Técnicas Inteligencia Artificial "Proyecto Final"
2017

Sebastián Correa Echeverri

1. Introducción

El proyecto final de la materia técnicas de inteligencia artificial del Master en inteligencia artificial, reconocimiento de formas e imágenes digitales de la Universidad Politécnica de Valencia, plantea el diseño y desarrollo de dos soluciones metaheurísticas (algoritmo genético y enfriamiento simulado) para el problema lista de espera de barcos para el atraque en una terminal de contenedores. Ante una lista de llegada de embarcaciones, se buscó obtener la mejor secuencia de asignación de atraque, menor tiempo de espera entre llegada y atranque de todos los buques.

Se evalúo y contrastó la utilidad de los métodos desarrollados (algoritmo genético y enfriamiento simulado), de los parámetros utilizados, la evaluación frente a tamaños del problema y evaluación del número de soluciones generadas vs. calidad de la solución. Se comparó el algoritmo genético con el enfriamiento simulado en búsqueda de una solución partiendo de una población aleatoria, así mismo se realizó la búsqueda de la solución con un método y la optimización de dicha con el otro método, genético-enfriamiento, y enfriamiento-genético.

La implementación de los algoritmos se realizó completamente desde cero utilizando el lenguaje de programación Python.

2. Problema

El problema descrito es una lista de espera de barcos para el atraque en una terminal de contenedores donde, serán cargados o descargados (import/export) en un tiempo específico. El muelle de atraque tiene una longitud disponible máxima de 700 metros y un número máximo de grúas asignables de 7.

El problema se simplifica para que cada barco se caracterice por:

- Tiempo de llegada previsto (a una lista de espera)
- Eslora (longitud de muelle a asignar).
- Movimientos (import/export) de contenedores. Implica su tiempo de servicio.
- Prioridad

El tiempo de atraque para carga y descarga de cada barco esta descrita por la función (en minutos):

$$\frac{M}{G}$$

Ecuación 1 Tiempo de Atranque

donde,
M el número de movimientos y G el número de grúas asignadas.
La métrica de evaluación: Coste de la secuencia de asignación de barcos es un criterio de optimalidad a minimizar, es:

$$\sum_{barcos} \left[k * t_{espera} + (1-k) * t_{atranque}\right]$$

Ecuación 2 Criterio de Optimalidad

donde,

k, es la prioridad del barco siendo 8 alta prioridad y 0 la más baja. $(0 \leq k \leq 8)$

$t_{espera,}$ es el tiempo de espera del barco el cual es igual al tiempo que se tarda en asignarle muelle a un barco menos el tiempo de llegada de esté.

$t_{atranque,}$ es el tiempo que toma al barco en finalizar toda su operación (import/export), que depende directamente de sus movimientos y de las grúas asignadas.

Se generó instancias aleatorias de prueba, con colas de 20 barcos, con densidades de llegada de [60' – 180'], número de movimientos [100 – 1000], y esloras de [100 – 500] metros.

Como simplificación del problema se propuso un método para seleccionar el número mínimo de grúas que se deben asignar a cada barco dependiendo de sus dimensiones. Se parte que el tamaño máximo del muelle es de 700 metros y dispone en un total de 7 grúas, por lo cual si existiera un barco de dicha dimensión se deberían de disponer la totalidad de grúas para su optimo descargue o cargue. Con esto se plantea una proporción o factor de grúas igual a:

$$\frac{longitud_{muelle}}{maximo_{\#\ gruas}}$$

Ecuación 3 Factor de Grúas

Para la solución del problema se plantea una matriz de asignación.

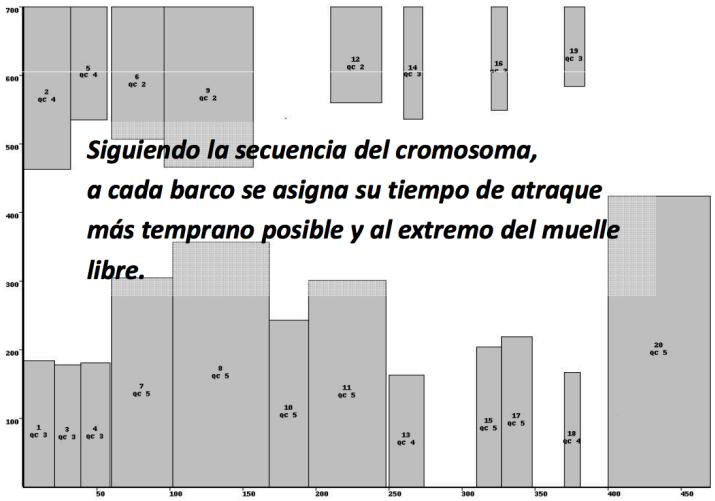

Ilustración 1 Asignación de barcos

La matriz cuenta con una columna de minutos, desde 0 hasta 200*(máxima cantidad de barcos) +180. Este valor máximo se calcula como el caso más pesimista en el cual todos los barcos llegan en una frecuencia alrededor de 180 minutos y el tiempo máximo de atranque por barco seria de 200 minutos, que sería el máximo número de movimientos sobre la máximo eslora posible sobre el factor de grúas (1000/ (500/100)).

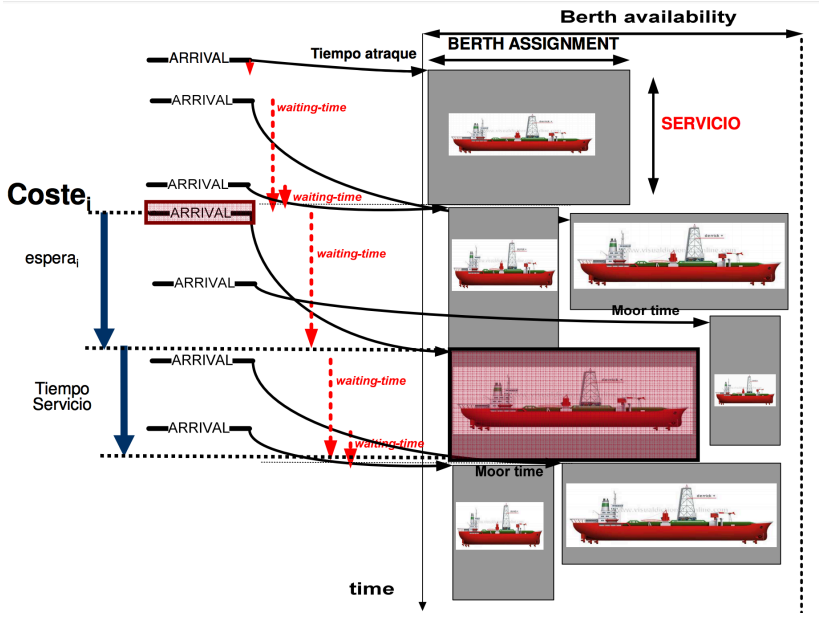

Ilustración 2 Asignación de barco, tiempo de atranque más temprano posible

3. Algoritmo Genético

En informática, un algoritmo genético (GA) es un algoritmo metaheurístico inspirado en el proceso de selección natural, este pertenece a la mayor clase de algoritmos evolutivos (EA).

El proceso se divide en, inicialización, en el que el tamaño de la población depende de la naturaleza del problema, a menudo la población inicial se genera aleatoriamente permitiendo toda la gama de posibles soluciones.

Selección, durante cada generación sucesiva, una parte de la población existente se selecciona para criar una nueva generación. Las soluciones individuales se seleccionan a través de un proceso basado en la aptitud.

Crossover y mutación, el siguiente paso es generar una población de segunda generación de soluciones de las seleccionadas a través de una combinación. En algunos casos se realiza mutación de los individuos para explorar posibles soluciones fuera del sector de búsqueda (exploración).

Terminación, este proceso generacional se repite hasta que se ha alcanzado una condición de terminación.

3.1. Función Fitness

Una función de aptitud o fitness es un tipo particular de función objetivo que se utiliza para evaluar lo cerca que está una solución de diseño a sus objetivos preestablecidos. La función fitness para la solución del problema, es el criterio de optimilidad [Ecuación 2], presentado previamente. El objetivo es minimizar el valor de dicha función, que representa el tiempo de espera de un barco, entre el momento que llega y el momento que es atendido.

Entre más alta la prioridad del barco más tiempo de espera tendrá aumentando el valor de la función fitness, mientras que cuando su prioridad es inferior o 0, se podría atender en cualquier momento y no afectaría de una forma tan significativa la función fitness. Finalmente, el valor entregado sería el costo de espera que tendría la combinación de barcos.

El tiempo de espera y de llegada está en minutos, por lo cual la función siempre está entre los rangos de 10’000 y 60’000.

Para la codificación de la función de fitness se partió de la creación de la matriz de asignación de forma vacía para esto se llenaron las casillas de capacidad con 700 (eslora máxima) y 7 (número máximo de grúas asignables).

El tamaño máximo de la matriz (filas), es el tamaño de tiempo que tomaría atender 20 barcos con llegadas de 180 minutos, con máxima capacidad y atendidos uno tras otros (peor escenario posible). Finalmente se crea una columna de barcos en el muelle el cual cada posición (x, y) es un vector que contiene los barcos presentes en ese minuto en el muelle ([1 3 19]).

```
matrix = [[],[],[],[]]
    for i in range(0,200*max_barcos+180):
        matrix[0].append(i+1)
        matrix[1].append([0])
        matrix[2].append(max_long)
        matrix[3].append(max_gruas)
```

Código Fuente 1 Creación vacía de Matriz de Asignación

Se evalúa el estado de la matriz en los campos correspondientes a la llegada del barco ($t_{espera} = 0$) hasta la duración de su atranque ($t_{llegada} + t_{atranque}$). Si los espacios de todas las casillas desde el tiempo de llegada hasta el tiempo de atranque más el tiempo de llegada en la matriz se encuentran libres, se les agregará a todas estas el barco correspondiente. Si el espacio se encuentra ocupado, pero hay suficientes grúas ($gruas_{total} - gruas_{barco} >= 0$) y suficiente espacio en el muelle ($eslora_{total} - eslora_{barco} >= 0$), a las casillas correspondiente se le adjuntara el nuevo barco formado un array del tipo [b_1 b_2 b_3 … b_n]. Si los espacios de todas las casillas desde el tiempo de llegada hasta el tiempo de atranque más el tiempo de llegada en la matriz se encuentran ocupados y no hay suficientes grúas o eslora disponible, se realizará la búsqueda en el siguiente minuto ($t_{espera} = 1$) hasta encontrar una posición disponible o con la suficiente eslora y grúas disponibles.

Con esto se calcula el tiempo real de espera de cada barco y la función de fitness para cada individuo.

```
matrix[1][t_llegada-1:t_llegada+t_atranque][i].append(barco)
matrix[2][t_llegada-1+i] = matrix[2][t_llegada-1+i] - eslora
matrix[3][t_llegada-1+i] = matrix[3][t_llegada-1+i] - gruas
fitness += prioridad*tiempo_espera + (1-prioridad)*t_atranque
```

Código Fuente 2 Calculo de Función Fitness

3.2. Individuos

Cada individuo está compuesto por una posible combinación de los N barcos del problema [15 2 3 10 9 8……] (1-20). El esquema de descodificación aplica la prioridad de atraque conforme al orden en la secuencia de genes del individuo. Esto quiere decir que se asignara el barco en la posición 0 y así sucesivamente. Todas las soluciones se basan en el problema de los siguiente 20 barcos (generado de forma aleatoria).

Nº	llegada	movimientos	eslora	prioridad
1	100	390	375	3
2	84	700	414	6
3	38	706	149	1
4	51	352	500	5
5	59	745	285	6
6	102	565	139	6
7	115	308	453	6
8	154	713	438	6
9	113	889	150	7
10	152	466	388	6
11	77	176	119	3
12	46	788	468	6
13	133	517	174	7
14	168	787	277	3
15	71	309	498	0
16	66	990	419	4
17	122	595	177	6
18	168	858	117	2
19	96	406	447	4
20	60	223	455	1

Tabla 1 Problema Inicial, Propiedades de los Barcos

Cada individuo es un array de tamaño 20 que contiene la combinación de los diferentes barcos. Para crear cada individuo era importante tener en cuenta que los números no pueden estar repetidos, ya que un barco no puede ser atendido dos veces.

El código usado para realizar la creación de individuos fue:

```
random.sample(range(1,21), 20)
```

Código Fuente 3 Generación de Individuos

3.3. Población

La población está formada por un número x de individuos y se puede iniciar de forma aleatoria (se generan x individuos aleatorios) o fija (se usan ya preestablecidos x individuos). Se usó población aleatoria para el problema y comparación de búsqueda de soluciones y la población fija se utilizó para comparar el proceso de mejoramiento de soluciones.

Para generar la población aleatoria se usó la función individuo que devuelve un array con una combinación aleatoria de barcos.

El código usado para realizar la creación de la población fue:

```
return [individual(1,largo) for i in range(num)]
```

Código Fuente 4 Creación de la población aleatoria

1.1. Selección y Cross-Over

Para la selección se escogió los n individuos con el mejor fitness para realizar la reproducción o Cross-Over. Para esto se evaluó y organizo el array de población.

```
puntuados = [ (calcularFitness(i), i) for i in population]
puntuados = [i[1] for i in sorted(puntuados, reverse=True)]
```
Código Fuente 5 Evaluación y ordenamiento de la función fitness

Luego se seleccionó de forma aleatoria dos padres de los n (porcentaje de población para padres) mejores individuos en la población. Los n mejores individuos pasaran a la siguiente generación y el resto de la población será reemplazada por los nuevos hijos generados.

Para la reproducción se utilizó la primera mitad de la carga genética del padre1 luego se utilizó la carga genética no repetida del padre1 de la mitad del padre2 para formar el nuevo individuo.

```
punto = round(largo/2)
padre = random.sample(selected, 2
population[i][:punto] =padre[0][:punto]
padre2 = [x for x in padre[1] if x not in padre[0][:punto]]
population[i][punto:] = padre2
```
Código Fuente 6 Reproducción CROSS-OVER

1.2. Mutación

La mutación se realiza en cada cambio de generación y depende de la probabilidad de mutación ósea, de tener un cambio en el material genético del individuo.

A partir de la probabilidad de mutación (mutation_chance) se realizó un cambio aleatorio de genes en los individuos de la población. Este cambio se realizó entre dos barcos de forma aleatoria, con esto se exploró diferentes soluciones y aportó diversidad a la población.

```
punto1 = random.randint(0,largo-1) #Se
punto2 = random.randint(0,largo-1)
while punto1 == punto2:
    punto2 = random.randint(0,largo-1)
aux = population[i][punto1]
population[i][punto1] = population[i][punto2]
population[i][punto2] = aux
```
Código Fuente 7 Mutación

2. Algoritmo Enfriamiento Simulado

El Enfriamiento o Recocido Simulado es un algoritmo de búsqueda por entornos con un criterio probabilístico de aceptación de soluciones basado en Termodinámica.

El procedimiento está establecido con una elección aleatoria de un vecino o sucesor del estado. Si es mejor, se elige incondicionalmente, si es peor, se acepta el estado con una probabilidad que depende de la temperatura (T) y el incremento de energía (ΔE), asimilado como el decremento del valor de la función de fitness f(n) [Ecuación 2] Código Fuente 2].

La temperatura (T) va bajando conforme avanza la búsqueda. Por ello, la probabilidad de aceptar un estado que empeore el actual va bajando a medida que avanza la búsqueda.

Típicamente,

$$\text{probabilidad} = \frac{e^{\Delta f}}{T}$$

Ecuación 4 Probabilidad de aceptación de malos individuos

El decremento α se puede definir de varias formas. Las más simples son:

a) $\alpha(i,T) = k * T_i$, $0 < k < 1$

Ecuación 5 Decremento (a) de la Temperatura

b) $\alpha(i,T) = \frac{T_i}{(1+k \cdot T_i)}$, k>0 y k <<

Ecuación 6 Decremento (b) de la Temperatura

Si la variación de T es lenta y T_i es alto, la probabilidad de llegar a una buena solución es mayor, pero la búsqueda es más costosa.

El algoritmo se realizó para cada individuo de la población elegida de igual manera que en el algoritmo genético [4]. Se usó la misma función fitness [3] para calificar a los individuos y se usó la mutación [6] del algoritmo genético como proceso de búsqueda de vecinos.

2.1. Temperatura

Se decidió usar la forma b [Ecuación 6] de reducción de temperatura debido a que presenta un comportamiento más agresivo de decrecimiento [].

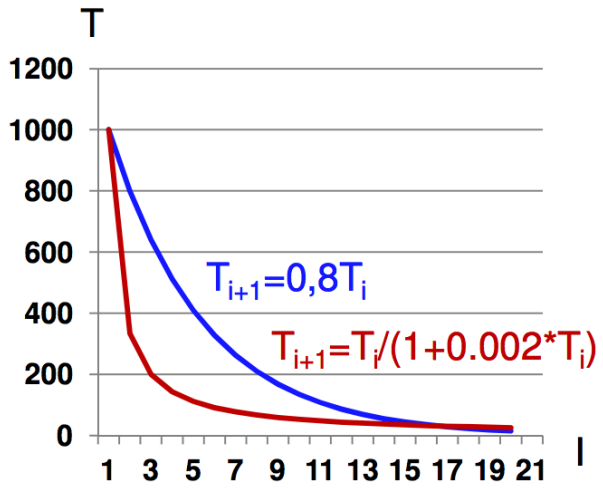

Gráfico 1 Comparación de decremento de Temperatura forma (a) y (b)

La función fitness devuelve valores en los rangos de 10.000 [3], se decidió realizar un ajuste a la forma b [Ecuación 6] para que la temperatura no llegara a cero de forma tan rápida afectando el resultado, finalmente se normalizo la temperatura con el valor 10.000. Este cambio también ayudo con el problema de tamaño de k (k<<) lo que facilito la observación y experimentos de este.

```
Tv[i] = Tv[i]/(1+k*(Tv[i]/10000))
```

Código Fuente 8 Decremento de Temperatura

3. Resultados

En esta sección se presentan los resultados obtenidos por algoritmo utilizado, y una tabla comparativa de resultados total.

3.1. Comportamiento de los parámetros del Algoritmo Genético

Se realizó un experimento de comparación de tamaños de población con el fin de encontrar un tamaño de población que cumpliera con un tiempo apropiado de iteración y otorgara una respuesta que redujera el valor de fitness de la población final.

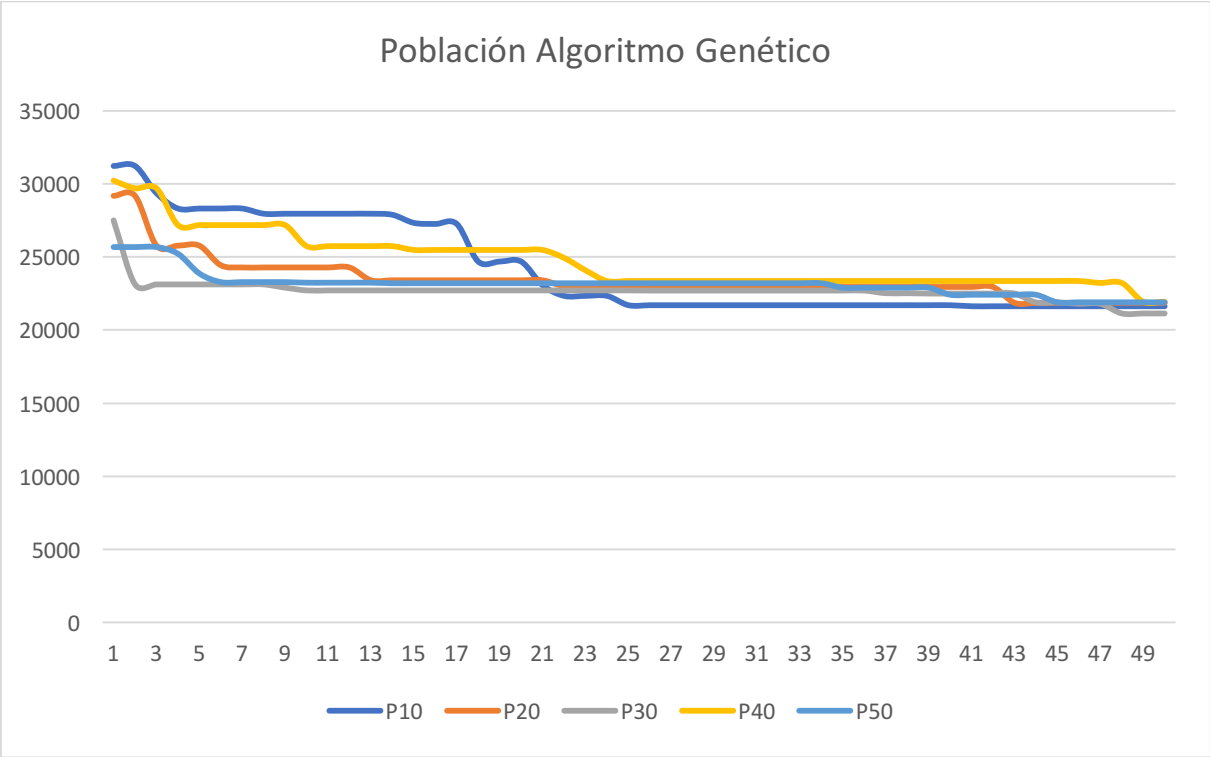

Gráfico 2 Evaluación del tamaño de población algoritmo Genético

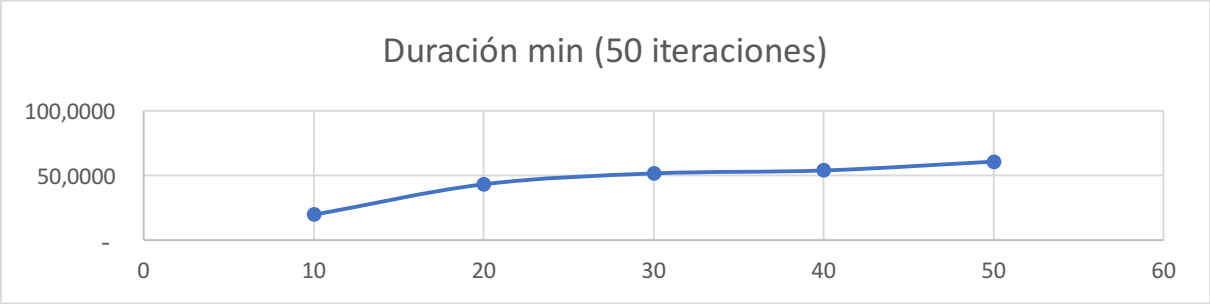

Gráfico 3 Duración de ejecución prueba de población Algoritmo Genético

Tamaño	Duración (min)
10	19,5791
20	43,0195
30	51,3838
40	53,6369
50	60,4244

Tabla 2 Duración de ejecución prueba de población

A partir del Gráfico 2 se puede observar que la población con un valor total de 30 individuos obtuvo un mayor resultado, su tiempo total de ejecución fue de 51 minutos con 23 segundos, lo cual es un valor aceptable para las pruebas realizadas, por lo tanto se seguirá realizando los experimentos de algoritmo genético con una población de 30 individuos.

Adicionalmente se realizó un experimento de comparación del parámetro probabilidad de mutación con el fin de encontrar un valor óptimo para el problema específico.

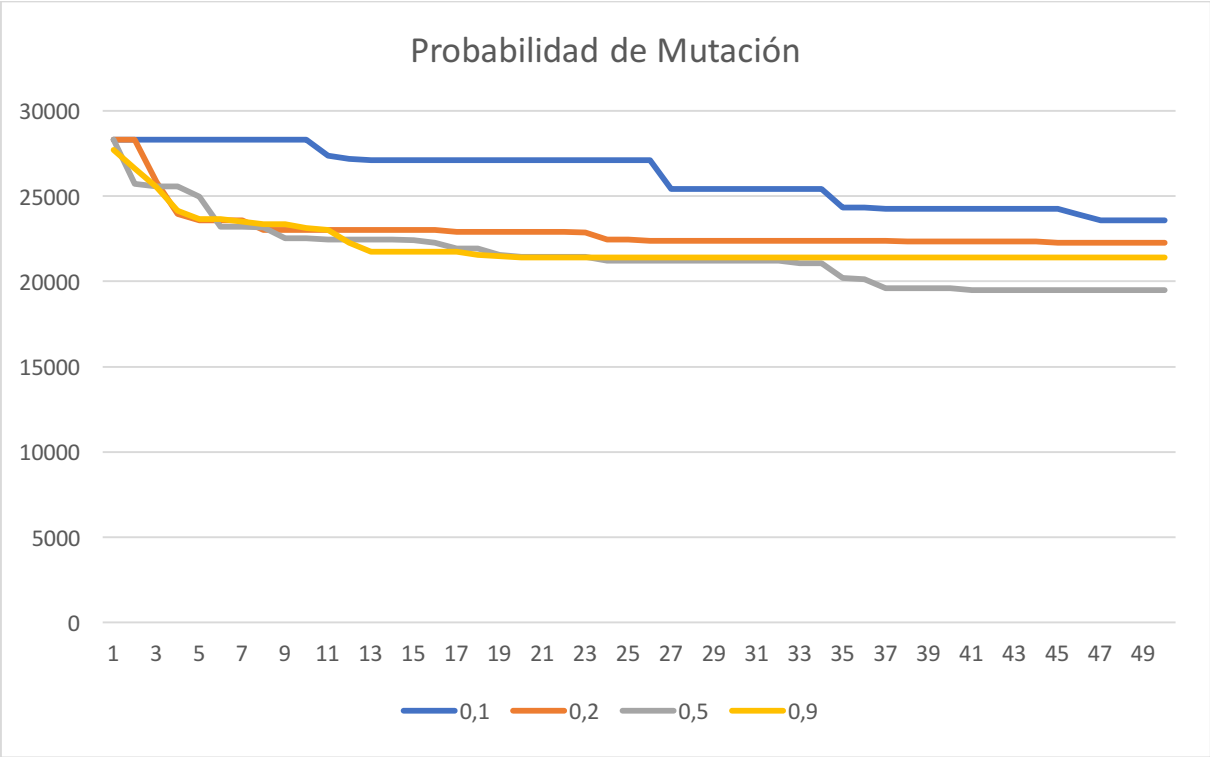

Gráfico 4 Variación de la probabilidad de mutación algoritmo Genético

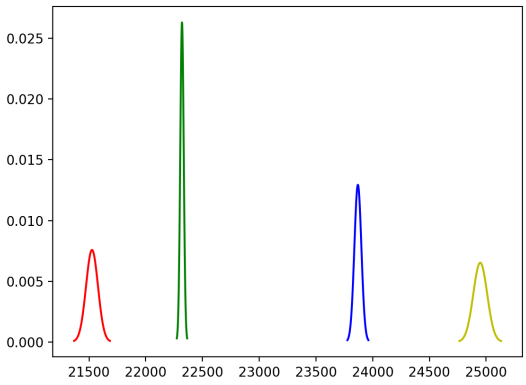

Gráfico 5 Distribuciones de las poblaciones según probabilidad de mutación

En el Gráfico 1 se pueden observar las distribuciones de población de los diferentes resultados a partir de una probabilidad de mutación rojo seria 0.5, verde 0.2, azul 0.1 y amarillo 0.9.

	deltaMin	deltaMediana	deltaMedia	deltaMax	deltaSigma	Promedio
Prueba 0.1	16,6908%	42,0245%	41,0032%	45,8179%	80,5748%	45,2222%
Prueba 0.2	21,2743%	45,2142%	44,8340%	54,2946%	95,3044%	52,1843%
Prueba 0.5	31,1658%	51,6428%	46,7938%	42,7640%	43,3929%	43,1519%
Prueba 0.9	24,3383%	42,1782%	38,3381%	31,5943%	23,6975%	32,0293%

Tabla 3 Reducción de características de población inicial

A partir de los gráficos se puede ver que el valor de probabilidad de mutación que mayor minimización de la función fitness de la población final es 0.2, puede que el valor mínimo obtenido por un valor 0.5 y 0.9 de probabilidad de mutación sea mayor, esto puede deberse a

una búsqueda más diversificada, pero el valor de 0.2 logra reducir de forma significativa la distancia entre los datos lo que con lleva a que la población final, la mayoría son buenos resultados, posteriormente se podrá visualizar nuevos resultados con mejores valores finales a partir del valor 0.2. Finalmente se escogió el valor 0.2 para la probabilidad de mutación.

3.2. Genético Población inicial Aleatoria

Se realizaron 5 experimentos de inicio de población aleatorio basados en el problema [Tabla 1] buscando la solución más óptima en 100 iteraciones (generaciones), se utilizó la probabilidad de mutación igual a 0.2 y una población total de 30 individuos.

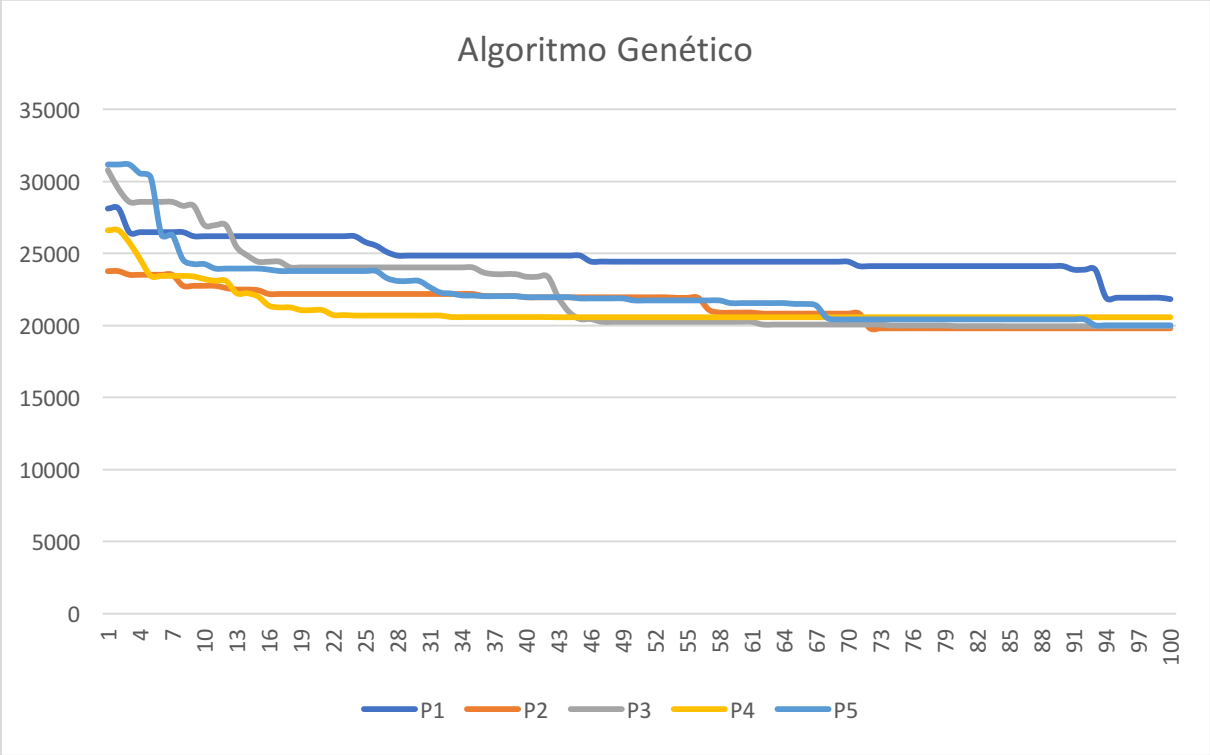

Gráfico 6 5 Experimentos Algoritmo Genético

Gráfico 7 Duración de los 5 experimentos Algoritmo Genético

	deltaMin	deltaMediana	deltaMedia	deltaMax	deltaSigma
Prueba 1	29,9114%	45,1380%	45,3871%	41,0956%	63,7950%
Prueba 2	35,6048%	47,1355%	44,7486%	39,3123%	51,9730%
Prueba 3	37,5850%	50,8633%	49,0136%	29,9057%	41,2623%
prueba 4	27,9379%	50,7198%	47,9990%	40,2072%	54,3343%
prueba 5	36,6845%	52,9118%	48,6043%	41,1036%	60,3870%
PROMEDIO	33,5447%	49,3537%	47,1505%	38,3249%	54,3503%

Tabla 4 Reducción Población Inicial experimentos Algoritmo Genético

	P1	P2	P3	P4	P5
Población Inicial	31152	30729	31917	28524	31621
Población Final	21834	19788	19921	20555	20021

Tabla 5 Mínimos valores alcanzados experimentos Algoritmo Genético

3.3. Comportamiento de los parámetros Enfriamiento Simulado

Se realizó un experimento de comparación de tamaños de población con el fin de encontrar un tamaño de población que cumpliera con un tiempo apropiado de iteración y otorgara una respuesta que redujera el valor de fitness de la población final.

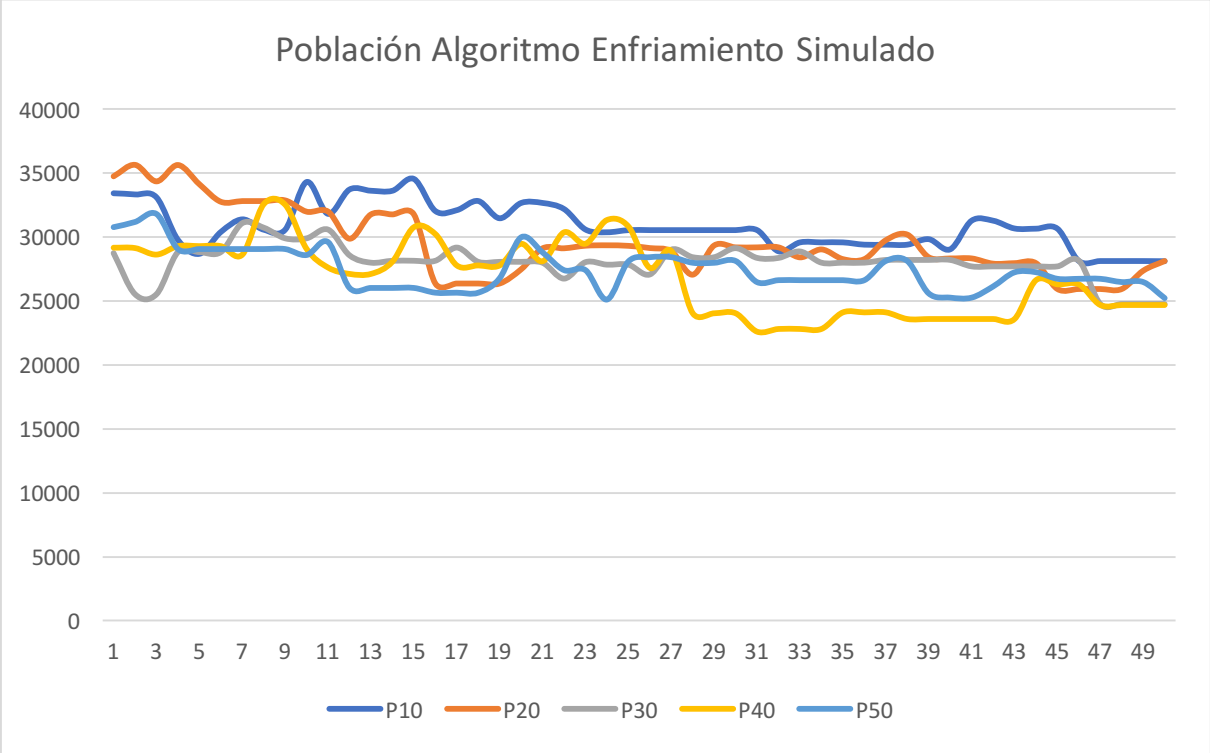

Gráfico 8 Evaluación del tamaño de población algoritmo Enfriamiento Simulado

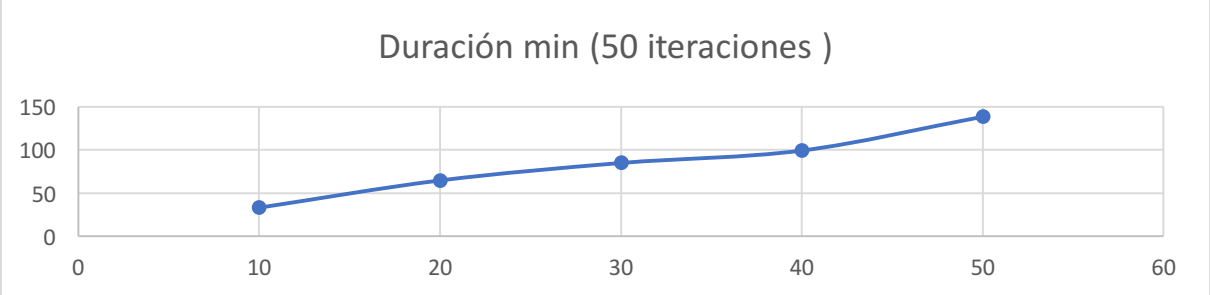

Gráfico 9 Duración de ejecución prueba de población Algoritmo Enfriamiento simulado

Tamaño	Duración (min)
10	33,40065697
20	64,89368605
30	85,08946505
40	99,46444645
50	138,5639574

Tabla 6 Duración de ejecución prueba de población Algoritmo Enfriamiento Simulado

Del Gráfico 8 se puede inferir que los tamaños de población con mejores resultados fueron los de 30 y 40 individuos, su diferencia en tiempo de ejecución es de 14,37 minutos, por lo cual se decide escoger el de menor tiempo, ya que obtiene un resultado óptimo y requiere de menos tiempo de ejecución.

Se realizó un experimento de comparación del parámetro K de disminución de la temperatura con el fin de encontrar un valor óptimo para el problema específico.

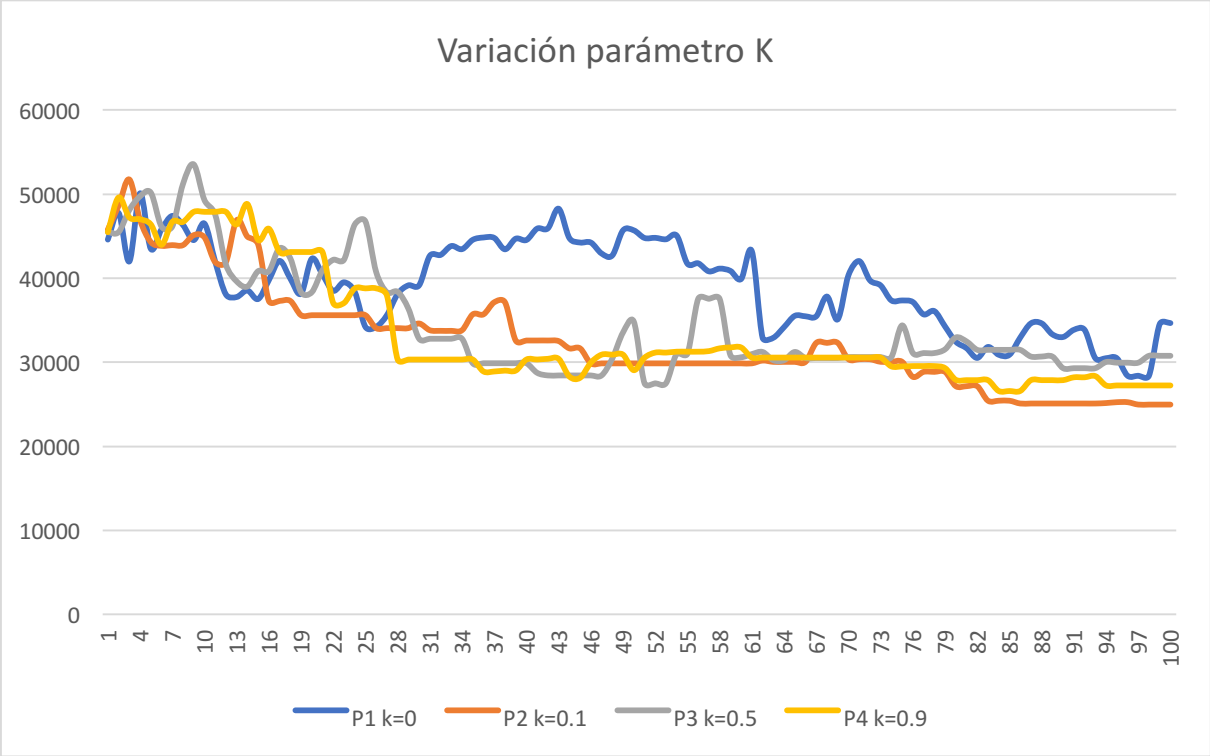

Gráfico 10 Variación del parámetro K algoritmo Enfriamiento Simulado

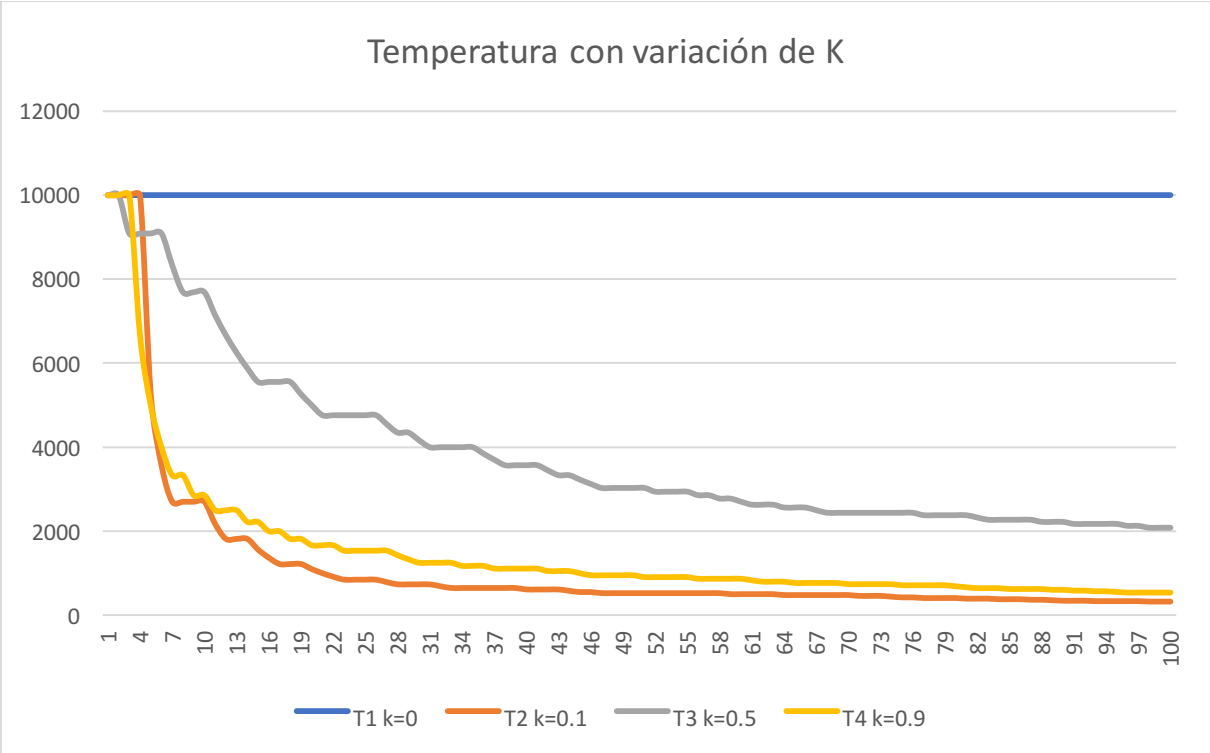

Gráfico 11 Variación de la temperatura algoritmo Enfriamiento Simulado

Del Gráfico 10 y del Gráfico 11 se puede observar que el mejor K sería el de 0.2, es el valor del parámetro que mejor resultado tiene final de fitness, adicionalmente la curva de reducción de temperatura es mucho más inclinada lo que hace que el algoritmo este menos expuesto a desmejorar. Debido a esto se elige a k = 0.2.

3.4. Enfriamiento Simulado Población inicial Aleatoria

Se realizaron 5 experimentos de inicio de población aleatorio basados en el problema [Tabla 1] buscando la solución más óptima en 100 iteraciones, se utilizó 0.2 como parámetro k, y $T_{i\ de}$ 10.000 y una población total de 30 individuos.

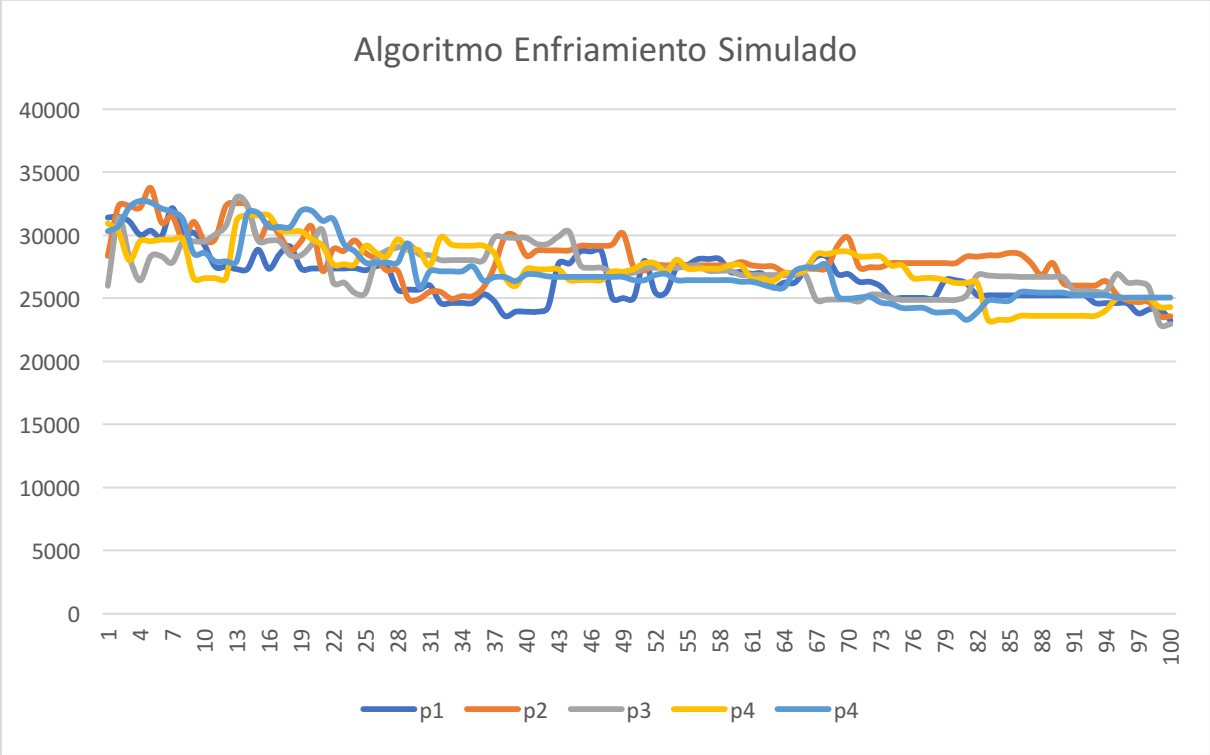

Gráfico 12 5 Experimentos Algoritmo Enfriamiento Simulado

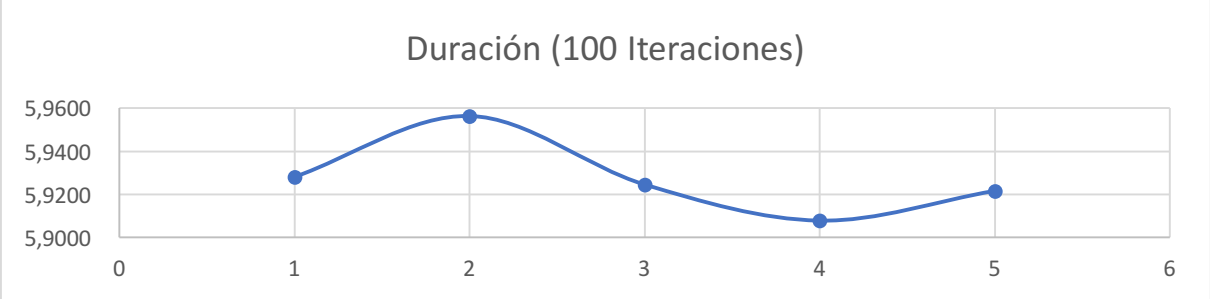

Gráfico 13 Duración 5 Experimentos Algoritmo Enfriamiento Simulado

	deltaMin	deltaMediana	deltaMedia	deltaMax	deltaSigma
Prueba 1	18,1453%	26,3020%	26,0171%	15,0182%	21,5116%
Prueba 2	21,2071%	24,9641%	22,4821%	19,5963%	-0,2876%
Prueba 3	11,6658%	16,1503%	19,0324%	21,7478%	11,3837%
prueba 4	25,1638%	24,5259%	20,7587%	15,1909%	1,7578%
prueba 5	21,9397%	23,6600%	21,4109%	19,2217%	6,3410%
PROMEDIO	19,6243%	23,1205%	21,9402%	18,1550%	8,1413%

Tabla 7 Reducción Población Inicial experimentos Algoritmo Enfriamiento Simulado

	P1	P2	P3	P4	P5
Población Inicial	28.371,0000	29.957,0000	25.999,0000	32.507,0000	32.097,0000
Población Final	23.223,0000	23.604,0000	22.966,0000	24.327,0000	25.055,0000

Tabla 8 Mínimos valores alcanzados experimentos Algoritmo Enfriamiento Simulado

3.5. Población inicial igual, Algoritmo Genético y Enfriamiento simulado

El siguiente experimento se partió de una población fija, no aleatoria, de individuos con el fin de realizar una comparación entre el algoritmo Genético y el Enfriamiento Simulado. Se realizaron 100 iteraciones, una probabilidad e mutación igual 0.2, un parámetro k igual 0.1 y una T_i igual 10.000.

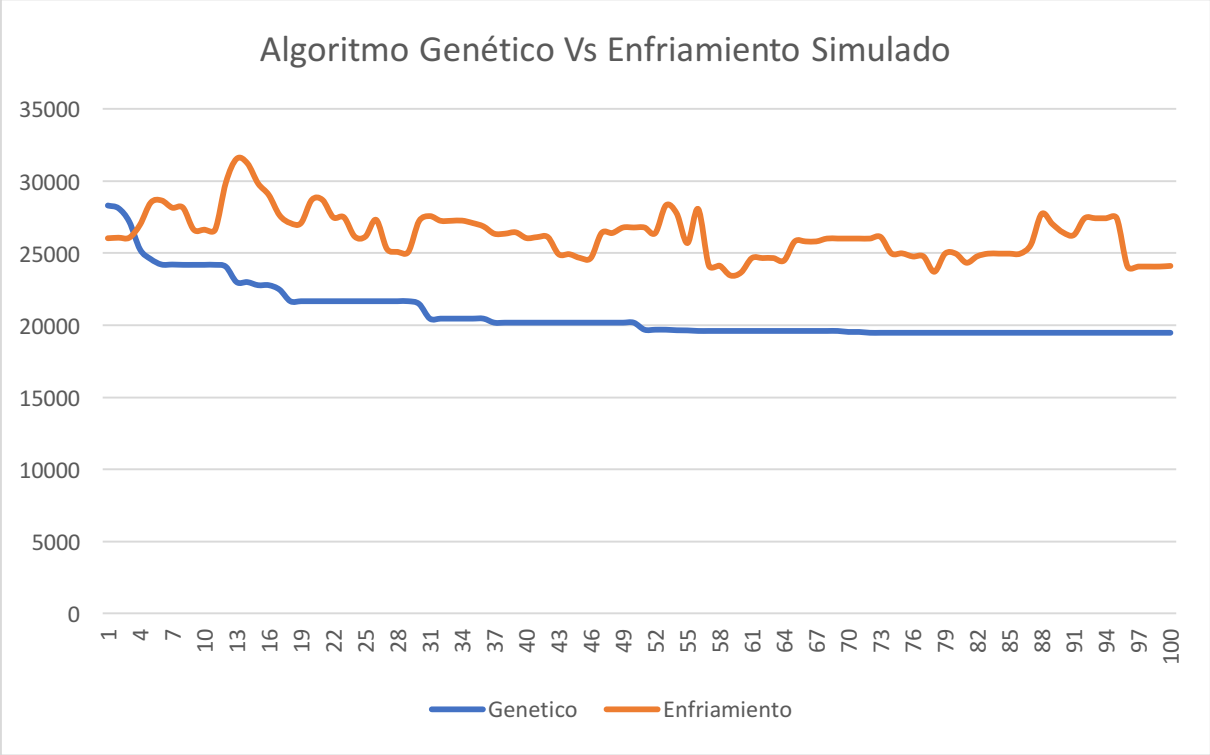

Gráfico 14 Algoritmo Genético Vs Enfriamiento Simulado

	Población Inicial	Genético	Enfriamiento
Min	28297,00	19499,00	24120,00
Mediana	40662,00	19499,00	30766,50
Media	40662,00	19499,00	30766,50
Max	51541,00	26854,00	37908,00
Sigma	4893,25	1322,63	3317,66

Tabla 9 Población Inicial Algoritmo Genético Vs Enfriamiento Simulado

	deltaMin	deltaMediana	deltaMedia	deltaMax	deltaSigma
Genético	31,0916%	52,0461%	52,0461%	47,8978%	72,9703%
Enfriamiento	14,7613%	24,3360%	24,3360%	26,4508%	32,1992%

Tabla 10 Reducción Población Inicial Algoritmo Genético Vs Enfriamiento Simulado

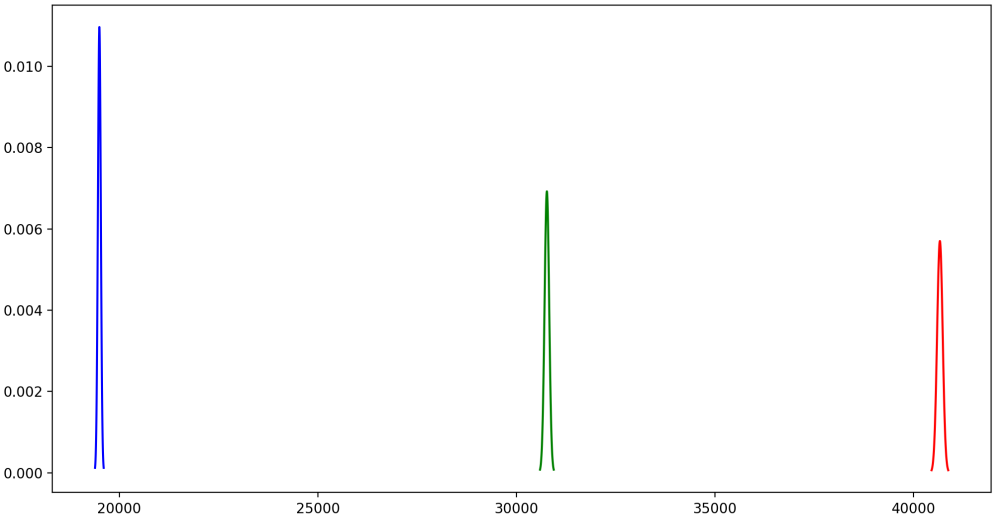

Gráfico 15 Comparación de poblaciones Algoritmo Genético Vs Enfriamiento Simulado

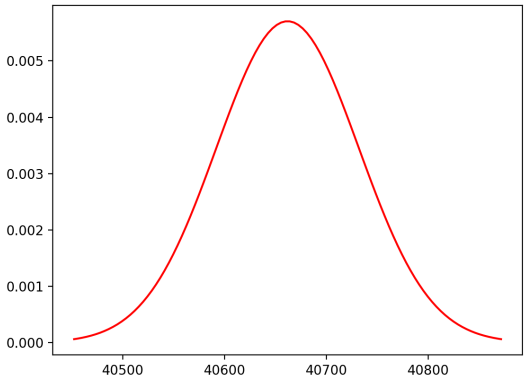

Gráfico 16 Distribución de población inicial

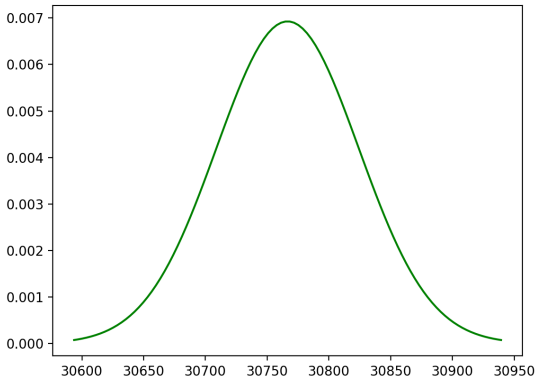

Gráfico 17 Distribución de población Enfriamiento Simulado

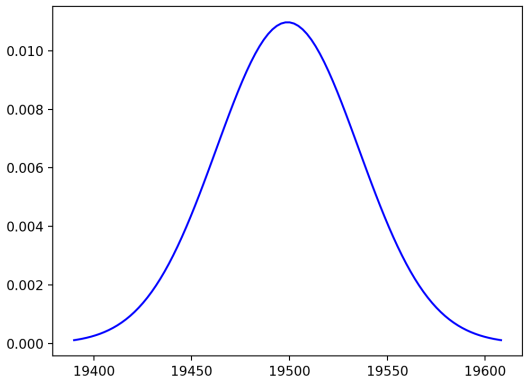

Gráfico 18 Distribución de población Genético

El tiempo de ejecución del algoritmo genético fue de 4,4537 horas y el tiempo de ejecución del Enfriamiento Simulado fue 5,9103 h con una diferencia de 1,4566 horas entre ambos.

En el Gráfico 14 se puede observar que la reducción de fitness en el algoritmo genético es más estable y más rápido que en el enfriamiento simulado. Se puede apreciar que el valor final mejor de fitness (el más bajo) en ambos casos sería el del a algoritmo genético haciéndolo mejor en cuestiones de reducción por iteración y valor final. El valor final del algoritmo genético fue de 19499 y el del enfriamiento simulado fue de 24120 con una reducción con respecto al mejor individuo inicial de 31,0916% y 14,7613% respectivamente (ver Tabla 9).

En el Gráfico 15 se puede apreciar que ambas soluciones mejoran de forma significativa la población inicial, pero es claro que el algoritmo genético finaliza con una población menos dispersa y con mejores soluciones que el algoritmo de enfriamiento simulado, pues el peor individuo del algoritmo genético sigue siendo mejor que el mejor individuo del enfriamiento simulado.

3.6. Comparación de mejora de solución, Algoritmo Genético y Enfriamiento simulado

Se realizó un experimento de comparación de soluciones para comparar y evaluar los dos algoritmos como buscadores de soluciones a partir de una población no muy buena inicial y como mejoradores de la solución obtenida del otro, genético-enfriamiento, y enfriamiento-genético. Se realizaron 100 iteraciones, una probabilidad e mutación igual 0.2, un parámetro k igual 0.1 y una T_i igual 10.000.

	Población Inicial	Búsqueda Enfriamiento	Mejora Genético	Búsqueda Genético	Mejora Enfriamiento
Min	28297	23177	21228	18488	22204
Mediana	40662	32549	21228	18488	31115
Media	40459	32063	21531	19161	31202
Max	51541	41040	23777	26859	43321
Sigma	4893,25	3853,56	727,21	2056,84	4691,23

Tabla 11 Comparación de mejora de solución de búsqueda y mejoramiento

	deltaMin	deltaMediana	deltaMedia	deltaMax	deltaSigma
Búsqueda Enfriamiento	18,0938%	19,9523%	20,7519%	20,3741%	21,2473%
Mejora Genético	8,4092%	34,7814%	32,8485%	42,0638%	81,1288%
Búsqueda Genético	34,6645%	54,5325%	52,6413%	47,8881%	57,9657%
Mejora Enfriamiento	-20,0995%	-68,2957%	-62,8427%	-61,2904%	-128,0795%

Tabla 12 Reducción Población Inicial en comparación de mejora de solución de búsqueda y mejoramiento

Como se puede apreciar en la Tabla 11 el algoritmo genético presenta mejores soluciones buscando una solución mejor a partir de una solución inicial no muy buena, y también mejora sustancialmente la solución previamente encontrada por el enfriamiento simulado. Igualmente, el enfriamiento simulado presenta un mejoramiento más significativo si la solución inicial no es buena, ya que si parte de una solución previamente optimizada, puede llegar a perjudicar la solución como se aprecia en la Tabla 12.

4. Conclusiones

Un valor alto de probabilidad de mutación puede ser volátil, esto significa que puede obtener tan buenos resultados como malos, esto se debe a que su búsqueda puede estar más diversificada pero también puede desviarse a valores que no son los más óptimos.

Aunque en algunos casos el algoritmo de enfriamiento simulado pueda otorgar una mejor solución, siempre va a haber una posibilidad de desmejoramiento ya que hay una probabilidad de escoger un vecino con un fitness más alto, que podría llevar a la solución fuera de un óptimo local y hallar el óptimo global, así mismo como es alejar la solución de un muy buen individuo.

El algoritmo genético reduce de forma significativa la distancia que hay entre los datos (sigma), haciendo que todos los individuos de la población presenten muy buenas soluciones, mientras que el algoritmo de enfriamiento simulado mantiene un sigma muy similar al de la población inicial, esto puede deberse a su comportamiento de mejoramiento a través de vecino, por lo que un individuo puede llegar a un valor local mejor no muy lejano del inicial (ver Gráfico 15).

En un caso práctico de necesitarse detener el algoritmo debido a cuestiones de tiempo y necesidad y trabajar con la última solución obtenida el algoritmo genético asegura que el individuo que entregara será mejor al anterior y a la inicial, mientras que en el enfriamiento simulado como hay una búsqueda de mejora la solución podría encontrarse en una desmejora en comparación con su solución anterior o inicial. Esto se puede apreciar en el Gráfico 14 ya que la curva de mejoramiento por iteración en el genético es decreciente, mientras que el enfriamiento simulado proyecta una especie de aleatoriedad.

Si se analiza el Gráfico 12 se puede apreciar que la línea p3 que inicia en una población con mejor fitness que el p1 llega a tener más aleatoriedad y búsqueda entre soluciones y vecinos, mientras que el p1 logra alcanzar una mejor solución rápidamente más rápido que p3. En alrededor de la iteración 40 p1 ha obtenido una muy buena solución a partir de una muy mala inicial, mientras que p3 se encuentra en una peor solución de la que inicio. Con esto se puede concluir que el enfriamiento simulado para este problema en específico encuentra mejores soluciones si parte de una solución inicial con un alto fitness y alta diversidad.

Para el problema planteado el algoritmo genético presenta mejores resultados que el algoritmo de enfriamiento simulado. El mejor individuo obtenido en todos los experimentos para la solución del problema [Tabla 1] fue [4, 7, 10, 11, 9, 6, 17, 2, 1, 5, 13, 19, 12, 8, 18, 16, 14, 3, 20, 15].

5. Bibliografía

[1] Akbari, Ziarati (2010). "A multilevel evolutionary algorithm for optimizing numerical functions" IJIEC 2 (2011): 419–430

[2] Ornby, G. S.; Linden, D. S.; Lohn, J. D., *Automated Antenna Design with Evolutionary Algorithms*.

[3] Deb, Kalyanmoy; Spears, William M. (1997). "C6.2: Speciation methods". *Handbook of Evolutionary Computation*. Institute of Physics Publishing.

[4] Schmitt, Lothar M (2004), *Theory of Genetic Algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling*, Theoretical Computer Science 310: 181–231

[5] John H. Holland (2001). *Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand*. Retrive http://www2.econ.iastate.edu/tesfatsi/holland.gaintro.htm.

[6] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220