Inteligencia artificial II

Trabajo práctico N°1

Integrantes:

- Costamagna Luciana
- Felicito Miled
- Martinez Ferran

EJERCICIO 1

Dado un almacén con un layout similar al siguiente, calcular el camino más corto (y la distancia) entre 2 posiciones del almacén, dadas las coordenadas de estas posiciones, utilizando el algoritmo A*.

Resolución:

Para la resolución de este ejercicio creamos la clase nodo, la cual tiene todos los atributos necesarios para armar la grilla que simula el almacén con los productos, sus estanterías y pasillos. Los métodos de la clase calculan el costo y la heurística, para la cual utilizamos la distancia euclidiana entre dos puntos.
El algoritmo propuesto en un principio setea las dimensiones del lugar, y las coordenadas de cada nodo de la grilla. Luego se asignan los vecinos de cada nodo, para luego poder hacer los cálculos necesarios para determinar el camino que se debe seguir.
Finalmente se guardan los lugares visitados, haciendo un backtracking para no volver a caer en ellos y ahorrar tiempo. Al avanzar se tiene en cuenta que el nodo no sea una estantería ni el punto de partida o llegada.
Usando la librería P5 se grafica la grilla con el camino recorrido.

Al comienzo utilizamos como costo de pasar de un nodo a otro un valor de 1, pero luego probamos con un valor de 0,5 y el algoritmo funcionaba de manera más precisa. Al avanzar al ejercicio 2, surgió la necesidad de hacerlo en forma de función, para poder utilizarlo más fácilmente cuando fuera requerido.

EJERCICIO 2

Dada una orden de pedido, que incluye una lista de productos del almacén anterior que deben ser despachados en su totalidad, determinar el orden óptimo para la operación de picking mediante Temple Simulado.

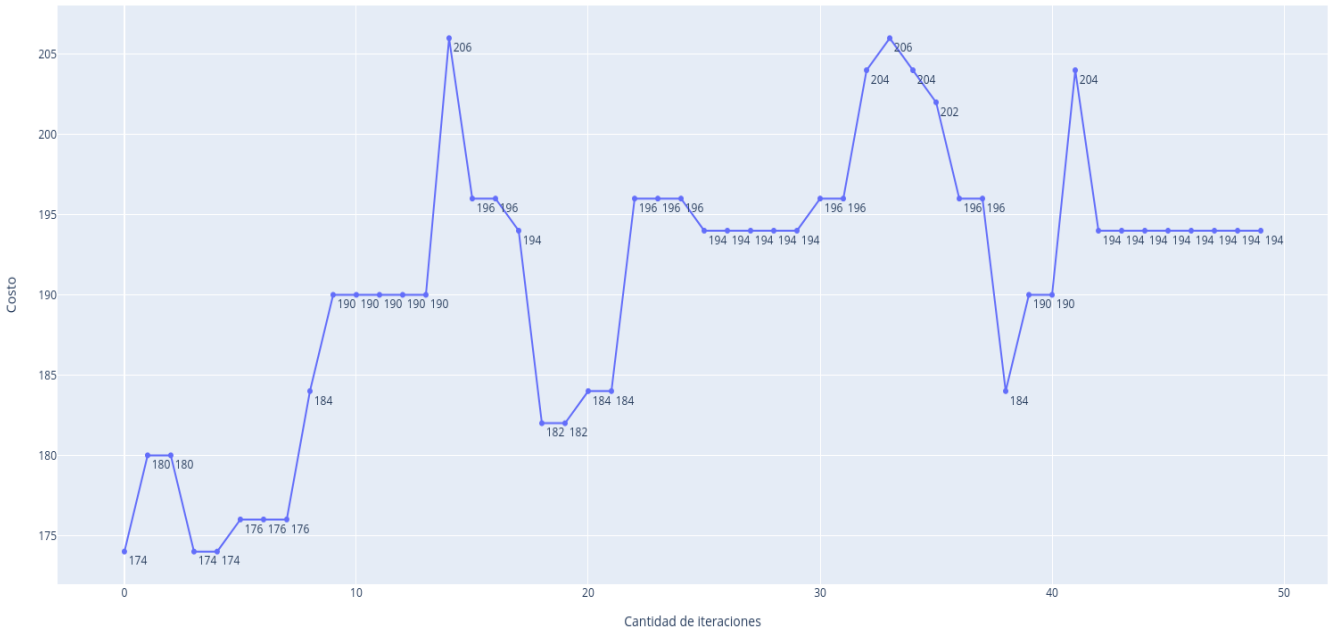
Resolución:

El algoritmo de Temple Simulado comienza recibiendo una lista de una orden de productos, la cual es luego pasada a una de nodos para poder después utilizar la función A* para calcular los costes. Primero sacamos el costo total de sin alterar la orden dada, es decir, recorriendo los productos según lo ingresado, lo cual nos servirá para comparar con otras combinaciones.
Lo siguiente es permutar los productos para ir generando distintas secuencias de recorrido, a las cuales volvemos a hacerles el cálculo del costo y vamos quedándonos con el más bajo a medida que los comparamos con los anteriores. En el caso de encontrar una secuencia de órdenes cuyo valor sea mayor que el previo, la aceptamos pero con una cierta probabilidad. Finalmente elegimos el menor de todos, que supone el camino más eficiente para realizar el picking.

En una primera instancia hicimos el algoritmo sin usar la probabilidad y funcionaba mejor, pero solo para un pequeño número de órdenes, para mayores cantidades se hacía ineficiente. Básicamente era un algoritmo de hill climbing con reinicio aleatorio, el cual iba saltando dentro de todo el espacio de búsqueda de manera random.

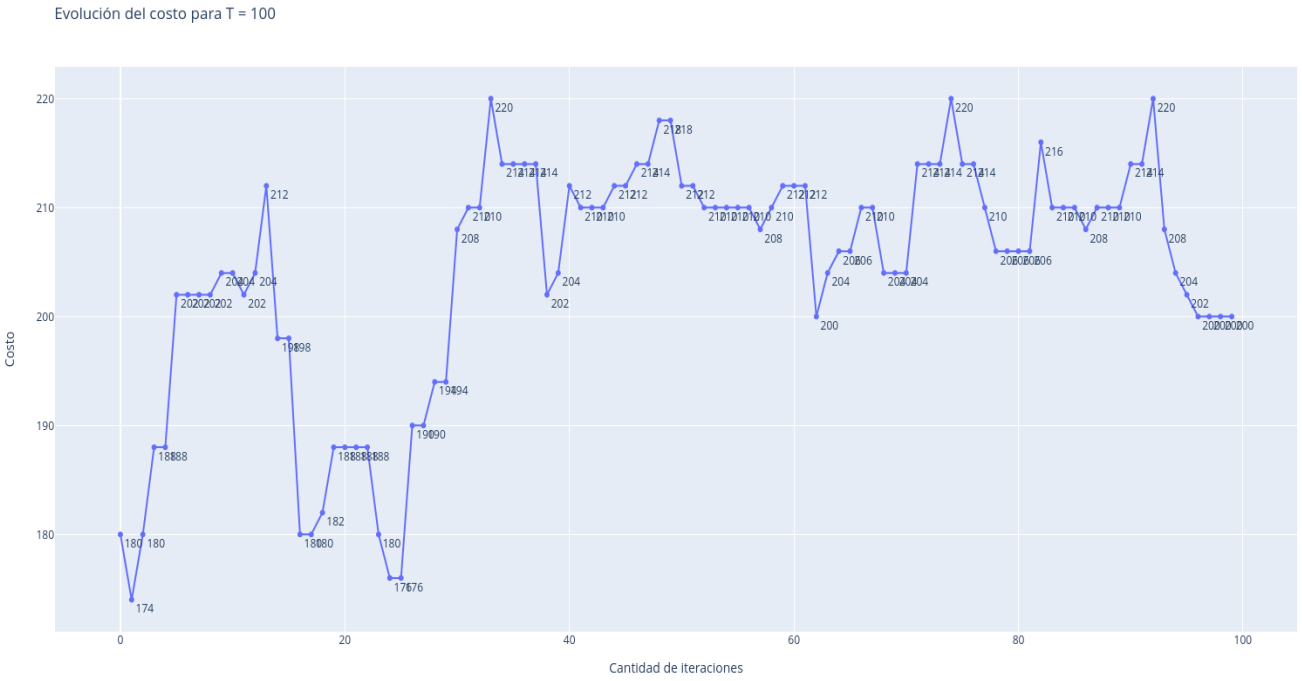
Estos son algunos ejemplos de cómo varía el costo de una orden con distintos valores de T. Cada imagen contiene un gráfico con la evolución y a su vez el dato del valor de cada iteración para mejor interpretación. Debido a que tiene una probabilidad a la hora de elegir el siguiente camino no siempre toma el menos costoso.

Para T = 50



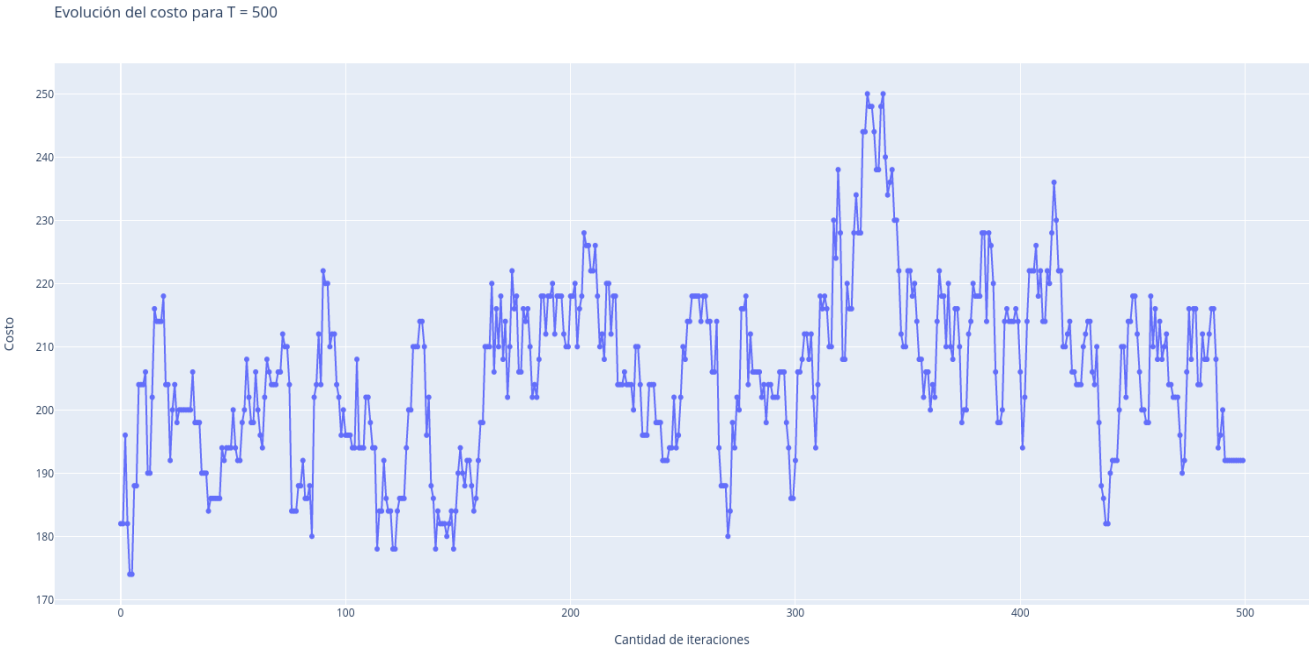

```
[174, 180, 180, 174, 174, 176, 176, 176, 184, 190, 190, 190, 190, 190, 206, 196, 196, 194, 182, 182, 18
4, 184, 196, 196, 196, 196, 194, 194, 194, 194, 194, 196, 196, 204, 206, 204, 202, 196, 196, 184, 190, 190,
204, 194, 194, 194, 194, 194, 194, 194, 194]
```

Para T = 100



```
[180, 174, 180, 188, 188, 202, 202, 202, 202, 204, 204, 202, 204, 212, 198, 198, 180, 180, 182, 188, 188,
 188, 188, 180, 176, 176, 190, 190, 194, 194, 208, 210, 210, 220, 214, 214, 214, 214, 202, 204, 212, 210,
 210, 210, 212, 212, 214, 214, 218, 218, 212, 212, 210, 210, 210, 210, 208, 210, 212, 212, 212, 200,
 204, 206, 206, 210, 210, 204, 204, 204, 214, 214, 214, 220, 214, 214, 210, 206, 206, 206, 206, 216, 210,
 210, 210, 208, 210, 210, 210, 214, 214, 220, 208, 204, 202, 200, 200, 200, 200]
```

Para T = 500




```
[182, 182, 196, 182, 174, 174, 188, 188, 204, 204, 204, 206, 190, 190, 202, 216, 214, 214, 214, 218, 204,
 204, 192, 200, 204, 198, 200, 200, 200, 200, 200, 200, 206, 198, 198, 198, 190, 190, 190, 184, 186, 186,
 186, 186, 186, 194, 192, 194, 194, 194, 200, 194, 192, 192, 198, 200, 208, 202, 198, 198, 206, 200, 196,
 194, 202, 208, 206, 204, 204, 204, 206, 206, 212, 210, 210, 204, 184, 184, 184, 188, 188, 192, 186, 186,
 188, 180, 202, 204, 212, 204, 222, 220, 220, 210, 212, 212, 204, 202, 196, 200, 196, 196, 196, 194, 194,
 208, 194, 194, 194, 202, 202, 198, 194, 194, 178, 184, 184, 192, 186, 184, 184, 178, 178, 184, 186, 186,
 186, 194, 200, 200, 210, 210, 210, 214, 214, 210, 196, 202, 188, 186, 178, 184, 182, 182, 182, 180, 182,
 184, 178, 184, 190, 194, 190, 188, 192, 192, 188, 184, 186, 192, 198, 198, 210, 210, 210, 220, 206, 216,
 210, 218, 208, 214, 202, 210, 222, 216, 218, 206, 206, 216, 214, 216, 210, 202, 204, 202, 208, 218, 218,
 212, 218, 218, 220, 212, 218, 218, 218, 212, 210, 210, 218, 218, 220, 210, 216, 218, 228, 226, 226, 222,
 222, 226, 218, 210, 212, 208, 220, 220, 212, 218, 218, 204, 204, 204, 206, 204, 204, 204, 200, 210, 210,
 204, 196, 196, 196, 204, 204, 204, 198, 198, 198, 192, 192, 192, 194, 194, 202, 194, 196, 202, 210, 208,
 214, 214, 218, 218, 218, 218, 214, 218, 218, 214, 214, 206, 206, 214, 194, 188, 188, 188, 180, 184, 198,
 194, 202, 200, 216, 216, 218, 204, 212, 206, 206, 206, 206, 202, 204, 198, 204, 204, 202, 202, 202, 206,
 206, 206, 198, 194, 186, 186, 192, 206, 206, 208, 212, 212, 208, 212, 202, 194, 204, 218, 216, 218, 216,
 210, 210, 230, 224, 238, 228, 208, 208, 220, 216, 216, 228, 234, 228, 228, 244, 244, 250, 248, 248, 244,
 238, 238, 248, 250, 240, 234, 236, 238, 230, 230, 222, 212, 210, 210, 222, 222, 218, 220, 214, 208, 208,
 202, 206, 206, 200, 204, 202, 214, 222, 218, 218, 210, 220, 210, 208, 216, 216, 210, 198, 200, 200, 212,
 214, 220, 218, 218, 218, 228, 228, 214, 228, 226, 220, 206, 198, 198, 200, 214, 216, 214, 214, 214, 216,
 214, 206, 194, 202, 214, 222, 222, 222, 226, 218, 222, 214, 214, 222, 220, 228, 236, 230, 222, 222, 210,
 210, 212, 214, 206, 206, 204, 204, 204, 210, 212, 214, 214, 206, 204, 210, 198, 188, 186, 182, 182, 190,
 192, 192, 192, 200, 210, 210, 202, 214, 214, 218, 218, 212, 206, 200, 200, 198, 198, 218, 210, 216, 208,
 214, 208, 210, 212, 204, 204, 202, 202, 202, 196, 190, 192, 206, 216, 208, 216, 216, 204, 204, 212, 208,
 208, 212, 216, 216, 208, 194, 196, 200, 192, 192, 192, 192, 192, 192, 192, 192, 192]
```

EJERCICIO 3

Implementar un algoritmo genético para resolver el problema de optimizar la ubicación de los productos en el almacén, de manera de optimizar el picking de los mismos.

Resolución:

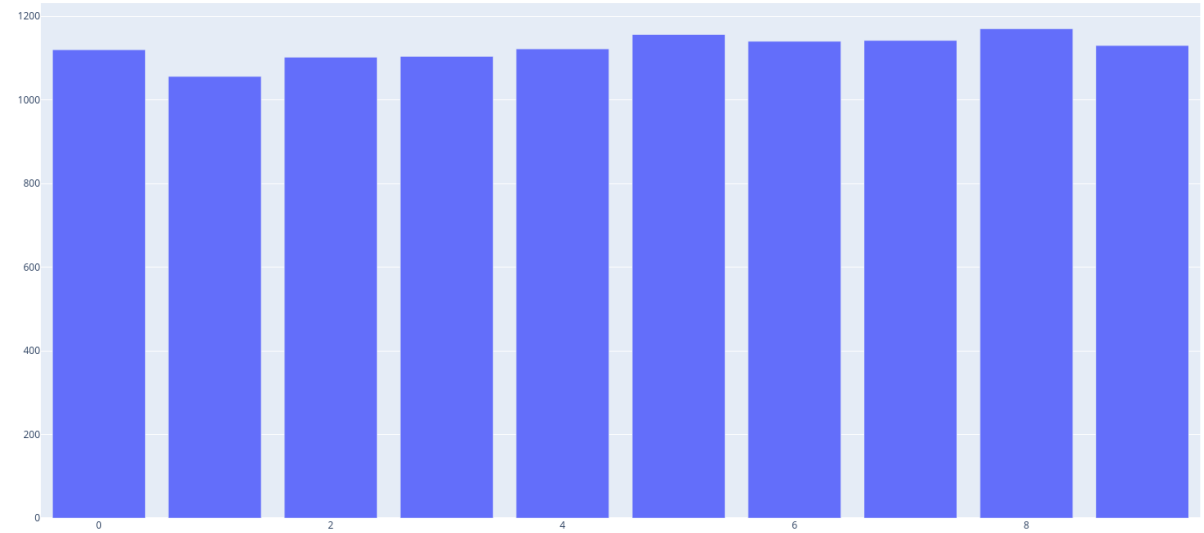
Para el algoritmo genético utilizamos la clase genoma, la cual tiene el fitness de cada individuo y la función para calcular el mismo.
Primero creamos los individuos de la primera generación, luego hacemos el crossover tipo cruce de orden tomando una porción de 3 elementos para hacer el cruce, teniendo en cuenta que no hay repetidos ni faltantes. A continuación, realizamos la mutación con una probabilidad del 30% aleatoria mutando de a 3 elementos, y volviendo a calcular su fitness.
Como queremos obtener resultados que vayan mejorando, mientras que el fitness obtenido sea peor que el peor de la generación anterior, hacemos las mutaciones, en caso contrario no lo hacemos.
Finalmente obtenemos una cierta combinación de productos con la que nos deberíamos guiar para ordenar el almacén.

CAPTURAS OBTENIDAS CON DISTINTOS PARÁMETROS

Siendo el eje horizontal el número de generación y el eje vertical el valor del mejor fitness de cada generación.

Imagen N°1

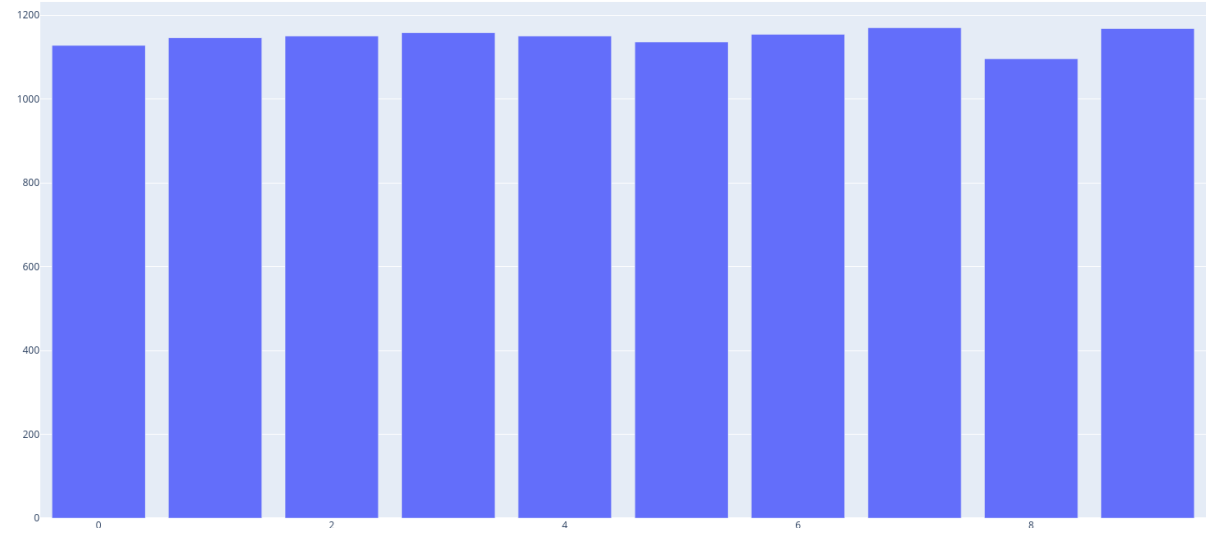
Cantidad de órdenes = 5
Cantidad de individuos por población = 8
T del temple simulado = 50
Cantidad de generaciones calculadas = 10




```
Mejor Fitness logrado =  1056
De la generacion  2
 [7, 91, 26, 77, 17, 46, 64, 53, 35, 85, 30, 51, 90, 59, 5, 80, 25, 32, 75, 23, 36, 57, 71, 81, 67, 21, 100, 82, 37, 29, 13, 33, 56, 4
2, 95, 24, 50, 87, 28, 19, 86, 93, 76, 65, 9, 6, 39, 94, 40, 54, 99, 88, 61, 34, 15, 58, 10, 74, 31, 27, 89, 22, 16, 69, 48, 45, 78, 1
4, 73, 92, 43, 41, 68, 96, 83, 84, 70, 49, 11, 1, 72, 3, 8, 60, 4, 63, 55, 2, 79, 97, 20, 52, 47, 12, 18, 66, 62, 98, 38, 44]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
10
```

Imagen N°2

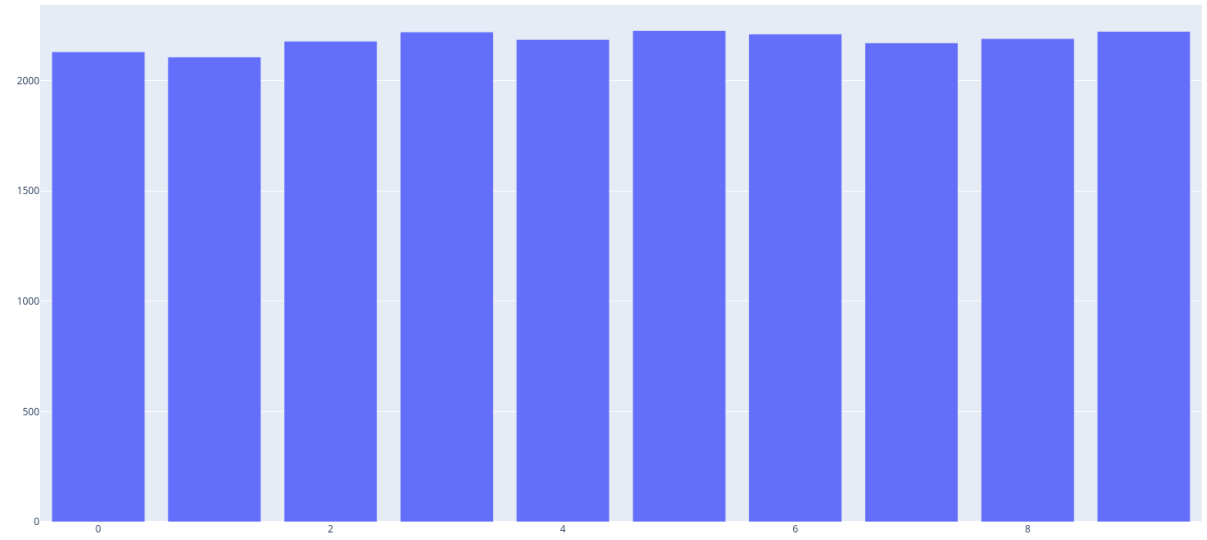
Cantidad de órdenes = 10
Cantidad de individuos por población = 8
T del temple simulado = 100
Cantidad de generaciones calculadas = 10

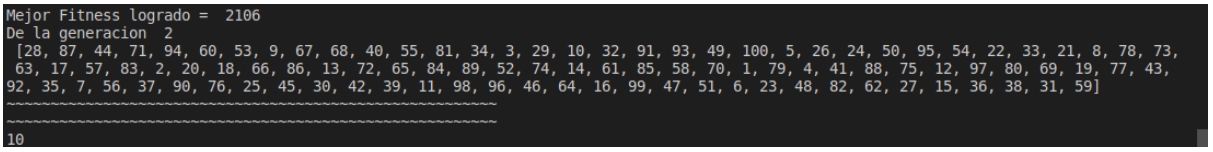



```
Mejor Fitness logrado =  1096
De la generacion  9
 [92, 21, 39, 56, 26, 78, 32, 66, 36, 18, 13, 71, 33, 93, 16, 43, 60, 87, 31, 80, 61, 64, 48, 4, 72, 91, 49, 3, 81, 58, 89, 70, 9, 51,
 55, 35, 2, 63, 82, 100, 94, 41, 47, 53, 73, 98, 25, 27, 83, 23, 44, 5, 75, 57, 85, 76, 29, 42, 7, 79, 67, 19, 14, 17, 20, 84, 90, 86,
 37, 28, 88, 69, 11, 34, 10, 15, 97, 77, 50, 74, 46, 22, 95, 6, 62, 54, 68, 65, 96, 99, 45, 24, 12, 8, 30, 59, 52, 38, 1, 40]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
10
```

Imagen N°3

Cantidad de órdenes = 5
Cantidad de individuos por población = 8
T del temple simulado = 50
Cantidad de generaciones calculadas = 10





Como podemos observar en las imágenes y datos obtenidos, al aumentar la cantidad de órdenes (de 5 a 10) que le pasamos al programa aumentan los valores de los fitness porque se recorren más nodos, sin embargo al parecer la diferencia entre fitness es menor, siendo de aproximadamente del 8% entre el mayor y el menor. Con 5 órdenes el porcentaje difiere en más o menos un 13%. Aumentando la cantidad de iteraciones del temple simulado (de 50 a 100) obtenemos resultados similares del 9% entre el mejor y peor valor.
Al no ser muy grandes las variaciones podríamos decir que no importa el orden en que pongamos los productos porque siempre vamos a obtener números parecidos lo cual no tiene mucho sentido y probablemente se deba a cómo están distribuidas las órdenes y las probabilidades de pedido de cada producto.