INTELIGENCIA ARTIFICIAL II

TRABAJO PRÁCTICO N°1

INTEGRANTES:

- Aldao, Antonella 12670
- Berridy, Ignacio 11987

INTRODUCCIÓN

El presente informe tiene como intención explicar brevemente los algoritmos desarrollados como solución a los ejercicios obligatorios propuestos por la cátedra y mostrar las pruebas realizadas para la optimización de estos. En él se encontrarán gráficos y conclusiones obtenidas de los mismos, con el fin de explicar el porqué de las decisiones tomadas para ciertos parámetros.

ALGORITMO A*

Para la codificación de este, al igual que para el resto de los ejercicios, se utilizó un paradigma orientado a objetos.
En este caso utilizamos 6 clases:

- box :se encuentran los atributos compartidos tanto por los espacios donde puede moverse como por las estanterías. Es la representación de los espacios que componen el almacén
- rack: hereda los atributos de box y le asigna dos atributos nuevos que son necesarios que no son necesarios para los espacios pero sí para las estanterías
- layout: crea la matriz principal donde cada elemento de ella puede ser box o puede ser rack dependiendo si es una estantería o no. Es la discretización del almacén espacialmente
- path: ejecuta el algoritmo A* propiamente dicho para el cual necesita de la matriz generada por layout
- interfaz_1: creamos una interfaz con tkinter para el ingreso de los valores y se conecta con interfaz_2
- interfaz_2: se muestra por pantalla el layout generado y sobre él se puede ver el camino dado como solución.

Cabe aclarar que las últimas dos clases solo se utilizaron con el fin de hacer más amigable el programa y poder visualizar si el algoritmo funcionaba correctamente.

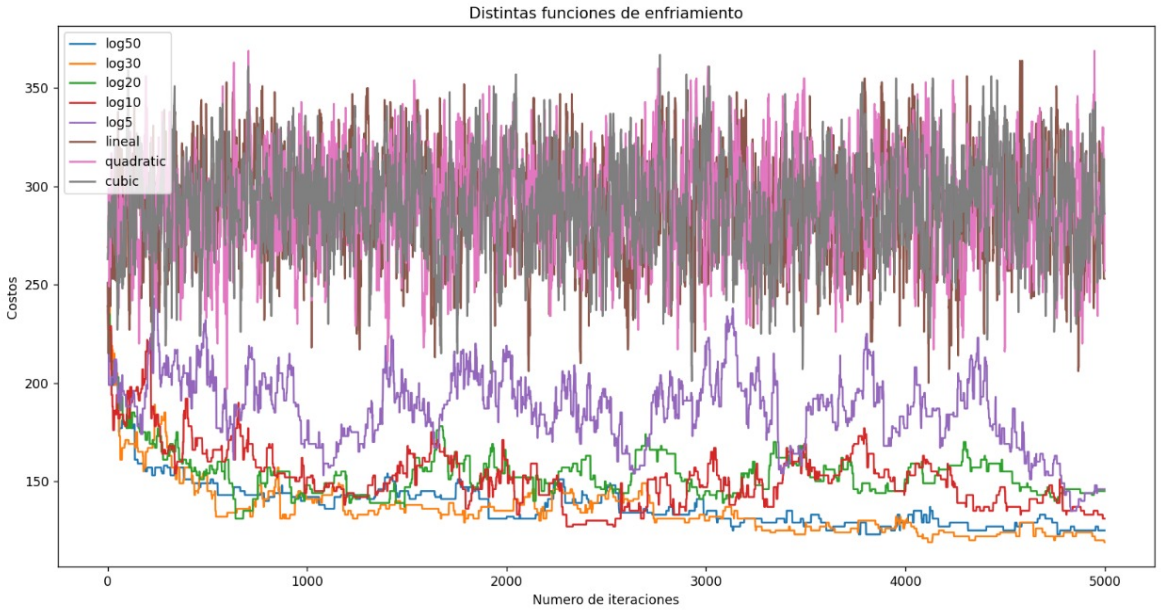
En este ejercicio no se presentaron complicaciones ni fue necesario elegir ningún parámetro.

SIMULATED ANNEALING

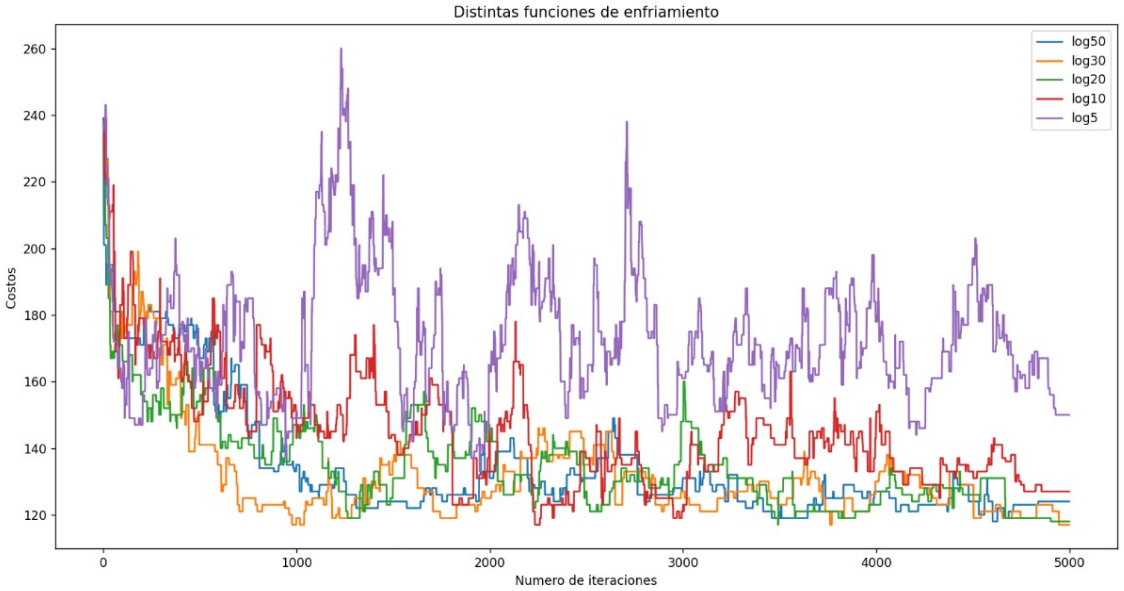
Reutilizando los códigos del ejercicio 1 solo se agregó una clase, simulated_annealing, la cual hace lo que su nombre indica.
En esta parte del trabajo se presentaron algunas complicaciones relacionadas a la selección de la función de enfriamiento y sus parámetros. A continuación se muestran las pruebas realizadas.

Test 1: Selección de la función de enfriamiento

Para esta parte se probaron varias funciones de enfriamiento para ver su repercusión en el resultado del algoritmo. Se obtuvo lo siguiente

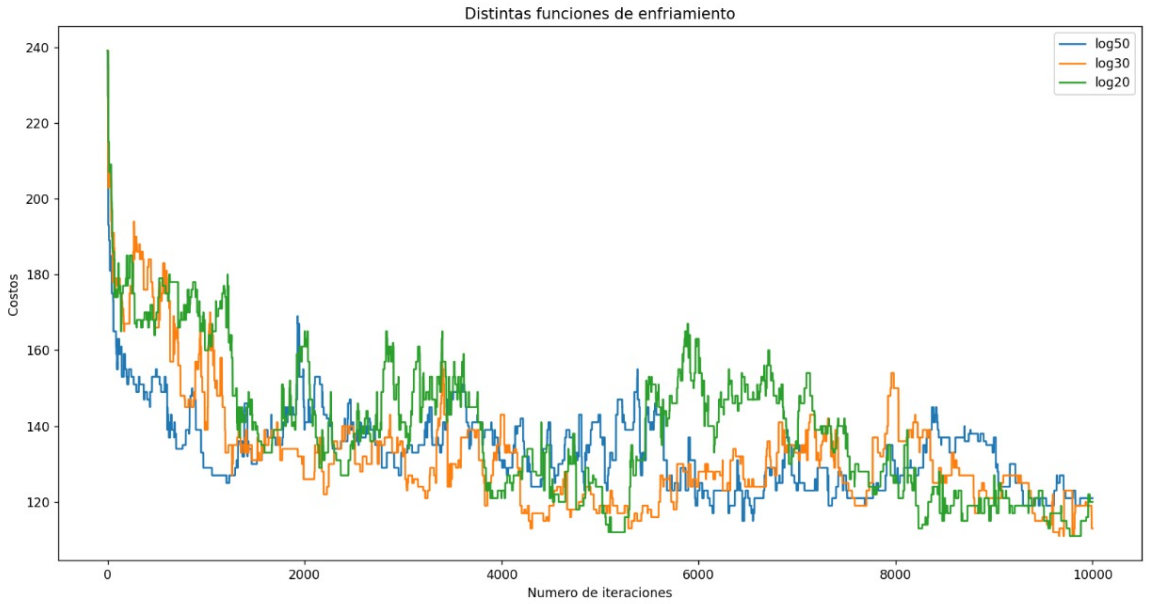


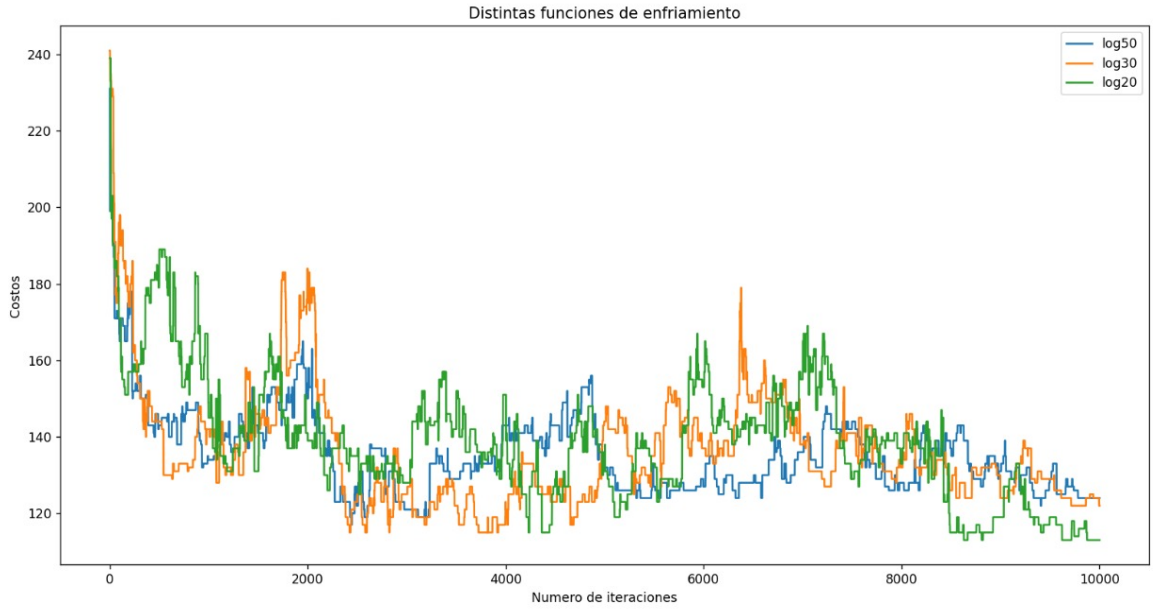
La funciones lineal, cuadrática y cúbica obtuvieron malos resultados frente a las funciones logarítmicas. Por ello se procedió a analizar solamente las últimas, obteniéndose lo siguiente:



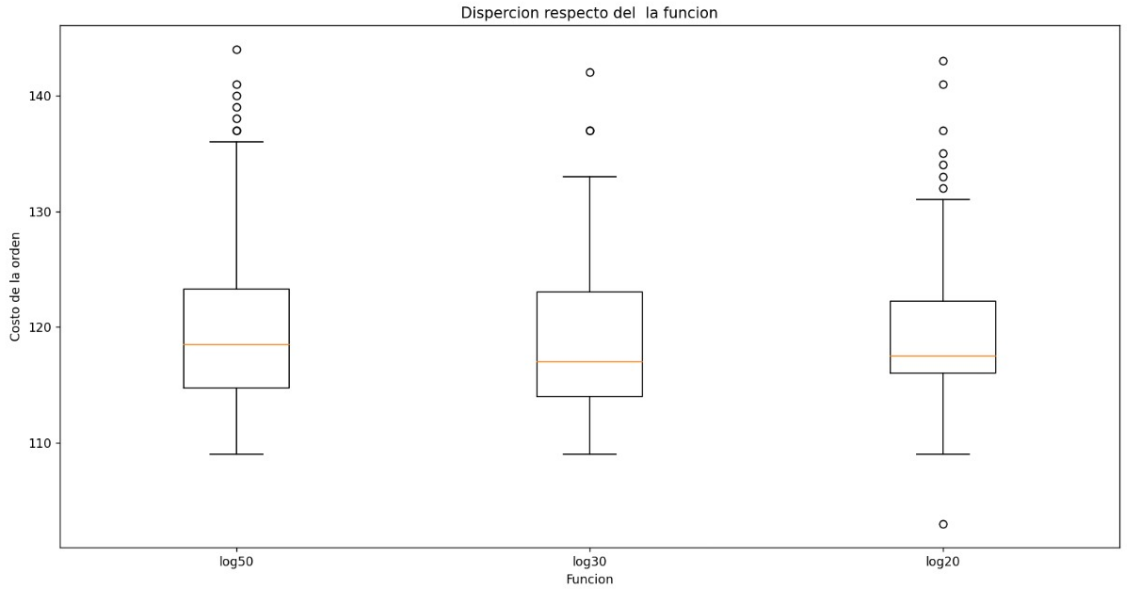
Se observa que al aumentar la base el resultado mejora considerablemente por lo cual las bases 5 y 10 fueron descartadas.

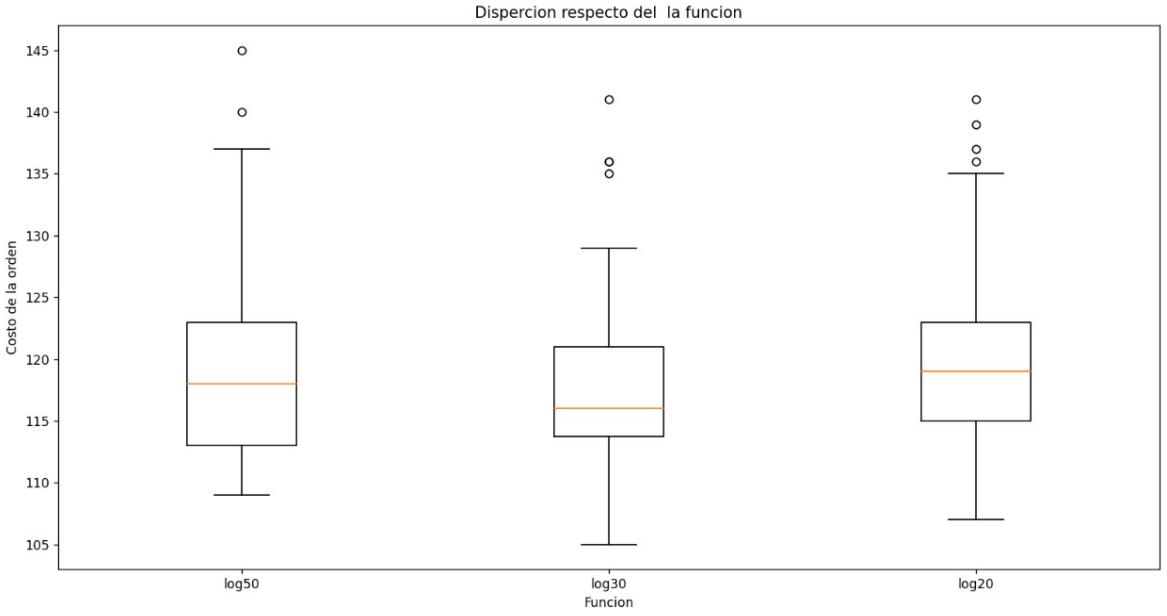
Analizando solamente las de 20, 30 y 50

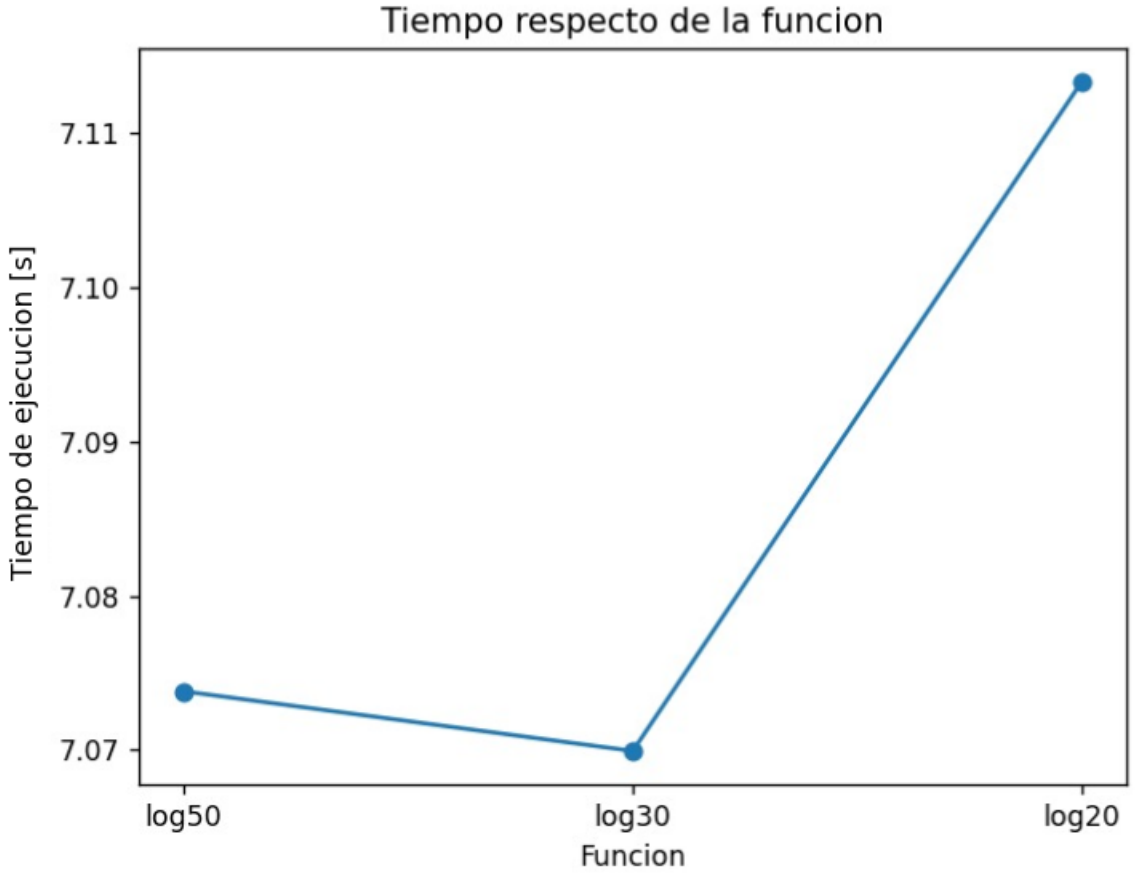




En estos gráficos se ve que cualquiera de las 3 es una buena opción si analizamos sólo el costo, por lo que se decidió analizar la dispersión y tiempo de ellas para decidir con cual quedarse. Se usó la orden 5 y se ejecutó 100 veces el algoritmo por cada función.


Dispercion respecto del la funcion
Costo de la orden
140
130
120
110
log50
log30
log20
Funcion

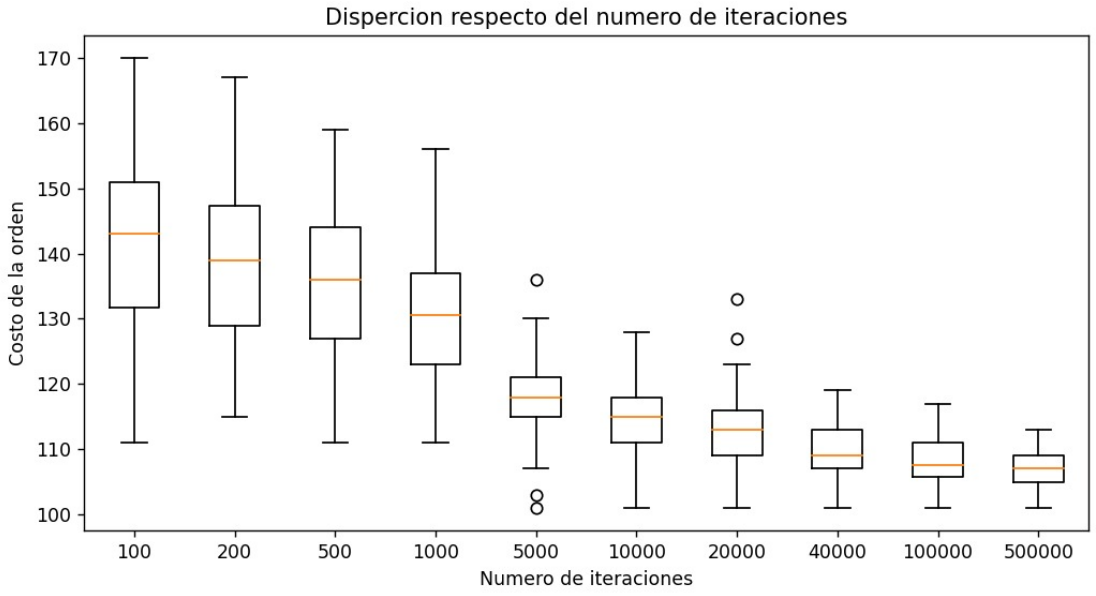

Dispercion respecto del la funcion
Costo de la orden
145
140
135
130
125
120
115
110
105
log50
log30
log20
Funcion

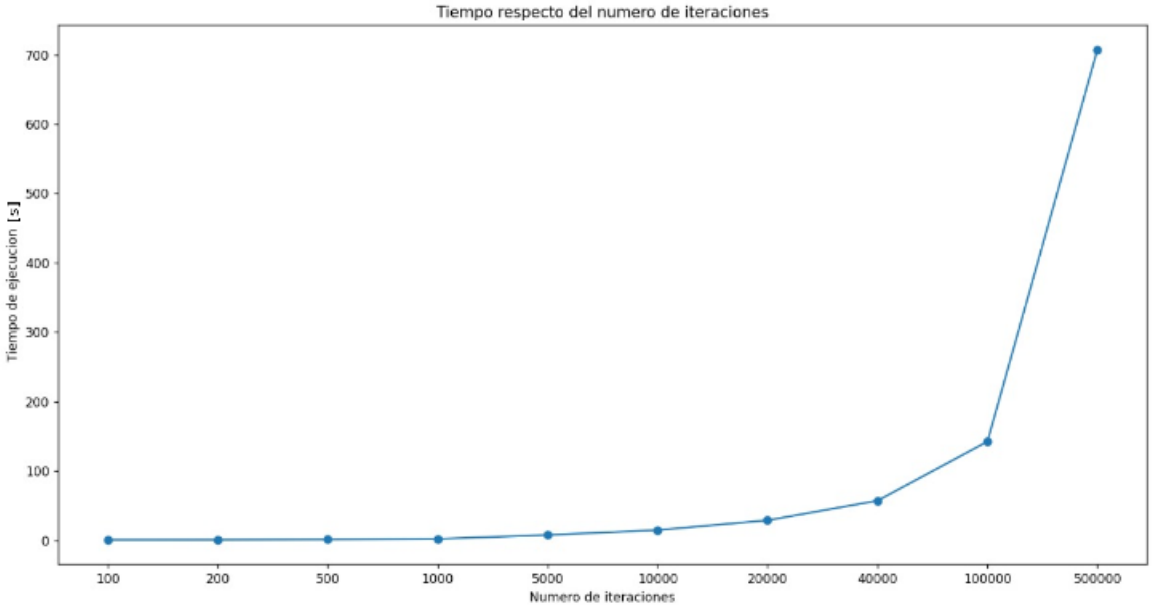


Al calcularse los valores de tiempo se vió que estos eran prácticamente iguales por lo cual al momento de la decisión no se tuvo en cuenta este valor.
Se optó por usar una base de 30 ya que si bien los gráficos eran muy parecidos entre ellos si se notaba una pequeña dispersión mayor en el de la base 50, y con respecto al de base 20 en él se encontraban más cantidad de valores aislados respecto al de 30.

Test 2: Selección de t

Luego de haber adoptado como función de enfriamiento logaritmo base 30 se procedió a determinar un valor de t, para ello se tuvo en cuenta el costo obtenido del simulated annealing, de la dispersión de los valores luego de hacer 100 veces la misma orden.





Tiempo con t = 100: 0.8485777378082275 [s]
Tiempo con t = 200: 0.999650239944458 [s]
Tiempo con t = 500: 1.39640474319458 [s]
Tiempo con t = 1000: 2.0986249446868896 [s]
Tiempo con t = 5000: 7.73996901512146 [s]
Tiempo con t = 10000: 14.791544675827026 [s]
Tiempo con t = 20000: 28.888020515441895 [s]
Tiempo con t = 40000: 57.16605186462402 [s]
Tiempo con t = 100000: 142.42023181915283 [s]
Tiempo con t = 500000: 707.2604374885559 [s]

Se puede ver cómo a medida que aumenta la cantidad de iteraciones el valor de costo disminuye y también lo hace la dispersión del resultado, pero al observar el segundo gráfico vemos que esto es a costa del incremento en el tiempo de ejecución del programa.
Se eligió usar los parámetros que dieron como resultado 10000 iteraciones ya que la relación costo-dispersión-tiempo era la más óptima. Esto se puede observar al ver los valores correspondientes a 10000 y 20000 iteraciones en donde, tanto el costo medio (marcado en naranja) como la dispersión de los resultados, son bastante parecidos, pero el tiempo de ejecución en el segundo caso es el doble

GENETIC ALGORITHM

Para la realización del algoritmo genético se creó una clase del mismo nombre, que recibe una distribución inicial del almacén, un conjunto de órdenes históricas y devuelve una distribución más óptima para dicho almacén.

Las principales funciones del algoritmo son la de selección de padres, el cálculo de fitness de la población y el cruce entre padres para formar una nueva población.

Fitness

A cada individuo de la población se le asigna un valor que se utiliza para hacer una comparación cuantitativa de los mismos, este valor es el fitness.
Para este caso el fitness de cada individuo se calcula como la suma de realizar todos los pedidos del historial de órdenes sobre la distribución del almacén que plantea el individuo en cuestión. Para esto primero se le aplica el algoritmo simulated annealing a cada orden para esta distribución y sobre eso se da el costo por orden. Es por esto la importancia que tiene el hecho de que al ejecutar repetidas veces el algoritmo simulated annealing sobre una misma orden y distribución los valores de costo obtenidos sean similares.

Selección de los padres

Primero se calcula un valor de fitness para cada uno de los individuos de la población y con base en eso se le asigna una probabilidad a cada uno. Se genera un número aleatorio y con él se elige al padre de la siguiente manera:

Por ejemplo:

Tenemos una población de 10 individuos con las siguientes probabilidades

1. 0.1
2. 0.15
3. 0.05
4. 0.2
5. 0.1
6. 0.05
7. 0.15
8. 0.05
9. 0.05
10. 0.1

	1	2	3	4	5	6	7	8	9	10
0	0.1	0.25	0.3	0.5	0.6	0.65	0.8	0.85	0.9	1

Depende de qué rango está el número aleatorio se selecciona el padre, como indica la figura anterior. Esto se repite hasta obtener dos padres distintos

Crossover:

Para la generación de los hijos se utilizó cruce de órdenes, para ello se seleccionaron dos padres de la forma antes mencionada. Para el cruce se tomaron aleatoriamente dos puntos de corte. Al primer hijo se le agregó entre el primer punto y el segundo punto los elementos del padre 2 y al hijo dos se le agregaron los del padre 1. Luego se completó al primer hijo desde el último elemento agregado completándolo con los valores del padre 1 desde la misma posición siempre y cuando no existieran ya dichos valores. Luego se completó desde el inicio hasta el primer elemento del hijo completándolo con los valores del padre que no estuvieran en el hijo. Lo mismo se hizo con el segundo hijo pero usando los elementos del segundo padre.
Ej: supongamos los siguientes datos

PADRE1	2	5	7	8	9	1	3	6	4
PADRE2	3	9	6	4	2	7	5	8	1

Se escoge la posición 2 y 4 aleatoriamente para el cruce, se agregan los valores del padre 1 al hijo 2 y viceversa

HIJO1			6	4	2				
HIJO2			7	8	9				

Se completa el hijo 1 con los valores del padre 1 desde la posición 4 (sin repetir elementos). Lo mismo con el hijo 2 y el padre 2

HIJO1			6	4	2	1	3		
HIJO2			7	8	9	5	1		

Se llenan los huecos (si es que existieran) con los primeros valores de los padres (sin repetir valores)

HIJO1			6	4	2	1	3	5	7
HIJO2			7	8	9	5	1	3	6

Se completan los valores faltantes de los hijos

HIJO1	8	9	6	4	2	1	3	5	7
HIJO2	4	2	7	8	9	5	1	3	6

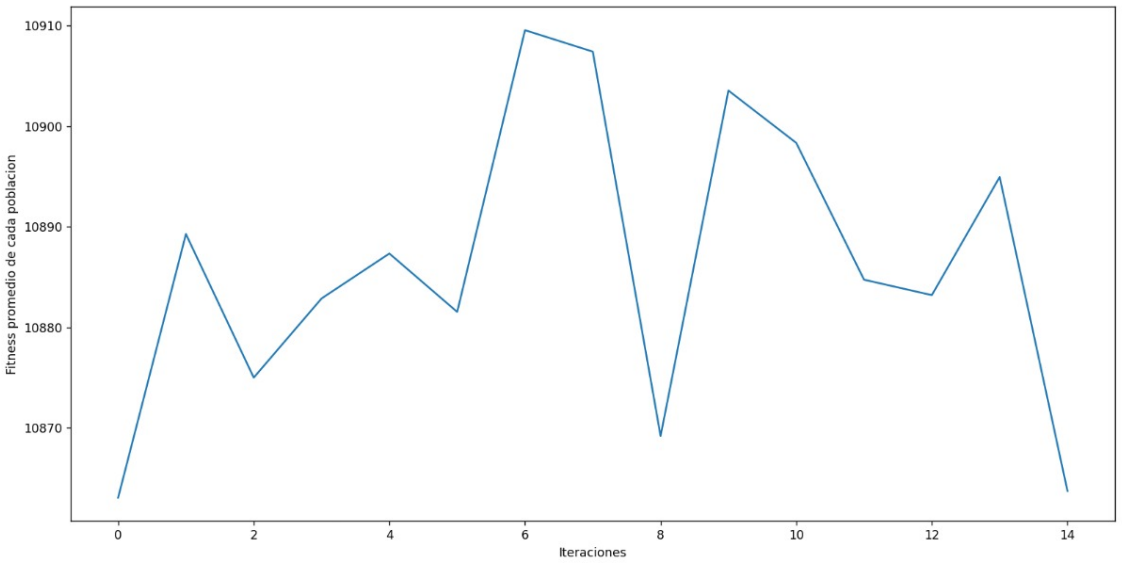
Luego de obtener los hijos, estos pueden mutar según una probabilidad.

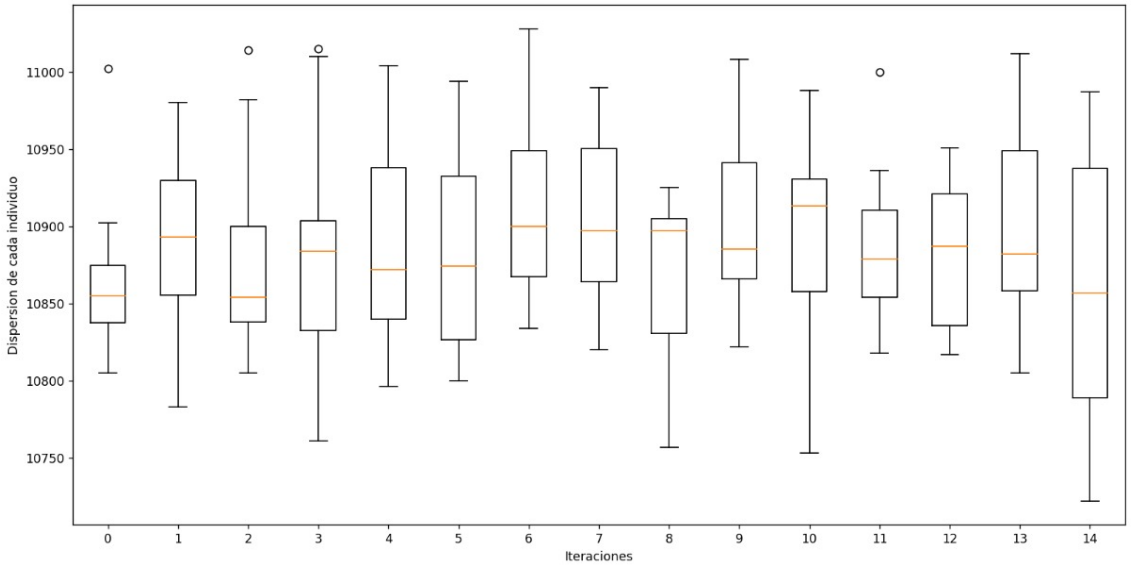
La ejecución del programa con los siguientes parámetros arrojó los siguientes resultados.

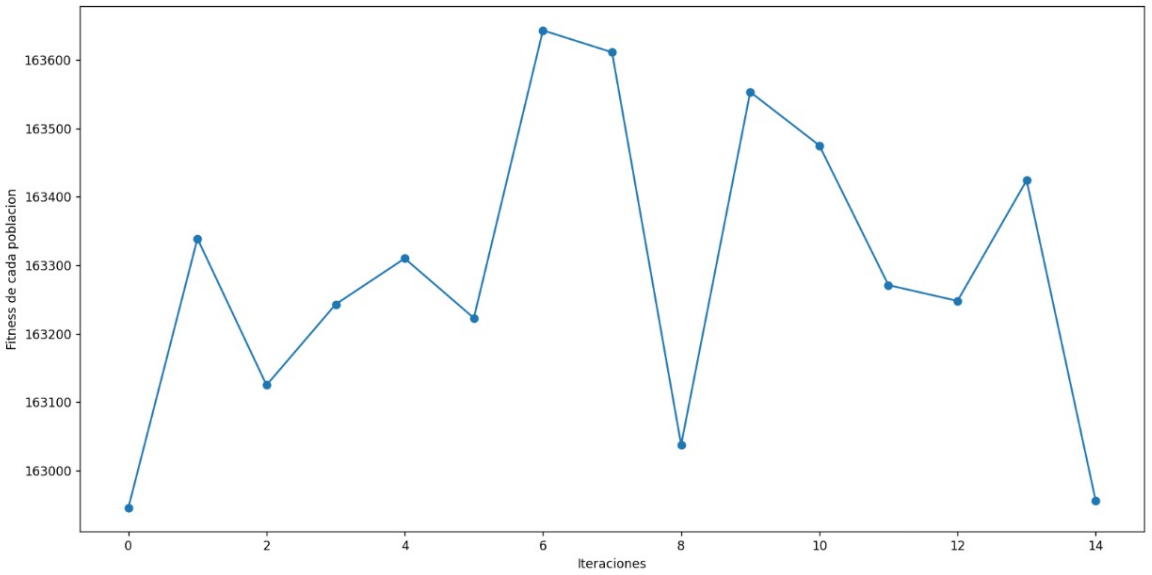
Parámetros:

- Tamaño de población = 15
- Iteraciones en simulated annealing = 10000
- Función de enfriamiento : log30(t), con t = t-1 en cada iteración
- Iteraciones máximas del algoritmo genético = 15

Resultados:




Dispersion de cada individuo
11000
10950
10900
10850
10800
10750
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Iteraciones


Fitness de cada poblacion
163600
163500
163400
163300
163200
163100
163000
0
2
4
6
8
10
12
14
Iteraciones

Grupo 6 - A* (Ejercicio 1)