Memoria de la práctica 2

Jaime Lorenzo Sánchez

Francisco Cruceira Lloret

Alejandro Del Moral Rodríguez

Francisco Julián Reyes Urbano

11 de febrero de 2022

Índice general

Capítulo 1

Codificación binaria

1.1. ¿Cómo se comporta este algoritmo a medida que cambiamos el número de soluciones, generaciones, tamaño del torneo, probabilidad de cruce y probabilidad de mutación? Amplía el problema con más objetos para ver el comportamiento del algoritmo

Realizamos un estudio inicial del comportamiento del algoritmo utilizando los siguientes datos:

1. pesos = [34, 45, 14, 76, 32]
2. precios = [340, 210, 87, 533, 112]
3. Peso máximo = 100
4. Tamaño de torneo selector: K=3
5. Número de generaciones = 3
6. Tamaño de la población = 10

7. Probabilidad de cruce = 0.1

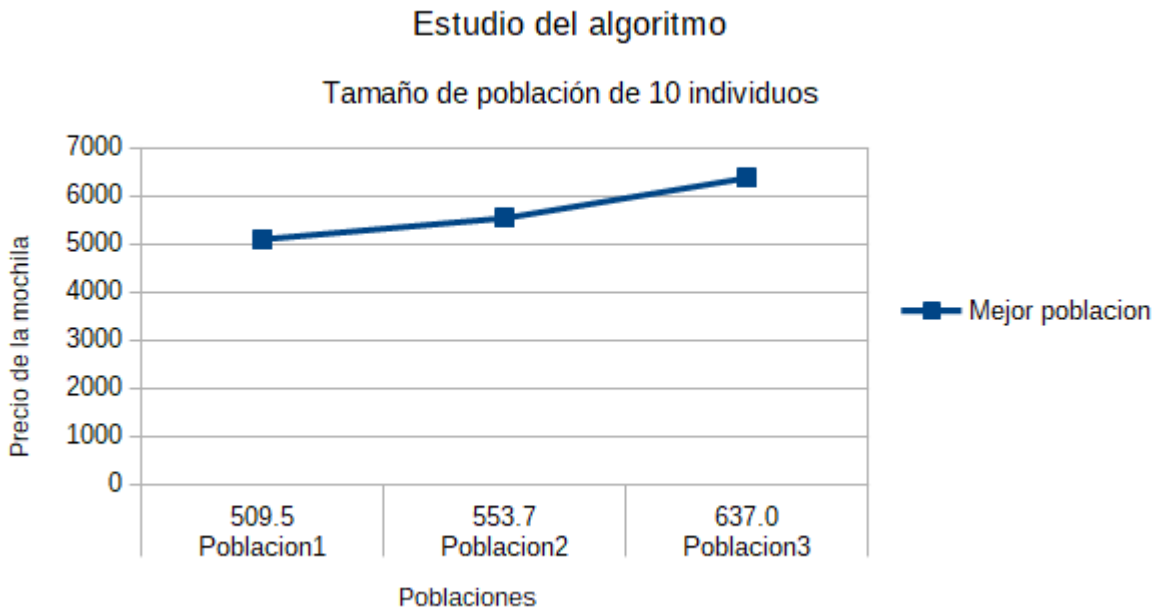
8. Probabilidad de mutación = 0.1

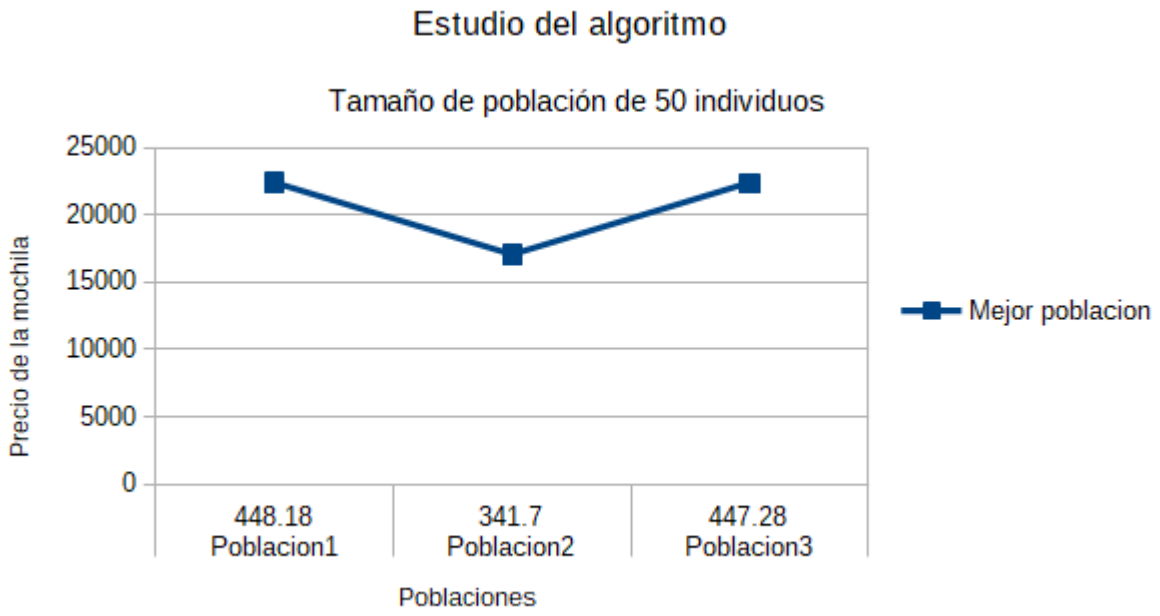
TAMAÑO DE LA POBLACIÓN

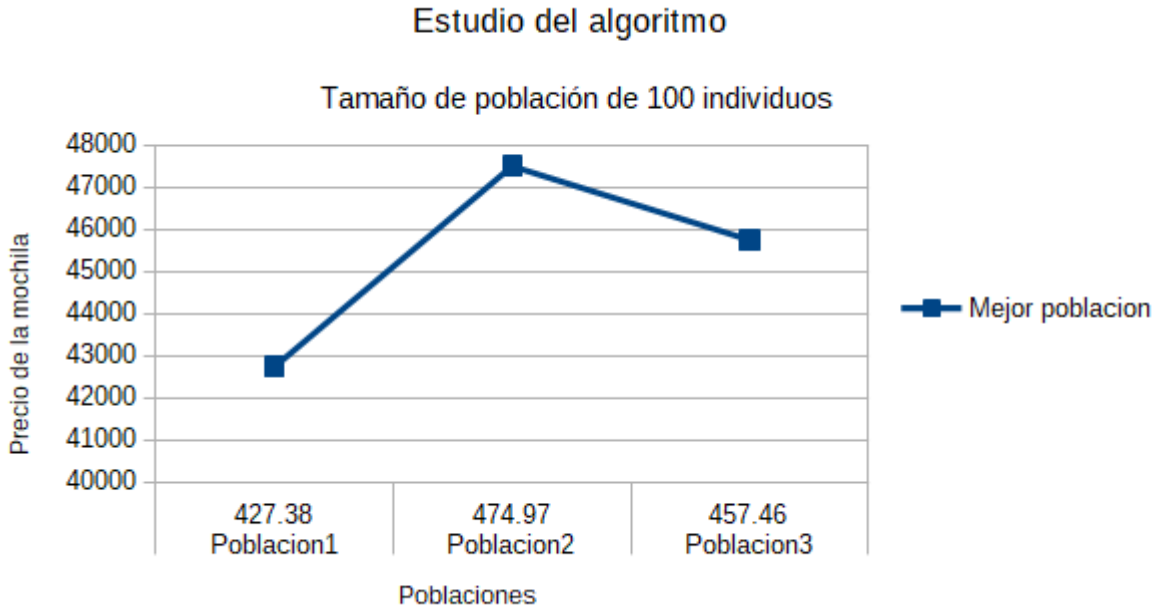
El número de soluciones determina el tamaño de la población. Por tanto, cuanto mayor sea el número de soluciones mayor será el número de individuos de las poblaciones.

El tamaño de la población es uno de los parámetros más determinantes en el buen funcionamiento del algoritmo y tiene gran influencia en la diversidad de la población y en el tiempo de computación.

Para realizar el estudio del comportamiento del algoritmo, realizaremos un estudio modificando el tamaño de la población, siendo un tamaño de población de 10 individuos, 50 individuos y 100 individuos respectivamente:







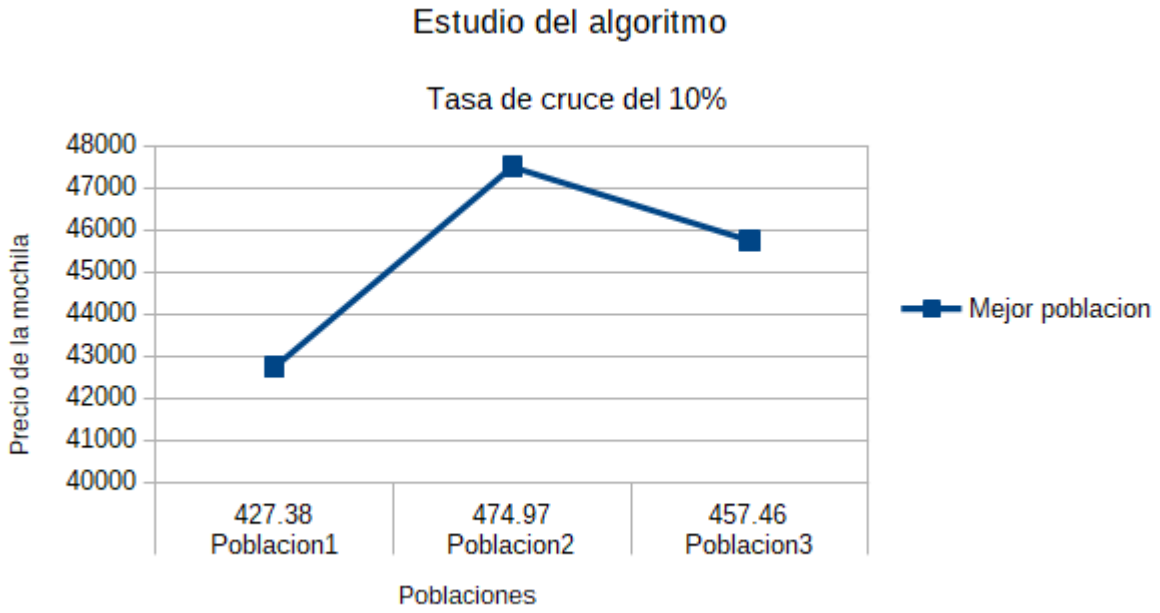
Como podemos observar en las gráficas anteriores, la solución más óptima (mejor población) se localiza cuando el número de individuos es de 10.

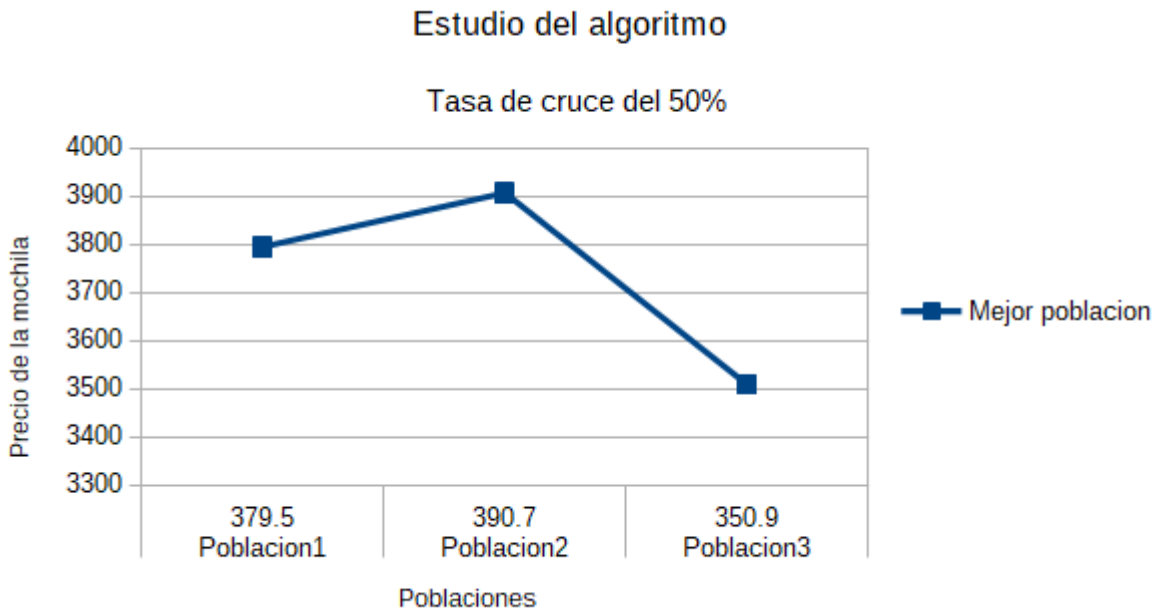
OPERACIÓN DE CRUCE

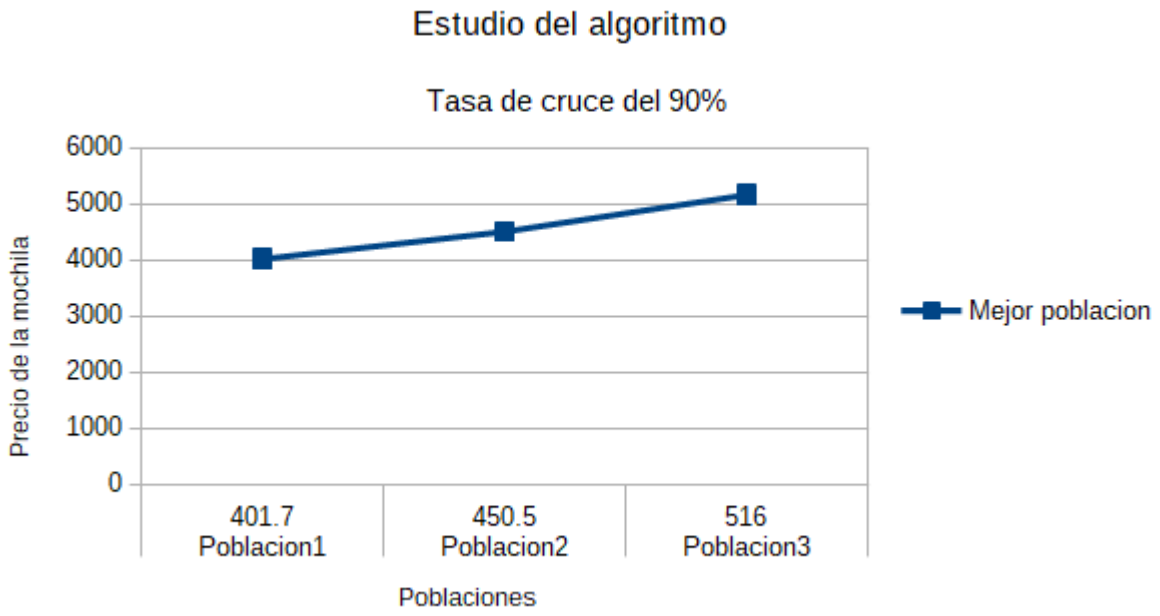
La operación de cruce permite generar nuevas soluciones a partir de otras 2 potenciales soluciones, guiando la población hacia el óptimo, ejerciendo una gran influencia en la convergencia del algoritmo.

Una tasa de cruce elevada hace que el algoritmo tienda a perder diversidad en pocas generaciones, de modo que sea más complicado obtener la solución más óptima.

Se muestran a continuación las poblaciones generadas para una tasa de cruce de 0.1, 0.5 y 0.9, respectivamente.







Si observamos la gráfica, podemos observar que la tasa de cruce influye en la diversidad de la población, de modo que cuanto mayor sea la tasa de cruce más tiende a perder la diversidad de la población.

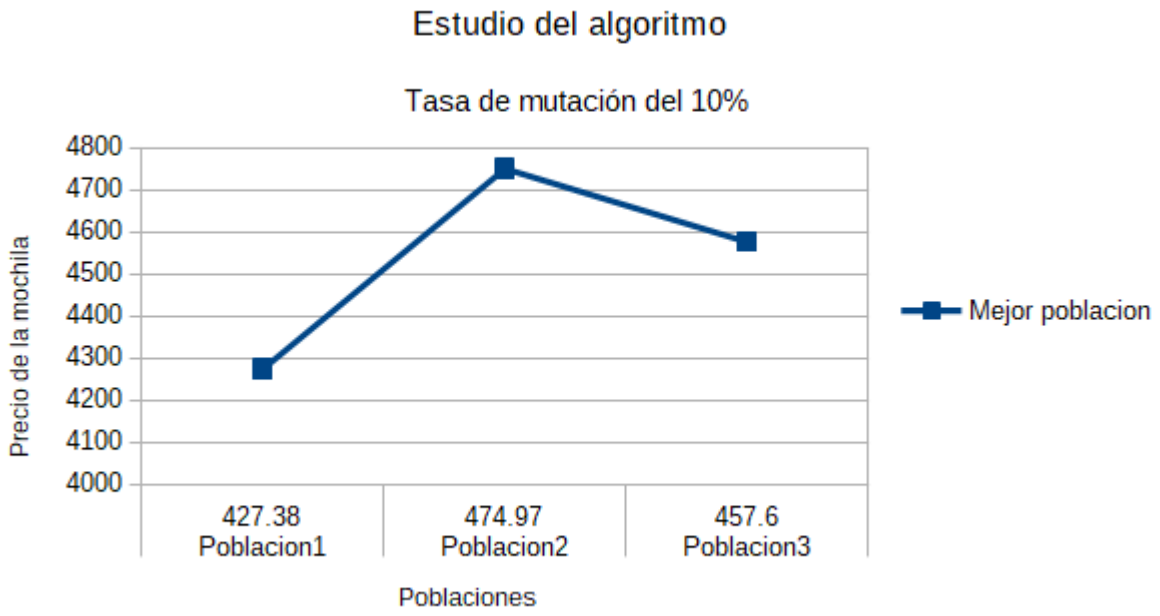
Por tanto, podemos comprobar que para poder obtener la solución óptima la tasa de cruce debe ser pequeña. Por tanto, consideraremos una tasa de cruce de 0,1.

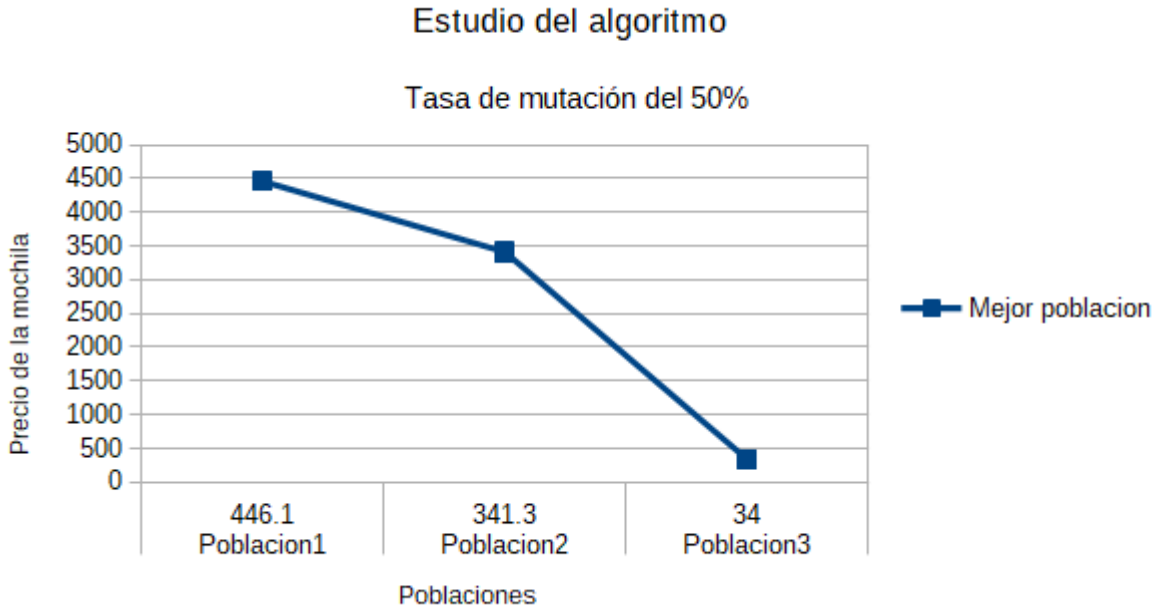
Tasa de mutación

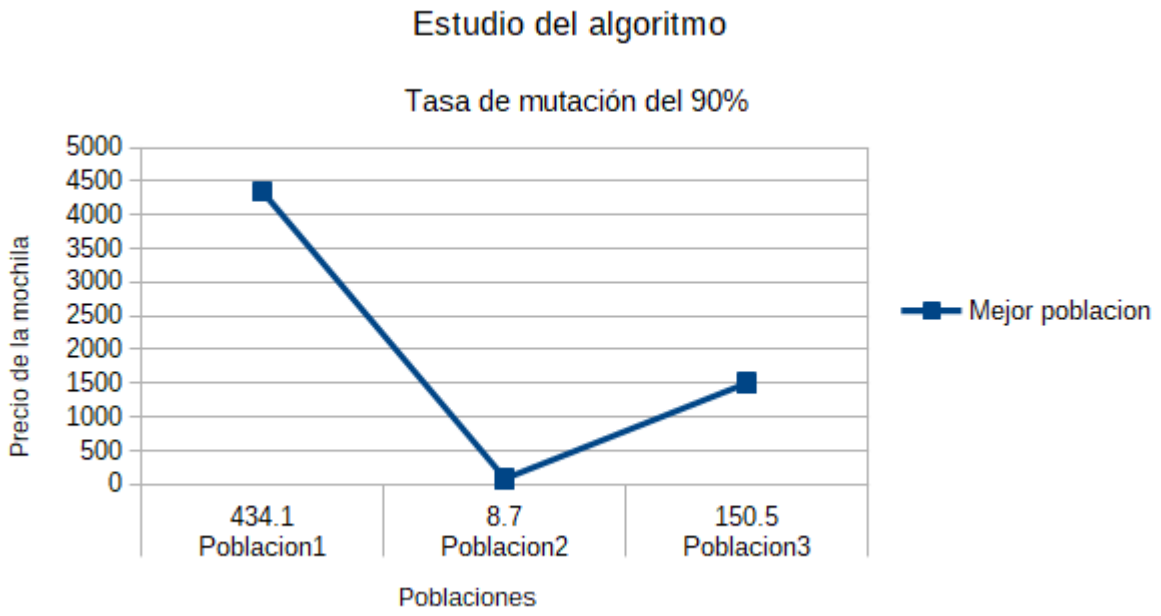
La tasa de mutación determina la probabilidad de que un individuo hijo sufra mas. Es decir, el algoritmo utiliza este porcentaje para cambiar los genes de un individuo de la población.

Sin embargo, los individuos mutados rara vez tienen un buen ajuste, pues la mutación no tiene en cuenta la historia del resto de generaciones.

Se muestran a continuación poblaciones generadas con una tasa de mutación de 0.1, 0.5 y 0.9, respectivamente:


Estudio del algoritmo
Tasa de mutación del 10%
Precio de la mochila
4800
4700
4600
4500
4400
4300
4200
4100
4000
427.38
Poblacion1
474.97
Poblacion2
457.6
Poblacion3
Poblaciones
Mejor poblacion


Estudio del algoritmo
Tasa de mutación del 50%
Precio de la mochila
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
446.1
Poblacion1
341.3
Poblacion2
34
Poblacion3
Poblaciones
Mejor poblacion



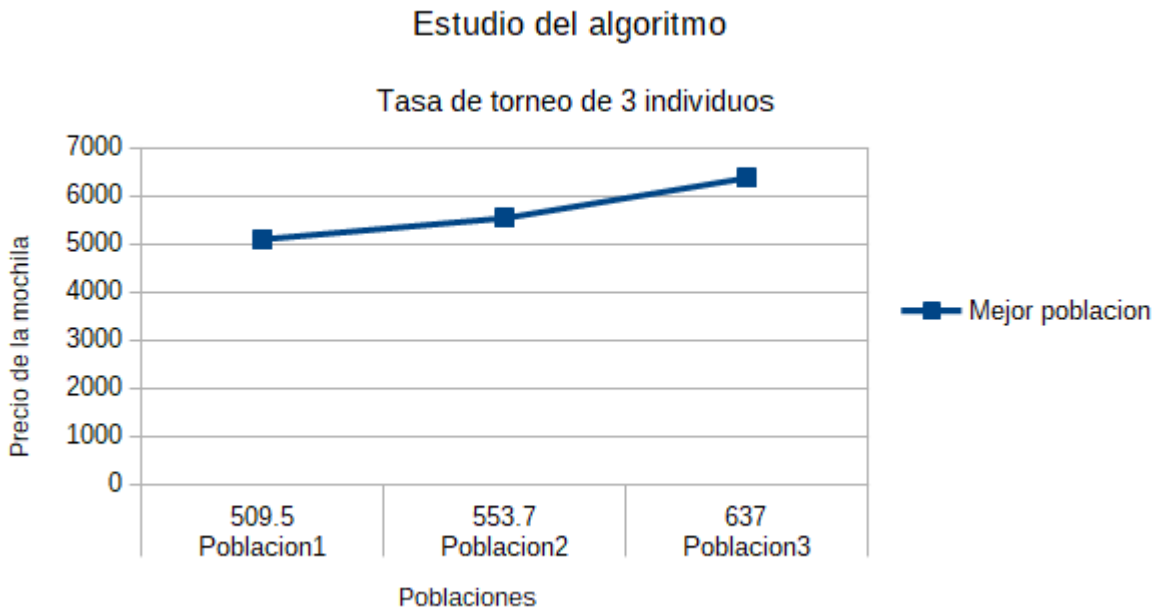
Si observamos las gráficas obtenidas, comprobamos que una elevada tasa de mutaciones introduce demasiados individuos no deseados, por lo que son recomendables probabilidades de mutación reducidos.

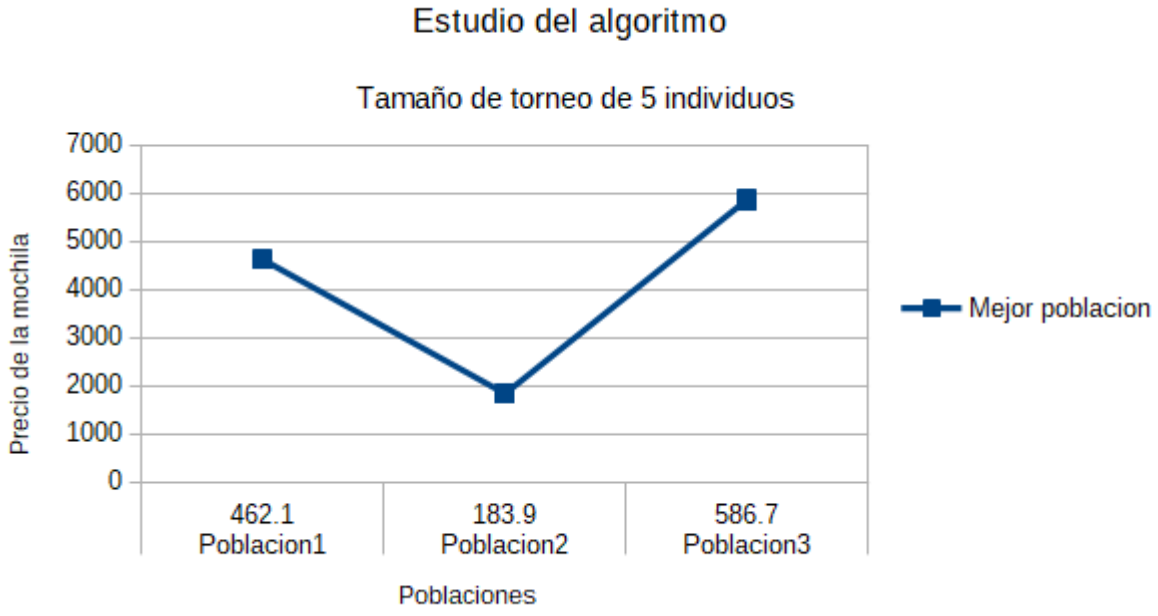
Por tanto, para poder obtener la mejor solución utilizaremos una tasa de mutaciones de 0,1.

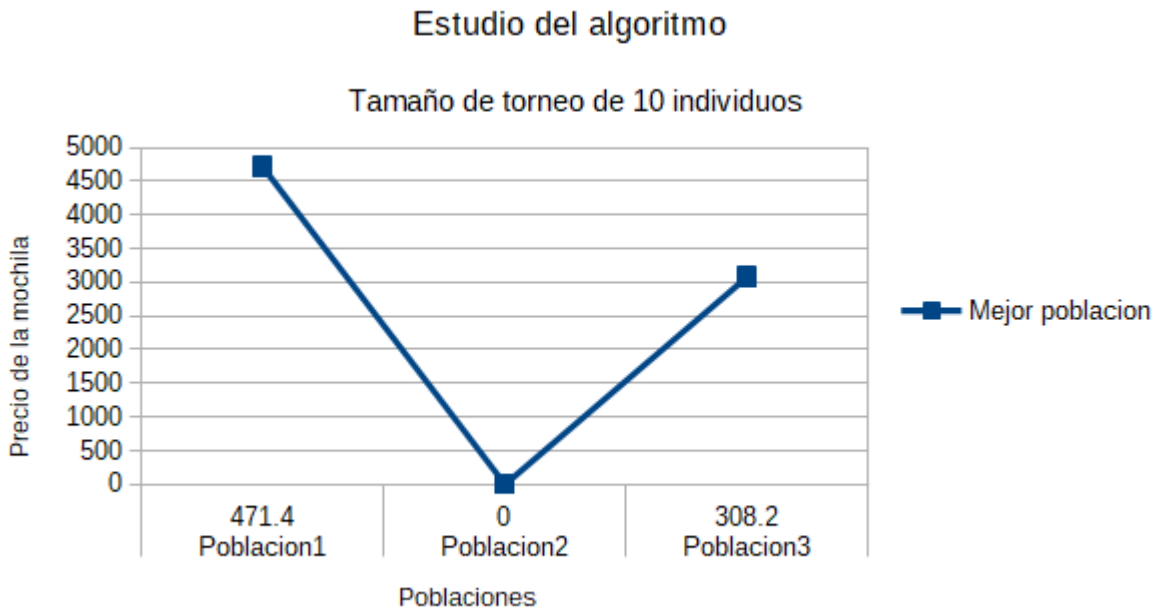
Tamaño de torneo

El tamaño del torneo determina el número de individuos de la población inicial que se seleccionan, siendo dichos individuos conocidos como padres candidatos. Una vez escogidos dichos padres candidatos, el algoritmo escogerá al padre considerado como el mejor, para posteriormente cruzarlo con otro padre.

Se muestran a continuación poblaciones generadas con un tamaño de torneo de 3, 5 y 10, respectivamente:


Estudio del algoritmo
Tasa de torneo de 3 individuos
7000
6000
5000
4000
3000
2000
1000
0
Precio de la mochila
509.5
Poblacion1
553.7
Poblacion2
637
Poblacion3
Poblaciones
Mejor poblacion


Estudio del algoritmo
Tamaño de torneo de 5 individuos
7000
6000
5000
4000
3000
2000
1000
0
Precio de la mochila
462.1
Poblacion1
183.9
Poblacion2
586.7
Poblacion3
Poblaciones
Mejor poblacion



El algoritmo utiliza el tamaño de torneo para seleccionar de la población un número de k individuos, los cuáles cruza y muta. Esto permite al algoritmo elegir de dichos k individuos el individuo mejor.

Cuanto mayor es este tamaño, podemos observar que los resultados de cada generación son cada vez peores, de modo que un número reducido del tamaño del torneo permite obtener una generación más óptima.

De modo que para conseguir la solución óptima, debemos utilizar un tamaño de torneo 3.

Numero de generaciones

El número de generaciones permite al algoritmo repetir su ciclo de acuerdo a dicha constante, de modo que un número mayor de generaciones permite al algoritmo encontrar una generación más óptima o aceptable que la generación anterior.

La demostración se puede observar en las gráficas mostradas anteriormente, donde podemos observar que cada nueva generación se acerca cada vez más a la solución óptima, siempre que los valores utilizados en el algoritmo sean los más óptimos.

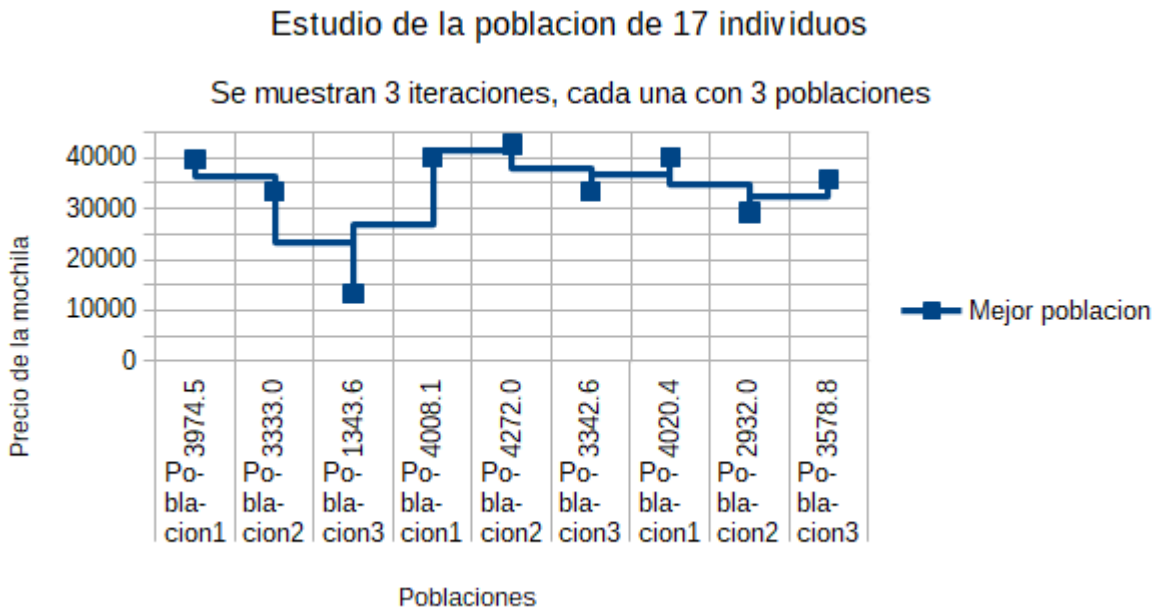
Sin embargo, este estudio del funcionamiento del algoritmo no es completo debido a que no hemos comprobado su funcionamiento para un número de objetos distintos. Para ello,

vamos a comprobar dicho funcionamiento utilizando las variables que permiten obtener el resultado más óptimo, pero con un número distintos de objetos y peso máximo de la mochila.

Realizamos un estudio para una población de 17 individuos, utilizando un peso máximo de mochila de 500 y las variables óptimas obtenidas anteriormente:

1. **Pesos:** 34, 45, 14, 76, 32,56,22,35,60,30,34,70,60,125,67,88,99.

2. **Precios:** 340, 210, 87, 533, 112,450,290,170,320,245,530,800,900,889,374,1050,780



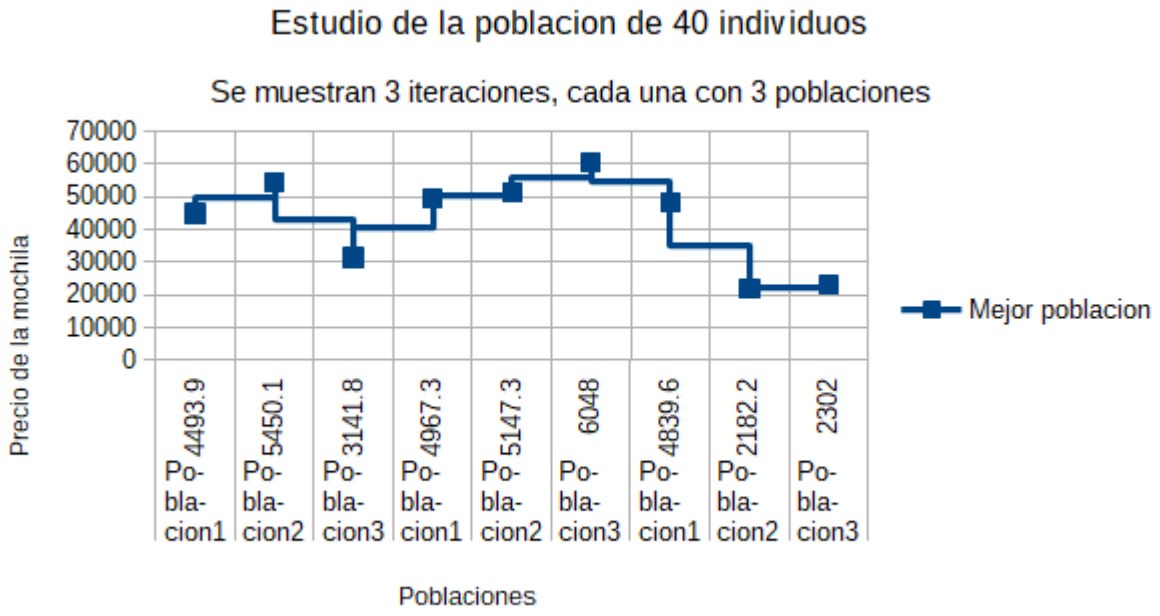
Tras realizar un número elevado de iteraciones, hemos comprobado que con estos datos la mejor población tiene un precio de mochila de 36000. Si observamos la gráfica, comprobamos que no siempre se obtiene la mejor solución, habiendo casos en los que la generación más óptima (la tercera población) sea incluso peor.

Esto se debe a que este algoritmo es generacional, es decir, en cada generación los individuos de la población son sustituidos por los resultantes de los operadores de selección, cruce y mutación de la población de la generación anterior.

Realizamos un estudio para una población de 40 individuos, utilizando un peso máximo de mochila de 1500:

1. **Pesos:** 120, 95,85,76,94,96,105,85,95,300,340,175,260,125,67,88,99,67,200,120,150,170,120,89,255,

2. **Precios:** 340, 210, 287, 533, 312,450,290,170,320,245,530,800,900,889,374,1050,780,588,245,760,45



En este caso también se cumple lo visto en la gráfica 'Estudio de la poblacion de 17 individuos', siendo la solución óptima aquella con un precio de la mochila de 60480.

1.2. ¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

Como hemos visto anteriormente, este algoritmo no siempre obtiene la mejor solución, sino la solución más óptima o la mejor posible.

Esto depende del numero de individuos de la población, del tamaño de torneo, la probabilidad de cruce, la probabilidad de mutación, el número de generaciones, etc. Los motivos son los explicados anteriormente.

1.3. Modifica el código para incorporar elitismo. La mejor solución se guarda en la élite y nunca se pierde hasta que venga una nueva mejor. Esta solución es la que se devuelve al final. ¿Has conseguido mejorar? ¿Por qué?

En este caso para analizar el efecto del uso del elitismo en el problema de la mochila mediante algoritmos genéticos, utilizaremos diferentes tamaños del problema, es decir variaremos:

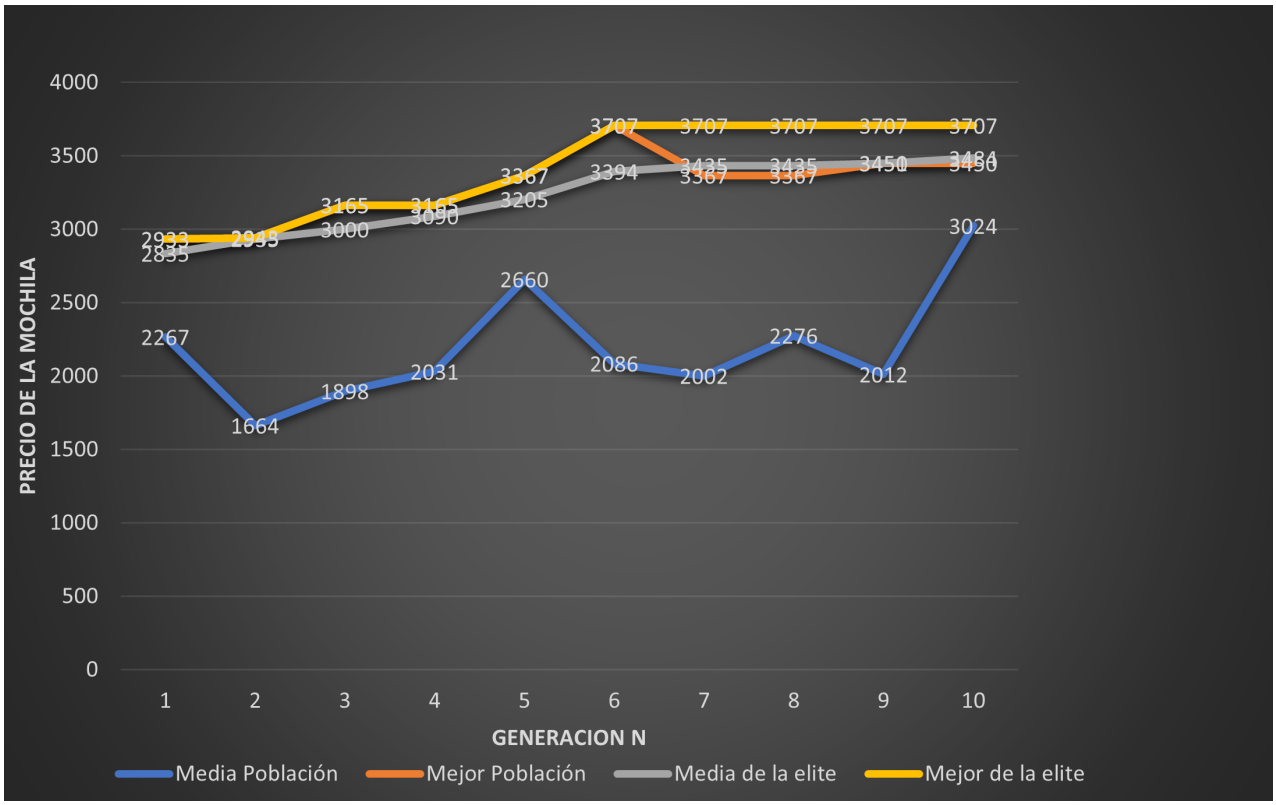
- El número de objetos
- Los precios de los objetos
- Los pesos de los objetos
- El peso máximo de la mochila

17 objetos

En un primer caso utilizamos un peso máximo de la mochila de 300 y 17 objetos a seleccionar los cuales tienen los siguientes valores de pesos y precios:

- **Pesos:** 34, 45, 14, 76, 32,56,22,35,60,30,34,70,60,125,67,88,99.
- **Precios:** 340, 210, 87, 533, 112,450,290,170,320,245,530,800,900,889,374,1050,780

Primeramente analizamos como evoluciona nuestro algoritmo genético con y sin elitismo mediante los datos obtenidos en una sola ejecución del algoritmo.Nuestro algoritmo posee inicialmente de los siguientes valores de los parámetros:
k=3 **nSoluciones**=30 **MaxGeneraciones**=10 **ProbabilidadCruce**=0.7 **ProbabilidadMutacion**=0.1 y **nSolucionesElite**=5.



En la gráfica si nos fijamos en las métricas de media de las soluciones elite y la mejor solución elite por cada generación o se mantiene constante o mejora los valores del precio de la mochila obtenidos en generaciones anteriores.En cambio no utilizando elitismo se puede observar que tanto la media como el mejor individuo de cada generación puede empeorar con respecto a los resultados obtenidos en generaciones anteriores.

Esto es debido a que nuestro algoritmo genético es de tipo generacional, es decir, en cada generación los individuos de la población son sustituidos por los resultantes de los operadores de selección ,cruce y mutación de la población de la generación anterior.En el caso de incluir elitismo a nuestro algoritmo por cada generación nos permite mantener los n mejores individuos (en nuestro caso 5) generados hasta el momento.

Tras ejecutar el algoritmo 200 veces, podemos considerar que el óptimo global es **3707** euros del precio de la mochila y que se alcanza (en la última generación el valor del mejor individuo es 3707) con una frecuencia muy baja de **4/200** con elitismo y **2/200** sin elitismo.Esta frecuencia tan baja nos indica que nuestro algoritmo para este problema intensifica mucho en zonas no prometedoras, para solucionarlo hay que aumentar la diversidad en nuestro algoritmo mediante:

- **K:** Si disminuimos el valor de k aumenta la probabilidad de elegir peores soluciones y por tanto nos permite conservar características útiles de peores soluciones para crear mejores progenitores en el proceso de cruce.

- **Probabilidad de mutación:** Si aumentamos el valor de la probabilidad de mutación nos permite generar muchas más diversas soluciones aunque tiene el inconveniente de que muchas de ellas pueden ser soluciones no validas y que por tanto solo aporten distorsión al problema.

- **Número de soluciones:** Si aumentamos el número de soluciones aumenta la diversidad de la población en detrimento de un mayor coste computacional.

Ejecutamos nuestro algoritmo 200 veces para diferentes valores de la probabilidad de mutación,numero de soluciones y el tamaño del torneo para ver como varia la frecuencia de obtener el óptimo global.

ProbMutación	Con elitismo	Sin elitismo
0.12	3/200	3/200
0.15	4/200	3/200
0.17	6/200	3/200
0.20	10/200	10/200

Número de soluciones	Con elitismo	Sin elitismo
30	4/200	2/200
40	12/200	10/200
100	39/200	34/200
150	60/200	54/200

Valor de k	Con elitismo	Sin elitismo
3	4/200	2/200
2	5/200	4/200
1	0/200	0/200

Analizando los diferentes parámetros que afectan a la diversidad de la población de un algoritmo genético para este problema podemos determinar:

- Que al aumentar la probabilidad de mutación por encima de 0.1 la probabilidad de explotar zonas prometedoras en la población apenas es apreciable ya que en muchos casos acaba generando soluciones no validas, por lo que lo más óptimo es mantener una probabilidad de mutación baja de 0.1

- Que al disminuir el valor de K , con k=2 apenas se aprecian diferencias en la probabilidad de explotar zonas prometedoras y con k=1 al tratarse de una selección aleatoria aumenta la probabilidad de que algoritmo vaya empeorando a lo largo de las diversas generaciones.´

- Al aumentar el número de soluciones aumenta de forma considerable la probabilidad de explotar zonas prometedoras ya que nos permite aumentar la diversidad de la población manteniendo una presión selectiva equilibrada (como ocurre con k=3) lo que nos permite mantener las mejores características a lo largo de las generaciones sin descartar buenas características de peores soluciones y evitando introducir demasiado ruido o distorsión, es decir, soluciones no validas como ocurre con valores altos de probabilidad de mutación.

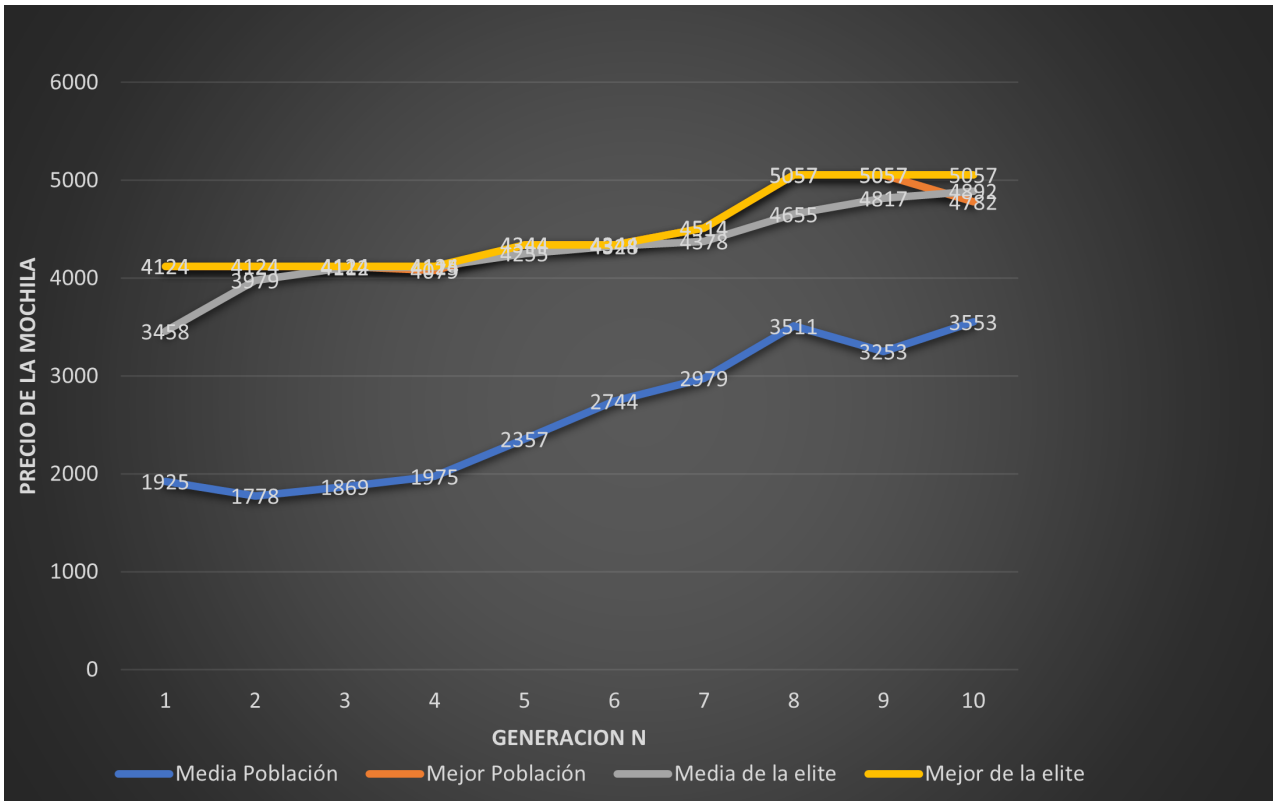
Por tanto lo más óptimo para este tamaño del problema es aumentar el número de soluciones de 30 a 150.

40 objetos

En este segundo caso utilizamos un peso máximo de la mochila de 715 y 40 objetos a seleccionar los cuales tienen los siguientes valores de pesos y precios:

- **Pesos:** 120, 95,85,76,94,96,105,85,95,300,340,175,260,125,67,88,99,67,200,120,150,170,120,89,255,
- **Precios:** 340, 210, 287, 533, 312,450,290,170,320,245,530,800,900,889,374,1050,780,588,245,760,45

Volvemos a analizar como evoluciona nuestro algoritmo en este caso para 40 objetos con y sin elitismo a través de una ejecución.Partimos inicialmente con estos valores de los parámetros del algoritmo:
k=3 **nSoluciones**=100 **MaxGeneraciones**=10 **ProbabilidadCruce**=0.7 **ProbabilidadMutacion**=0.1 y **nSolucionesElite**=5.



Se puede observar como las dos métricas que representan el uso del elitismo evolucionan siempre a hacia mejores o iguales soluciones a lo largo de la generaciones, mientras que las métricas que miden el uso del no elitismo muestran como la población a lo largo de la generaciones puede llegar a empeorar como ocurre en la última generación si nos fijamos en la métrica de el mejor individuo de la población.

Para este tamaño del problema ejecutamos de nuevo 200 veces el algoritmo y tras ello los resultados obtenidos son:

- Podemos considerar como óptimo global para este tamaño del problema al valor 5755.
- Se alcanza el óptimo global con una tasa muy baja de 1 entre 200 con y sin elitismo,

En este caso debemos aumentar la diversidad para evitar converger demasiado rápido hacia zonas no prometedoras.

Para aumentar la diversidad probamos a aumentar el número de soluciones ya que como se observo para el tamaño de 17 objetos fue el parámetro que mejor resultados obtuvo.

Número de soluciones	Con elitismo	Sin elitismo
100	1/200	1/200
200	6/200	4/200
400	11/200	11/200

Como se puede observar si utilizamos un tamaño de 400 soluciones se aumenta la probabilidad de alcanzar el óptimo global (en la última generación el valor del mejor individuo de la población es 5755) para este tamaño del problema.

Para este tamaño del problema se ha observado que en algunas ejecuciones hasta en la última generación la población va mejorando hacia zonas cercanas al óptimo global, por tanto modificamos el número máximo de generaciones para comprobar si nos permite aumentar la probabilidad de alcanzar el óptimo global.

MaxGeneraciones	Con elitismo	Sin elitismo
10	15/200	15/200
20	61/200	60/200

Si observamos los datos anteriores podemos determinar que al aumentar el número de soluciones y el número máximo de generaciones a 400 y 20 respectivamente la probabilidad de alcanzar el óptimo global para este tamaño del problema crece forma considerable.Esto es debido a al aumento de la diversidad en la población debido al uso de un número mayor

de soluciones y al aumento de la explotación de zonas prometedoras gracias al uso de un número mayor de máximas generaciones.

Análisis de elitismo VS No elitismo

Como se ha ido observando en lo largo del ajuste de los parámetros de los dos tamaños del problema considerados, la incorporación del elitismo permite mejorar el rendimiento del algoritmo ya que en los casos en los que el algoritmo descarta o elimina buenas soluciones en generaciones intermedias, este permite conservarlas.

En concreto con los dos tamaños del problema considerados, las diferencias entre el uso y no uso del elitismo no son muy significativas. Esto es debido principalmente a:

- Trabajamos con un tamaño de k=3, esto nos permite mantener una presión selectiva relativamente alta y por tanto se reduce la probabilidad de escoger peores soluciones de generación en generación.
- Trabajamos con una probabilidad de mutación de 0.1 lo que nos permite que se apenas produzca cambios en los individuos de la población , evitando así convertir muchas buenas soluciones en malas a través de la mutación.

En conclusión, el elitismo siempre va a permitir mejorar o al menos mantener el rendimiento de un algoritmo genético generacional aunque su verdadero potencial esta en problemas en los que se produzcan cambios muy bruscos o dispares entre las diversas generaciones.

1.4. Hasta ahora, hemos partido de soluciones válidas al comenzar el algoritmo. Cámbialo para que pueda generarse cualquier solución. ¿Afecta al rendimiento? Analiza esta nueva situación

Como podemos observar, al permitir generar cualquier solución se introduce inicialmente mucha distorsión al algoritmo genético generacional ya que la gran parte de soluciones generadas son invalidas. Para comprobar su efecto ejecutamos 200 veces nuestro algoritmo para los tamaños de problema de 17 y 40 objetos partiendo de cualquier solución para determinar si afecta

17 objetos

	Con elitismo	Sin elitismo
Solucione validas	60/200	54/200
Cualquier solución	21/200	19/200

40 objetos

	Con elitismo	Sin elitismo
Solucione validas	61/200	60/200
Cualquier solución	11/200	9/200

En ambos tamaños del problema se observa que al partir de cualquier solución la probabilidad de alcanzar el óptimo global de cada tamaño del problema disminuye de forma considerable y esto es debido principalmente a:

- Al partir de cualquier solución se generan muchas soluciones invalidas inicialmente y estas pueden llegar a cruzarse con buenas soluciones y generar de esta forma soluciones invalidas y por tanto obtener peor rendimiento del algoritmo.

- Se pierde la "buena diversidad"que nos aportaba trabajar con un número de soluciones altas, ya que al partir desde soluciones validas generábamos bastantes soluciones de un valor de precio bajo pero que poseían buenas características que eran útiles para poder generar mejores individuos que sean similares al óptimo global .En cambio al partir de cualquier solución, mucha de esta diversidad se pierde y se introduce muchas soluciones que no aportan ninguna característica o gen útil a nuestro algoritmo.

Capítulo 2

Codificación entera

2.1. ¿Cómo se comporta este algoritmo a medida que cambiamos el número de soluciones, generaciones, tamaño del torneo, probabilidad de cruce y probabilidad de mutación? Amplía el problema con más objetos para ver el comportamiento del algoritmo.

Empezaremos realizando un primer estudio del comportamiento del algoritmo utilizando los siguientes datos:

1. pesos = [34, 45, 14, 76, 32]
2. precios = [340, 210, 87, 533, 112]
3. Peso máximo = 1000
4. Tamaño de torneo selector: K=3
5. Número de generaciones = 3
6. Tamaño de la población = 10

7. Probabilidad de cruce = 0.1

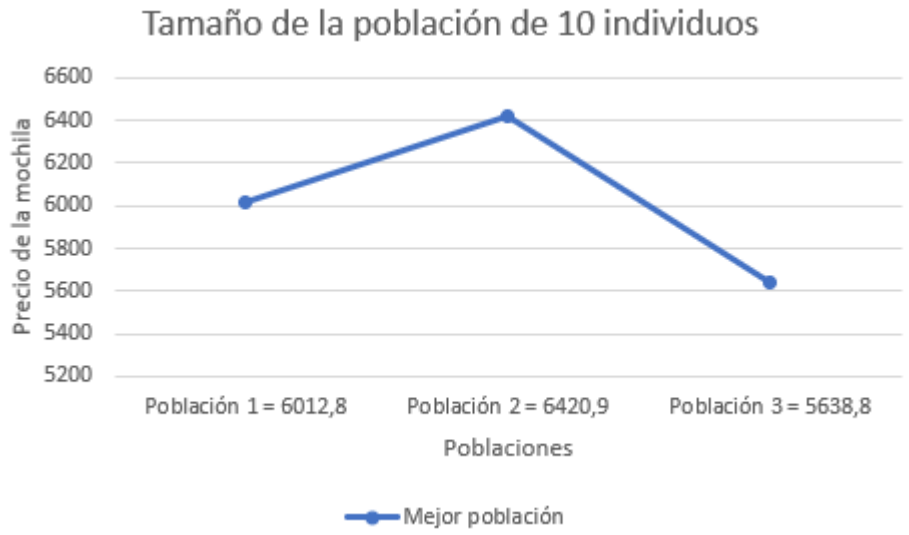
8. Probabilidad de mutación = 0.1

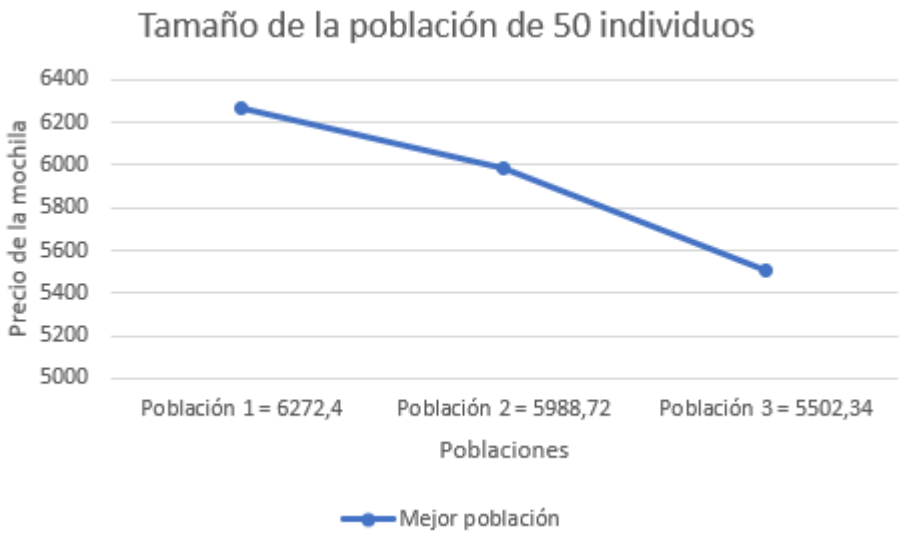
Como esta vez los objetos de las poblaciones pueden aparecer más de una vez es necesario aumentar el peso máximo de la mochila, en caso contrario provocará que se den bastantes más casos inválidos según el valor que le asignemos al peso máximo, cuanto menor sea más casos inválidos aparecerán. Como se puede ver en la asignación anterior de valores, se le ha asignado el valor 1000 como peso máximo, ya que con ese valor es poco probable que aparezcan casos inválidos.

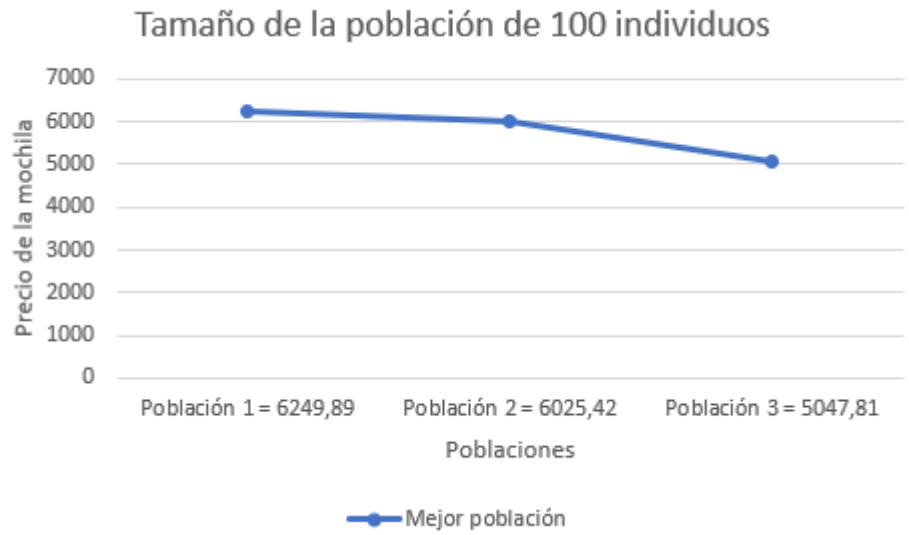
TAMAÑO DE LA POBLACIÓN

Como ya se menciono en la parte de la práctica con codificación binaria, el número de soluciones estará determinado por el tamaño de la población, de tal manera que cuanto mayor sea el número de soluciones mayor será el número de individuos de las poblaciones.

Para la realización de este estudio del algoritmo, se realizará un estudio modificando el tamaño de la población utilizando los mismos valores que en la parte binaria, pero aplicados con codificación entera. Los tamaños serán de 10 individuos, 50 individuos y 100 individuos:





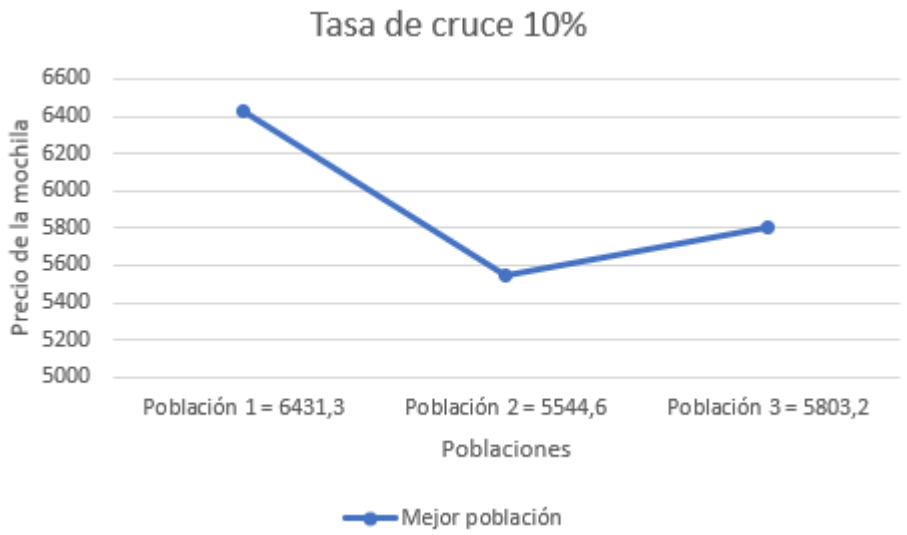


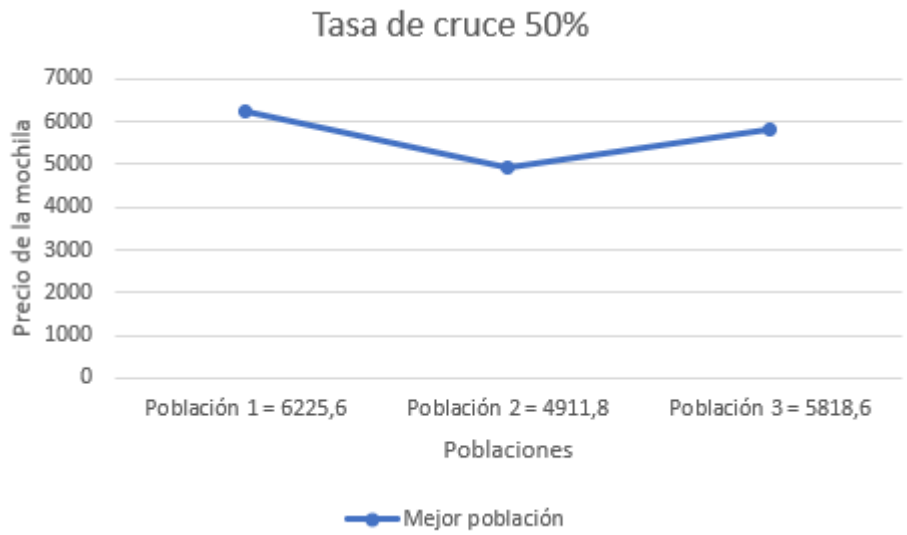
Como se puede observar, la solución más óptima se da en la gráfica con un tamaño de población de 10 individuos.

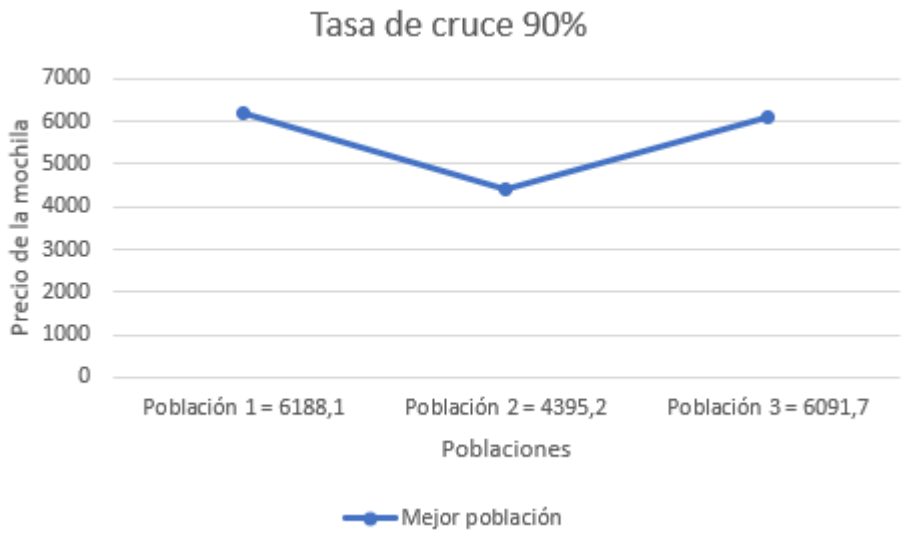
OPERACIÓN DE CRUCE

Como en la codificación binaria, en la operación de cruce se generarán otras soluciones partiendo de 2 soluciones potenciales, lo cual, guiaran a la población hasta su óptimo ejerciendo un gran peso en la convergencia del algoritmo.

Que el algoritmo tenga una tasa de cruce alta provoca la perdida de diversidad en pocas generaciones en el algoritmo. Para su comparación con la otra codificación se han tomado los valores 0.1, 0.5 y 0.9 para su tasa de cruce.







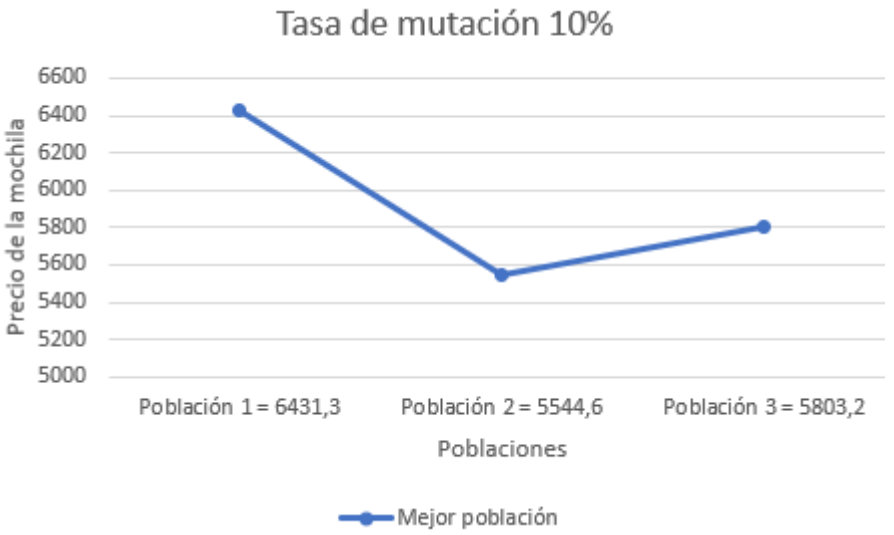
Como podemos observar, la tasa de cruce influye en la diversidad de la población, de la misma manera que en la binaria, cuanto mayor sea la tasa de cruce más tiende a ir

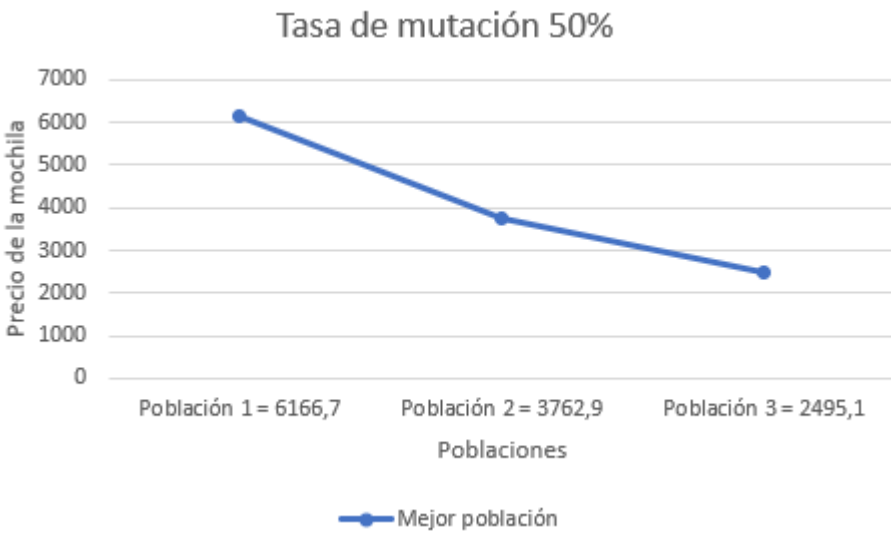
perdiendo la diversidad de población. En conclusión, para obtener una solución óptima la tasa de cruce implementando esta codificación tiene que ser del 0.1.

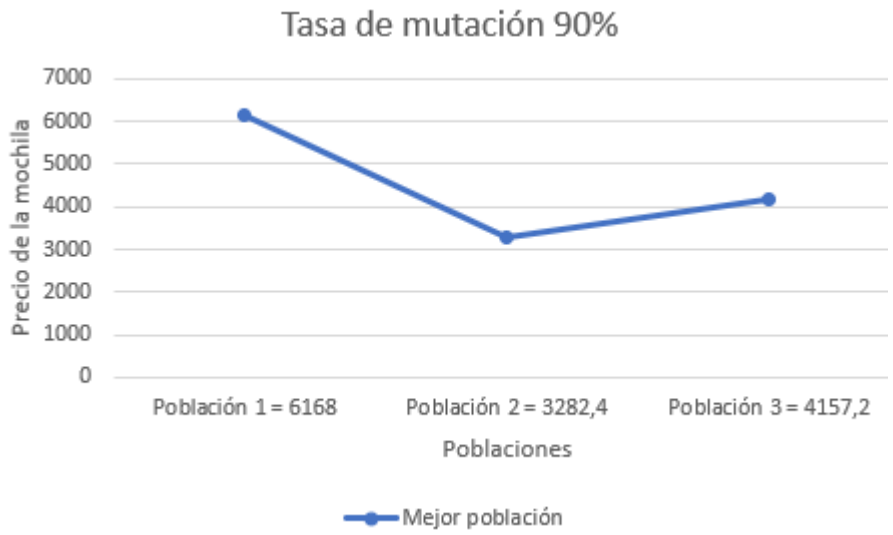
Tasa de mutación

Como ya se comento en la parte binaria, el algoritmo utiliza la tasa de mutación para cambiar los genes de un individuo de la población, pero estos rara vez tienen un buen ajuste, ya que la mutación no tiene en cuenta la historia del resto de generaciones.

Para poder comparar las gráficas se han escogido los valores 0.1, 0.5 y 0.9 para la tasa de mutación.







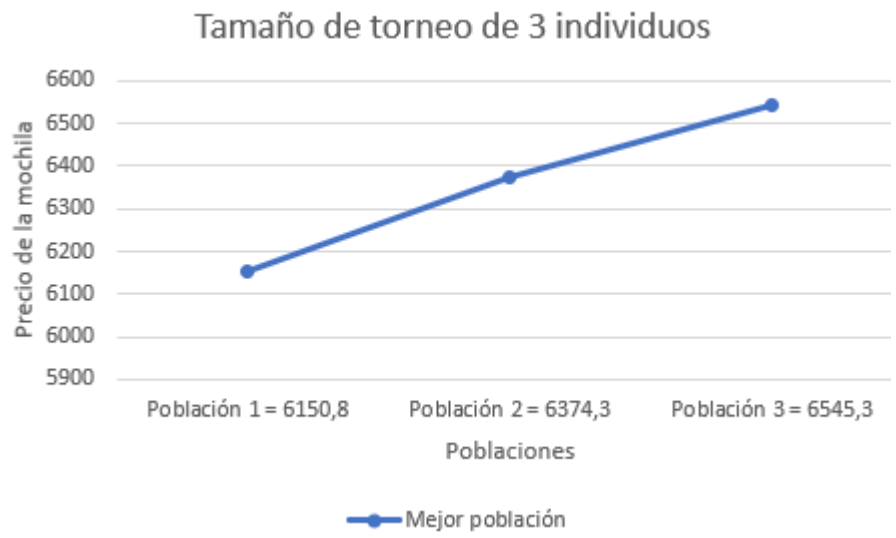
Si observamos las gráficas obtenidas, se puede apreciar que una elevada tasa de mutaciones introduce bastantes individuos no deseados, por lo que son recomendables probabilidades de mutación reducidos. Podemos concluir que una mayor tasa de mutaciones introducirá una cantidad mayor de individuos no deseados, por lo tanto lo más recomendable sería utilizar tasas de mutación reducidas, en este caso la del 0.1.

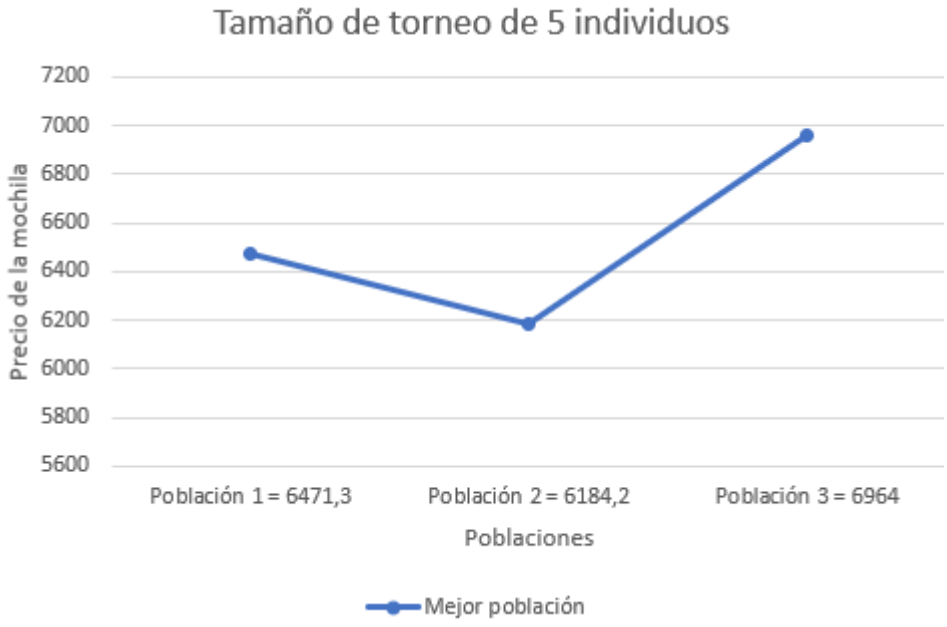
TAMAÑO DEL TORNEO

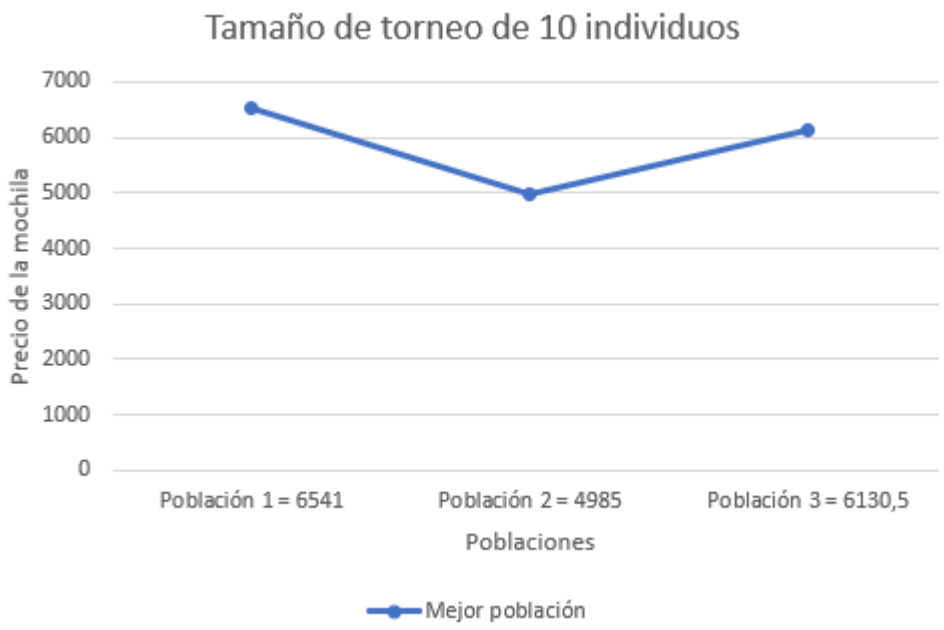
Como se menciona en la parte de codificación binaria, con el tamaño de torneo podremos proceder a determinar el número de padres candidatos, de los cuales luego el algoritmo escogerá al que considere mejor para cruzarlo con otro padre más tarde.

Seleccionará un número k de individuos que luego cruzará y mutará para poder escoger al mejor padre.

A continuación se muestran poblaciones generadas con un tamaño de torneo de 3, 5 y 10.


Tamaño de torneo de 3 individuos
6600
6500
6400
6300
6200
6100
6000
5900
Precio de la mochila
Población 1 = 6150,8
Población 2 = 6374,3
Población 3 = 6545,3
Poblaciones
Mejor población


Tamaño de torneo de 5 individuos
7200
7000
6800
6600
6400
6200
6000
5800
5600
Precio de la mochila
Población 1 = 6471,3
Población 2 = 6184,2
Población 3 = 6964
Poblaciones
Mejor población



Como se puede observar, el resultado de cada generación será peor o mejor en función del tamaño. Por lo que podemos deducir que cuanto reducido sea el tamaño del torneo, más probabilidades hay de obtener una generación más óptima.

De esta manera podemos llegar a la conclusión de que la gráfica con la solución más óptima es la que tiene un tamaño de torneo de 3 individuos.

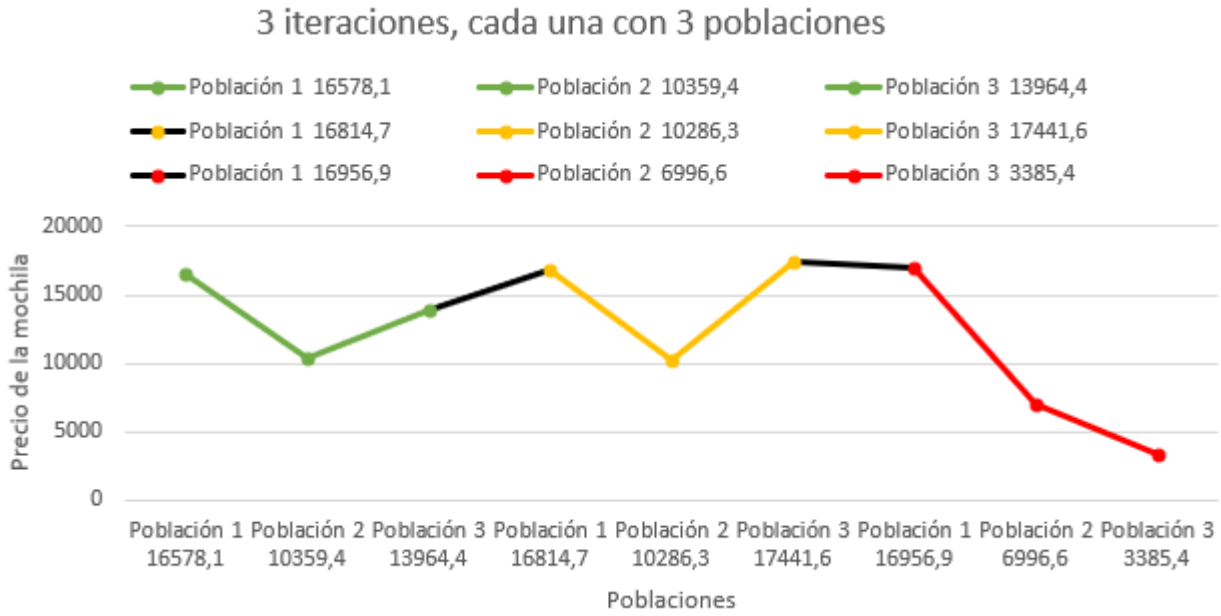
NÚMERO DE GENERACIONES

Como ya se explicó en el apartado de codificación binaria, el número de generaciones sirve para que el algoritmo pueda repetir el ciclo con la finalidad de obtener una generación que sea mejor que la anterior. Para ello, la nueva generación deberá usar los valores que se hayan establecido como óptimos con el objetivo de que cada nueva generación se acerque más a la solución óptima.

Para el estudio de este caso, y con el objetivo de poder observar las posibles diferencias entre ambas codificaciones, se realizará con los mismos parámetros que en la codificación binaria. Es decir, utilizaremos las variables que nos permiten dar con el resultado más óptimo, pero variando la cantidad de objetos y el peso máximo de la mochila. Recordemos que, a diferencia de la codificación binaria, en la codificación entera se estableció el valor 1000 como peso máximo ya que un peso menor a ese hará que aparezcan más casos inválidos cuanto menor sea el peso máximo.

Para el primer caso de estudio se tomará una población de 17 individuos, se tomarán las variables óptimas obtenidas y el peso máximo será de 2000.

- **Pesos:** 34, 45, 14, 76, 32,56,22,35,60,30,34,70,60,125,67,88,99.
- **Precios:** 340, 210, 87, 533, 112,450,290,170,320,245,530,800,900,889,374,1050,780



Para representar cada iteración se ha marcado cada una de un color diferente:

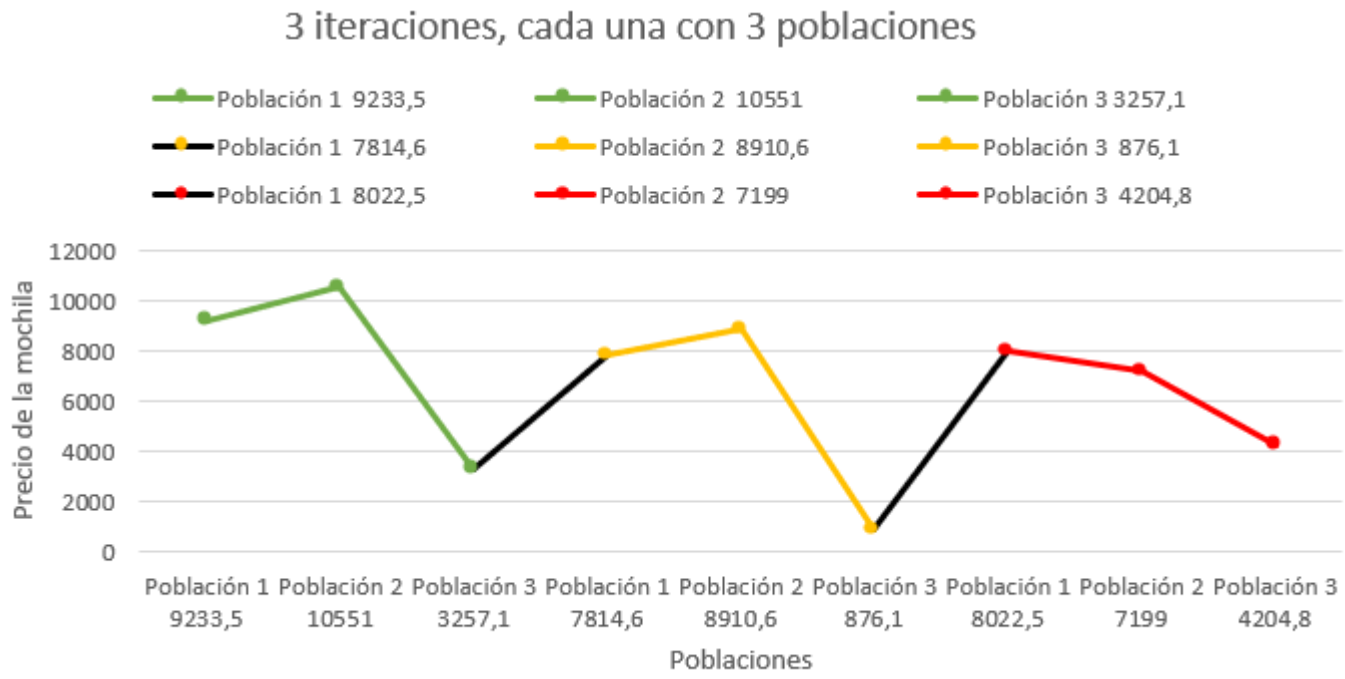
1. La primera iteración está representada en verde
2. La segunda iteración está representada en amarillo
3. La tercera iteración está representada en rojo

Tras realizar las iteraciones podemos observar que la mejor población ha sido la población 3 de la segunda iteración ya que ha obtenido un valor de 17441,6 que ha resultado ser mejor.

Para el segundo caso de estudio tomaremos una población de 40 individuos y el peso máximo sera 2500

1. **Pesos:** 120, 95,85,76,94,96,105,85,95,300,340,175,260,125,67,88,99,67,200,120,150,170,120,89,255, 124,320,220,120,75,45,130,400,310,220,130,98,115,320,145.
2. **Precios:** 340, 210, 287, 533, 312,450,290,170,320,245,530,800,900,889,374,1050,780,588,245,760,

450,145,670,230,1030,970,490,320,150,180,275,550,320,730,560,430,289,945,670,845.



Como en la gráfica anterior, se han indicado cada iteración con un color. Se les ha asignado el mismo color a cada iteración para una mejor identificación.

1. La primera iteración está representada en verde
2. La segunda iteración está representada en amarillo
3. La tercera iteración está representada en rojo

Como se puede ver en la gráfica, en este caso la mejor población sería la población 3 de la tercera iteración, que tiene un precio de mochila de 4204,8. A diferencia de la gráfica del caso anterior, esta vez no hemos obtenido la mejor solución y se nos ha dado el caso en el que la solución óptima ha resultado peor. La razón de esto es porque es un algoritmo generacional, el cual ya ha sido explicado en la parte de codificación binaria.

2.2. ¿Obtiene siempre la mejor solución? ¿Por qué? ¿De qué depende?

Como hemos podido ver, este algoritmo, de la misma manera que en la codificación binaria, busca obtener la solución más óptima posible en lugar de obtener siempre la mejor

solución.

Los motivos son los que se han ido mencionando en el apartado anterior. En lo referente a que depende, dependerá de parámetros como el tamaño de la población, la tasa de cruce, la tasa de mutación, el tamaño del torneo y el número de generaciones.

2.3. Modifica el código para incorporar elitismo. La mejor solución se guarda en la élite y nunca se pierde hasta que venga una nueva mejor. Esta solución es la que se devuelve al final. ¿Has conseguido mejorar? ¿Por qué?

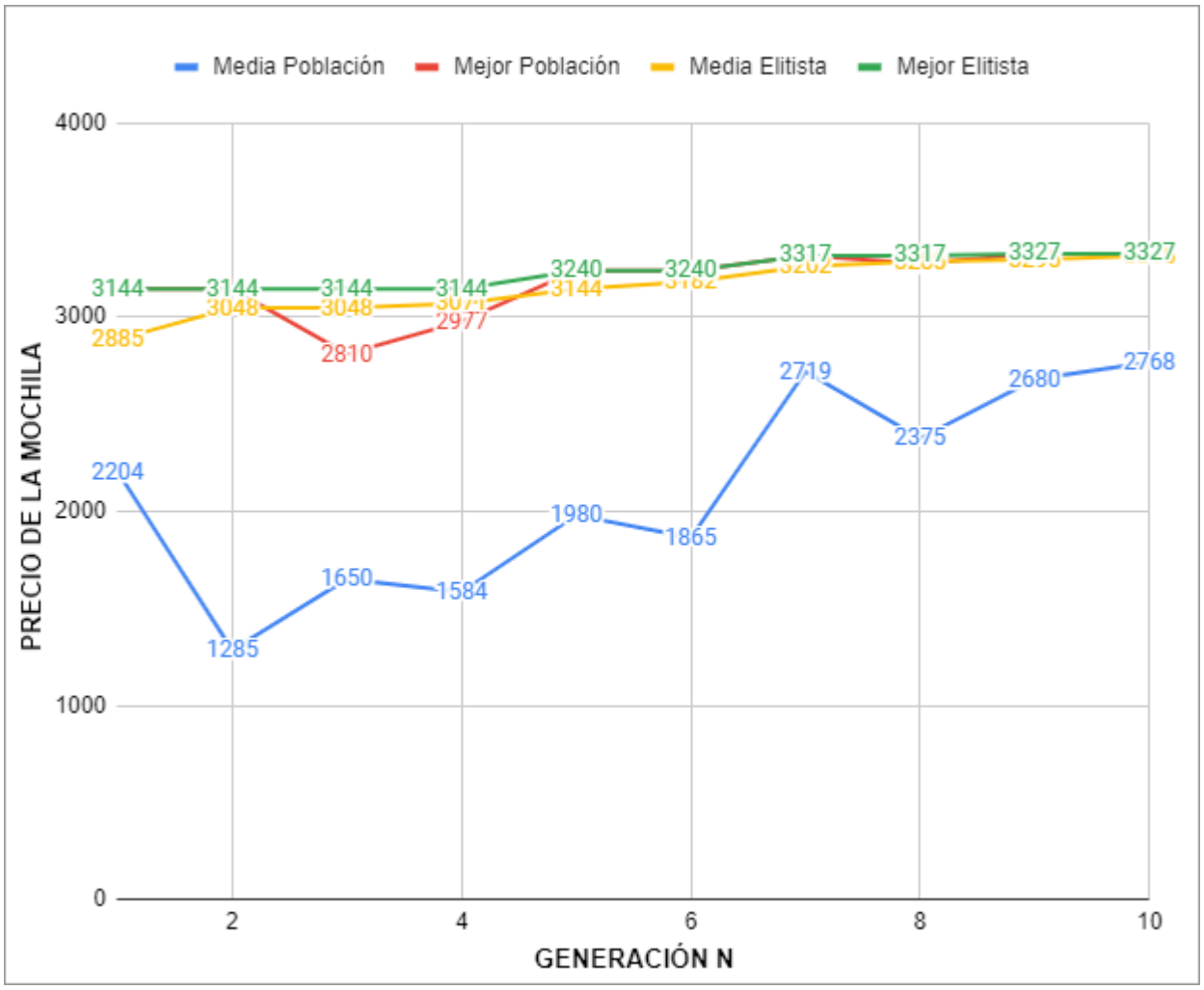
Partiremos inicialmente en los mismos parámetros de tamaños usados en la codificación binaria:

17 objetos

Utilizamos un peso máximo de la mochila de 300 y 17 objetos a seleccionar los cuales tienen los siguientes valores de pesos y precios:

- **Pesos:** 34, 45, 14, 76, 32,56,22,35,60,30,34,70,60,125,67,88,99.
- **Precios:** 340, 210, 87, 533, 112,450,290,170,320,245,530,800,900,889,374,1050,780

Analizamos como evoluciona nuestro algoritmo genético con y sin elitismo mediante los datos obtenidos en una sola ejecución del algoritmo. Haremos uso a su vez de los mismos valores que se usaron con anterioridad:
k=3 **nSoluciones**=30 **MaxGeneraciones**=10 **ProbabilidadCruce**=0.7 **ProbabilidadMutacion**=0.1 y **nSolucionesElite**=5.



Tras ejecutar el algoritmo 200 veces, vemos que el elitista consigue mejorar la solución 66 veces en comparación al método normal tras preservar las mejores soluciones generación tras generación. Podemos apreciar que aplicando la nueva codificación con los tamaños usados en la codificación binaria que es más complicado alcanzar el óptimo global en estas circunstancias, además con dicho peso máximo de la mochila y los pesos que poseen dichos objeto (no hacen falta usar muchos simultáneamente para llegar al peso máximo) apenas se hace uso de poder repetir un mismo objeto para las soluciones válidas.

```
2, 0, 0, 0, 0, 0, 14, 12, 0, 0, 3, 2, 12, 17, 2, 0], 0], [[0, 0, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 0, 0, 1, 0, 0, 1], 0], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 5, 0, 6, 13, 0, 0, 12, 0],
 0], [[0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 0], [[0, 0, 7, 1, 2, 9, 16
, 16, 3, 0, 6, 2, 0, 4, 4, 14, 0], 0]]
Poblacion 6
--------------
[[[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 3190], [[1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 3190], [[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0,
0, 0], 3190], [[1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 3190], [[2, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 3080], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
 3, 0, 0, 0, 0, 0], 2933], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2933
], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2933], [[0, 0, 0, 1, 0, 0, 0
, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2933], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0
, 0, 0], 2933], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2933], [[0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2933], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 3, 0, 0, 0, 0, 0], 2933], [[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 28
50], [[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 2740], [[1, 0, 0, 0, 0, 1,
 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0], 2660], [[1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
 0, 0, 0], 2660], [[1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0], 2660], [[2, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0], 2380], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0
, 0, 1, 1, 0, 0, 0, 0], 2233], [[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
2233], [[3, 7, 1, 0, 4, 4, 5, 14, 2, 0, 4, 0, 4, 3, 17, 0, 0], 0], [[0, 0, 0, 1, 0, 0
, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1], 0], [[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0
, 0, 0], 0], [[0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0], 0], [[0, 0, 0, 1,
0, 0, 0, 0, 13, 0, 0, 3, 4, 0, 0, 9, 0], 0], [[3, 7, 1, 0, 4, 4, 5, 14, 2, 0, 4, 0, 4
, 3, 17, 0, 0], 0], [[3, 7, 1, 0, 4, 4, 5, 14, 2, 0, 4, 0, 4, 3, 17, 0, 0], 0], [[16,
 0, 13, 0, 0, 0, 0, 4, 11, 0, 0, 3, 0, 0, 2, 11, 10], 0], [[0, 0, 14, 1, 0, 0, 9, 0,
0, 0, 0, 3, 0, 0, 0, 11, 0], 0]]
```

Por ello y para reducir el número de soluciones inválidas habrá que incrementaremos el peso máximo y nSoluciones como hemos comprobado a lo largo de las gráficas. intermedias, este permite conservarlas.

2.4. Hasta ahora, hemos partido de soluciones válidas al comenzar el algoritmo. Cámbialo para que pueda generarse cualquier solución. ¿Afecta al rendimiento? Analiza esta nueva situación

A diferencia de la codificación binaria tendremos que determinar un límite para prevenir repeticiones de objetos de manera indefinida (peso máximo / objeto con menor peso). A pesar de ello resulta mucho más probable que aceptar soluciones inválidas en estas situaciones afecte de mala manera en gran medida al rendimiento en comparación.

	Con elitismo	Sin elitismo
17 Objetos	51/200	43/200
40 Objetos	48/200	43/200

- Una vez más la propagación de soluciones inválidas a lo largo de las generaciones junto a la reducida amplitud de soluciones válidas acaban siendo de gran impacto para el algoritmo.