Computación Neuronal y Evolutiva

(Práctica4 – Algoritmos Genéticos Básicos)

UNIVERSIDAD DE BURGOS

Autores:

Mario Ubierna San Mamés

Jorge Navarro González

Índice de Contenido

Descripción de la carga y almacenamiento de los datos

Para la realización de la carga de los datos hemos usado el fichero LecturaDatos.py, el cual se ha adjuntado en el zip.

En cuanto a cómo guardamos los datos que empleamos en el problema, hemos usado las siguientes variables:

- **Videos**: es un entero en el que almacenamos los números de videos que tiene nuestro problema. Hemos considerado esta opción ya que es la más fácil de almacenar un número entero.

- **Endpoints**: al igual que antes es un entero, en el que guardamos los diferentes dispositivos finales. Hemos considerado esta opción ya que es la más fácil de almacenar un número entero.

- **Request**: es un entero en el que almacenamos el número de peticiones estimada de cuantas veces se va a pedir cada vídeo concreto desde un determinado punto de acceso. Hemos considerado esta opción ya que es la más fácil de almacenar un número entero.

- **Caches**: es un entero en el que guardamos el número de cachés que va a disponer nuestro problema. Hemos considerado esta opción ya que es la más fácil de almacenar un número entero.

- **Tam_cache**: es un entero en el que almacenamos el tamaño de las cachés, solo usamos un entero porque el problema viene determinado para que todas las cachés tengan el mismo tamaño. Hemos considerado esta opción ya que es la más fácil de almacenar un número entero.

- **Memorias**: es una lista donde guardamos para cada uno de los videos el tamaño que ocupa. Lo ideal sería hacerlo con numpy, pero nosotros no nos manejamos muy bien con ello, así que hemos usado la manera tradicional, la cual es algo menos eficiente, sin embargo no hay mucha diferencia en problemas pequeños.

- **Endpointcache**: es una lista donde almacenamos para cada uno de los endpoints la latencia que tiene a la/s caché/s, es decir, es una lista de listas. Si no tiene conexión a alguna caché entonces almacenamos un 0. Lo ideal sería hacerlo con numpy, pero nosotros no nos manejamos muy bien con ello, así que hemos usado la manera tradicional, la cual es algo menos eficiente, sin embargo no hay mucha diferencia en problemas pequeños.

- **Tiempo_endpoint_central**: es una lista donde guardamos para cada uno de los endpoint la latencia que tiene al servidor central. Lo ideal sería hacerlo con numpy, pero nosotros no nos manejamos muy bien con ello, así que hemos usado la manera tradicional, la cual es algo menos eficiente, sin embargo no hay mucha diferencia en problemas pequeños.

- **Descripcion_request**: es una lista donde almacenamos para cada uno de los endpoints las peticiones que tiene al video/s, es decir, es una lista de listas. Si no tiene conexión a algún video entonces almacenamos un 0. Lo ideal sería hacerlo con numpy, pero nosotros no nos manejamos muy bien con ello, así que hemos usado la manera tradicional, la cual es algo menos eficiente, sin embargo no hay mucha diferencia en problemas pequeños.

Como se puede apreciar en el texto anterior, hemos creado todas las variables necesarias para poder almacenar todos los datos que se nos proporcionaba por el fichero.

Descripción de la solución del problema

¿Cómo se representa su genotipo?

Nuestro genotipo se representa como un vector binario, cuyo tamaño es el número de cachés por el número de videos.

Imaginemos que tenemos 3 vídeos y 4 cachés, nuestro genotipo sería de la siguiente forma:

Caché 1			**Caché 2**			**Caché 3**			**Caché 4**		
Vid1	Vid2	Vid3	Vid1	Vid2	Vid3	Vid1	Vid2	Vid3	Vid1	Vid2	Vid3

Caché 1			**Caché 2**			**Caché 3**			**Caché 4**		
1	0	0	1	1	0	0	0	1	1	0	1

Dada la representación anterior, el significado de la misma sería el siguiente:

- La caché1 tiene el vídeo1.
- La caché2 tiene el vídeo1 y el vídeo2.
- La caché3 tiene el vídeo3.
- La caché4 tiene el vídeo1 y el vídeo3.

¿Cómo se calcula la adaptación?

Para poder realizar este paso hemos creado una función fitness a la que la pasamos un individuo, y nos devuelve el valor de fitness para ese individuo, teniendo en cuenta la penalización.

Esta función fitness lo que hace es evaluar un genotipo y nos devuelve un valor, el cuál puede ser el mínimo de las latencias de un determinado endpoint sobre una caché que contenga el vídeo por el número de peticiones que hay desde el endpoint a ese video, o la latencia desde un determinado endpoint sobre el servidor central (ya que éste siempre contiene a los vídeos) por el número de peticiones que hay desde el endpoint a ese video, esto es para cada uno de los endpoints y para cada uno de los videos, por lo tanto el resultado es el sumatorio de cada una de las soluciones.

```python
def fitness(genotipo):
    mstotal = 0
    for i in range(dg.endpoints):
        for j in range(dg.videos):
            peticion = dg.descripcion_request[i][j]
            if peticion != 0:
                conjunto_latencias = set()
                for c in range(dg.caches):
                    latencia = dg.endpointcache[i][c]
                    if latencia != 0:
                        if genotipo[c*dg.videos+j] == 1:
                            conjunto_latencias.add(latencia)
                if len(conjunto_latencias) > 0:
                    mstotal+=min(conjunto_latencias)*peticion
                else:
                    mstotal+=dg.tiempo_endpoint_central[i]*peticion
    mstotal*=penalizar_genotipo(genotipo)
    return mstotal,
```

Por otro lado, tal y como se comentó en clase, penalizamos el genotipo dependiendo de cuánto se pasen los videos sobre la caché, es decir, si los vídeos no llegan a llenar la caché o la llenan de forma completa no penalizamos, pero si se pasa el tamaño de los vídeos sobre el tamaño de la caché sí que penalizamos de forma proporcional, es decir, si el tamaño de los vídeos es de 120 y el de la caché de 100 la penalización sería de 1,2.

Para un problema sencillo la penalización tal y como la hemos mencionado antes nos serviría, pero si el problema es algo complejo tenemos que penalizar mucho más, es decir, multiplicamos por 100. Siguiendo el caso anterior, en vez de la penalización de 1,2 sería de 120.

El código correspondiente a la penalización es el siguiente:

```python
def penalizar_genotipo(genotipo):
    suma_pen = 1
    for i in range(dg.caches):
        desplazamiento = i*dg.videos
        tamano = 0
        for j in range(dg.videos):
            tamano+=dg.memorias[j]*genotipo[desplazamiento+j]
        resta = tamano - dg.tam_cache
        if resta > 0:
            suma_pen+=resta/dg.tam_cache
    if suma_pen != 1:
        suma_pen*=100
    return suma_pen
```

Ventajas y desventajas del genotipo y el fitness

En cuanto al genotipo la principal ventaja es que procesar la información es muy sencilla, ya que son número binarios, es decir, 0 ó 1.

Otra de las principales ventajas del genotipo es que no nos tenemos que preocupar por el cruce, ya que si fueran enteros sí que tendríamos que comprobar que un mismo vídeo no puede estar dos veces en la misma caché por ejemplo.

La mayor desventaja que tenemos es que no hemos usado numpy, por lo que nuestra representación no es todo lo eficiente que podría ser, aun así para el tamaño de problemas que hemos resuelto no hemos tenido ningún inconveniente.

En cuanto a la función fitness tenemos la ventaja que tratamos de que la latencia para consumir un conjunto de vídeos desde un endpoint determinado sea la menor posible, lo cual es lo que tendríamos que resolver desde el primer día que se nos presentó la práctica.

El principal inconveniente que tiene nuestra fitness es que quedan números muy grandes, ya que multiplicamos la latencia de cada video por el número de peticiones que tiene dicho vídeo, este resultado a la vista no es del todo bonito ya que te queda 7 cifras o más como resultado y no es fácil de apreciar, sin embargo, si el resultado fuera 3 dígitos sería más sencillo.

Fenotipo que devuelve el algoritmo

Para realizar este apartado nos hemos fijado en la representación que se indicaba en el fichero del desafío HashCode.

Dicha representación nos indica por cada caché los vídeos que contiene, por ejemplo, si tuviéramos el siguiente ejemplo:

Caché 0			**Caché 1**			**Caché 2**			**Caché 3**		
1	0	0	1	1	0	0	0	1	1	0	1

Dada la representación anterior, el significado de la misma sería el siguiente:

- La caché0 tiene el vídeo1.
- La caché1 tiene el vídeo1 y el vídeo2.
- La caché2 tiene el vídeo3.
- La caché3 tiene el vídeo1 y el vídeo3.

Y la salida por fichero sería la siguiente, donde su significado es lo comentado en la página anterior:

4

0 1

1 1 2

2 3

3 1 3

El código correspondiente a la traducción del genotipo al fenotipo es el siguiente:

```
def fenotipo(genotipo):

    fenotipo = []
    for i in range(dg.caches):
        fenotipo.append([])
        for j in range(dg.videos):
            if genotipo[i*dg.videos+j] == 1:
                fenotipo[i].append(j)

    with open("fenotipo.out",'w') as out:
        count = 0
        for i in fenotipo:
            if len(i) != 0:
                count += 1
        out.write(str(count)+"\n")
        for i in range(len(fenotipo)):
            if len(fenotipo[i]) != 0:
                out.write(str(i))
                for j in fenotipo[i]:
                    out.write(" "+str(j))
                out.write("\n")
```

Para saber si es una solución válida o no, lo que hemos hecho ha sido por cada experimento imprimimos la penalización que tiene la solución, si dicha penalización es 1 es que realmente esa solución es correcta, por lo contrario, si es distinto de 1 es que no es una solución válida.

Por otro lado, para saber si la solución es óptima o no, lo que hemos hecho ha sido ejecutar varias veces el mismo fichero con los mismos parámetros, si nos daba un resultado similar entonces es que era óptima la solución, por lo contrario, si sí que había diferencia entonces es que no siempre encuentra la solución óptima.

Experimentos

Para la realización de los experimentos he seguido las pautas que nos indicaron en ubuvirtual, las cuales son las siguiente:

1. Seleccionar una configuración base. Basta con que aporte una solución medianamente buena. Simplemente es para tener un punto de partida.

2. Seleccionar un aspecto a comprobar (Ej: Tipo de Selección, Numero de Individuos, Probabilidad de Cruce, etc)

3. Elegir el rango disponible para probar. El resto de los parámetros quedan fijos según la configuración base elegida en (1):

    1. Si es un número, se debe comprobar un rango de valores. 4 o 5 son suficientes (Ej: num individuos = [50, 100, 150, 200, 250])

    2. Si es una selección, simplemente se eligen entre los disponibles (Ej: Selección = [Torneo, Ruleta, Restos])

4. Seleccionar 2 o 3 configuraciones del problema / ficheros.

5. Repetir la ejecución del algoritmo una vez por cada opción en el punto (3) y cada fichero en (4).

6. Agrupar los datos de cada experimento en tablas o gráficos para presentarlos. Por cuestiones de resumen de información, incluso se pude comprobar solamente el valor óptimo encontrado en cada ejecución, en lugar de la gráfica completa de la búsqueda.

Cabe mencionar que hicimos 4 experimentos, y en cada experimento hicimos 9 casos, hemos ejecutado esos 9 casos para los 4 ficheros de cada experimento.

Experimento 1

Para la realización de este experimento hemos usado el fichero *"me_at_the_zoo"*, éste es un fichero sencillo ya que no tiene muchos datos.

Caso 1

El primer caso siempre es algo sencillo, es decir, una configuración inicial. Los parámetros con los que hemos realizado este caso son los siguientes:

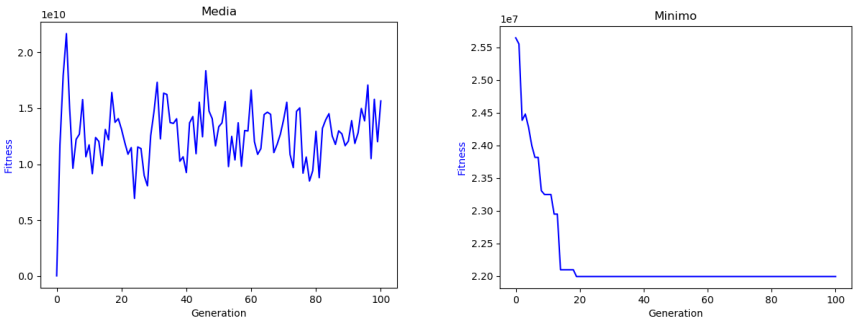
alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)




```
90      88      1.20665e+10     2.36192e+10     2.19956e+07     7.88261e+10
91      83      1.38942e+10     2.57527e+10     2.19956e+07     7.93493e+10
92      74      1.18604e+10     2.32317e+10     2.19956e+07     6.91245e+10
93      83      1.28132e+10     2.40651e+10     2.19956e+07     7.43124e+10
94      72      1.49847e+10     2.57237e+10     2.19956e+07     9.45166e+10
95      80      1.38774e+10     2.58013e+10     2.19956e+07     8.7879e+10
96      81      1.70771e+10     2.75705e+10     2.19956e+07     7.9341e+10
97      88      1.05048e+10     2.21549e+10     2.19956e+07     7.4496e+10
98      87      1.5802e+10      2.62383e+10     2.19956e+07     7.39697e+10
99      79      1.20166e+10     2.41822e+10     2.19956e+07     7.5839e+10
100     82      1.56574e+10     2.54162e+10     2.19956e+07     7.27828e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(21995616,)
```

Como podemos apreciar en el primer gráfico, exceptuando las primeras generaciones el resto sí que se mantienen entre 1 y 1.5 de fitness, lo cual es algo normal. Por otro lado, vemos en el siguiente gráfico que en apenas 20 generaciones ya se ha alcanzado el mínimo, esto es algo bueno ya que converge muy rápido el problema, esto se debe a que se penaliza muy fuerte a aquellos genotipos que no cumplen las condiciones del problema.

Caso 2

En este caso realizamos solo la modificación en el número de generaciones, pasa de 100 a 300. Los parámetros con los que hemos realizado este caso son los siguientes:

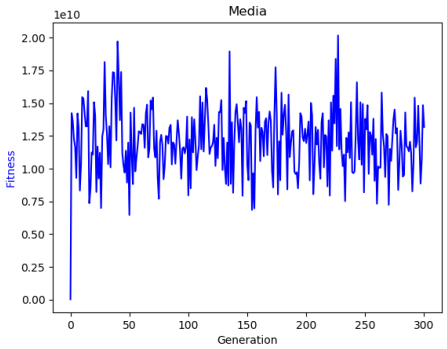
alg_param['cxpb'] = 0.75

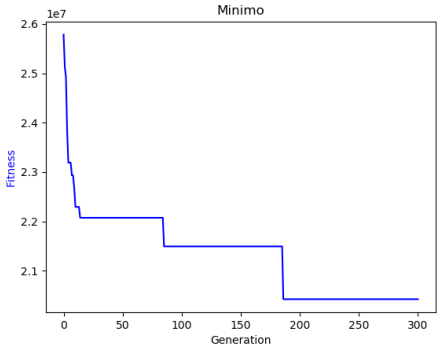
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 300

init_pop = toolbox.population(n=100)






```
290     63      8.26902e+09     2.06153e+10     2.0422e+07      7.45312e+10
291     73      1.04358e+10     2.2257e+10      2.0422e+07      7.56004e+10
292     84      1.54232e+10     2.69904e+10     2.0422e+07      8.15253e+10
293     70      1.16371e+10     2.30168e+10     2.0422e+07      6.7414e+10
294     76      1.20476e+10     2.359e+10       2.0422e+07      7.26259e+10
295     83      1.48079e+10     2.50971e+10     2.0422e+07      8.25118e+10
296     77      1.21248e+10     2.31059e+10     2.0422e+07      6.84141e+10
297     81      8.86172e+09     2.0547e+10      2.0422e+07      7.09217e+10
298     86      1.06584e+10     2.22585e+10     2.0422e+07      7.31016e+10
299     84      1.48509e+10     2.52169e+10     2.0422e+07      8.37399e+10
300     84      1.31757e+10     2.42011e+10     2.0422e+07      7.28673e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(20421989,)
```

En cuanto al primer gráfico, podemos comentar que al tener más generaciones tenemos más genotipos que no siguen a la media aun así la gran mayoría ha conseguido estar entre 1.2 y 1.4, reduciendo algo más que en el caso anterior.

Por otro lado, en el segundo gráfico vemos que al igual que antes converge muy rápido, sin embargo, en este caso continúa bajando más allá de la generación 20, algo que no sucedía en el caso anterior.

Como conclusión, cabe destacar que a medida que aumentamos las generaciones mejora la solución del algoritmo.

Caso 3

En este caso realizamos solo la modificación en el número de generaciones, pasa de 300 a 600. Los parámetros con los que hemos realizado este caso son los siguientes:

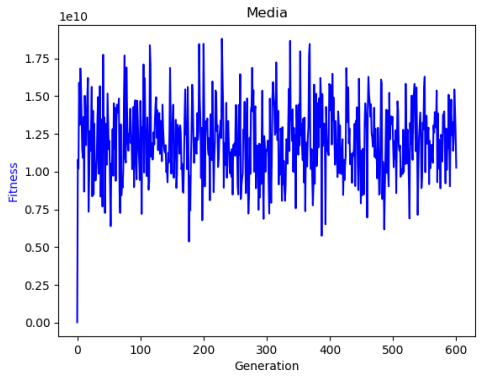
alg_param['cxpb'] = 0.75

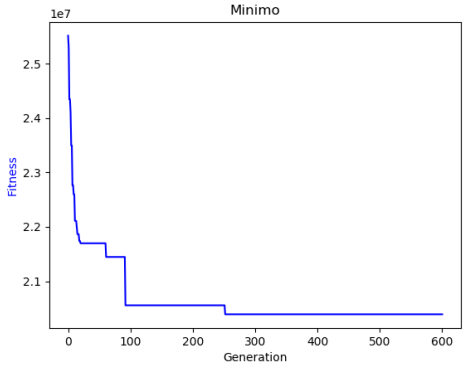
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 600

init_pop = toolbox.population(n=100)






```
590     85      9.03035e+09     2.07874e+10     2.03935e+07     6.47379e+10
591     78      1.33267e+10     2.45871e+10     2.03935e+07     7.57207e+10
592     81      1.47709e+10     2.53351e+10     2.03935e+07     7.00281e+10
593     85      1.24111e+10     2.33784e+10     2.03935e+07     6.96412e+10
594     79      1.32749e+10     2.51398e+10     2.03935e+07     7.93023e+10
595     83      1.13767e+10     2.28531e+10     2.03935e+07     6.74961e+10
596     83      1.21283e+10     2.36997e+10     2.03935e+07     7.03362e+10
597     84      1.54411e+10     2.61489e+10     2.03935e+07     7.43457e+10
598     80      1.42048e+10     2.54863e+10     2.03935e+07     6.79566e+10
599     84      1.26172e+10     2.35579e+10     2.03935e+07     7.47383e+10
600     74      1.02605e+10     2.07656e+10     2.03935e+07     6.13254e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(20393504,)
```

En el primer gráfico podemos observar, que las medias están más concentradas, ahora es más difícil de sacar un intervalo, aun así podemos ver que tiene un intervalo menor al caso anterior.

Por otro lado, el gráfico del mínimo vemos que converge parecido al caso anterior, sin embargo, ahora antes de llegar a la generación 300 se produce una mejora del algoritmo, por lo tanto ahora obtenemos una mejor solución que en los casos anteriores.

Caso 4

En este caso y en el siguiente volvemos a la configuración inicial, lo que pasa que ahora modificamos la población inicial. Los parámetros con los que hemos realizado este caso son los siguientes:

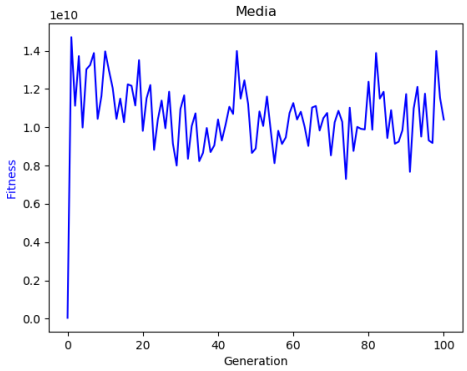
alg_param['cxpb'] = 0.75

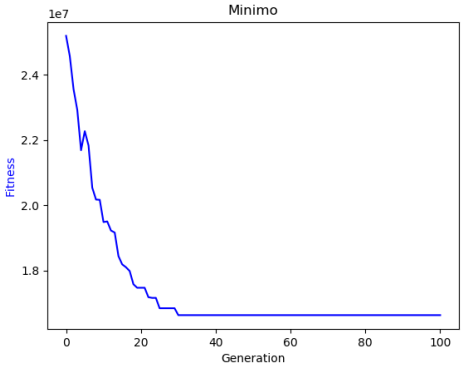
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=300)






```
90      248     1.17317e+10     2.1239e+10      1.66253e+07     6.36687e+10
91      248     7.67247e+09     1.77614e+10     1.66253e+07     6.25034e+10
92      230     1.09815e+10     2.10435e+10     1.66253e+07     6.99507e+10
93      246     1.21109e+10     2.20056e+10     1.66253e+07     9.13252e+10
94      245     9.51238e+09     1.99246e+10     1.66253e+07     7.45489e+10
95      248     1.17531e+10     2.17246e+10     1.66253e+07     6.99169e+10
96      236     9.31402e+09     1.96008e+10     1.66253e+07     6.21145e+10
97      230     9.1781e+09      1.97894e+10     1.66253e+07     6.91202e+10
98      246     1.39839e+10     2.34953e+10     1.66253e+07     7.28914e+10
99      236     1.15376e+10     2.08374e+10     1.66253e+07     6.64269e+10
100     252     1.0399e+10      2.04711e+10     1.66253e+07     7.43757e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(16625283,)
```

Tal y como vemos en la primera gráfica, la media sí que es un poco dispersa, aun así podemos decir que el intervalo en el que se encuentra la media sería de 1 a 1.2, mejoramos dicho intervalo.

Por otro lado, vemos que la función mínimo es más bonita, ya que va algo más curvada y no es tan brusca como los casos anteriores, además como se puede apreciar en el fitness hemos mejorado considerablemente la solución respecto a los casos anteriores.

Caso 5

En este caso volvemos a la configuración inicial, lo que pasa que ahora modificamos el número de generaciones de 100 a 400. Los parámetros con los que hemos realizado este caso son los siguientes:

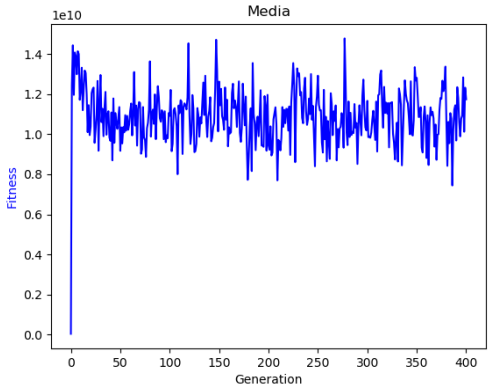
alg_param['cxpb'] = 0.75

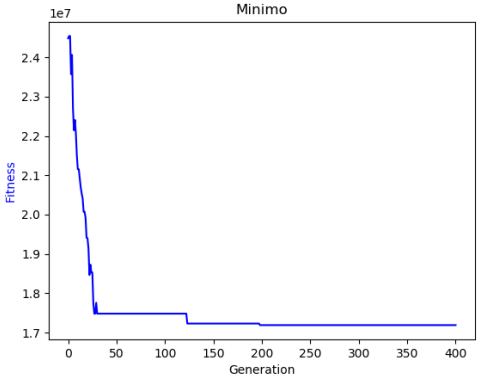
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 400

init_pop = toolbox.population(n=300)






```
390     227     9.68719e+09     2.00227e+10     1.71885e+07     7.02065e+10
391     253     1.23495e+10     2.17389e+10     1.71885e+07     7.68953e+10
392     240     1.18155e+10     2.18521e+10     1.71885e+07     7.52871e+10
393     222     1.06014e+10     2.12856e+10     1.71885e+07     7.45327e+10
394     241     9.89211e+09     1.99348e+10     1.71885e+07     6.84767e+10
395     249     1.07782e+10     2.07007e+10     1.71885e+07     7.59712e+10
396     220     1.09298e+10     2.14808e+10     1.71885e+07     6.93515e+10
397     230     1.28376e+10     2.27021e+10     1.71885e+07     6.58843e+10
398     233     1.01248e+10     2.07189e+10     1.71885e+07     7.50512e+10
399     235     1.23243e+10     2.2281e+10      1.71885e+07     6.8179e+10
400     232     1.17432e+10     2.17659e+10     1.71885e+07     6.54291e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(17188485,)
```

Como podemos observar en la primera gráfica, las medias están más concentradas, sin embargo, el intervalo en el que se encuentra la media sería de 0.9 a 1.3, el cual es un pelín más amplio que en el caso anterior.

Por otro lado, en la segunda gráfica podemos ver que converge muy rápido a la solución, pero aun así no nos proporciona una mejor solución que en el caso anterior, por lo que es mejor el anterior caso.

Caso 6

En este caso volvemos a la configuración inicial, lo que pasa que ahora modificamos la probabilidad de cruce, la reducimos a 0.4. Los parámetros con los que hemos realizado este caso son los siguientes:

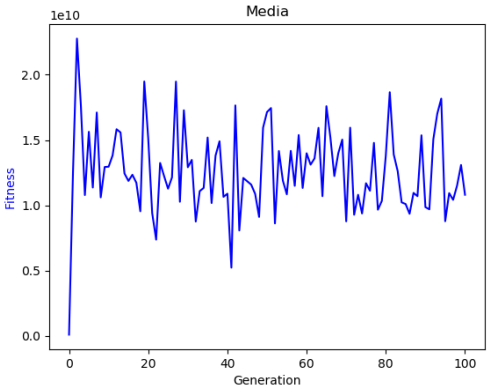
alg_param['cxpb'] = 0.4

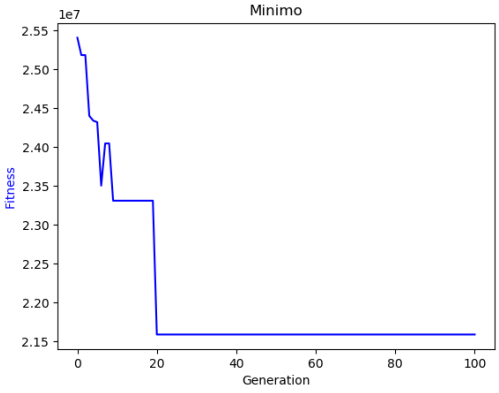
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      49      9.85661e+09     2.2808e+10      2.15891e+07     7.76767e+10
91      48      9.68967e+09     2.24223e+10     2.15891e+07     8.23256e+10
92      54      1.50582e+10     2.63611e+10     2.15891e+07     7.56738e+10
93      59      1.70193e+10     2.78224e+10     2.15891e+07     8.39142e+10
94      56      1.81737e+10     2.85904e+10     2.15891e+07     8.5717e+10
95      61      8.77795e+09     2.11601e+10     2.15891e+07     8.20142e+10
96      54      1.09329e+10     2.34215e+10     2.15891e+07     7.56389e+10
97      50      1.0416e+10      2.24877e+10     2.15891e+07     8.34335e+10
98      59      1.15163e+10     2.41307e+10     2.15891e+07     7.62104e+10
99      47      1.30853e+10     2.48985e+10     2.15891e+07     8.06272e+10
100     47      1.08054e+10     2.31205e+10     2.15891e+07     6.72492e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(21589130,)
```

En este caso, en el primer gráfico vemos que las medias están más dispersas, el intervalo en el que encontramos la media podría ser el siguiente, de 1.2 a 1.6. Hemos aumentado un poco la media, esto se debe a que estamos poniendo casi la misma probabilidad para el cruce que para la mutación, generando así una zona de inestabilidad.

En cuanto al segundo gráfico, podemos observar que a partir de la generación 20 ya ha encontrado el óptimo, en cuanto a los casos anteriores podemos ver que el fitness que no da es peor por lo comentado en el párrafo anterior.

Caso 7

En este caso volvemos a la configuración inicial, lo que pasa que ahora modificamos la probabilidad de cruce, la aumentamos a 0.95. Los parámetros con los que hemos realizado este caso son los siguientes:

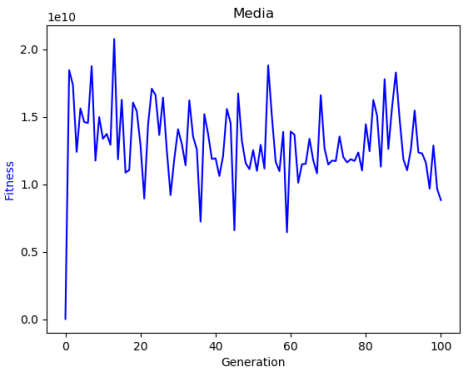
alg_param['cxpb'] = 0.95

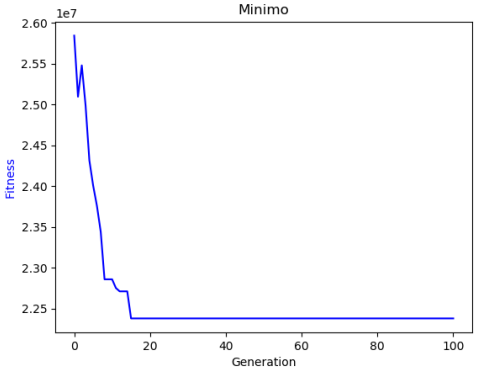
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      95      1.18635e+10     2.34379e+10     2.238e+07       7.27072e+10
91      96      1.10428e+10     2.34201e+10     2.238e+07       8.30199e+10
92      100     1.25742e+10     2.46786e+10     2.238e+07       7.40709e+10
93      100     1.54646e+10     2.70818e+10     2.238e+07       8.13551e+10
94      100     1.23601e+10     2.36086e+10     2.238e+07       7.19294e+10
95      98      1.22851e+10     2.47616e+10     2.238e+07       7.35201e+10
96      98      1.15986e+10     2.33393e+10     2.238e+07       6.94506e+10
97      97      9.67736e+09     2.16922e+10     2.238e+07       7.34523e+10
98      96      1.2878e+10      2.551e+10       2.238e+07       8.18684e+10
99      96      9.64586e+09     2.18911e+10     2.238e+07       8.00792e+10
100     96      8.83328e+09     2.11946e+10     2.238e+07       6.81518e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(22380031,)
```

Al aumentar la probabilidad de cruce, lo que estamos consiguiendo es que los hijos siempre vengan de las mismas características que los padres, por lo que al apenas mutar no hay nuevas características, esto lo podemos ver en todas las capturas anteriores ya que en el primer gráfico el intervalo es algo mayor que el caso anterior, y en el segundo gráfico no baja tanto la fitness con el paso de las generaciones.

Además podemos observar que el valor fitness del mejor genotipo es mayor que en el caso anterior, por lo que una alta probabilidad de cruce no proporciona los mejores resultados.

Caso 8

En este caso volvemos a la configuración inicial, y lo que hemos hecho es modificar la probabilidad de mutación, bajando su valor considerablemente. Los parámetros que hemos utilizado son los siguientes:

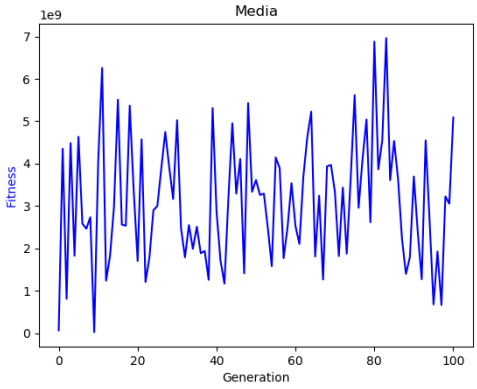
alg_param['cxpb'] = 0.75

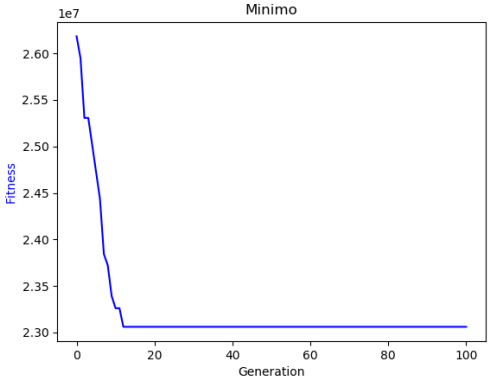
alg_param['mutpb'] = 0.05

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      86      3.697e+09       1.47388e+10     2.30605e+07     7.34636e+10
91      77      2.38409e+09     1.16145e+10     2.30605e+07     6.80399e+10
92      76      1.26997e+09     8.74087e+09     2.30605e+07     6.56718e+10
93      75      4.5487e+09      1.6719e+10      2.30605e+07     8.3728e+10
94      75      2.6462e+09      1.29569e+10     2.30605e+07     7.38426e+10
95      84      6.78266e+08     6.51922e+09     2.30605e+07     6.55436e+10
96      72      1.92603e+09     1.09568e+10     2.30605e+07     7.74727e+10
97      80      6.63478e+08     6.37208e+09     2.30605e+07     6.40648e+10
98      80      3.22717e+09     1.40037e+10     2.30605e+07     6.98011e+10
99      75      3.05478e+09     1.33298e+10     2.30605e+07     7.05091e+10
100     66      5.08972e+09     1.73818e+10     2.30605e+07     8.46019e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(23060462,)
```

Cuando se baja la probabilidad de mutación, implica que las principales características del genotipo van a ser siempre similares a lo de los padres, esto es algo nefasto ya que no aparecen nuevas características, por lo tanto la solución no mejor.

Esto lo podemos observar en el primer gráfico, ya que si tenemos unos padres malísimos, los hijos también van a salir muy mal, es por ello que el intervalo de la media es muy amplio. Por otro lado, vemos que el fitness tampoco es el mejor que hemos encontrado.

Caso 9

En este caso realizamos nos basamos en la configuración anterior, y lo que hacemos es aumentar la probabilidad de mutación a 0.4. Los parámetros que hemos usado para realizar este caso son los siguientes:

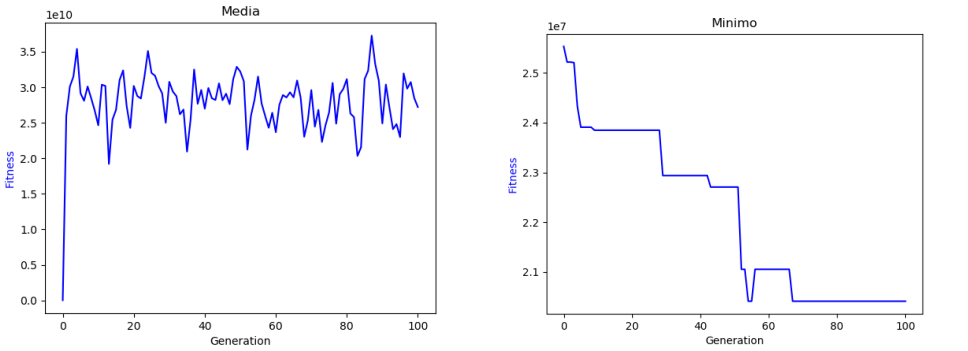
alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.4

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)




```
90      84      2.49085e+10     2.68658e+10     2.04141e+07     8.27091e+10
91      93      3.03801e+10     2.85429e+10     2.04141e+07     8.12935e+10
92      84      2.71579e+10     2.66787e+10     2.04141e+07     7.42692e+10
93      84      2.40983e+10     2.74666e+10     2.04141e+07     7.3686e+10
94      88      2.48131e+10     2.89237e+10     2.04141e+07     7.41046e+10
95      74      2.29895e+10     2.83914e+10     2.04141e+07     8.8369e+10
96      94      3.19442e+10     2.93651e+10     2.04141e+07     8.49056e+10
97      90      2.98043e+10     2.94591e+10     2.04141e+07     7.94893e+10
98      86      3.0727e+10      2.90024e+10     2.04141e+07     8.35466e+10
99      87      2.84568e+10     2.84207e+10     2.04141e+07     7.76454e+10
100     83      2.72146e+10     2.92811e+10     2.04141e+07     8.08341e+10

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(20414073,)
```

Al poner una probabilidad de mutación algo mayor de lo normal, lo que provoca es que aparezcan todo el rato nuevas características, con esto conseguimos nuevas soluciones, y como dicha probabilidad no es tan alta, conseguimos un resultado bastante mejor que en el anterior caso. Aun así vemos en el segundo gráfico que a la función le cuesta converger, además en la generación 56 aproximadamente aparecen nuevas características, las cueles provocan que vuelva a subir el fitness, sin embargo, sobre la generación 70 vuelve a bajar el fitness gracias a las características conseguidas en las generaciones anteriores.

Experimento 2

Para la realización de este experimento hemos usado el fichero *"datos_1.txt"*, éste es un fichero sencillo ya que no tiene muchos datos, aunque es algo mayor al caso anterior.

Caso 1

Al igual que sucedía en el caso 1 del anterior experimento, éste es un caso inicial con unos parámetros "normales", los parámetros que hemos usado para realizar este caso son los mismos que el caso 1 del anterior experimento:

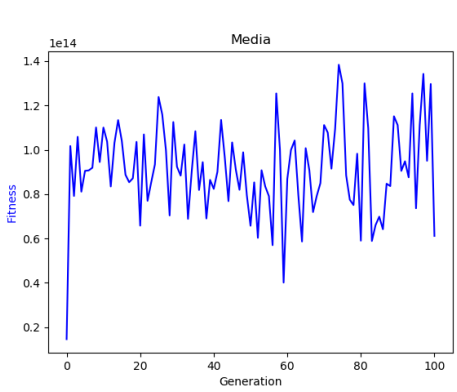
alg_param['cxpb'] = 0.75

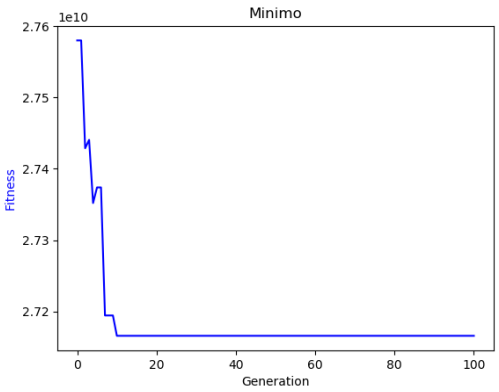
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      80      1.11158e+14     1.86427e+14     2.71658e+10     5.29011e+14
91      89      9.04856e+13     1.7338e+14      2.71658e+10     5.80621e+14
92      69      9.47989e+13     1.78914e+14     2.71658e+10     4.73042e+14
93      76      8.75819e+13     1.7553e+14      2.71658e+10     5.02025e+14
94      75      1.2544e+14      1.97489e+14     2.71658e+10     5.45536e+14
95      81      7.3614e+13      1.63046e+14     2.71658e+10     4.86403e+14
96      90      1.11182e+14     1.87989e+14     2.71658e+10     5.00081e+14
97      85      1.34249e+14     1.99071e+14     2.71658e+10     4.93e+14
98      78      9.50398e+13     1.75727e+14     2.71658e+10     5.92426e+14
99      86      1.29695e+14     1.96475e+14     2.71658e+10     5.25604e+14
100     81      6.11308e+13     1.5176e+14      2.71658e+10     4.80205e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27165751943,)
```

Como podemos observar con estos parámetros la función converge muy rápido, alcanzando el óptimo sobre la generación 10.

Caso 2

Para este caso hemos usado la configuración anterior, pero hemos modificado el número de generaciones de 100 a 300. Los parámetros que hemos usado son los siguientes:

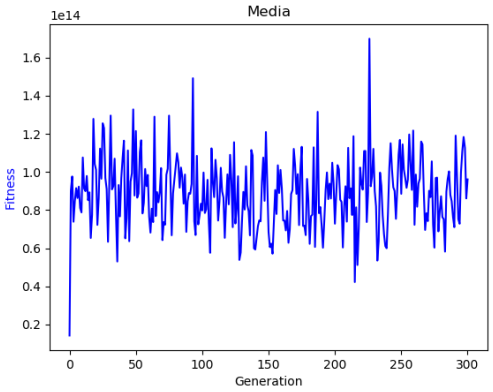
alg_param['cxpb'] = 0.75

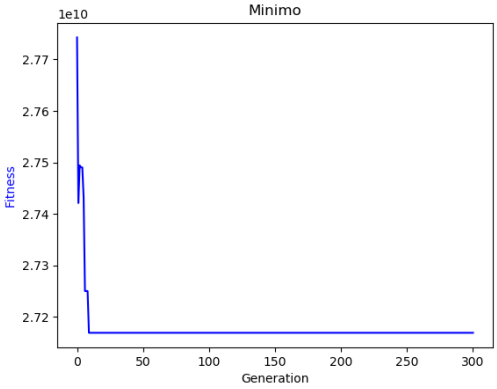
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 300

init_pop = toolbox.population(n=100)






```
290     82      7.10539e+13     1.57361e+14     2.71691e+10     4.75545e+14
291     81      1.19042e+14     1.88272e+14     2.71691e+10     4.64195e+14
292     70      9.79967e+13     1.82817e+14     2.71691e+10     5.90823e+14
293     79      7.52351e+13     1.67391e+14     2.71691e+10     6.07083e+14
294     85      7.28474e+13     1.61311e+14     2.71691e+10     4.6582e+14
295     79      9.97885e+13     1.79397e+14     2.71691e+10     4.72205e+14
296     79      1.10762e+14     1.8627e+14      2.71691e+10     5.33826e+14
297     82      1.18325e+14     1.88152e+14     2.71691e+10     4.75056e+14
298     75      1.12638e+14     1.91049e+14     2.71691e+10     5.79353e+14
299     72      8.61358e+13     1.73659e+14     2.71691e+10     6.45391e+14
300     83      9.60919e+13     1.76528e+14     2.71691e+10     5.23404e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27169137484,)
```

Al igual que sucedía en el caso anterior, la función converge muy rápido, lo bueno que la media permanece estable, es decir la gran mayoría de ellas no están dispersas.

El punto negativo es que el fitness del mejor genotipo es un poco peor que en el caso anterior, posiblemente si volviéramos a ejecutar estos parámetros podríamos obtener una solución igual o un poco mejor que en el caso anterior.

Caso 3

Al igual que en el caso anterior, hemos usado la misma configuración, pero hemos aumentado el número de generaciones a 600. Los parámetros de este caso son los siguientes:

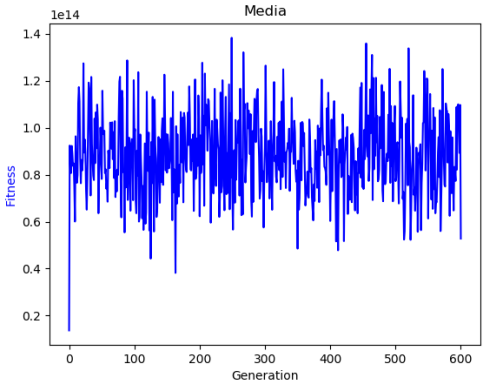
alg_param['cxpb'] = 0.75

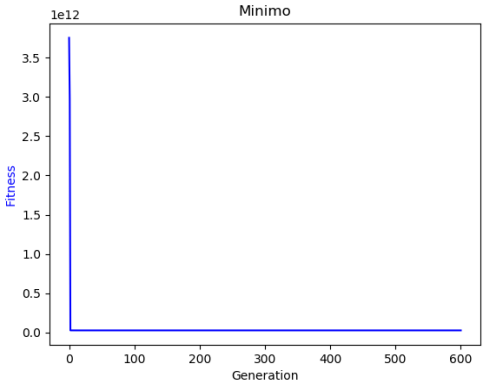
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 600

init_pop = toolbox.population(n=100)






```
590     83      8.3309e+13      1.72383e+14     2.74132e+10     4.7896e+14
591     78      8.1346e+13      1.68376e+14     2.74132e+10     4.74872e+14
592     73      7.73458e+13     1.6276e+14      2.74132e+10     4.89258e+14
593     87      1.08681e+14     1.89165e+14     2.74132e+10     5.34858e+14
594     86      8.21402e+13     1.67229e+14     2.74132e+10     4.81579e+14
595     86      1.09123e+14     1.8954e+14      2.74132e+10     4.89859e+14
596     74      1.09974e+14     1.88983e+14     2.74132e+10     5.47188e+14
597     73      1.05946e+14     1.80953e+14     2.74132e+10     5.02085e+14
598     73      8.9234e+13      1.73928e+14     2.74132e+10     4.8896e+14
599     86      1.09682e+14     1.85762e+14     2.74132e+10     4.81188e+14
600     78      5.27762e+13     1.37838e+14     2.74132e+10     4.73739e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27413225153,)
```

En este caso podemos apreciar que la medias están un poco más dispersas, además la función converge muy rápido, esto significa que el mejor genotipo se obtiene muy pronto, como consecuencia podemos observar que el fitness del mejor genotipo esta vez sí que es más alto que en los casos anteriores.

En conclusión, aumentar el número de generaciones no significa que vayas a obtener mejores resultados.

Caso 4

En este caso hemos usado la configuración inicial del caso 1, pero hemos modificado la población inicial. Los parámetros para realizar este caso son los siguientes:

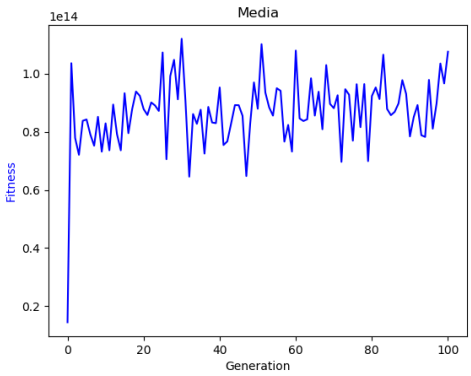
alg_param['cxpb'] = 0.75

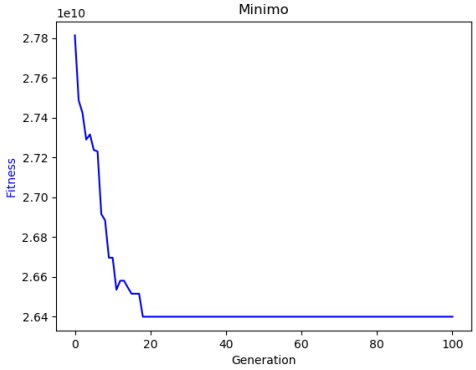
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=300)






```
90      230     7.84366e+13     1.65124e+14     2.63998e+10     5.06194e+14
91      236     8.50978e+13     1.7205e+14      2.63998e+10     6.14687e+14
92      240     8.92327e+13     1.74741e+14     2.63998e+10     4.90697e+14
93      235     7.88322e+13     1.6414e+14      2.63998e+10     5.20877e+14
94      247     7.82727e+13     1.63135e+14     2.63998e+10     5.37022e+14
95      258     9.78956e+13     1.80484e+14     2.63998e+10     4.84001e+14
96      207     8.11025e+13     1.68793e+14     2.63998e+10     6.37247e+14
97      230     8.97838e+13     1.7601e+14      2.63998e+10     5.91534e+14
98      249     1.03534e+14     1.84585e+14     2.63998e+10     4.83992e+14
99      245     9.6681e+13      1.77055e+14     2.63998e+10     5.79421e+14
100     247     1.07604e+14     1.85636e+14     2.63998e+10     5.06295e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(26399834408,)
```

Cuando aumentamos la población inicial, obtenemos una mejora de la solución, esto se debe a que hay más soluciones disponibles que en los casos anteriores, es decir, hay más padres y más mutaciones.

En el primer gráfico vemos que la media está un poco dispersa, pero el intervalo se mueve entre 0,8 y 1. Además podemos ver que el fitness también converge muy rápido, ya que a partir de la generación 19 aproximadamente alcanzamos el mínimo de dicha función.

Caso 5

Este caso está basado en el anterior, lo que pasa es que en vez de aumentar la población inicial, hemos aumenta el número de generaciones. Los parámetros que hemos utilizado son los siguientes:

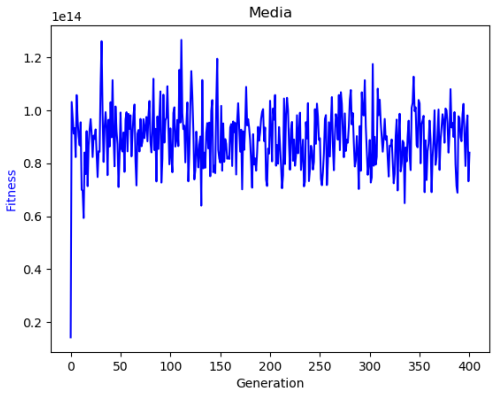
alg_param['cxpb'] = 0.75

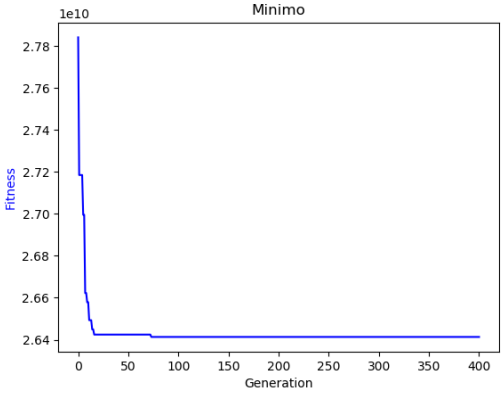
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 400

init_pop = toolbox.population(n=300)






```
390     218     9.70826e+13     1.78265e+14     2.64133e+10     4.90591e+14
391     225     8.98725e+13     1.73847e+14     2.64133e+10     4.95942e+14
392     225     8.83444e+13     1.72946e+14     2.64133e+10     5.02704e+14
393     237     9.84848e+13     1.79888e+14     2.64133e+10     5.88283e+14
394     243     1.02465e+14     1.80724e+14     2.64133e+10     5.25157e+14
395     247     9.12297e+13     1.72629e+14     2.64133e+10     5.69492e+14
396     254     7.89741e+13     1.65222e+14     2.64133e+10     4.94801e+14
397     258     9.34881e+13     1.75992e+14     2.64133e+10     5.05861e+14
398     231     9.81058e+13     1.8009e+14      2.64133e+10     5.02243e+14
399     230     7.31864e+13     1.61016e+14     2.64133e+10     5.82959e+14
400     238     8.402e+13       1.70166e+14     2.64133e+10     4.93206e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(26413270642,)
```

Como podemos observar, el fitness del mejor es algo peor que en caso anterior, por lo tanto, seguimos demostrando que no por aumentar el un poco el número de generaciones vamos a obtener un mejor resultado.

En cuanto a la primera gráfica podemos ver que la medias están muy concentradas, y en cuanto a la segunda gráfica que la función sigue convergiendo muy rápido, alcanzando el mínimo muy pronto.

Caso 6

En este caso hemos vuelto a utilizar la configuración inicial, pero esta vez hemos modificado la probabilidad de cruce. Los parámetros que hemos usado para realizar este caso son los siguientes:

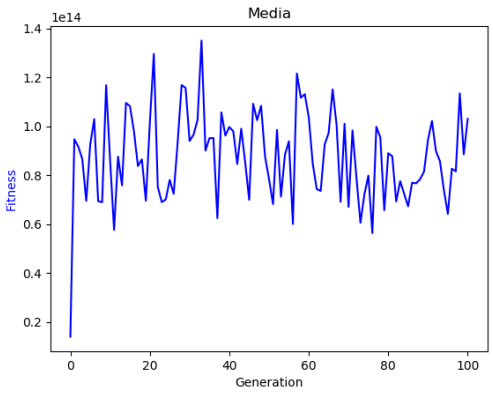
alg_param['cxpb'] = 0.4

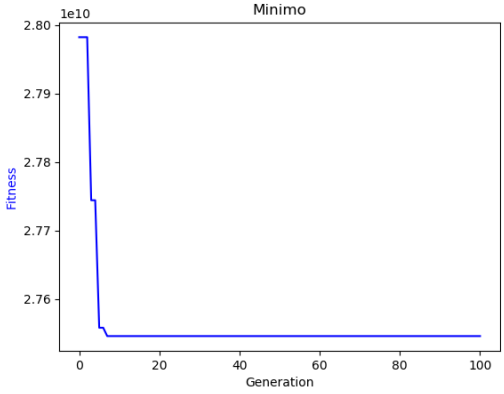
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      51      9.45458e+13     1.78497e+14     2.75462e+10     4.76119e+14
91      46      1.02279e+14     1.82619e+14     2.75462e+10     4.74174e+14
92      51      8.99385e+13     1.7491e+14      2.75462e+10     4.7854e+14
93      49      8.58784e+13     1.72322e+14     2.75462e+10     4.84874e+14
94      59      7.40153e+13     1.63773e+14     2.75462e+10     4.91801e+14
95      50      6.42588e+13     1.53388e+14     2.75462e+10     5.04815e+14
96      45      8.27418e+13     1.7217e+14      2.75462e+10     5.92714e+14
97      53      8.16865e+13     1.69287e+14     2.75462e+10     4.98836e+14
98      62      1.13567e+14     1.87114e+14     2.75462e+10     4.64627e+14
99      50      8.86293e+13     1.67626e+14     2.75462e+10     5.01184e+14
100     58      1.03153e+14     1.84304e+14     2.75462e+10     5.18355e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27546164639,)
```

Como podemos apreciar en el primer gráfico, la media es muy dispersa, más que en los casos anteriores, además converge el mínimo muy rápido, todo esto provoca que el fitness del mejor genotipo sea peor que en los casos anteriores.

En conclusión, si reduces la probabilidad de cruce, estás eliminando características nuevas, por lo que empeoras la solución.

Caso 7

Este caso es similar al anterior, pero en vez de disminuir la probabilidad de cruce, la hemos aumentado. Los parámetros que hemos usado para realizar este caso son los siguientes:

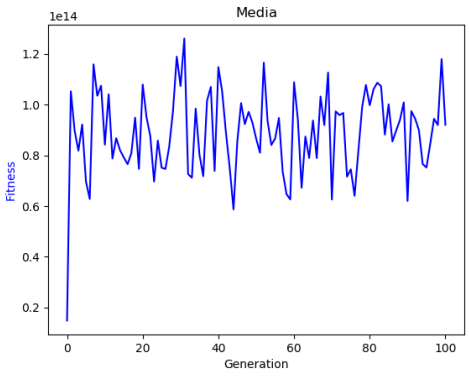
alg_param['cxpb'] = 0.95

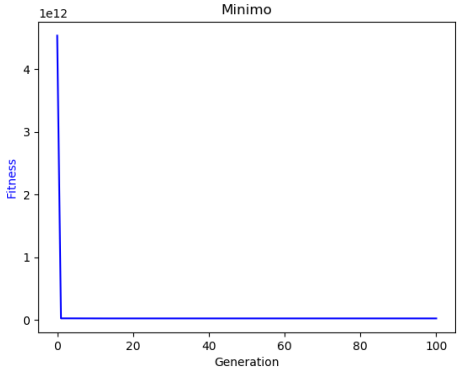
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      94      6.20128e+13     1.48836e+14     2.65835e+10     4.70521e+14
91      98      9.73905e+13     1.78308e+14     2.65835e+10     4.73958e+14
92      96      9.44798e+13     1.78306e+14     2.65835e+10     4.77137e+14
93      92      8.9958e+13      1.75195e+14     2.65835e+10     4.99476e+14
94      98      7.65564e+13     1.63869e+14     2.65835e+10     4.89553e+14
95      94      7.51988e+13     1.61004e+14     2.65835e+10     4.89369e+14
96      96      8.45402e+13     1.67706e+14     2.65835e+10     4.64009e+14
97      100     9.44094e+13     1.7814e+14      2.65835e+10     4.79877e+14
98      99      9.19269e+13     1.79537e+14     2.65835e+10     6.56215e+14
99      94      1.17944e+14     1.88384e+14     2.65835e+10     4.81328e+14
100     98      9.20088e+13     1.74941e+14     2.65835e+10     5.16958e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(26583534496,)
```

Como podemos ver en el fitness del mejor genotipo, lo hemos reducido bastante al caso anterior, esto se debe a que la probabilidad de cruce es mayor, sin embargo, al tener una probabilidad tan alta de cruce la mutación apenas hace efecto, por lo que si se consigue balancear bien ambas probabilidades obtendremos la mejor solución.

Caso 8

Para realizar este caso hemos usado la configuración inicial, y lo que hemos hecho ha sido disminuir la probabilidad de mutación, para realizar este caso hemos usado los siguientes parámetros:

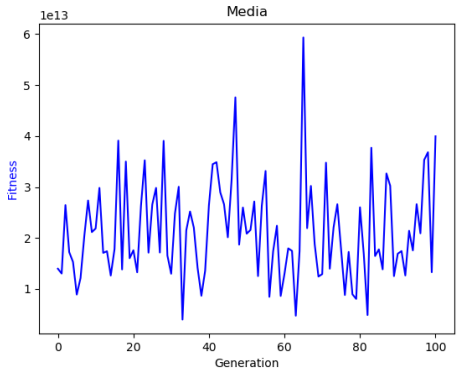
alg_param['cxpb'] = 0.75

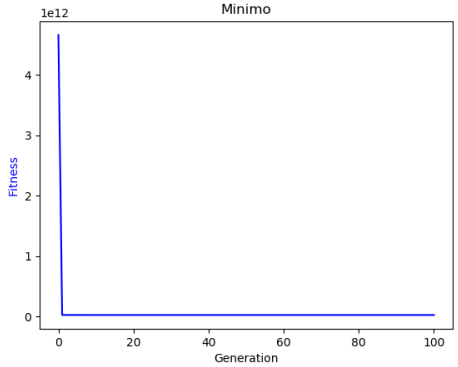
alg_param['mutpb'] = 0.05

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      75      1.69136e+13     8.29363e+13     2.74092e+10     4.56358e+14
91      79      1.74295e+13     8.53431e+13     2.74092e+10     4.60677e+14
92      76      1.26465e+13     7.19633e+13     2.74092e+10     4.57567e+14
93      71      2.14095e+13     9.32488e+13     2.74092e+10     4.49003e+14
94      61      1.75655e+13     8.59974e+13     2.74092e+10     4.58158e+14
95      79      2.66284e+13     1.05665e+14     2.74092e+10     4.85935e+14
96      74      2.09123e+13     9.11886e+13     2.74092e+10     4.47179e+14
97      79      3.53243e+13     1.19894e+14     2.74092e+10     4.86927e+14
98      70      3.68277e+13     1.25351e+14     2.74092e+10     5.24494e+14
99      76      1.32836e+13     7.57503e+13     2.74092e+10     4.95173e+14
100     83      3.99612e+13     1.27381e+14     2.74092e+10     5.12281e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27409161682,)
```

Como era de esperar, al reducir la probabilidad de mutación estamos eliminando nuevas características para nuestra población, por lo que empeora la solución. Además si observamos el primer gráfico, las medias están muy dispersas, lo cual no es lo idóneo para encontrar una buena solución.

Caso 9

En este caso nos hemos basado en el caso anterior, sin embargo lo que hemos hecho ha sido aumentar la probabilidad de mutación, por lo que los parámetros de este caso son los siguientes:

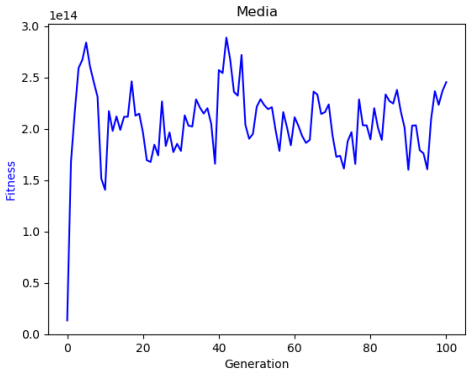
alg_param['cxpb'] = 0.75

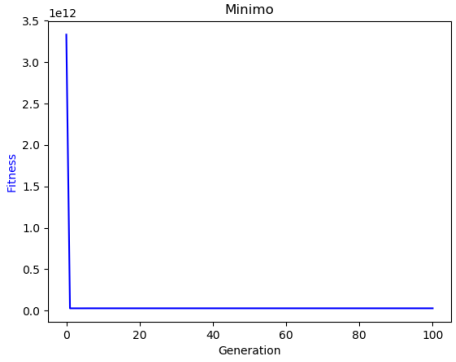
alg_param['mutpb'] = 0.4

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      81      1.60137e+14     2.09045e+14     2.77955e+10     6.09138e+14
91      90      2.0314e+14      2.19285e+14     2.77955e+10     6.0236e+14
92      87      2.03431e+14     2.1632e+14      2.77955e+10     6.50151e+14
93      86      1.79232e+14     2.2473e+14      2.77955e+10     6.45082e+14
94      81      1.76119e+14     2.13785e+14     2.77955e+10     5.93972e+14
95      84      1.60626e+14     2.09489e+14     2.77955e+10     5.95828e+14
96      86      2.09271e+14     2.23473e+14     2.77955e+10     6.05376e+14
97      95      2.36683e+14     2.27036e+14     2.77955e+10     6.63314e+14
98      83      2.23418e+14     2.23478e+14     2.77955e+10     6.24753e+14
99      87      2.3689e+14      2.20722e+14     2.77955e+10     5.83394e+14
100     83      2.45497e+14     2.26115e+14     2.77955e+10     6.30768e+14

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(27795532945,)
```

Podemos observar que el fitness del mejor genotipo es peor que en el caso anterior, esto se debe a que las medias están muy dispersas otra vez, ya que al subir la probabilidad de mutación generamos nuevas características, las cuales pueden empeorar nuestra solución, y esto es lo que sucede.

Además, podemos ver que la función mínimo converge muy rápido, lo cual tampoco es lo idóneo, ya que alcanzamos la solución muy pronto.

Experimento 3

Para la realización de este experimento hemos usado el fichero *"datos_2.txt"*, éste es un fichero normal ya que tiene más datos que en los experimentos probados anteriormente.

Caso 1

Para realizar este caso hemos seguido la configuración que hemos realizado en experimentos anteriores, cuyos parámetros son:

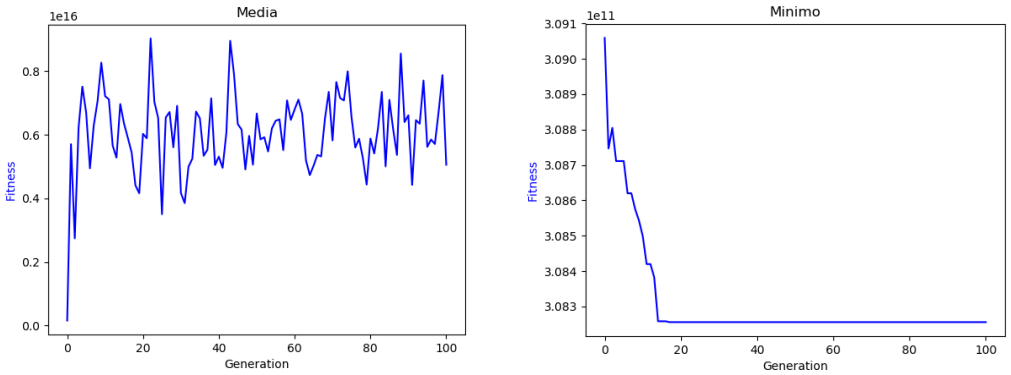
alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)




```
90      79      6.61231e+15     1.22221e+16     3.08256e+11     3.11611e+16
91      76      4.42178e+15     1.04089e+16     3.08256e+11     3.07943e+16
92      74      6.4585e+15      1.21691e+16     3.08256e+11     3.06868e+16
93      80      6.34665e+15     1.20952e+16     3.08256e+11     3.68587e+16
94      89      7.70346e+15     1.30022e+16     3.08256e+11     3.22575e+16
95      75      5.61936e+15     1.16091e+16     3.08256e+11     3.15878e+16
96      86      5.84833e+15     1.15316e+16     3.08256e+11     3.10796e+16
97      77      5.70961e+15     1.15477e+16     3.08256e+11     3.13349e+16
98      87      6.73932e+15     1.23357e+16     3.08256e+11     3.07631e+16
99      80      7.8752e+15      1.33243e+16     3.08256e+11     4.43612e+16
100     77      5.05741e+15     1.11781e+16     3.08256e+11     3.11653e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308255671721,)
```

Tal y como podemos apreciar en la función mínimo a partir de la generación 19 aproximadamente obtenemos el mínimo.

Caso 2

Para realizar este caso no hemos basado en el caso anterior, pero hemos modificado el número de generaciones, los parámetros de este caso son los siguientes:

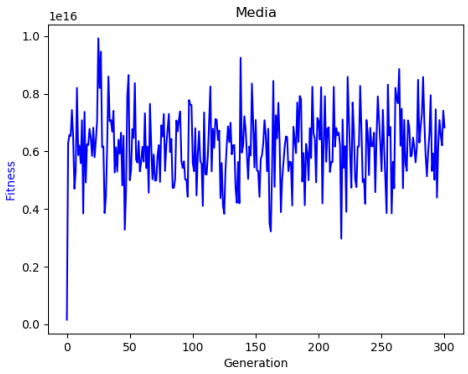
alg_param['cxpb'] = 0.75

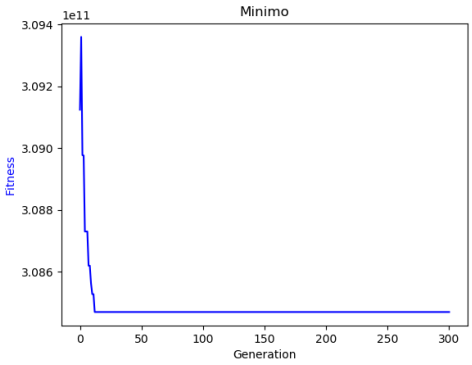
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 300

init_pop = toolbox.population(n=100)






```
290     76      5.32353e+15     1.11265e+16     3.0847e+11      3.16078e+16
291     79      5.92756e+15     1.17568e+16     3.0847e+11      3.13538e+16
292     66      5.01819e+15     1.09099e+16     3.0847e+11      3.06618e+16
293     78      7.4574e+15      1.29269e+16     3.0847e+11      3.13631e+16
294     75      4.39841e+15     1.04787e+16     3.0847e+11      3.16235e+16
295     71      6.18359e+15     1.19996e+16     3.0847e+11      3.14083e+16
296     83      7.08795e+15     1.26321e+16     3.0847e+11      3.27468e+16
297     85      6.5575e+15      1.24419e+16     3.0847e+11      4.4402e+16
298     83      6.20948e+15     1.20508e+16     3.0847e+11      3.15365e+16
299     81      7.41367e+15     1.27346e+16     3.0847e+11      3.09168e+16
300     72      6.82872e+15     1.24632e+16     3.0847e+11      3.26117e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308470113338,)
```

Al igual que pasaba en experimentos anteriores, no por aumentar el número de generaciones vas a obtener un mejor fitness, como se puede apreciar obtenemos un peor fitness que en el caso anterior.

Algo positivo es que en la primera gráfica las medias están muy juntas, es decir, no hay mucha dispersión.

Caso 3

Para realizar este caso hemos usado la configuración inicial, pero hemos aumentado aún más el número de generaciones, los parámetros de este caso son los siguientes:

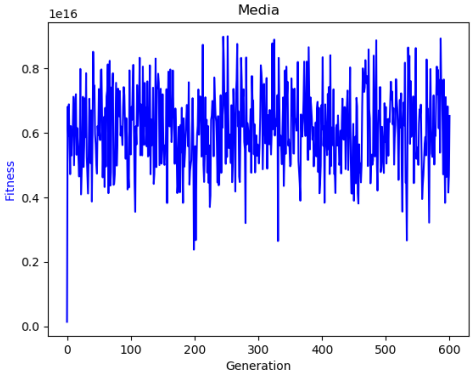
alg_param['cxpb'] = 0.75

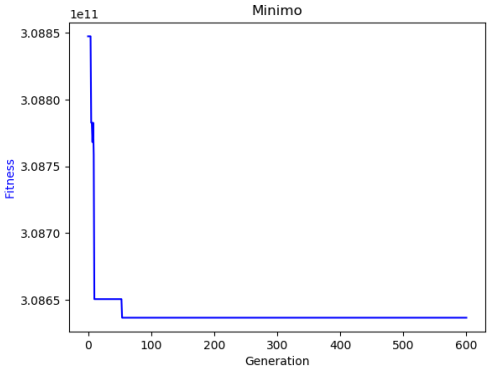
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 600

init_pop = toolbox.population(n=100)






```
590     78      4.71367e+15     1.08025e+16     3.08637e+11     3.06915e+16
591     88      7.65395e+15     1.29191e+16     3.08637e+11     3.16936e+16
592     69      5.32742e+15     1.14236e+16     3.08637e+11     3.26485e+16
593     68      3.83181e+15     9.91379e+15     3.08637e+11     3.02635e+16
594     72      7.10744e+15     1.24938e+16     3.08637e+11     3.20252e+16
595     73      4.63019e+15     1.07143e+16     3.08637e+11     3.60439e+16
596     63      4.7293e+15      1.08401e+16     3.08637e+11     3.15985e+16
597     81      6.81962e+15     1.24906e+16     3.08637e+11     3.16189e+16
598     75      4.15119e+15     1.02923e+16     3.08637e+11     3.09722e+16
599     79      4.70128e+15     1.07788e+16     3.08637e+11     3.10074e+16
600     83      6.52557e+15     1.22843e+16     3.08637e+11     3.18905e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308636684332,)
```

Como bien hemos comentado en el caso anterior, no por aumentar el número de generaciones vamos a obtener una mejor solución, en este caso vuelve a suceder lo mismo el fitness es peor que en los casos anteriores.

Al igual que sucedía en el caso anterior, obtenemos algo bueno y es que la media está bien limitada, no hay mucha desviación. Por lo contrario, la función mínimo decrece medianamente rápido ya que para la generación 80 hemos alcanzado el mínimo.

Caso 4

En este caso hemos vuelto a la configuración inicial, sin embargo, hemos modificado la población inicial, por lo que los parámetros para este caso son los siguientes:

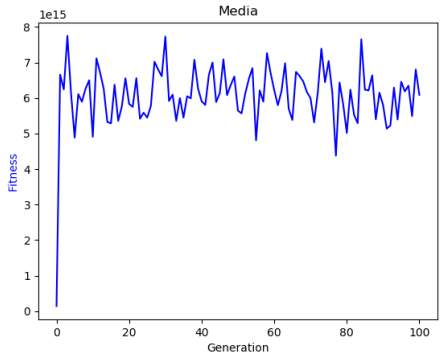
alg_param['cxpb'] = 0.75

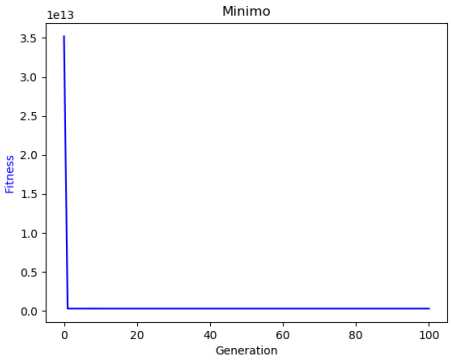
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=300)






```
90      236     5.80794e+15     1.16554e+16     3.08376e+11     3.10579e+16
91      220     5.14064e+15     1.12938e+16     3.08376e+11     4.29934e+16
92      243     5.22509e+15     1.12878e+16     3.08376e+11     3.18846e+16
93      245     6.29231e+15     1.20925e+16     3.08376e+11     3.21457e+16
94      249     5.39584e+15     1.13959e+16     3.08376e+11     3.20009e+16
95      249     6.46089e+15     1.2303e+16      3.08376e+11     3.70759e+16
96      239     6.18759e+15     1.1887e+16      3.08376e+11     3.17785e+16
97      243     6.34871e+15     1.21856e+16     3.08376e+11     3.20334e+16
98      237     5.49114e+15     1.16174e+16     3.08376e+11     4.07788e+16
99      249     6.80456e+15     1.24335e+16     3.08376e+11     3.18109e+16
100     228     6.09143e+15     1.18796e+16     3.08376e+11     3.12561e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308376389209,)
```

Al aumentar la población inicial tenemos más padres, por lo que podemos obtener mejores hijos, sin embargo, no mejoramos el fitness del primer caso.

Un punto negativo es que converge muy rápido la función mínimo, consiguiendo así un mínimo bastante pronto.

En conclusión, aunque aumentemos la población inicial depende del problema si mejora la solución o no.

Caso 5

En este caso usamos la configuración anterior, pero aumentamos el número de generaciones para saber si mejora la solución, los parámetros que hemos usado son los siguientes:

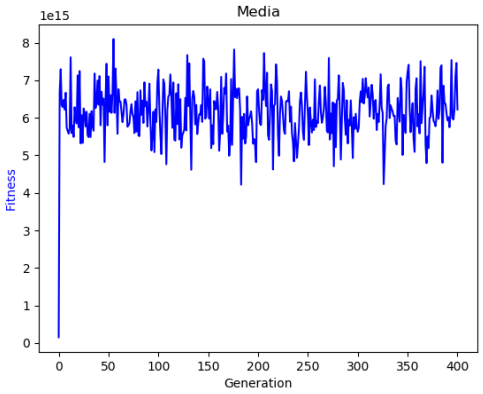
alg_param['cxpb'] = 0.75

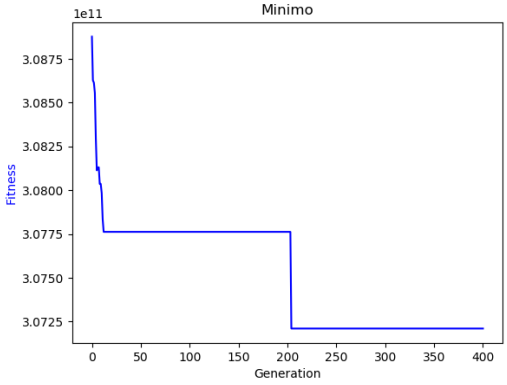
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 400

init_pop = toolbox.population(n=300)






```
390     248     5.93137e+15     1.17372e+16     3.07211e+11     3.17707e+16
391     246     6.01371e+15     1.18747e+16     3.07211e+11     4.37105e+16
392     242     5.7483e+15      1.17637e+16     3.07211e+11     4.4171e+16
393     237     6.10446e+15     1.1947e+16      3.07211e+11     4.15069e+16
394     235     7.53732e+15     1.30827e+16     3.07211e+11     4.49196e+16
395     240     5.98512e+15     1.19038e+16     3.07211e+11     4.23557e+16
396     244     5.95476e+15     1.17597e+16     3.07211e+11     3.17804e+16
397     229     6.23597e+15     1.204e+16       3.07211e+11     4.30492e+16
398     247     7.11789e+15     1.24759e+16     3.07211e+11     3.11533e+16
399     235     7.45882e+15     1.26373e+16     3.07211e+11     3.33404e+16
400     241     6.22027e+15     1.19326e+16     3.07211e+11     3.40476e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(307210539534,)
```

Como se puede observar en el fitness del mejor genotipo, hemos conseguido una mejor solución que en el caso anterior, esto se debe a que a partir de la generación 200 el fitness vuelve a bajar, lo cual es lo que estamos buscando.

Por lo tanto, como ya hemos mencionado, dependiendo del problema aumentar el número de generaciones nos puede beneficiar para poder encontrar una mejor solución.

Caso 6

En este caso hemos vuelto a la configuración inicial, lo que pasa es que hemos modificado la probabilidad de cruce, disminuyéndola. Los parámetros que hemos usado para realizar este caso son los siguientes:

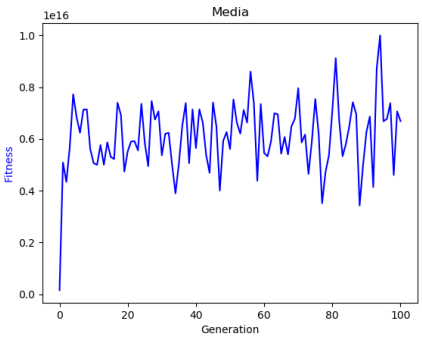
alg_param['cxpb'] = 0.4

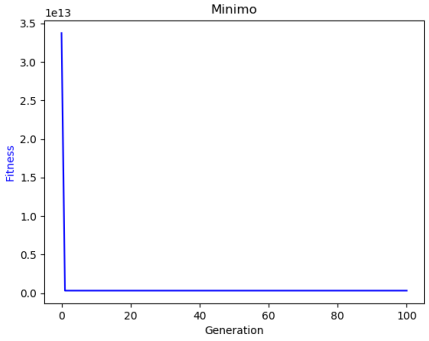
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      55      6.27361e+15     1.21728e+16     3.08795e+11     3.11111e+16
91      55      6.86556e+15     1.25656e+16     3.08795e+11     3.11292e+16
92      39      4.141e+15       1.0267e+16      3.08795e+11     3.12785e+16
93      53      8.70364e+15     1.36252e+16     3.08795e+11     3.1768e+16
94      59      9.99643e+15     1.41001e+16     3.08795e+11     4.62276e+16
95      46      6.68104e+15     1.23557e+16     3.08795e+11     3.90527e+16
96      58      6.77297e+15     1.24013e+16     3.08795e+11     3.23827e+16
97      48      7.38016e+15     1.27889e+16     3.08795e+11     3.18972e+16
98      47      4.60642e+15     1.10631e+16     3.08795e+11     4.45474e+16
99      58      7.06084e+15     1.25739e+16     3.08795e+11     3.1535e+16
100     51      6.68888e+15     1.2674e+16      3.08795e+11     4.31936e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308794718530,)
```

En este caso volvemos a empeorar el fitness, esto se debe a que al tener una baja probabilidad de cruce estamos eliminando muchas posibles soluciones que podrían mejorar nuestra solución, por lo tanto, no es bueno tener una baja probabilidad de cruce.

Caso 7

En este caso hemos usado la misma configuración que en el caso anterior, pero hemos aumentado la probabilidad de cruce, los parámetros que hemos utilizado son los siguientes:

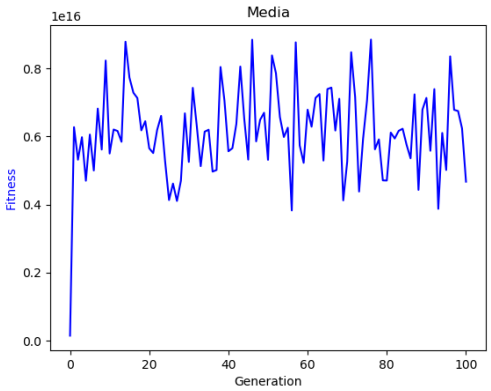
alg_param['cxpb'] = 0.95

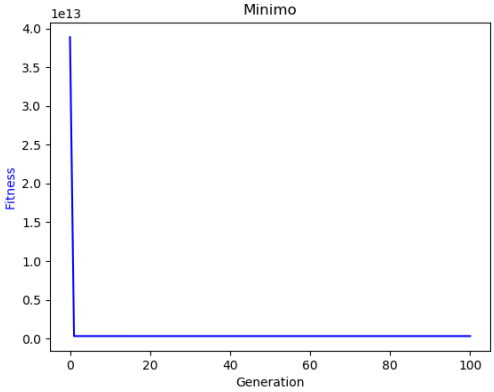
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      96      7.13115e+15     1.26985e+16     3.0773e+11      3.14572e+16
91      98      5.57871e+15     1.14023e+16     3.0773e+11      3.23347e+16
92      96      7.38979e+15     1.27916e+16     3.0773e+11      3.12281e+16
93      99      3.87282e+15     1.00208e+16     3.0773e+11      3.10782e+16
94      96      6.10167e+15     1.18414e+16     3.0773e+11      3.1508e+16
95      100     5.01454e+15     1.071e+16       3.0773e+11      3.10095e+16
96      95      8.35108e+15     1.34031e+16     3.0773e+11      3.22601e+16
97      95      6.781e+15       1.22682e+16     3.0773e+11      3.11242e+16
98      99      6.74588e+15     1.23461e+16     3.0773e+11      3.04716e+16
99      96      6.23517e+15     1.20971e+16     3.0773e+11      3.09769e+16
100     94      4.67078e+15     1.07029e+16     3.0773e+11      3.02363e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(307729681935,)
```

Como podemos observar hemos mejorado el fitness respecto al caso anterior, gracias a que la probabilidad alta de cruce genera nuevas características, las cuales mejoran la solución.

Sin embargo, si observamos la segunda gráfica, obtenemos el mínimo de la función muy pronto, lo cual no es lo ideal.

Caso 8

Para realizar este caso hemos usado la configuración inicial, pero hemos cambiado la probabilidad de mutación, disminuyéndola. Los parámetros que hemos usado para realizar este caso son los siguientes:

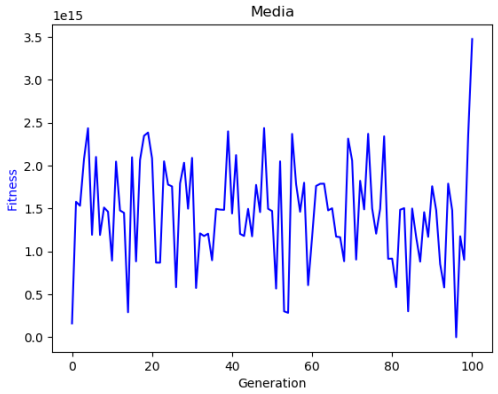
alg_param['cxpb'] = 0.75

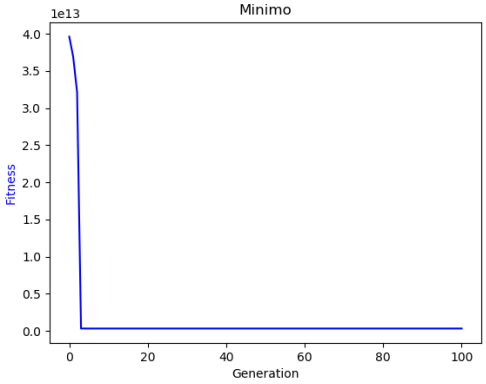
alg_param['mutpb'] = 0.05

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      87      1.76036e+15     6.96765e+15     3.08908e+11     2.99394e+16
91      72      1.48164e+15     6.45773e+15     3.08908e+11     3.01177e+16
92      78      8.52432e+14     4.84657e+15     3.08908e+11     2.92811e+16
93      77      5.80509e+14     4.0614e+15      3.08908e+11     2.90283e+16
94      83      1.79061e+15     7.08989e+15     3.08908e+11     3.11788e+16
95      81      1.48076e+15     6.45822e+15     3.08908e+11     3.08564e+16
96      74      3.08908e+11     0               3.08908e+11     3.08908e+11
97      71      1.17701e+15     5.76795e+15     3.08908e+11     3.03892e+16
98      86      9.02639e+14     5.13404e+15     3.08908e+11     3.15468e+16
99      78      2.35752e+15     7.99768e+15     3.08908e+11     3.07695e+16
100     83      3.47609e+15     9.71174e+15     3.08908e+11     4.21026e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308908050757,)
```

Tal y como observamos el fitness, hemos obtenido una solución poco eficiente, ya que si disminuimos demasiado la probabilidad de mutación estamos quitando nuevas características a nuestra población, lo cual no es lo deseable.

Caso 9

Para realizar este caso nos hemos basado en el caso anterior, pero esta vez hemos aumentado la probabilidad de mutación. Para realizar este caso hemos usado los siguientes parámetros:

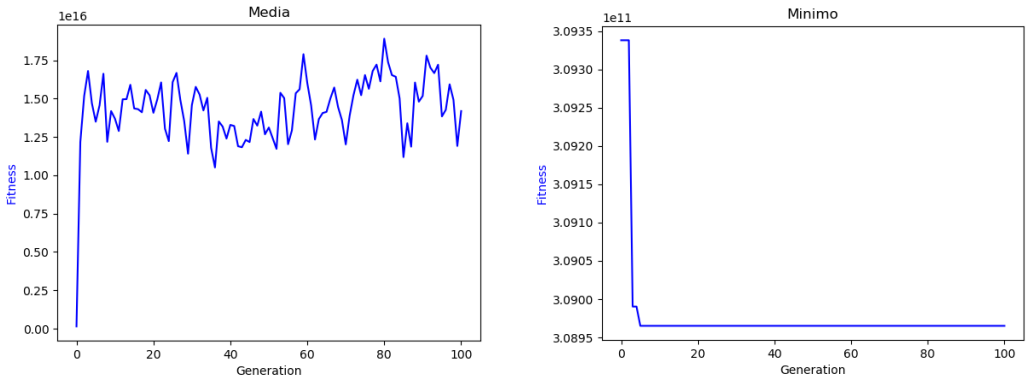
alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.4

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)




```
90      90      1.51704e+16     1.54982e+16     3.08965e+11     4.46301e+16
91      89      1.77998e+16     1.42189e+16     3.08965e+11     4.69352e+16
92      79      1.70153e+16     1.53181e+16     3.08965e+11     4.48296e+16
93      85      1.66609e+16     1.51601e+16     3.08965e+11     4.56286e+16
94      91      1.72067e+16     1.49797e+16     3.08965e+11     4.52784e+16
95      85      1.38374e+16     1.51129e+16     3.08965e+11     4.46295e+16
96      83      1.42676e+16     1.47773e+16     3.08965e+11     4.07026e+16
97      82      1.59294e+16     1.60079e+16     3.08965e+11     4.69934e+16
98      87      1.49195e+16     1.58003e+16     3.08965e+11     4.63011e+16
99      78      1.19099e+16     1.47523e+16     3.08965e+11     4.52456e+16
100     90      1.41865e+16     1.48413e+16     3.08965e+11     4.16053e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(308965146900,)
```

Otra cosa que debemos de tener en cuenta es que no por aumentar mucho la probabilidad de mutación vamos a conseguir una mejor solución, es más todo lo contrario obtenemos una peor solución que en el caso anterior, esto se debe a que genera todo el rato nuevas característica, por lo que no se quedan unas características “fijas” si no que éstas están todo el rato cambiando, lo cual no es lo deseable.

Experimento 4

Para la realización de este experimento hemos usado el fichero *"datos_5.txt"*, éste es un fichero complejo ya que tiene más datos que en los experimentos probados anteriormente.

Caso 1

Al igual que en experimentos anteriores en este caso hemos ejecutado la misma configuración inicial, cuyos parámetros son los siguientes:

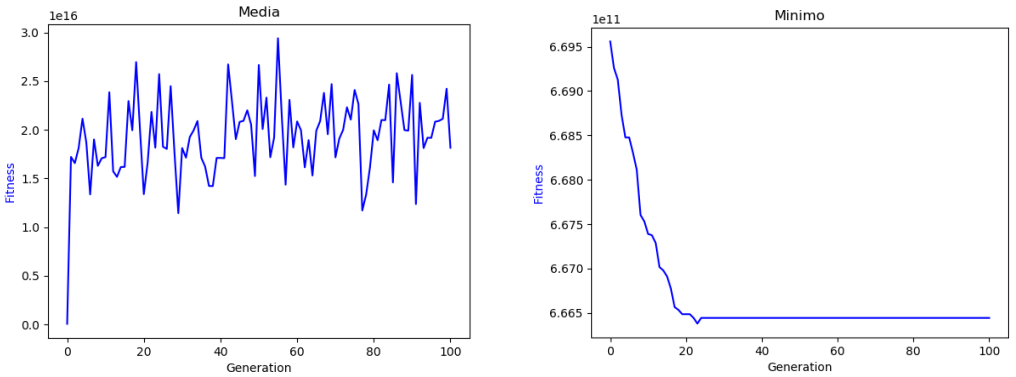
alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)




```
90      85      2.56497e+16     4.21846e+16     6.66444e+11     9.79414e+16
91      78      1.23729e+16     3.20107e+16     6.66444e+11     9.68493e+16
92      83      2.27842e+16     4.0553e+16      6.66444e+11     9.88951e+16
93      87      1.81279e+16     3.74333e+16     6.66444e+11     9.75632e+16
94      73      1.9189e+16      3.79404e+16     6.66444e+11     9.77776e+16
95      78      1.91797e+16     3.79164e+16     6.66444e+11     9.73995e+16
96      78      2.08362e+16     3.92374e+16     6.66444e+11     9.72777e+16
97      79      2.09227e+16     3.93991e+16     6.66444e+11     9.78199e+16
98      89      2.11375e+16     3.90809e+16     6.66444e+11     9.93345e+16
99      86      2.42315e+16     4.04936e+16     6.66444e+11     1.00012e+17
100     75      1.81574e+16     3.77911e+16     6.66444e+11     1.28291e+17

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(666444448189,)
```

Tal y como podemos observar en el primer gráfico, las medias están un poco dispersas, por otro lado, el segundo gráfico alcanza el mínimo en la generación 21 aproximadamente.

Caso 2

Para realizar este caso nos hemos basado en la configuración anterior, pero hemos modificado el número de generaciones, los parámetros que hemos utilizado son los siguientes:

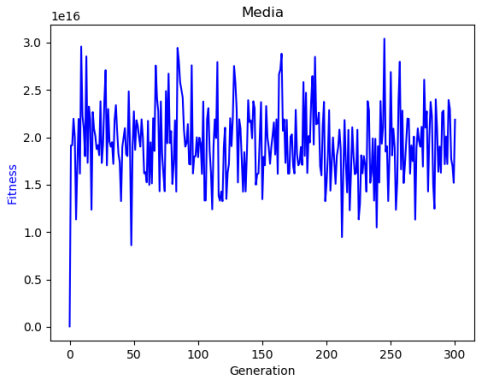
alg_param['cxpb'] = 0.75

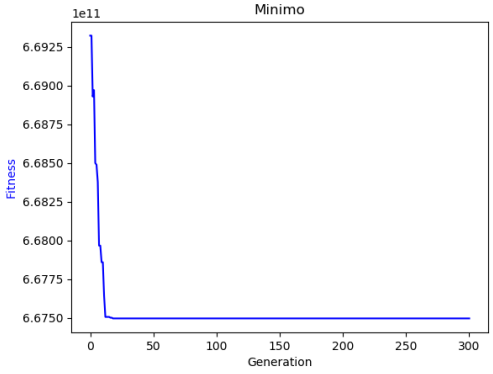
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 300

init_pop = toolbox.population(n=100)






```
290     85      2.26184e+16     3.98917e+16     6.67496e+11     1.06084e+17
291     83      2.28144e+16     4.06018e+16     6.67496e+11     9.75852e+16
292     79      1.71671e+16     3.6645e+16      6.67496e+11     9.74494e+16
293     83      2.00889e+16     3.89703e+16     6.67496e+11     9.99495e+16
294     83      1.71826e+16     3.66765e+16     6.67496e+11     9.70585e+16
295     87      2.39377e+16     4.07799e+16     6.67496e+11     9.78322e+16
296     83      2.29369e+16     4.00669e+16     6.67496e+11     9.71469e+16
297     88      1.77236e+16     3.66135e+16     6.67496e+11     1.19512e+17
298     80      1.69665e+16     3.62381e+16     6.67496e+11     9.69923e+16
299     89      1.52053e+16     3.42528e+16     6.67496e+11     9.73092e+16
300     78      2.18611e+16     3.98832e+16     6.67496e+11     9.75364e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(667496121446,)
```

Al igual que sucedía en casos anteriores, no por aumentar un poco el número de generaciones vamos a obtener una mejor solución, en este caso sigue sucediendo lo mismo.

Por otro lado, la medias sí que están un poco más limitadas y no están tan dispersas.

Caso 3

Para la realización de este caso hemos usado la configuración inicial, lo único que hemos cambiado ha sido el número de generaciones, aumentándola. Los parámetros que hemos usado son los siguientes:

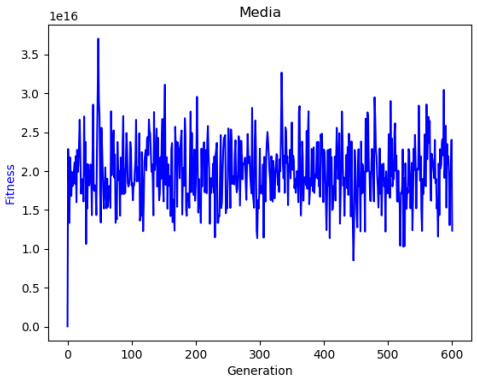
alg_param['cxpb'] = 0.75

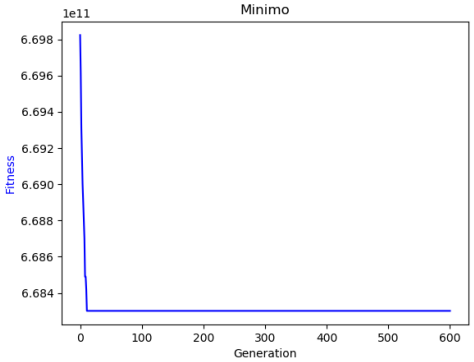
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 600

init_pop = toolbox.population(n=100)






```
590     81      2.58265e+16     4.22486e+16     6.68301e+11     9.89256e+16
591     89      1.53097e+16     3.4215e+16      6.68301e+11     9.66557e+16
592     91      2.34795e+16     4.08058e+16     6.68301e+11     9.79733e+16
593     82      2.09023e+16     3.93635e+16     6.68301e+11     9.75952e+16
594     84      2.19124e+16     4.00972e+16     6.68301e+11     9.76153e+16
595     79      2.00765e+16     3.83123e+16     6.68301e+11     9.74636e+16
596     70      1.30556e+16     3.28484e+16     6.68301e+11     1.20511e+17
597     85      1.98558e+16     3.85171e+16     6.68301e+11     9.81571e+16
598     84      1.99716e+16     3.87402e+16     6.68301e+11     9.87486e+16
599     94      2.40139e+16     4.10955e+16     6.68301e+11     9.85727e+16
600     83      1.23467e+16     3.19442e+16     6.68301e+11     9.74601e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(668301134520,)
```

Aunque las medias están más acotadas que en los casos anteriores, la solución es peor, al igual que antes no por aumentar un poco el número de generaciones mejoramos la solución.

Caso 4

En este caso hemos usado la configuración inicial, pero esta vez lo que hemos aumentado ha sido la población inicial, los parámetros para la realización de este caso son los siguientes:

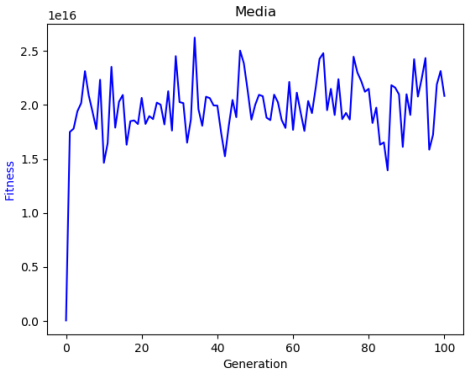
alg_param['cxpb'] = 0.75

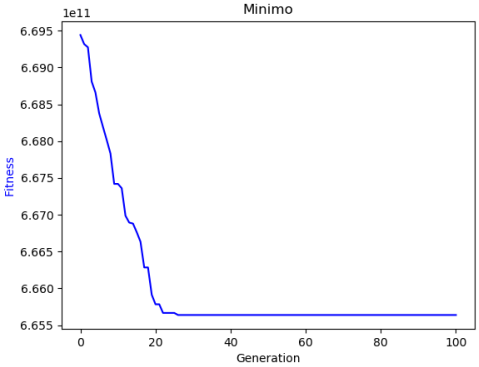
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=300)






```
90      227     2.09908e+16     3.95311e+16     6.65639e+11     1.00889e+17
91      235     1.90784e+16     3.78305e+16     6.65639e+11     9.84389e+16
92      242     2.42334e+16     4.14756e+16     6.65639e+11     1.21409e+17
93      229     2.07773e+16     3.91389e+16     6.65639e+11     1.28394e+17
94      258     2.24218e+16     4.01105e+16     6.65639e+11     1.23549e+17
95      231     2.43414e+16     4.13109e+16     6.65639e+11     1.06319e+17
96      234     1.58639e+16     3.50649e+16     6.65639e+11     1.19293e+17
97      247     1.72398e+16     3.68573e+16     6.65639e+11     1.3123e+17
98      243     2.1893e+16      4.00291e+16     6.65639e+11     9.98553e+16
99      242     2.31417e+16     4.05337e+16     6.65639e+11     1.29344e+17
100     233     2.08377e+16     3.90459e+16     6.65639e+11     9.76547e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(665638737921,)
```

Tal y como podemos observar en el fitness, ésta es la mejor solución que hemos encontrado de momento para este problema, es decir, que para este problema aumentar la población inicial sí que nos ha servido para mejorar la solución, esto se debe a que hay más padres, por lo tanto hay mejores características que pueden heredar los hijos.

Caso 5

En este caso hemos usado la configuración anterior, pero hemos aumentado el número de generaciones. Los parámetros para la realización de este caso de uso son los siguientes:

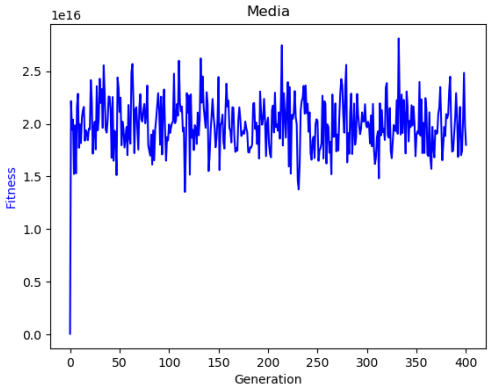
alg_param['cxpb'] = 0.75

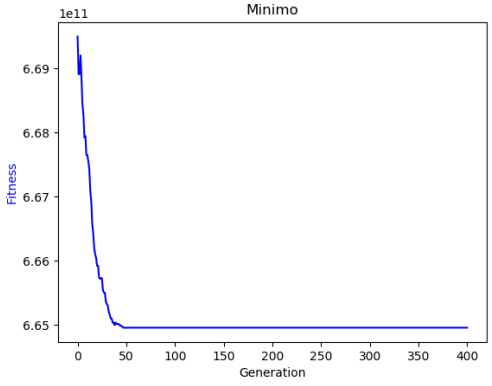
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 400

init_pop = toolbox.population(n=300)






```
390     249     2.29093e+16     4.0359e+16      6.64958e+11     9.84789e+16
391     240     2.05643e+16     3.87887e+16     6.64958e+11     1.29522e+17
392     234     1.68517e+16     3.6211e+16      6.64958e+11     9.9185e+16
393     255     1.96217e+16     3.83674e+16     6.64958e+11     1.14923e+17
394     235     2.16014e+16     3.95085e+16     6.64958e+11     9.83e+16
395     241     1.7023e+16      3.65604e+16     6.64958e+11     1.11224e+17
396     243     1.74156e+16     3.70865e+16     6.64958e+11     1.33926e+17
397     248     2.03693e+16     3.89207e+16     6.64958e+11     9.82525e+16
398     239     2.48383e+16     4.12255e+16     6.64958e+11     9.84129e+16
399     242     1.98206e+16     3.84586e+16     6.64958e+11     1.31781e+17
400     237     1.79975e+16     3.72565e+16     6.64958e+11     1.10368e+17

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(664957543874,)
```

Al igual que sucedía en el caso anterior, hemos conseguido volver a mejorar la solución aumentando el número de generaciones, esto se debe a que por norma general, a mayor número de generaciones más posibilidad de evolucionar tienes.

Además podemos apreciar que en la primera gráfica las medias están muy bien acotadas.

Caso 6

Para la realización de este caso hemos usado la configuración inicial, y le hemos disminuido la probabilidad de cruce. Los parámetros para la realización de este caso son los siguientes:

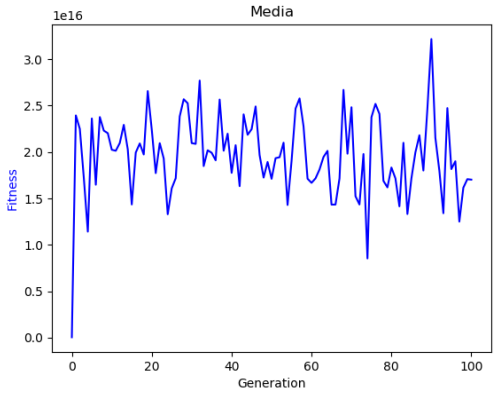
alg_param['cxpb'] = 0.4

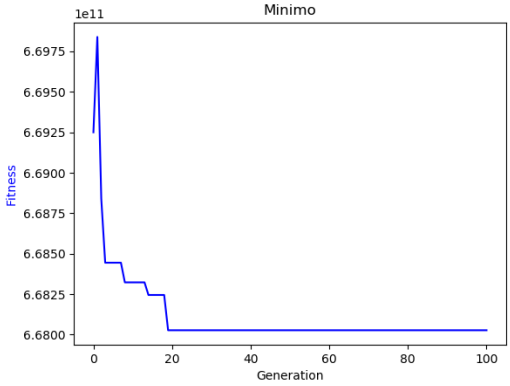
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      57      3.21764e+16     4.49179e+16     6.68027e+11     1.23534e+17
91      59      2.15375e+16     3.94965e+16     6.68027e+11     9.7351e+16
92      53      1.79899e+16     3.66673e+16     6.68027e+11     9.65284e+16
93      51      1.34139e+16     3.32486e+16     6.68027e+11     9.95086e+16
94      53      2.47381e+16     4.17393e+16     6.68027e+11     9.83564e+16
95      56      1.81669e+16     3.75121e+16     6.68027e+11     9.74148e+16
96      46      1.90092e+16     3.80226e+16     6.68027e+11     9.73679e+16
97      40      1.25122e+16     3.1452e+16      6.68027e+11     9.66203e+16
98      64      1.61653e+16     3.57223e+16     6.68027e+11     9.78267e+16
99      57      1.70821e+16     3.64637e+16     6.68027e+11     9.70137e+16
100     54      1.70322e+16     3.63574e+16     6.68027e+11     9.76425e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(668027338201,)
```

Como podemos apreciar el fitness empeora, esto se debe a que hay un menor cruce entre los padres, y por lo tanto tenemos menos características “buenas” que seguir evolucionando.

Además como se puede apreciar en el primer gráfico las medias no están acotadas, es decir, hay un poco de dispersión.

Caso 7

Este caso está basado en el anterior, pero esta vez lo que hacemos es aumentar la probabilidad de cruce, los parámetros de este caso son los siguientes:

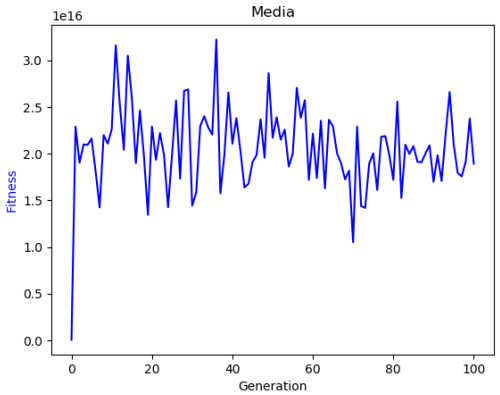
alg_param['cxpb'] = 0.95

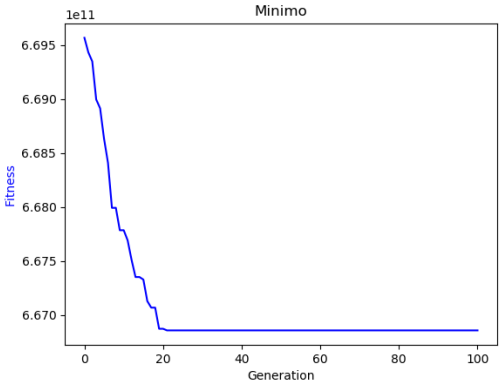
alg_param['mutpb'] = 0.2

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      90      1.70083e+16     3.63048e+16     6.66856e+11     9.80164e+16
91      98      1.98422e+16     3.80009e+16     6.66856e+11     1.00662e+17
92      96      1.70935e+16     3.64887e+16     6.66856e+11     9.82164e+16
93      93      2.21511e+16     3.98302e+16     6.66856e+11     9.92814e+16
94      97      2.66209e+16     4.38802e+16     6.66856e+11     1.32312e+17
95      96      2.1017e+16      3.90123e+16     6.66856e+11     9.71277e+16
96      98      1.79554e+16     3.69848e+16     6.66856e+11     1.01738e+17
97      96      1.7578e+16      3.74023e+16     6.66856e+11     1.23826e+17
98      96      1.91766e+16     3.70077e+16     6.66856e+11     1.11062e+17
99      98      2.37738e+16     4.11983e+16     6.66856e+11     1.04496e+17
100     94      1.89359e+16     3.73947e+16     6.66856e+11     1.05945e+17

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(666856161883,)
```

Como podemos observar, obtenemos un mejor fitness que en el caso anterior, esto se debe a lo comentado en el caso anterior, al tener mayor probabilidad de cruce, las características “buenas” sí que se pasan a los hijos, permitiendo así mejorar la solución.

Caso 8

Este caso utiliza la configuración inicial, pero hemos hecho un cambio sobre la probabilidad de mutación, disminuyéndola. Los parámetros de este caso son los siguientes:

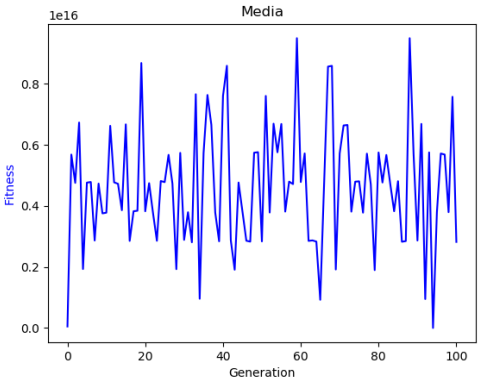
alg_param['cxpb'] = 0.75

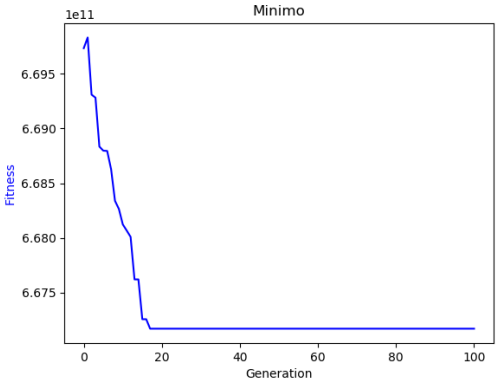
alg_param['mutpb'] = 0.05

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      80      2.86744e+15     1.63039e+16     6.67169e+11     9.78211e+16
91      67      6.68994e+15     2.43847e+16     6.67169e+11     9.71548e+16
92      67      9.48976e+14     9.43555e+15     6.67169e+11     9.48316e+16
93      78      5.7558e+15      2.27806e+16     6.67169e+11     9.70623e+16
94      78      6.67169e+11     0               6.67169e+11     6.67169e+11
95      80      3.80017e+15     1.86147e+16     6.67169e+11     9.64426e+16
96      77      5.71951e+15     2.26368e+16     6.67169e+11     9.5977e+16
97      75      5.68522e+15     2.25021e+16     6.67169e+11     9.64535e+16
98      70      3.79719e+15     1.85994e+16     6.67169e+11     9.5705e+16
99      73      7.57935e+15     2.5704e+16      6.67169e+11     9.69767e+16
100     72      2.82137e+15     1.60393e+16     6.67169e+11     9.44351e+16

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(667169085517,)
```

Cuando disminuimos la probabilidad de mutación, estamos eliminando características nuevas, por lo que la solución que nos da, por norma general va a ser peor.

Además, como podemos apreciar en el primer gráfico la medias no están acotadas, lo cual, no es lo ideal.

Caso 9

Para realizar este caso hemos usado la configuración anterior, aunque hemos modificado la probabilidad de mutación, aumentándola. Los parámetros para realizar este caso son los siguientes:

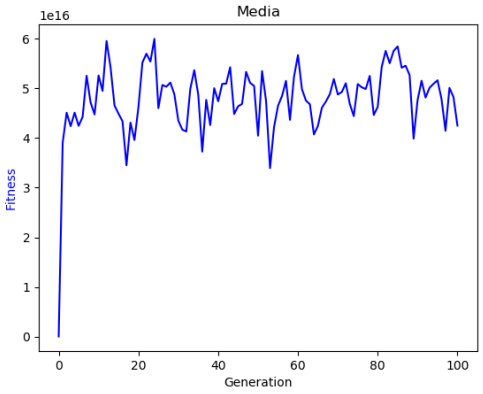
alg_param['cxpb'] = 0.75

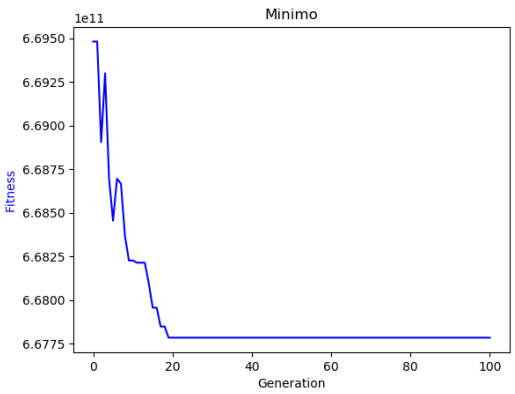
alg_param['mutpb'] = 0.4

alg_param['pop_size'] = 25

alg_param['ngen'] = 100

init_pop = toolbox.population(n=100)






```
90      86      4.76248e+16     4.70648e+16     6.67786e+11     1.16061e+17
91      74      5.15426e+16     4.61867e+16     6.67786e+11     1.32418e+17
92      83      4.81555e+16     4.63612e+16     6.67786e+11     1.30762e+17
93      85      5.01175e+16     4.84022e+16     6.67786e+11     1.32503e+17
94      84      5.09314e+16     4.88411e+16     6.67786e+11     1.33506e+17
95      88      5.1634e+16      4.60353e+16     6.67786e+11     1.28511e+17
96      78      4.78799e+16     4.75726e+16     6.67786e+11     1.33163e+17
97      84      4.1458e+16      4.62353e+16     6.67786e+11     1.27102e+17
98      89      5.00999e+16     4.84602e+16     6.67786e+11     1.31095e+17
99      82      4.82412e+16     4.93135e+16     6.67786e+11     1.33026e+17
100     82      4.25045e+16     4.93918e+16     6.67786e+11     1.31961e+17

Penalización Mejor Genotipo: 1

Fitness Mejor Genotipo(667785588286,)
```

Si aumentamos la probabilidad de mutación no significa que vaya a mejorar la solución del problema, como sucede en este caso. Esto se debe a que si aumentamos mucho la probabilidad de mutación generar todo el rato nuevas características, por lo que los hijos están todo el rato cambiando de características, lo cual no es lo ideal.

Tabla resumen de los experimentos

En la siguiente tabla vamos a indicar para cada experimento y para cada caso, el valor fitness que nos devuelve nuestra función, el objetivo de esta tabla es agrupar todos los resultados en una misma página.

	Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**
Caso 1	21995616	27165751943	308255671721	666444448189
Caso 2	20421989	27169137484	308470113338	667496121446
Caso 3	20393504	27413225153	308636684332	668301134520
Caso 4	16625283	26399834408	308376389209	665638737921
Caso 5	17188485	26413270642	307210539534	664957543874
Caso 6	21589130	27546164639	308794718530	668027338201
Caso 7	22380031	26583534496	307729681935	666856161883
Caso 8	23060462	27409161682	308908050757	667169085517
Caso 9	20414073	27795532945	308965149600	667785588286

Como se puede apreciar en la tabla, a medida que usamos ficheros más complejos, el fitness va aumentando, lo cual es algo normal.