PRÁCTICA 1

Algoritmos basados en Entornos y Trayectorias

José María Prieto Crespo

Índice

1. **Greedy** 2

Ejecución Greedy con floor (a la baja): 2

Ejecución Greedy con round (a la alza): 2

2. **Búsqueda Aleatoria** 2

Semilla 1: 2

Semilla 2: 2

Semilla 3: 3

Semilla 4: 3

Semilla 5: 3

100 iteraciones: 3

200 iteraciones: 3

500 iteraciones: 3

1000 iteraciones: 3

2000 iteraciones: 4

3. **Búsqueda local** 4

4. **Enfriamiento simulado** 5

Mejor S1: 6

Mejor S2: 6

Mejor S3: 6

Mejor S4: 6

Mejor S5: 6

Media kms : 6

5. **Búsqueda tabú** 6

Mejor S1: 7

Mejor S2: 7

Mejor S3: 7

Mejor S4: 7

Mejor S5: 7

Media kms : 7

6. **Comparación** 7

1. Greedy

Ejecución de la función “evalua” para una solución Greedy. Ésta ha sido calculada a partir de la primera fila de movimientos del fichero csv “deltas_5m”, donde manteniendo la proporción inicial, hacemos que el número máximo de slots sea de 220.

Estos valores se pueden realizar redondeando a la alza o a la baja, a continuación muestro ambas ejecuciones:

Ejecución Greedy con floor (a la baja):

- Capacidad: [6, 9, 17, 8, 10, 17, 10, 12, 8, 13, 13, 24, 10, 17, 20, 18]
- Slots máximos: 212
- Km: 654.711536775757

Ejecución Greedy con round (a la alza):

- Capacidad: [7, 9, 18, 8, 11, 18, 11, 12, 8, 13, 13, 24, 11, 18, 20, 19]
- Slots máximos: 220
- Km: 574.5513854177129

2. Búsqueda Aleatoria

Consiste en generar aleatoriamente una solución en cada iteración debiéndose ejecutar 100 iteraciones con cada semilla, devolviendo la mejor de las iteraciones.

Uso 5 semillas: 10, 20, 30, 40, 50.

Para el caso de las 100 iteraciones los resultados son los siguientes:

Semilla 1:

Capacidad inicial: [20, 2, 17, 20, 14, 7, 6, 20, 6, 4, 18, 25, 2, 14, 21, 16]

Km inicial: 685.3495431372869

Capacidad final: [19. 21. 15. 23. 17. 2. 8. 20. 11. 5. 4. 19. 15. 12. 8. 15.]

Km final: 556.2034498501058

Semilla 2:

Capacidad inicial: [14, 20, 20, 18, 2, 16, 9, 12, 15, 5, 7, 16, 18, 19, 18, 2]

Km inicial: 828.0669475685492

Capacidad final: [18. 5. 17. 10. 17. 18. 18. 14. 15. 15. 10. 12. 12. 2. 16. 14.]

Km final: 539.1707251602488

Semilla 3:

Capacidad inicial: [16, 10, 16, 5, 23, 9, 24, 7, 15, 11, 5, 14, 13, 19, 22, 4]

Km inicial: 736.2133562154206

Capacidad final: [10. 11. 16. 18. 18. 4. 16. 20. 18. 5. 8. 24. 9. 14. 4. 17.]

Km final: 540.7972458095894

Semilla 4:

Capacidad inicial: [10, 3, 18, 8, 11, 8, 13, 15, 18, 16, 22, 14, 19, 16, 2, 21]

Km inicial: 588.1938187472992

Capacidad final: [18. 9. 16. 11. 13. 14. 9. 16. 12. 10. 17. 19. 18. 6. 17. 9.]

Km final: 513.8436548893304

Semilla 5:

Capacidad inicial: [13, 7, 7, 11, 10, 24, 11, 19, 19, 9, 9, 9, 5, 24, 22, 14]

Km inicial: 792.041682806271

Capacidad final: [19. 21. 15. 23. 17. 2. 8. 20. 11. 5. 4. 19. 15. 12. 8. 15.]

Km final: 537.1335685350334

Además, he hecho una serie de pruebas para ver como funciona el algoritmo para distinto número de iteraciones, siendo estos los resultados tanto en tiempo como en kilómetros recorridos:

100 iteraciones:

Media kms : 537.4297288488615

Segundos ejecución: 4.345508813858032

200 iteraciones:

Media kms : 522.7364590041163

Segundos ejecución: 8.78053593635559

500 iteraciones:

Media kms : 505.81369053187757

Segundos ejecución: 22.011014699935913

1000 iteraciones:

Media kms : 503.0588170243357

Segundos ejecución: 43.99536085128784

2000 iteraciones:

Media kms : 494.83455593986844

Segundos ejecución: 87.62870121002197

La conclusión final es que a partir de las 500 iteraciones no renta seguir, ya que se estabiliza el número de kilómetros sobre los 500km, siendo el tiempo mucho mayor cada vez.

3. Búsqueda local

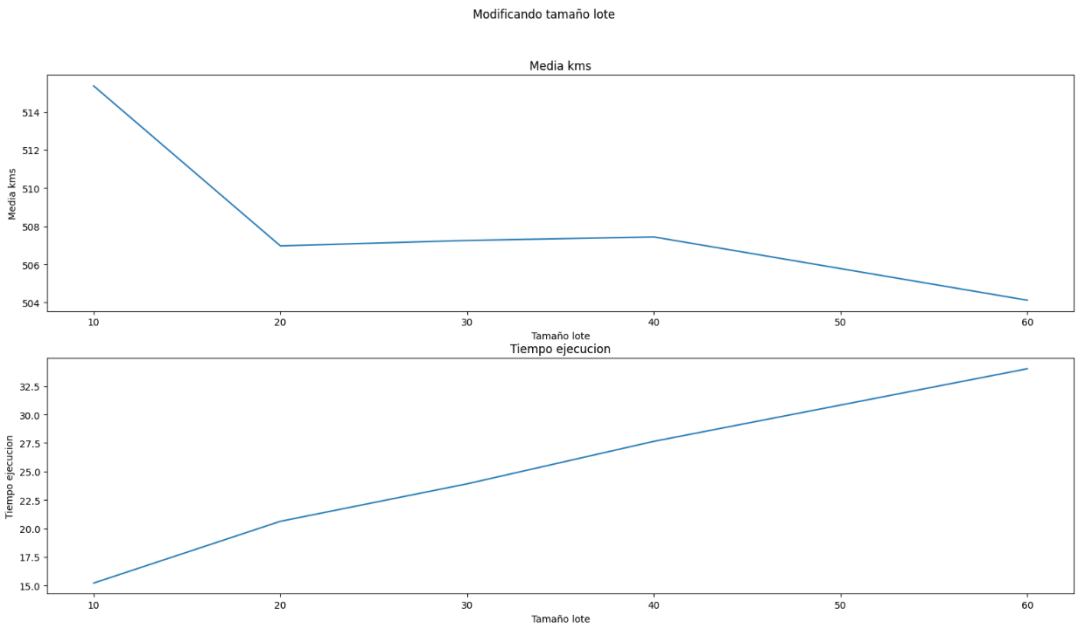
Algoritmo de búsqueda local, implementado siguiendo el esquema de el primer mejor vecino. Para ello, dividimos el número total de vecinos en bloques y buscamos el primer mejor vecino de cada bloque. Además, tenemos otra condición de parada donde se limita el número de llamadas a la función de coste a 3000.

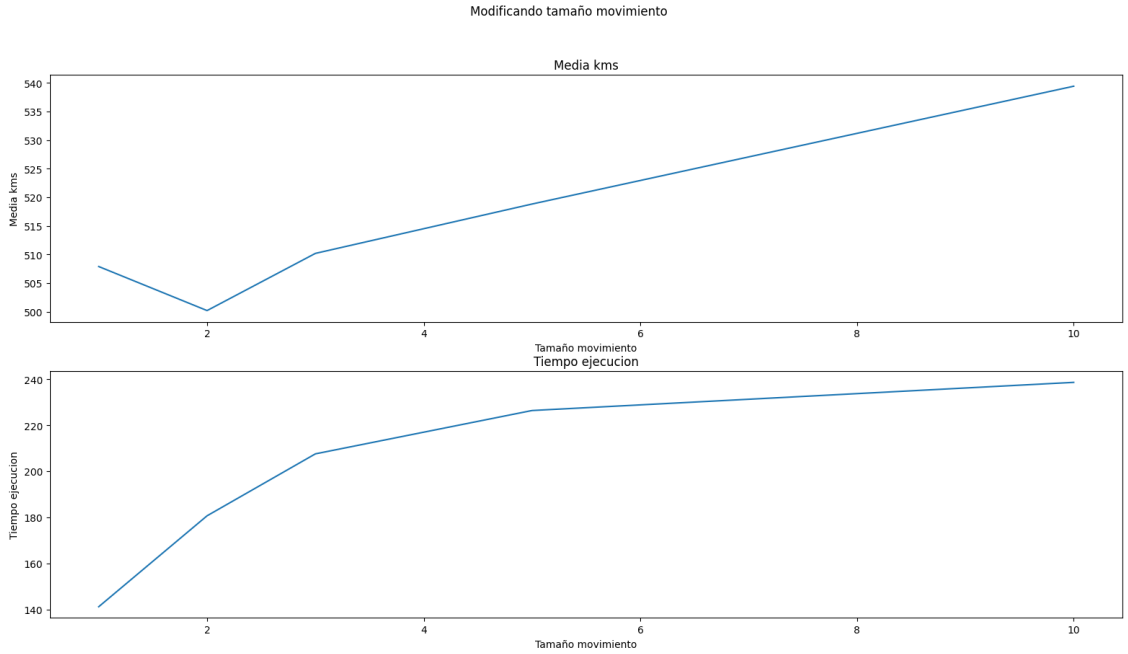
Para este caso he hecho una serie de pruebas para diferente tamaño de lotes y diferente tamaño de movimiento de vecinos:

Tamaño de lotes: 10, 20, 30, 40, 60

Numero de movimientos: 1, 2, 3, 5, 10

Las gráficas obtenidas son:





Media km/lote	515.35926666	**506.97078415**	507.25568102	507.44038977	504.12261855
Tiempo/lote	15.20506501	20.62619352	23.9153223	27.64383864	34.01472712
Media km/movimiento	507.89158951	**500.19113783**	510.18976147	518.820831	539.41890138
Tiempo/movimiento	141.16999364	180.64903402	207.54597783	226.37016916	238.59131694

Como podemos observar, donde mejores valores obtenemos (teniendo en cuenta el tiempo) es para 20 lotes y 2 movimientos para el cálculo de vecinos.

4. Enfriamiento simulado

Se ha de implementar el algoritmo de Enfriamiento Simulado, donde la temperatura inicial se calculará a partir de una fórmula cuyos parámetros principales son el coste Greedy, “phi” y “mu”. Estos dos últimos se encuentran en un rango entre 0.1 y 0.3, por lo que los calcularemos de forma que el número de soluciones iniciales no aceptadas sea de un 20%.

Tras mi propia experimentación realizando 100 iteraciones, los mejores valores que he obtenido han sido mu = 0.2 y phi = 0.1.

La ejecución para las 5 semillas ha dejado los siguientes resultados:

Mejor S1:

Capacidad final: [14, 11, 17, 14, 20, 13, 12, 8, 12, 16, 12, 22, 2, 5, 21, 13]

Km recorridos: 470.3252044633543 km

Mejor S2:

Capacidad final: [14, 11, 20, 21, 20, 13, 12, 12, 12, 14, 10, 19, 3, 10, 9, 11]

Km recorridos: 484.4928624905891 km

Mejor S3:

Capacidad final: [13, 7, 19, 17, 20, 15, 12, 10, 12, 14, 11, 23, 4, 16, 7, 13]

Km recorridos: 454.62072289612684 km

Mejor S4:

Capacidad final: [13, 8, 21, 29, 20, 11, 13, 12, 12, 13, 10, 23, 4, 10, 3, 12]

Km recorridos: 435.8808179001181 km

Mejor S5:

Capacidad final :[13, 10, 19, 14, 19, 15, 11, 10, 13, 15, 12, 24, 2, 12, 13, 11]

Km recorridos: 458.42693033945767 km

Media kms :

460.7493076179292 km

5. Búsqueda tabú

Para esta se implementa una versión de la BT donde se usa una lista de movimientos tabú y tres estrategias de reinicialización, así como una lista de frecuencias.

- **Lista de movimientos tabú:** almacena el nuevo valor que tendrían las dos posiciones obtenidas al generar el mejor de los vecinos.
- **Lista de frecuencias:** guarda la frecuencia con la que se ha producido un valor en cada estación. Para reducir el tamaño de la misma, he almacenado los valores en intervalos de 10.
- **Estrategias de reinicialización:** hay que ejecutarlas 4 veces en total durante la ejecución. Para cada una de las reinicializaciones, hay un 25% de que se reinicialice construyendo una solución aleatoria, un 50% de generar una nueva solución greedy y un 25% restante de reinicializar a partir de la mejor solución obtenida.

Los resultados de la ejecución han sido los siguientes:

Mejor S1:

Capacidad final: [14, 11, 17, 17, 20, 13, 12, 8, 12, 16, 12, 22, 2, 5, 18, 13]

Km recorridos: 470.3252044633543

Mejor S2:

Capacidad final: [14, 11, 20, 18, 20, 13, 12, 12, 12, 14, 10, 19, 3, 7, 15, 11]

Km recorridos: 484.4928624905891

Mejor S3:

Capacidad final: [16, 10, 22, 17, 20, 15, 12, 10, 12, 14, 11, 17, 4, 19, 4, 10]

Km recorridos: 455.9181048337314

Mejor S4:

Capacidad final: [13, 8, 21, 26, 20, 11, 13, 12, 12, 13, 10, 23, 4, 13, 3, 12]

Km recorridos: 435.8808179001181

Mejor S5:

Capacidad final: [13, 10, 22, 17, 19, 15, 14, 10, 13, 15, 12, 18, 2, 9, 13, 11]

Km recorridos: 457.48561922980565

Media kms :

460.82052178351967 km

6. Comparación

Algoritmo	Ev. Medias	Ev. Mejor	Desv. Ev	Mejor Kms	Media Kms	Desv. Kms
Greedy	1	1	0	574.55	574.55	0
Aleatoria	100	100	0	513.84	537.43	15.19
BL	813.8	497	197.63	483.29	506.52	18.83
Enf. Simulado	800	800	0	435.88	460.75	15.65
Tabú	8000	8000	0	435.88	460.83	16.99

- **Búsqueda aleatoria:** a pesar de no tener la mejor media de Kms, 537kms, esto cambia según la semilla que uses y puede darse el caso de que la aleatoria llegue a una media de kms muy baja donde cuantas más iteraciones realice, más probabilidades hay de mejora.
- **Búsqueda local:** único caso en el que usamos el "Primer mejor vecino", principalmente por temas de eficiencia, aunque puede llegarse a dar casos en los que se produzcan más de 800 iteraciones, ya que alguna vez se recorren prácticamente todos los vecinos de una solución. En cuanto a kms no garantiza la mejor solución debido al uso del

"Primer mejor vecino", que no encuentra la mejor solución del entorno, sin embargo si que obtenemos unos valores bastante mejores que con la búsqueda aleatoria.

- **Enfriamiento simulado:** para este caso tenemos un número fijo de evaluaciones, 800, debido a que lo ejecutamos para 20 vecinos durante 40 iteraciones, por eso que la desviación típica de evaluaciones sea 0. En cuanto a los kms, como en este caso usamos el mejor vecino y no solo nos quedamos con el primero mejor que nos encontremos pues mejoramos mucho la solución respecto a la anterior, obteniendo una media casi de 50kms menos.
- **Búsqueda tabú:** caso muy similar anterior, solo que en este caso el limite de iteraciones totales lo he establecido en 200, por lo que 200 iteraciones por 40 vecinos cada una daría ese resultado. Tal y como pasa en el enfriamiento simulado, cuantas más iteraciones tengamos aparentemente mayor probabilidad hay de encontrar la mejor solución, quedando así reflejado al estabilizarse ambas sobre los 460kms de media.

Una vez analizados los distintos resultados, el que mejores resultados obtiene es el enfriamiento simulado ya que, a pesar de tener unos valores similares a los de la búsqueda tabú, lo hace en mucho menos tiempo e iteraciones, por lo que es más eficiente.