Práctica de Búsqueda Local : Bicing

Inteligencia Artificial

Autores : Momin Miah Begum & Muhammad Yasin Khokhar
Curso : 2023/2024 1Q

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índice

1. Problema

Descripción

En esta práctica se ha planteado un problema relacionado con los servicios de préstamo de bicicletas. En concreto, debemos simular la empresa Bicing, que al mismo tiempo presta los servicios de Bicicletas a sus clientes. La empresa que debemos de simular ha de dar una optimización sobre la distribución de las bicicletas en las diferentes estaciones que tiene la empresa Bicing. Tenemos información de las estaciones y necesitamos cubrir la demanda a la hora siguiente en todas las estaciones posibles.

Elementos y información que disponemos de parte de la empresa Bicing
1. **Previsión de número de bicicletas** que no van a ser utilizadas en la hora actual.
2. **Previsión de número de bicicletas** que una estación tendrá en la siguiente hora.
3. Una posible demanda que indica el número de bicicletas que esa estación debería de tener.
4. Existen dos tipos de escenarios, demanda equilibrada y hora punta. La demanda equilibrada significa que la demanda que tienen las estaciones son muy parecidas. Hora punta es cuando la demanda de una estación comparada con otra puede diferenciarse bastante.
5. No hay un límite para el almacenaje de bicicletas en una estación, es decir una estación puede tener infinitas bicicletas aparcadas.
6. El número de bicicletas es constante, durante la simulación no se pueden añadir o quitar nuevas bicicletas del escenario.

Restricciones propias de la empresa encargada de optimización:
1. Una furgoneta no puede cargar más de 30 bicicletas.
2. Una furgoneta solo tiene un origen donde puede cargar las bicicletas y puede llegar a tener como máximo dos destinos. El número de bicicletas en la furgoneta, después de acabar el trayecto, ha de ser cero.
3. El coste del transporte de Bicicletas es ((nb+9)div 10) siendo nb el número de bicicletas que lleva la furgoneta para un determinado destino.

1.1 La descripción/justificación de la implementación del estado

El primer paso que se ha dado es discutir cómo representar el estado de este problema. Un primer planteamiento nos basamos en utilizar estaciones como estados. Escogiendo esa idea nos equivocamos, ya que los estados se tienen que ir expandiendo a medida que el algoritmo de búsqueda avanza. Por lo tanto, después de unos planteamientos más sofisticados, llegamos al estado siguiente:

double[][] furgos;
Una matriz que almacenará todas las furgonetas que tiene la empresa. Esta matriz Fx5. El primer elemento de la matriz es el origen de la furgoneta, -1 si no está asignada para una ruta. El segundo elemento es el destino 1 de la furgoneta, el tercer elemento es el destino 2 dos de la furgoneta. Cabe destacar que una furgoneta no puede ir al destino 2 si no tiene destino 1 y además, el orden del trayecto es Origen -->; Destino 1 -->; Destino 2. Por otro lado, también se ha de recordar que el destino 2 es totalmente opcional en una ruta.

double[]bicisQueFaltan;
Vector de tamaño de número de estaciones que almacenará las bicicletas que faltan en cada estación i. Se calcula de la siguiente manera, bicisQueFaltan[i] = Math.max(Demanda - NumBicicletasNext,0). Una estación necesita aquellas bicicletas que le faltan para cubrir o igualar la demanda.

double[]biciQueSobran;
Vector de tamaño de número de estaciones que almacenará las bicicletas que sobran en cada estación i. Se calcula de la siguiente manera, biciQueSobran[i] = Math.min(Math.min((NumBicicletasNext-Demanda),30),NumBicicletasNoUsadas). A una estación le sobran aquellas bicicletas que tiene actualmente tiene, pero se ha de tener en cuenta que al escoger estas bicicletas no se está alejando de la posible demanda que tendrá esta estación de origen, por eso primero se debe de calcular las bicicletas si la estación necesita más bicicleta en la hora siguiente o no, a partir de este calcula mirar el número de bicicletas que una furgoneta puede cargar. Obviamente, el numero de bicicletas máximo que la furgoneta cargará son las no usadas.Cabe destacar que en el cálculo de biciQueSobran solo se añaden si las bicicletas que tendrá la estación, menos las que tiene ahora, menos demanda, es mayor que 0.

double[]estacionOrigenFurgo;
Vector de tamaño de número de furgonetas, indica el origen que tiene cada furgoneta, útil para poder mantener un registro de qué origen está en uso y qué origen se puede asignar, ya que dos furgonetas **no** pueden tener orígenes iguales.

int tipoHeuristica; valor para decidir qué tipoHeuristica (con costes o sin costes).

Estaciones estaciones;
Esta representación del estado nos permite cumplir con los objetivos que el algoritmo de búsqueda necesita, cuyo volumen dependerá de la mejora que haga el algoritmo aplicando los operadores necesarios. Gracias a esta representación podemos satisfacer las restricciones que hay en la simulación del escenario.

1.2 La descripción/justificación de los operadores que habéis elegido

Dado que nuestra representación de estados consiste en furgonetas con rutas, los operadores son fundamentales para variar estos estados. Aquí detallamos los operadores que hemos planteado.

1. Asignar/Cambiar Destino 1: Este operador asigna un destino a una furgoneta si no tiene uno previamente asignado. En caso ya tener uno, se intercambia por el nuevo destino, dejando libre el destino anterior al que estaba asignado. Este operador es crucial ya que permite que las furgonetas tengan rutas asignadas, lo cual es necesario para el algoritmo. La condición de aplicabilidad es que la furgoneta tenga un origen distinto al destino a asignar.
Ramificación del operador: (FXE)Por cada furgoneta, se decide si asignar o cambiar el destino 1 basado en la condición de aplicabilidad.

2. Asignar/Cambiar Destino 2: Similar al operador anterior, pero se aplica al destino 2. La condición de aplicabilidad es que la furgoneta tenga un origen distinto al destino 2 y que el destino 1 sea diferente del destino 2.
Ramificación del operador:((FXE) Por cada furgoneta, se decide si asignar o cambiar el destino 2 según la condición de aplicabilidad.

3. Intercambiar Destinos: Este operador intercambia los destinos de dos furgonetas. Es útil para explorar soluciones en las que las furgonetas tienen asignados dos destinos, de los cuales el segundo está más cerca del origen que el primero. Al implementar la heurística 2 (que incluye el costo), este operador puede reducir el costo total.
Condición de aplicabilidad: Cada furgoneta debe tener un origen, un destino 1 y un destino 2, necesariamente.
Ramificación del operador: (F)Para cada furgoneta, se intercambia el destino 1 con el destino 2, y el número de bicicletas también se intercambia.

4. Cambiar Origen: Este operador permite cambiar el origen de las furgonetas, lo que puede ser útil para explorar diferentes combinaciones de orígenes, ya sea con rutas asignadas o con furgonetas sin rutas. Es importante ver que al aplicar este

operador, se deben recalcular las asignaciones de bicicletas para las rutas a destino 1 y destino 2, ya que suponemos que la furgoneta a la que se aplique este operador ya tenía un origen.
Condición de aplicabilidad: La furgoneta debe tener destinos asignados y el nuevo origen debe ser diferente a los destinos existentes. Además, el nuevo origen no debe estar en uso por otra furgoneta.
Ramificación del operador: (FXE) Por cada furgoneta, se decide si cambiar el origen o no, según la condición de aplicabilidad.

5. Desasignar Destino 1: Este operador desasigna el destino 1 de la ruta, lo que significa que la furgoneta ya no tendrá una ruta asignada.
Condición de aplicabilidad: La furgoneta debe tener un origen y un destino 1 asignado, pero no debe tener un destino 2 asignado.
Ramificación del operador: (F) Para cada furgoneta, se elimina el destino 1.

6. Desasignar Destino 2: Este operador desasigna el destino 2 de la ruta, lo que significa que la furgoneta ya no tendrá una ruta asignada.
Condición de aplicabilidad: La furgoneta debe tener un origen, un destino 1 y un destino 2 asignados.
Ramificación del operador: (F) Para cada furgoneta, se elimina el destino 2.

1.3 La descripción/justificación de las estrategias para hallar la solución inicial

En cuanto a la solución inicial del problema, hemos tenido discusiones sobre cuál debería ser la solución inicial más trivial. Después de varios planteamientos, llegamos a la conclusión de que la solución inicial trivial debería ser una solución vacía, lo que significa que inicialmente ninguna furgoneta tiene ningún origen asignado. Pero, nos dimos cuenta de que con esta solución inicial, el algoritmo no avanzaba como debería. Esto se debía a que asignar un origen no generaba pérdidas ni ganancias, lo que hacía que el algoritmo no progresara.
A partir de esta experiencia , decidimos que inicialmente todas las furgonetas deberían tener un origen asignado. A continuación, describiremos dos enfoques para la solución inicial:

Solución Inicial 1 (Trivial):

En esta solución, asignamos un origen a todas las furgonetas de manera secuencial, asegurándonos de que dos furgonetas no tengan el mismo origen. Si el número de estaciones es mayor que el número de furgonetas, asignamos un origen

a cada furgoneta de forma secuencial. De lo contrario, asignamos un origen a cada furgoneta de acuerdo al número de estaciones disponibles.

Solución Inicial 2 (Greedy):

En esta solución inicial más elaborada, ordenamos las estaciones de acuerdo a la demanda y la disponibilidad de bicicletas en orden creciente. Esto significa que las estaciones con mayor demanda y menor disponibilidad de bicicletas ocupan las primeras posiciones en la lista, y las estaciones con menor demanda y mayor disponibilidad de bicicletas ocupan las últimas posiciones.

Luego, asignamos los orígenes de las furgonetas según esta ordenación, de manera que las furgonetas tendrán origen a las estaciones con menor demanda y mayor disponibilidad de bicicletas.

1.4 La descripción/justificación de las funciones heurísticas

Las dos posibles heurísticas que pueden extraerse del enunciado de la práctica son las siguientes:

La primera heurística solo tendrá en cuenta las bicicletas que una furgoneta transporta y que se acercan a la demanda. Esta heurística busca maximizar el beneficio al evaluar una única unidad de furgoneta. El cálculo de esta heurística es sencillo: suma el número de bicicletas (nb1 y nb2) destinadas a los dos destinos y busca maximizar esta cantidad.

La segunda heurística es una variante de la primera, y se diferencia en que tiene en cuenta el coste que supone para la empresa el transporte de las bicicletas. En esta heurística, el cálculo es más elaborado. Si la furgoneta tiene un solo destino, se resta del beneficio el coste de transporte, que se calcula como ((nb1+9) div 10) multiplicado por la distancia entre el origen y el destino 1. Si hay dos destinos, el cálculo se complica, ya que se debe tener en cuenta que la furgoneta aún tiene la carga de las bicicletas para el posible segundo destino. El coste se calcula como (((nb1+nb2)+9) div 10) multiplicado por la distancia entre el origen y el destino 1, más el coste de (nb2+9) div 10 por la distancia entre el destino 1 y el destino 2.

Ambas heurísticas son correctas, ya que buscan maximizar el beneficio y trabajan con una sola unidad. La elección entre ellas dependerá de las necesidades específicas del problema a resolver.

2. Experimentación

2.1 Experiment 1: Determinar conjunto de operadores óptimos.

Observación:Existe la posibilidad de que algunos conjuntos de operadores conduzcan a soluciones superiores en comparación con otros.
Planteamiento: El objetivo de este experimento es determinar conjuntos de operadores diferentes, evaluando tanto la calidad de los resultados obtenidos como el tiempo de ejecución.
Hipòtesis: En nuestra hipótesis nula, consideramos que todos los conjuntos de operadores generan resultados similares. En la hipótesis alternativa, postulamos que ciertos conjuntos de operadores sobresalen en términos de beneficio y tiempo de ejecución.
Hipótesis alternativa: Hay ciertos conjuntos de operadores que predominan en términos de beneficio y tiempo.
Metodología:

- Generar doce conjuntos de operadores:
  - Sean los operadores :
    1. Cambiar Destino 1
    2. Cambiar Destino 2
    3. Cambiar Origen
    4. Intercambiar la Ruta
    5. Desasignar Destino 1
    6. Desasignar Destino 2
- Generamos 10 semillas aleatorias, una para cada réplica.
- Para cada semilla, se hace el cálculo de tiempo y beneficio para cada conjunto de operadores.
- Tanto el heurístico, como el número de estaciones, bicicletas y furgonetas son constantes.
- - Conjuntos considerados
- {1_},{1_2},{1_2_3},{1_2_3_4},{1_2_3_4_5},{1_2_3_4_5_6},{1_3},{1_2_4},{1_5},{1_3_5},{1_2_5_6},{1_2_4_5_6}.
-

En la Figura 1, se aprecia claramente el beneficio de cada conjunto de operadores en 10 semillas diferentes. La presencia de conjuntos repetidos en los resultados sugiere que, en este problema, los operadores 1 y 2 son los que prevalecen y se utilizan prioritariamente, independientemente de la combinación con otros operadores.

Es importante notar que aunque el número total de combinaciones de operadores sería de 6!, muchas de estas combinaciones carecen de sentido práctico. Por ejemplo, asignar el Destino 2 debe ir siempre de la mano con asignar el Destino 1. Por lo tanto, se redujo el conjunto de combinaciones consideradas.

Para determinar el conjunto óptimo, la Figura 2 muestra el tiempo de ejecución. La elección se basó en encontrar un equilibrio entre el beneficio y el tiempo. Aunque el conjunto 1_2_5_6 tiene la menor media en beneficios, se observó experimentalmente que los operadores 5 y 6 rara vez, incluso podríamos decir que no se aplican en la práctica, lo que lleva a la elección del conjunto 1_2_3 como el más adecuado. La diferencia en las medias de tiempo de ejecución entre estos dos conjuntos es mínima, menos de una unidad en milisegundos.

Esta decisión se tomó considerando diversas semillas y configuraciones del problema, lo que respalda la elección del conjunto de operadores 1_2_3 como el más eficiente.

Esta información refuerza la validez de la selección del conjunto de operadores y su impacto en la eficacia del algoritmo.

Operador/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
1_	93	85	57	91	87	82	81	67	73	106	**82,2**
1_2	103	92	57	91	105	82	81	99	73	115	**89,8**
1_2_3	103	92	57	91	105	82	81	99	73	115	**89,8**
1_2_3_4	103	92	57	91	105	82	81	99	73	115	**89,8**
1_2_3_4_5	103	92	57	91	105	82	81	99	73	115	**89,8**
1_2_3_4_5_6	103	92	57	91	105	82	81	99	73	115	**89,8**
1_3	*93*	85	57	91	87	82	81	67	73	106	**82,2**
1_2_4	103	92	57	91	105	82	81	99	73	115	**89,8**
1_5	93	85	57	91	87	82	81	67	73	106	**82,2**
1_3_5	93	85	57	91	87	82	81	67	73	106	**82,2**
1_2_5_6	103	92	57	91	105	82	81	99	73	115	**89,8**
1_2_4_5_6	103	92	57	91	105	82	81	99	73	115	**89,8**

Figura 1: *Tabla de medias del beneficio para diferentes conjuntos de operadores*

Operador/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
1_	0,974	0,813	0,935	0,763	0,82	0,708	0,48	0,47	0,446	0,431	**0,684**
1_2	2,641	1,604	1,197	1,246	2,186	1,069	0,859	1,533	0,771	1,15	**1,4256**
1_2_3	3,828	2,514	1,881	1,857	3,258	3,612	1,093	2,232	1,251	1,781	**2,3307**
1_2_3_4	4,396	2,411	1,772	2,352	3,394	1,084	1,12	2,236	1,129	1,722	**2,1616**
1_2_3_4_5	3,895	2,477	1,781	1,94	3,329	1,15	1,141	2,318	1,099	1,76	**2,089**
1_2_3_4_5_6	4,223	1,388	1,811	1,808	3,293	1,137	1,094	2,183	1,116	1,747	**1,98**
1_3	2,542	1,787	1,366	1,41	1,408	0,931	0,854	0,83	0,83	0,831	**1,2789**
1_2_4	2,731	0,782	1,14	1,195	2,26	0,737	0,717	1,529	0,696	1,171	**1,2958**
1_5	0,959	1,417	0,811	0,751	0,794	0,515	0,465	0,494	0,228	0,449	**0,6883**
1_3_5	1,466	1,658	1,452	1,41	1,342	0,86	0,848	0,883	0,894	0,895	**1,1708**
1_2_5_6	2,304	0,935	1,211	1,178	2,136	0,713	0,707	1,642	0,685	1,272	**1,2783**
1_2_4_5_6	2,288	1,197	1,28	1,375	2,094	0,722	0,712	1,698	0,749	1,303	**1,3418**

***Figura 2**: Tabla de medias de tiempos para diferentes conjuntos de operadores*

2.2 Experimento 2: Determinar qué solución inicial es mejor.

Observación: Es posible que uno de los generadores de soluciones iniciales sea más efectivo en términos de beneficio y tiempo en comparación con el otro.
Planteamiento: Observar con qué generador de solución inicial, independientemente de las condiciones del problema, funciona mejor.
Hipòtesis: Creemos que el generador de solución Greedy será mejor que el secuencial.
Alternativa: El beneficio o el tiempo de ejecución no depende del generador de solución planteado.
Método:

- Generamos 10 semillas aleatorias, que serán las réplicas del experimento.
- Para cada réplica, usamos el algoritmo Hill Climbing pero cambiando el generador de solución inicial.
- Con los beneficios y tiempos de ejecución de cada réplica, hacemos la media.

En **la Figura 3**, se muestra la media de beneficios según el generador de solución inicial. En esa tabla, podemos observar una diferencia notable en los beneficios, con el generador Greedy superando consistentemente al generador secuencial.

En lo que respecta al tiempo de ejecución para cada generador, notamos que el generador Greedy requiere un poco más de tiempo. Esto puede deberse a la fase de ordenación que implica.

Basándonos en los resultados, concluimos que el generador de solución inicial más efectivo es el Greedy. Ofrece un beneficio medio que es casi el doble en comparación con el generador de solución inicial Trivial (secuencial), mientras que el tiempo medio de ejecución es prácticamente el mismo (como se muestra en **la Figura 4**).

Esta elección tiene sentido, ya que el generador Greedy nos proporciona una solución inicial desde la cual es más probable alcanzar un máximo local superior en comparación con el generador de solución inicial Trivial (secuencial).

Ini/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
Trivial	62	17	25	65	79	69	39	46	23	30	45,5
Greedy	103	92	57	91	105	82	81	99	73	115	89,8

***Figura 3** : Beneficios con diferentes generadores de solución inicial*

Ini/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
Trivial	4,798	2,798	6,608	2,714	4,507	5,316	4,488	2,843	0,562	1,386	3,602
Greedy	7,354	5,194	2,573	2,135	3,735	2,892	3,680	6,981	1,774	2,919	3,924

***Figura 4** : Tiempos con diferentes generadores de solución inicial*

2.3 **Experimento 3:** Determinar mejores parámetros para Simulated Annealing.

Observación : Tiene que haber una combinación de los parámetros (K,Lambda) para el algoritmo de Simulated Annealing que dé mejores resultados que los demás.
Planteamiento: Para cada combinación observar qué beneficio nos da y compararlos.
Hipòtesis: Creemos que todos los resultados serán iguales.
Alternativa: Habrá una configuración de K y Lambda que será de mejor beneficio que todas las otras posibles configuraciones.
Método:

- Con 10 seeds aleatorios hacer 15 iteraciones con Simulated Annealing con una configuración posible.
- Coger la media de los 15 iteraciones con los 10 seeds.
- Los posibles valores de K son {1,5,25,125} y de Lambda son {10,1,0.01,0.001}

Hemos determinado hacer 10000 iteraciones con 10 iteraciones por cambio de temperatura, que son números suficientes para que converge el algoritmo.
En la Figura 5 podemos observar que el mejor beneficio es para K = 5 y Lambda = 0.001.

Lambda / K	K = 1	K = 5	K = 25	K = 125
λ = 10	64	51	41	39
λ = 1	92	84	75	73
λ = 0.01	95	88	66	51
λ = 0.001	96	63	44	46

***Figura 5**: Tabla con los beneficios para todas las posibles configuraciones de K y Lambda*

2.4 **Experimento 4**: Determinar la evolución del tiempo dependiendo del número de estaciones del problema.

Observación: Creemos que hay una tendencia a que el programa tarde más dependiendo del número de estaciones que hay en el problema.
Planteamiento: Ver como aumenta el tiempo de ejecución a medida que aumentamos el número de estaciones.
Hipòtesis: A más número de estaciones, más tiempo de ejecución.
Alternativa : El número de estaciones no afecta al tiempo de ejecución del algoritmo.
Método:

- Para 10 semillas generadas aleatoriamente, hacemos la réplica.
- Consiste, para cada semilla, ejecutar el Hill Climbing 15 veces, pero cada iteración aumenta el número de estaciones en 25.
- Para cada iteración con cada semilla, coger el tiempo de ejecución.
- Con la ayuda de un software estadístico (R) , ver la correlación entre el tiempo que tarda y el número de estaciones.

Simplemente hemos creado el programa que extraiga toda la información necesaria y la hemos pasado a R.

La Figura 5 nos muestra cómo nos han quedado los resultados para un seed determinado.

	E	B	F	TIEMPO
1	25	1250	5	2871
2	50	2500	10	15169
3	75	3750	15	52272
4	100	5000	20	132288
5	125	6250	25	275470
6	150	7500	30	513787
7	175	8750	35	873728
8	200	10000	40	1349754
9	225	11250	45	1941294
10	250	12500	50	2933339
11	275	13750	55	4181825
12	300	15000	60	5828366
13	325	16250	65	8012454
14	350	17500	70	10622829

***Figura 5**: Tabla con la relación tiempo - número de estaciones.*

Hemos hecho lo mismo para todos las semillas. Y hemos obtenido la media para la regresión lineal.

La fórmula es : TIEMPO = -2558914 + 29265.137 * E (Estaciones)

Podemos afirmar que hay una relación lineal entre el tiempo de ejecución y el número de estaciones del problema.

2.5 Experimento 5: Diferencias entre Hill Climbing y Simulated Annealing con las distintas Heurísticas

Observación: Pensamos que la aplicación de una de las heurísticas es mejor que la otra.
Planteamiento: Observamos las diferencias de las medias tanto del beneficio, como de los tiempos y los kilómetros con las diferentes Heurísticas en cada algoritmo.
Hipótesis: Creemos que la Heurística 1 dará mejores resultados
Alternativa: La Heurística 2 es mejor.
Método:

- Generar 10 semillas aleatoriamente, que serán las réplicas.
- Para cada semilla se usa los dos algoritmos.
- Se calcula los beneficios, los tiempos y los kilómetros recorridos
- Se hace la media y se evalúan los datos
- Para Simulated Annealing hacemos 15 iteraciones para cada seed
- Hacemos este procedimiento para ambos Heuristicos.

En las siguientes tablas observamos los resultados de este experimento.

Heurística /Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	103	92	57	91	105	82	81	99	73	115	89,8
H2	49	69,7	46,9	57,8	49,3	52	59,4	44,2	51,3	67,8	54,74

Tabla 1: *Beneficios con Hill Climbing según la heurística.*

Heurística /Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	122,67	86,53	71	104,27	107,73	93,73	102,8	39,8	87,27	111,4	**92,72**
H2	51	46,96	32,55	47,47	49,76	50,63	46,83	43,36	40,29	53,75	**46,26**

Tabla 2: *Beneficios con Simulated Annealing según la heurística.*

Heurística /Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	3	2	1	1	2	1	1	2	1	1	2
H2	2	2	1	1	1	1	1	1	1	1	1

Tabla 3 : *Tiempos con Hill Climbing según la heurística.*

Heurística /Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	8,533	7,4	7,1333	6,1333	5,1333	5,1333	5,0666	5,066	5,066	5,066	5,97308
H2	8,133	8,666	7,6	6,666	5,4666	5,7333	5,266	5,6	5,4	5,7333	6,42642

Tabla 4: *Tiempos con Simulated Annealing según la heurística.*

Heurística /Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	74,7	38,1	41,5	28,8	43	21,6	32,3	45	37,1	49,7	41,18
H2	13,5	8	8,1	12,4	14,2	10	9,4	17,4	10,3	10,2023	11,35023

Tabla 5: *Kilómetros recorridos con Hill Climbing según la heurística.*

Heurística/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
H1	38,42	32,99	29,24	34,63	33,84	28,13	27,53	39,80	31,73	44,58	34,09
H2	11,97	11,48	10,94	12,02	11,48	9,20	13,65	10,36	10,64	11,89	11,36

***Tabla 6**: Kilómetros recorridos con Simulated Annealing según la heurística.*

En términos de beneficio, se observa claramente que H1 supera a H2, principalmente debido a que H1 no tiene en cuenta los costes de transporte. Este resultado se refleja consistentemente en ambos algoritmos.

En relación al tiempo de ejecución, con Hill Climbing, se evidencia que H2 requiere más tiempo en comparación con H1, mientras que en Simulated Annealing, la situación se invierte. Sin embargo, las diferencia en el tiempo de ejecución no son particularmente significativas. En general, se podría afirmar que independientemente del heurístico empleado, el tiempo de ejecución es relativamente consistente.

Respecto a los kilómetros recorridos, es interesante notar que, independientemente del algoritmo utilizado, la implementación de H2 resulta en una reducción sustancial en la distancia recorrida. Este hallazgo es coherente, ya que H2 tiene en cuenta los costes de transporte y, por lo tanto, tiende a generar soluciones más eficientes en términos de distancia.

Estos resultados sugieren que la elección del heurístico tiene un impacto significativo en los beneficios y en la distancia recorrida, mientras que el tiempo de ejecución se mantiene relativamente constante independientemente del heurístico utilizado.

2.7 **Experimento 7:** Determinar el número de furgonetas óptimas para el mejor beneficio dependiendo del tipo de demanda.

Observación: Queremos averiguar si a partir de un determinado número de furgonetas ya es imposible tener más beneficio.
Planteamiento: Generar diferentes problemas con un número diferente de furgonetas, manteniendo todo constante.
Hipòtesis: Habrá un punto en el que aumentar las furgos ya no dará mejor beneficio.
Alternativa: Aumentar furgos siempre dará mejor beneficio.

Método:

- Generamos 10 semillas aleatoriamente, que serán las réplicas de los experimentos
- Usamos Hill Climbing sobre el programa con las variables iniciales.
- Para cada semilla, hacemos el algoritmo con 5 furgonetas y miramos el beneficio. Luego ponemos 5 furgonetas más y así hasta que el beneficio se estanque.

Hemos visto que hay un número de furgonetas dónde ir añadiendo más no mejora el beneficio.
En la siguiente tabla vemos que ronda entre los 19 y 18 furgonetas.

Demanda/Seed	3756	583	84787	263	7085	25	2884	9683	25654	4983	media
Equilibrada	20	15	20	15	20	20	20	20	20	20	19
Hora punta	20	15	15	15	20	20	15	20	20	20	18

Figura 6: *Número de furgonetas máximas para alcanzar el beneficio máximo.*