más de un mejor cromosoma en cada generación.

Gráfico

Descripción generada automáticamente

Ilustración - Evolución best n

Gráfico, Histograma, Gráfico de cajas y bigotes

Descripción generada automáticamente

Ilustración - Evolucion BEST%

Esto indica que, en cada generación, los hijos se van pareciendo mas a los padres, encontrando la misma solución a la hora de mutar.

Texto

Descripción generada automáticamente

Ilustración - Muestreo generación E menor

Por último, en los muestreos de generación, podemos observar que el cromosoma consenso es igual al mejor cromosoma de la población en la generación 15. El cromosoma consenso contiene los genes más repetidos en cada posición dentro de una población generada.

Los hijos se van pareciendo cada vez más a la mejor solución, el algoritmo ha convergido en la solución óptima.

* Número de estaciones alto, E = 16. Cuando el número de estaciones es alto, es muy probable que el algoritmo converja lentamente debido a que probará las distintas combinaciones posibles de todos los genes encontrando la solución óptima.

En este caso se deberían ajustar los parámetros para aumentar las generaciones permitiendo que el algoritmo genético genere soluciones óptimas hasta que converja en la solución más óptima.

Gráfico

Descripción generada automáticamente

Ilustración - Fitness/Media E mayor

En esta solución podemos observar que la media del fitness sigue aumentando, pero no llega a ser igual que el mejor fitness, esto significa que los hijos siguen implementado variaciones respectos a los padres mejorando el fitness.

Gráfico

Descripción generada automáticamente

Ilustración - Evolucion mayor fitness E mayor

Con un E menor, cercano a 0, podíamos observar que a una generación muy temprana el algoritmo ya convergía, ahora podemos observar que, al realizar muchas más combinaciones de genes, el algoritmo tarda más generaciones en encontrar nuevas soluciones óptimas.

Gráfico

Descripción generada automáticamenteGráfico

Descripción generada automáticamente

Ilustración 3- Best% en E mayor

Ilustración 2- Bestn E mayor

Vemos que el algoritmo todavía no converge, el número de mejores cromosomas se mantiene, esto significa que el algoritmo todavía está encontrando variaciones de los mejores padres.

Texto

Descripción generada automáticamente

Ilustración - Resumen generación 50 E mayor

Por último, observamos que el cromosoma consenso tiene la gran mayoría de genes distintos al mejor cromosoma, lo que significa que el resto de la población todavía sigue probando diferentes combinaciones.

## Cuestión 4

Realizaremos tres pruebas y graficaremos los resultados obtenidos comparando las gráficas donde se vea la comparación con los siguientes valores:

|  |  |  |
| --- | --- | --- |
| Alpha | NPOB | NGEN |
| 0.2 | 25 | 100 |
| 0.5 | 50 | 400 |
| 0.8 | 100 | 1000 |

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 26- Comparativa 2

Ilustración 25- Comparativa 1

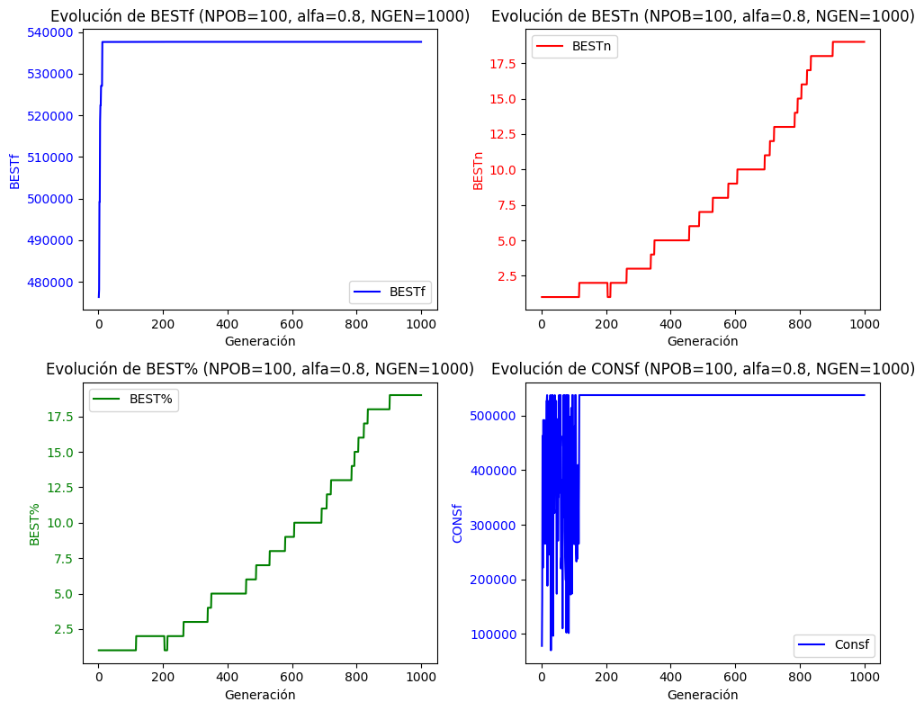


Ilustración 7- Comparativa 3

Analizando las pruebas realizadas, podemos concluir la siguiente relación:

* Si Alpha cercano a 0, entonces el número de generaciones y de individuos de la población debe ser mayor.
* Si Alpha cercano a 1, entonces el número de generaciones y de individuos debe ser menor.

Esto se debe a la fórmula del fitness, si Alpha es cercano a 1, entonces el sumatorio de las poblaciones tiene más relevancia, en este caso, para realizar el sumatorio de poblaciones, el orden de los genes en el cromosoma no tiene importancia. Lo que genera cromosomas de distintos genes con un fitness equitativo. Haciendo que el algoritmo converja más rápidamente.

Mientras que con un Alpha cercano a 0, genera que el sumatorio de las distancias tenga más relevancia. En el sumatorio de distancias, es crucial el orden de los genes del cromosoma, dependiendo de la ordenación de los genes se obtiene un fitness distinto. Por eso con un Alpha menor, se necesitan muchas más generaciones para que el algoritmo converja, debido a la importancia de la ordenación del cromosoma, es más difícil que distintos cromosomas puedan tener un fitness similar.

## Cuestión 5

Para esta cuestión vamos a analizar cómo afectan las probabilidades crecientes de cambio (Pc) y probabilidades crecientes de intercambio (Pi) a BESTf, BESTn, BEST% y CONSf.

Probaremos los siguientes valores:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Probabilidades | PC | PI | PNada | Alpha | NGEN | NPOB |
| Prueba 1 | 0.1 | 0.1 | 0.8 | 0.5 | 200 | 50 |
| Prueba 2 | 0.3 | 0.3 | 0.4 | 0.5 | 200 | 50 |
| Prueba 3 | 0.5 | 0.5 | 0 | 0.5 | 200 | 50 |

Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

Ilustración 28- Prueba 1

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 29- Prueba 3

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 30- Prueba 3

Como podemos observar en las gráficas, Pc y Pi son las probabilidades de cambio que afectan a los cromosomas padres. Probabilidades bajas de estos factores, provocan que la nueva población no haya sufrido mutaciones y sea similar a los padres. Esto provocará que el algoritmo converja rápidamente, debido a que los hijos de los padres se parecerán cada vez más entre sí, evitando que se puedan generar nuevas combinaciones de genes que produzcan un fitness mayor.

A medida que se van aumentando las probabilidades de los factores de cambio, podemos observar como el algoritmo sigue creando nuevas y nuevas combinaciones de genes, podemos observar como a medida que se aumentan las probabilidades de cambio, se reduce la estabilidad del cromosoma consenso.

Al no estabilizarse el cromosoma consenso, indica que el resto de la población ha sufrido una mutación y el algoritmo todavía está encontrando nuevas combinaciones de genes que son distintas a la de los padres, generando mayores posibilidades de encontrar hijos con mejores fitness.

## Cuestión 6

Prueba con distintos porcentajes de elitismo y encuentra cual es el mejor. Haz una tabla comparativa y explica los detalles de la configuración elegida.

Para estas pruebas debemos probar con unos datos iniciales iguales para los tres porcentajes de elitismo que vamos a calcular (10%, 20% y 30%):

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Ilustración 31- Parámetros seleccion

* Elitismo 10%

En esta ocasión estaremos obteniendo los resultados con el 10% de elitismo, es decir, el 10% de los hijos serán exactamente igual que el 10% de los padres con más fitness.

Esto supondrá que los nuevos hijos construidos contarán con un porcentaje menor de padres exactamente iguales. Sin embargo, se generará un 10% más de hijos a partir de los padres seleccionados, es decir, el 70% de la población que construimos en la función generar70 se dividirá en 10% (exactamente igual que el 10% de la élite de la generación anterior) y en 60% (construidos a partir del 10%).

La función que hemos modificado de la implementación original en Python es la siguiente:

Texto

Descripción generada automáticamente

Ilustración 32- Generar 70 python

Gráfico, Histograma

Descripción generada automáticamenteGráfico, Histograma

Descripción generada automáticamenteLos resultados obtenidos se ven en las siguientes gráficas:

Ilustración 33- BEST% y BESTn

En estas gráficas podemos observar la progresión de las variables BestN y Best%. En esta ocasión a partir de la generación 350 aproximadamente comienza a aumentar el número de cromosomas cuyo fitness es el Best Fitness hasta llegar a la generación 800 donde tenemos hasta un 4.5% de cromosomas cuyo fitness es BestF, es decir, un total de 9 dentro de cada generación.

Gráfico

Descripción generada automáticamente

Ilustración 34- comparación fitness/media

En esta otra gráfica podemos ver la comparativa de la media de los fitness de cada generación con el mejor fitness de cada generación. Podemos ver como rápidamente la media sube hasta llegar a valores muy similares que el BestF. Sin embargo, estos valores no convergen hasta la generación 350 aproximadamente. Momento en el que el valor de BestN empieza a dispararse. Eso quiere decir que, hasta entonces, el programa continuaba encontrando mejores cromosomas.

* Elitismo 20

En esta ocasión estaremos obteniendo los resultados con el 20% de elitismo, es decir, el 20% de los hijos serán exactamente igual que el 20% de los padres con más fitness. Estos datos son los mismos que con los que se trabaja en el enunciado original.

El código es el mismo que el que observamos en la (ilustración 32) pero cambiando los valores de los porcentajes.

Los resultados obtenidos se ven en las siguientes gráficas:

Gráfico, Histograma

Descripción generada automáticamenteGráfico, Histograma

Descripción generada automáticamente

Ilustración 35- BEST% y BESTn

En esta ocasión, podemos ver en la gráfica como el valor de BestN empieza a variar antes que en el Elitismo 10%. Sin embargo, el valor de BestN sufre altibajos (ya que surgen nuevos cromosomas mejores) y no es hasta la generación 600 que el valor de la variable BestN aumenta considerablemente. Es decir, el cromosoma con mayor fitness no vuelve a ser superado por otro con mejor fitness.

Gráfico

Descripción generada automáticamente

Ilustración 36- Comparación fitness/media

En esta gráfica vemos que la media del fitness tarda un poco más en alcanzar los valores de BestF que en el Elitismo 10%. Esto se debe a que la élite es de peor calidad en el Elitismo 20% ya que selecciona padres con una media de fitness más baja que en el Elitismo 10%.

* Elitismo 30%

En esta ocasión estaremos obteniendo los resultados con el 30% de elitismo, es decir, el 30% de los hijos serán exactamente igual que el 30% de los padres con más fitness.

Esto supondrá que los nuevos hijos construidos contarán con un porcentaje menor de padres exactamente iguales. Sin embargo, se generarán un 10% menos de hijos a partir de los padres seleccionados, es decir, el 40% de la población que construimos en la función generar70 se dividirá en 30% (exactamente igual que el 30% de la élite de la generación anterior) y en 40% (construidos a partir del 30%).

El código es el mismo que el que observamos en la (ilustración 32) pero cambiando los valores de los porcentajes.

Los resultados obtenidos se ven en las siguientes gráficas:

Gráfico, Histograma

Descripción generada automáticamenteGráfico, Histograma

Descripción generada automáticamente

Ilustración 37- BEST% y BESTn

En este ejemplo se puede ver claramente como el BestN empieza a crecer pasada la generación 500, es decir, pasan más generaciones hasta que el fitness más alto se estanca, es decir, que no se genera un individuo mejor.

Gráfico

Descripción generada automáticamente

Ilustración 38. comparación fitness/media

Comparando esta gráfica podemos ver como lo que hemos comentado para el Elitismo 20% se repite, pero esta vez de una forma más clara. La media fitness tarda en crecer ya que los padres seleccionados son más variados y la media de su fitness es más baja.

### Generación 300:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 1 | 0.5 | 200 | 0.5 |
| Elitismo 20% | 2 | 1 | 200 | 0.5 |
| Elitismo 30% | 1 | 0.5 | 200 | 0.5 |

### Generación 400:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 3 | 1.5 | 200 | 0.5 |
| Elitismo 20% | 2 | 1 | 200 | 0.5 |
| Elitismo 30% | 1 | 0.5 | 200 | 0.5 |

### Generación 500:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 4 | 2 | 200 | 0.5 |
| Elitismo 20% | 2 | 1 | 200 | 0.5 |
| Elitismo 30% | 1 | 0.5 | 200 | 0.5 |

### Generación 600:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 7 | 3.5 | 200 | 0.5 |
| Elitismo 20% | 2 | 1 | 200 | 0.5 |
| Elitismo 30% | 3 | 1.5 | 200 | 0.5 |

### Generación 700:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 7 | 3.5 | 200 | 0.5 |
| Elitismo 20% | 3 | 1.5 | 200 | 0.5 |
| Elitismo 30% | 4 | 2 | 200 | 0.5 |

### Generación 800:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Elitismo | BestN | Best% | NPOB | Alpha |
| Elitismo 10% | 9 | 4.5 | 200 | 0.5 |
| Elitismo 20% | 6 | 4 | 200 | 0.5 |
| Elitismo 30% | 5 | 2.5 | 200 | 0.5 |

### Conclusión

En conclusión, tras haber probado los tres porcentajes de elitismo y haberlos comparado gráficamente y, posteriormente, a través de unas tablas, concluimos que el Elitismo del 10% es el mejor de cara a generar más rápidamente (menor número de generaciones) un mejor fitness.

Cuanto menor es el elitismo, menos son los padres seleccionados a la hora de construir más hijos (a partir de ellos mismos), pero estos padres aseguran ser los mejores dentro de la población, es decir, tendrán un fitness sin mucha desviación.

Si la élite es más reducida, es de más calidad, su fitness es más alto (de media) y por lo tanto los hijos también lo van a ser ya que se van a parecer a los mejores cromosomas de la generación anterior.

# Bibliografía

Tejedor, Á. J. (s.f.). *IAII\_Canvas*. Obtenido de https://ufv-es.instructure.com/courses/26539/pages/sobre-la-asignatura

Gómez, R. E. L., & Guerrero, M. H. A. (2013). Los Algoritmos Genéticos en el Modelo de Cournot. *Economía Informa*. https://doi.org/10.1016/s0185-0849(13)71307-0

KeepCoding, R. (2023, 27 abril). ¿Qué es un algoritmo genético? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. https://keepcoding.io/blog/que-es-un-algoritmo-genetico/

López, J. C. (2010, 7 octubre). *Introducción a los algoritmos genéticos: como implementar un algoritmo genético en JAVA - Adictos al trabajo*. Adictos al trabajo. https://www.adictosaltrabajo.com/2010/10/07/jgap/

Ossipov, E. (2017). Algoritmos Genéticos: Búsqueda y Optimización por Selección Natural. *Toptal Engineering Blog*. https://www.toptal.com/algorithms/algoritmos-geneticos-busqueda-y-optimizacion-por-seleccion-natural