Modelos Bioinspirados y Heurísticas de Búsqueda

PRÁCTICA 1

ANÁLISIS DE ALGORITMOS DE BÚSQUEDA BASADOS EN ENTORNOS Y TRAYECTORIAS

Pablo Cordón Hidalgo

ÍNDICE

INTRODUCCIÓN

En este documento se verán y analizarán los resultados de la aplicación de los algoritmos Voraz (Greedy), Búsqueda Aleatoria, Búsqueda local, el primer mejor vecino, Enfriamiento Simulado y Búsqueda Tabú para resolver un problema de optimización de uso de una red de bicicletas en la ciudad de Santander.

Para ello partimos de una red de 16 estaciones de bicicletas con una capacidad máxima total de 220 bicicletas, donde se ha extraído el comportamiento medio a lo largo de un día capturando el movimiento de bicicletas en cada estación cada 5 minutos desde las 8:00 am hasta las 7:am del día siguiente, es decir , el número de bicicletas que cada cinco minutos en un día se han tomado o dejado en cada estación.

Ejecutaremos cada algoritmo 5 veces, cada una de ellas con una semilla diferente, para comparar resultados. Las semillas usadas en esta práctica son: [1111, 2222, 3333, 4444, 5555]

DESCRIPCIÓN DEL DATASET

El dataset para esta práctica se compone de tres matrices de datos:

- **Matriz delta de movimientos:**

El primer registro muestra los Id de las estaciones, del 0 al 15.

El segundo registro es un movimiento con el estado inicial de la red al comienzo del periodo de estudio.

Del segundo en adelante se tienen los movimientos cada 5 minutos para cada estación en la forma: (- 5, +5 ……., 0). Cojo 5 bicis de la primera, dejo 5 en la segunda , ….. no hago nada en la 16. (-10, +8,…….....-1)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	7	13	6	8	13	8	9	6	10	10	18	8	13	15	14
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

- **Cercanas_indices:**

Matriz con las estaciones más cercanas a cada índice.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	4	6	10	8	1	12	15	11	5	2	9	3	14	13
1	12	6	10	5	7	0	15	8	11	4	2	9	3	14	13
2	9	11	15	3	5	12	8	6	1	7	0	10	4	14	13
3	9	2	11	5	15	12	1	6	8	14	7	10	13	0	4
4	0	7	8	6	10	15	1	12	11	5	2	9	3	14	13
5	12	1	11	6	15	2	9	8	10	7	3	0	4	14	13
6	1	12	7	10	0	8	5	15	11	4	2	9	3	14	13
7	0	10	6	1	4	8	12	15	5	11	2	9	3	14	13
8	15	6	0	11	7	12	4	1	10	5	2	9	3	14	13
9	3	2	11	5	15	12	1	6	8	7	10	0	4	14	13
10	7	6	1	0	12	4	8	5	15	11	2	9	3	14	13
11	15	2	5	12	8	9	6	1	3	7	0	10	4	14	13
12	5	1	6	15	11	7	10	8	0	2	9	4	3	14	13
13	14	3	9	2	5	11	12	15	1	6	8	10	7	0	4
14	13	3	9	2	5	11	12	15	1	6	8	10	7	0	4
15	11	8	12	6	2	5	1	7	0	9	10	4	3	14	13

Por ejemplo, la estación más cercana de la 1 es la 12, la siguiente más cercana es la 6, y así sucesivamente.

- **Cercanas_kms:**

Matriz con la distancia entre cada una de las estaciones de la matriz previa.

0	1	2	3	4	5
0.0	0.4612275771434886	0.716556975483184	0.9313621873428618	0.9477275970723462	1.0635186361556144
0.0	0.5530185954345544	0.5939211595070305	0.819978299475489	0.9210339622316458	1.0045346448337067
0.0	0.661118187182893	0.7719908030770463	1.1490439477242063	1.1609208865928937	1.3478521492750146
0.0	0.6315818426361309	1.1609208865928937	1.883933147494817	2.007667350665301	2.2781766815791635
0.0	0.716556975483184	1.1686787287120894	1.3444067143388139	1.6148276615883383	1.6188773495262376
0.0	0.4897223724780759	0.9210339622316458	1.0037473059397939	1.1220529184477213	1.1643931713197897
0.0	0.5939211595070305	0.6328831596461355	0.6335764884966517	0.7987412528665032	0.9313621873428618
0.0	0.4612275771434886	0.5073157312018843	0.6335764884966517	1.0045346448337067	1.1686787287120894

Por ejemplo, la estación 1 estará a 0.553 kms de la 12, a 0.593 kms de la 6, y así sucesivamente.

ALGORITMO GREEDY (VORAZ)

El algoritmo voraz (Greedy) es un algoritmo constructivo que sigue la heurística de asignar las capacidades proporcionalmente al vector inicial de estado de la red de bicicletas prestando atención a que la cantidad total de plazas sea la indicada, en nuestro caso 220.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Greedy	1	1	-	574.551	574.551	-	0.035

El Algoritmo Greedy es el más eficiente de todos porque no realiza un procedimiento iterativo de aproximación hacia la solución. Utiliza un criterio que le indica en cada momento cuál es el mejor candidato para añadir a la solución y, cuando consigue construir una solución, finaliza. En términos de eficiencia, el algoritmo Greedy es el mejor.

Como el Algoritmo Greedy siempre devuelve la misma solución para el mismo problema, por tanto, la desviación típica que presenta en el coste es 0.

ALGORITMO ALEATORIO

El algoritmo aleatorio genera una solución completamente aleatoria en cada iteración, devolviendo al final la mejor encontrada.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Aleatorio	100	100	-	438.129	462.034	11.935	4.478

En este algoritmo el número de evaluaciones es constante. Debido a esto no se analizarán las columnas referentes a estos datos.

Se observa que este algoritmo, aun siendo rápido, genera soluciones muy malas. Esto, es debido a:

- El algoritmo selecciona soluciones al azar sin criterio ninguno y, la probabilidad de alcanzar la solución óptima es $\frac{1}{tam.\,solución!}$. En nuestro caso, la probabilidad de alcanzar la solución óptima usando este algoritmo es de $\frac{1}{16!} = 4.78 * 10^{-14}$

- El incremento de posibles soluciones crece de manera factorial mientras el incremento de evaluaciones se amplía linealmente.

ALGORITMO DE BÚSQUEDA LOCAL, EL PRIMER MEJOR VECINO

Este algoritmo analiza N vecinos de la solución inicial en cada iteración. Una vez que se encuentre un mejor vecino, se actualiza la solución actual y se reinician los vecinos a analizar. Este algoritmo pertenece al grupo de algoritmos de búsqueda en profundidad.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
BL Primer	258	264	3.949	384.847	393.459	6.662	11.139

Los Algoritmos de Búsqueda Local están planteados de forma que siguen generando vecinos hasta que no haya ningún vecino que mejore la solución actual, es decir, no paran hasta llegar a un óptimo local.

ENFRIAMIENTO SIMULADO

El Enfriamiento Simulado (Simulated Annealing) es un método de Búsqueda Global, proveniente de modificar la Búsqueda Local, permitiendo algunos movimientos hacia soluciones peores para escapar de óptimos locales.

Este algoritmo usa un criterio probabilístico de aceptación de soluciones basado en Termodinámica

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Enf. Simulado	821	821	0	392.781	411.345	13.945	38.125

Los resultados del enfriamiento simulado son los de peor calidad, pues la desviación típica es muy alta, y obviando los algoritmos greedy y aleatorio, su coste es el más alto. Además, en este caso un mayor número de llamadas a la función de evaluación no ha mejorado en absoluto el resultado con respecto a la búsqueda local

BÚSQUEDA TABÚ

Este algoritmo busca una mejor solución descartando movimientos repetidos a corto plazo a menos que mejore la solución y, cada cierto número de iteraciones reinicia la búsqueda a partir de la mejor solución encontrada intensificando el coste de esta, una solución aleatoria o en función de la memoria a largo plazo para explorar zonas no visitadas.

Hemos usado versiones diferentes de la búsqueda tabú, una con sólo movimientos en la lista tabú, y otra con los elementos que se cambian y sus respectivos movimientos.

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Tabú_movs	7842	7844	1.095	351.875	364.056	12.026	368.201
Tabú_elem_mov	7842	7844	1.095	353.829	367.958	10.471	339.983

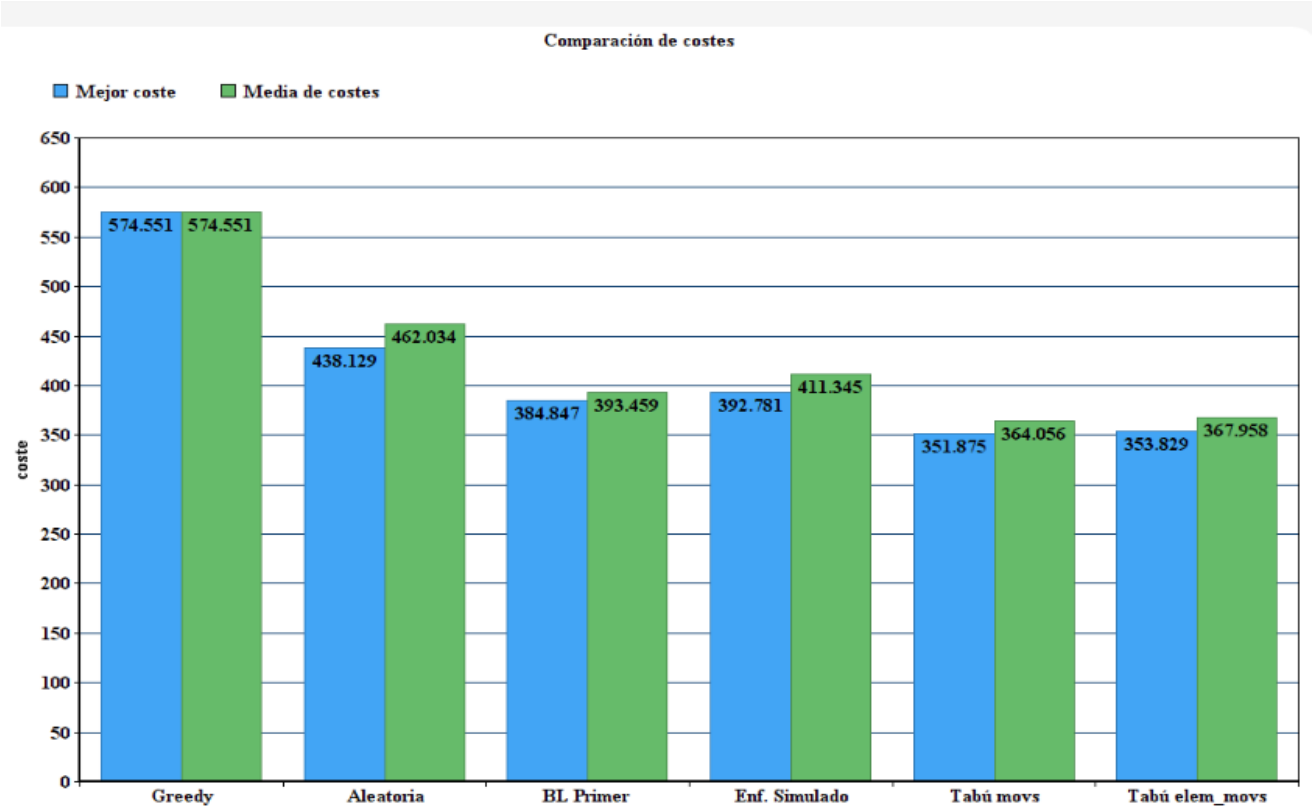
Se observa que los resultados de la búsqueda tabú son los mejores en cuanto a coste con diferencia, pero esto también se ve reflejado en el número de llamadas a la función de evaluación, que es considerablemente más alto que en el resto de algoritmos.

COMPARACIÓN DE ALGORITMOS

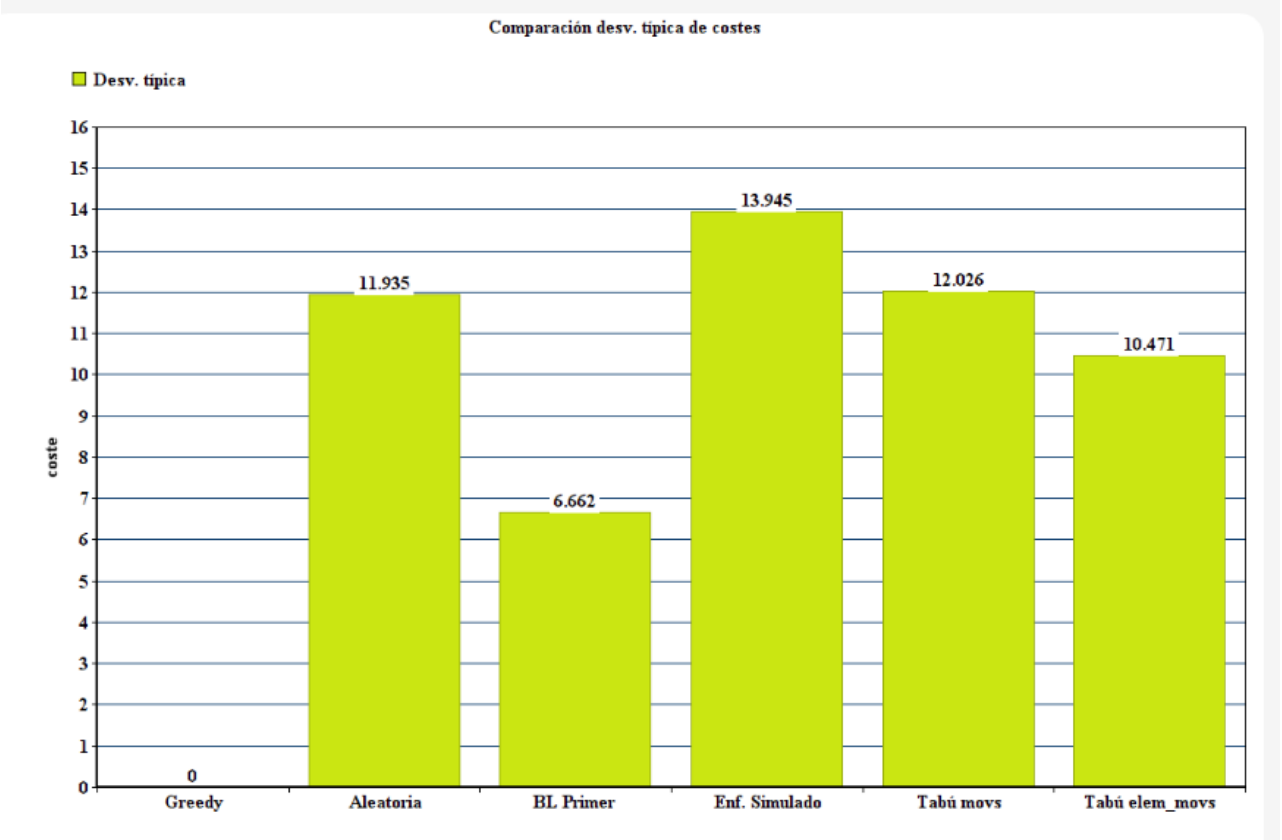
- **Tabla comparativa**

Algoritmo	Ev. Mejor	Ev. Media	Ev. Desv	Kms. mejor	Kms. Media	Kms. Desv	Tiempo
Greedy	1	1	-	574.551	574.551	-	0.035
Aleatoria	100	100	-	438.129	462.034	11.935	4.478
BL Primer	258	264	3.949	384.847	393.459	**6.662**	11.139
Enf. Simulado	821	821	0	392.781	411.345	13.945	38.125
Tabú_movs	7842	7844	1.095	**351.875**	**364.056**	12.026	368.201
Tabú_elem_mov	7842	7844	1.095	353.829	367.958	10.471	339.983

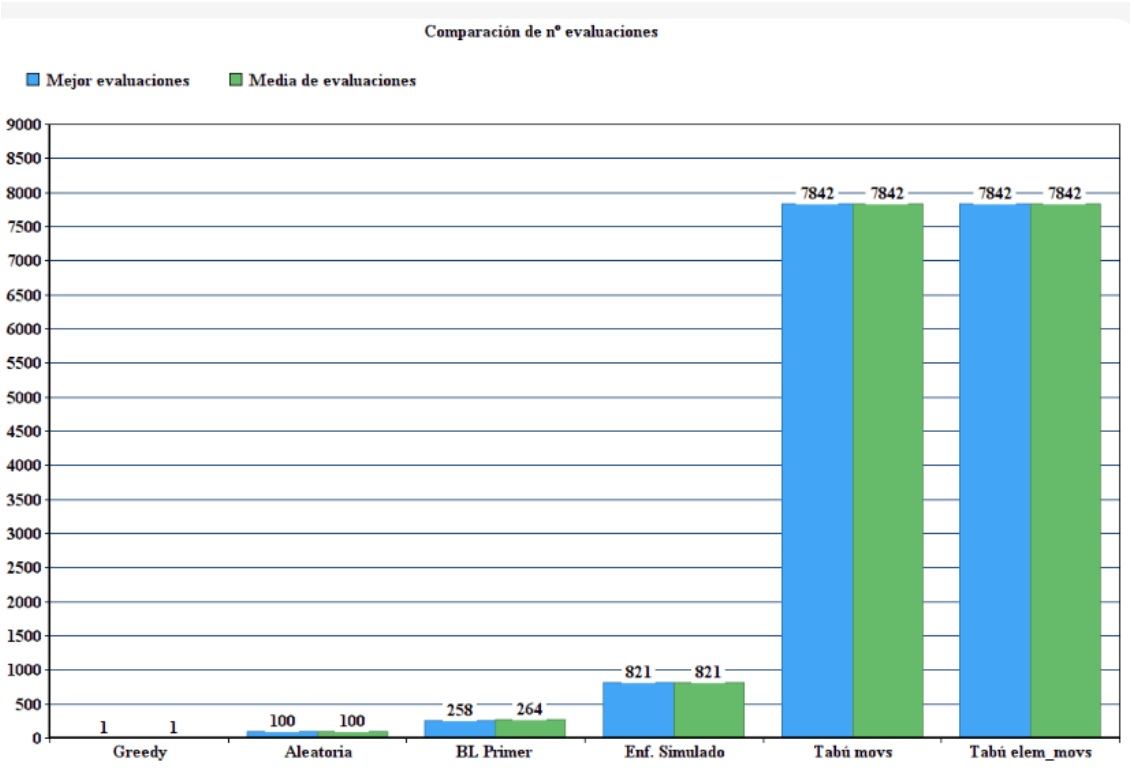
- **Gráfica de comparación de costes**



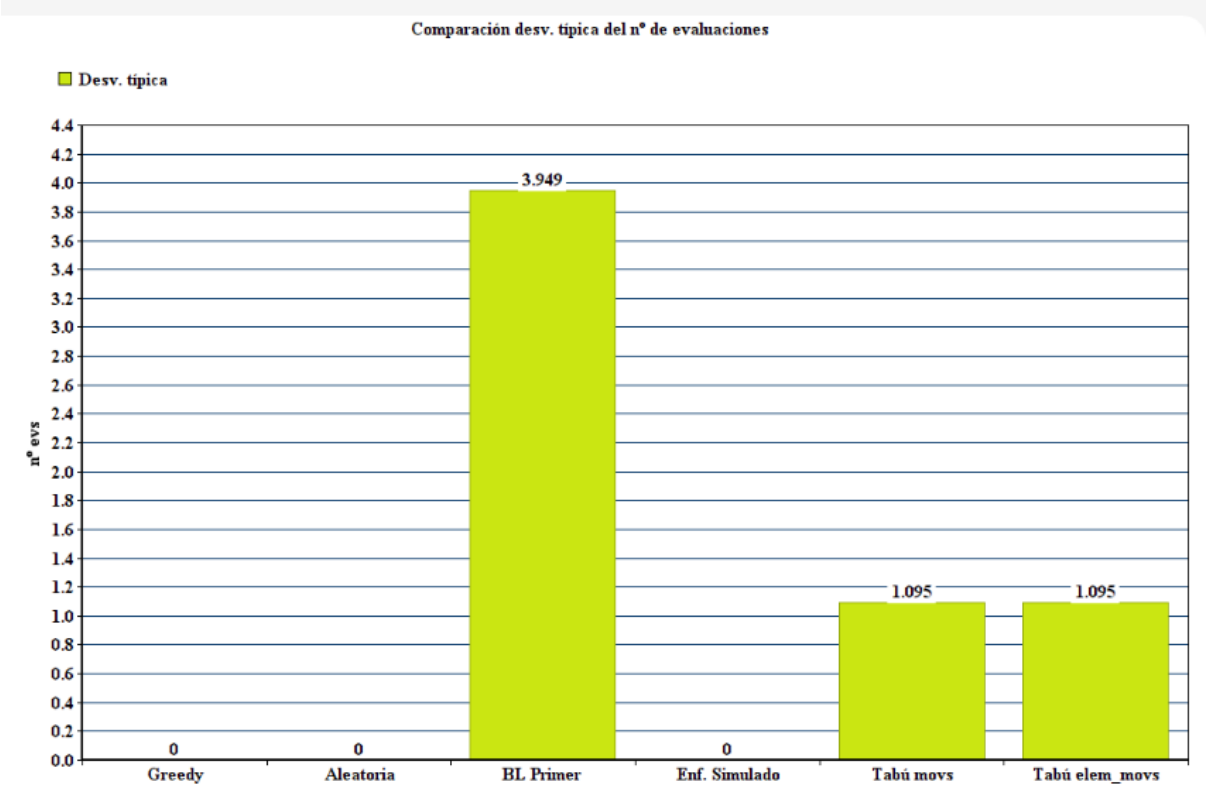
- **Gráfica de comparación de desviación típica de los costes**



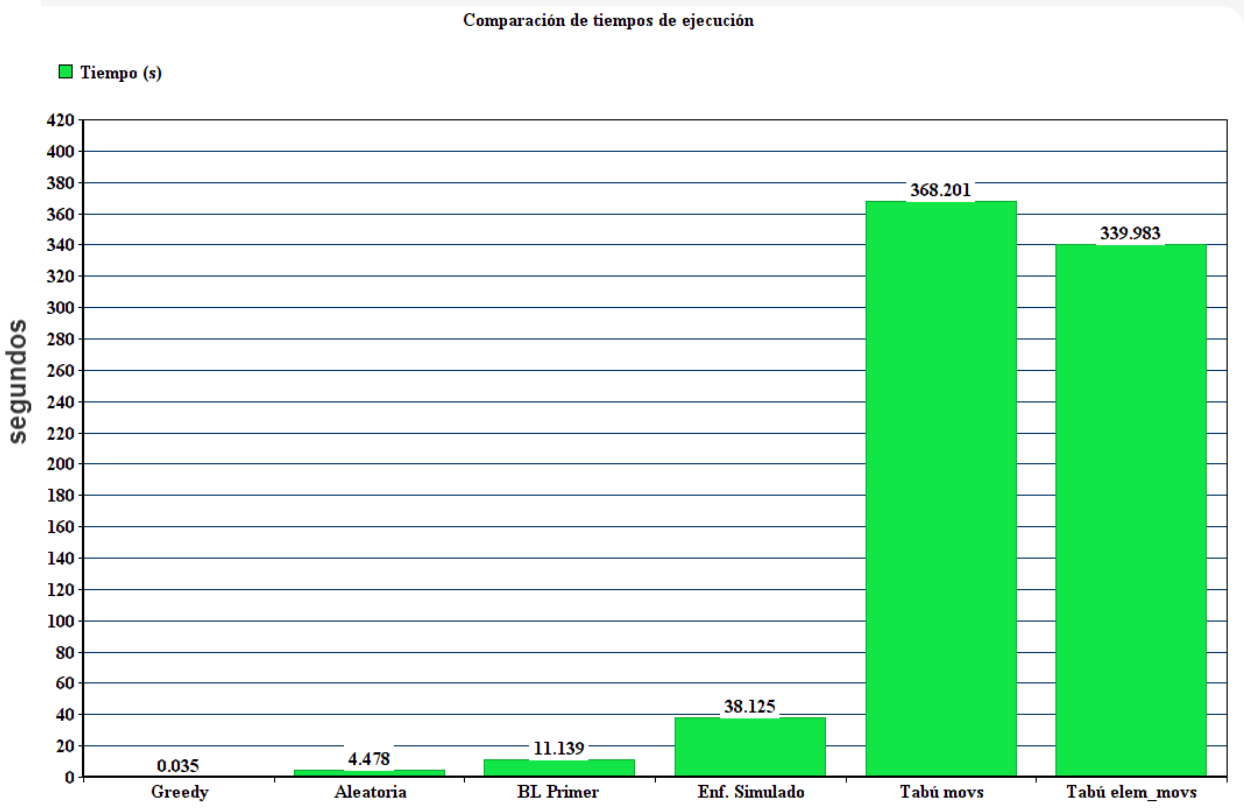
- **Gráfica de comparación de número de llamadas a la función de evaluación**



- **Gráfica de comparación de desviación típica en el número de llamadas a la función de evaluación**



- **Gráfica de comparación de tiempos**



OBSERVACIONES

- Debido a que el algoritmo “Greedy” siempre genera la misma solución su desviación típica respecto al coste es 0.

- El tiempo de ejecución del algoritmo y el número de llamadas a la función de evaluación son directamente proporcionales, pues lo más costoso de todos estos algoritmos son las llamadas a la función de evaluación.

- La desviación típica respecto al número de llamadas a la función de evaluación es 0 en los algoritmos Greedy, Aleatorio y Enfriamiento Simulado, ya que estos siempre realizarán el mismo número de llamadas a la función. En la Búsqueda Local, varía el número de llamadas a la función de coste ya que dependerán de si encuentra un mejor vecino para centrarse en encontrar el mínimo local. La búsqueda tabú varía por criterio aleatorio de las reinicializaciones, en las que en 2 de los 3 casos de reinicialización, se vuelve a calcular el coste de una nueva solución.

- Por otro lado, se observa que el algoritmo más consistente con la generación solución es el algoritmo “Búsqueda local: el primer mejor vecino”. Esto es debido a que este algoritmo, partiendo de una solución, no se centra tanto en explorar el entorno sino en buscar el primer mejor vecino para profundizar en la búsqueda de la solución.

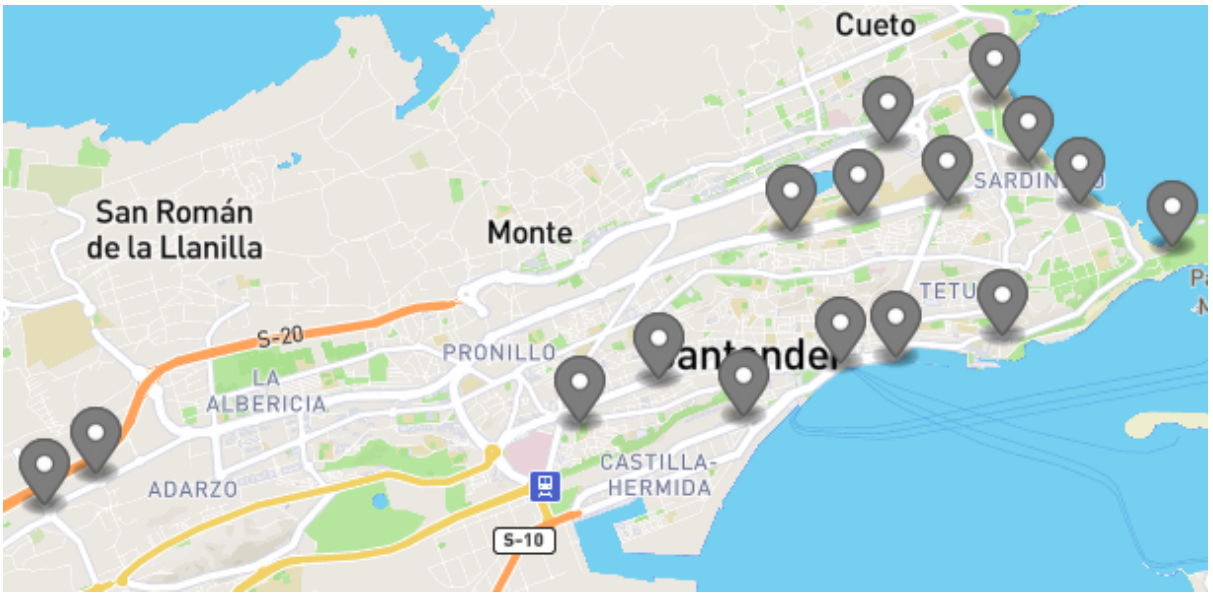
CONCLUSIONES

Si bien las mejores soluciones las aporta la búsqueda tabú, esta búsqueda es también la que tarda mayor tiempo, por lo que en el caso de tener un equipo con menor capacidad computacional , o un dataset mayor, el tiempo de evaluación crecería mucho.

Por lo general, la desviación típica del coste no es muy grande en este dataset, pero teniendo en cuenta la robustez de los algoritmos, el más robusto es la Búsqueda Local Primero el mejor vecino. Sin embargo, este algoritmo da también una de las peores soluciones en cuanto a coste.

REPRESENTACIÓN DE LAS ESTACIONES EN EL MAPA

Para representar la solución final en un mapa hemos usado el repositorio interactivo de github “geojson” (http://geojson.io/#map=13/43.4836/-3.8353) , que nos permite indicar puntos en un mapa con un json dadas unas coordenadas y definir para cada uno de estos puntos ciertos elementos, en nuestro caso el id, la posición en la que se encuentra en el dataframe, el número de bicicletas en la solución inicial, y el número de bicicletas en la mejor solución de todas las estudiadas




```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      -3.817978245,
      43.46072554
    ]
  },
  "properties": {
    "id": 13,
    "pos_dataframe": 9,
    "sol_ini": 10,
    "sol_final": 15
  }
```