Práctica de búsqueda local
Curso 2017-2018 Q1

Inteligencia Artificial - FIB

Documentación de la realización de la práctica y los experimentos

Sebastián Sánchez Menéndez
Daniel Martínez Bordes
Ferran Martínez Felipe
Sophia Lichtenberg

Contenido

Introducción

Uno de los problemas tratables desde el conjunto de algoritmos de búsqueda local es la planificación de rutas de abastecimiento. En nuestro caso, más concretamente, trataremos el problema del abastecimiento de gasolineras dentro de una distribución geográfica por parte de un conjunto de centros de distribución.

La primera pregunta a resolver consiste en porque utilizar algoritmos de búsqueda local. El caso de los problemas de planificación de abastecimiento es uno en el que no existe una única solución. Lo importante en este tipo de problemas no es tanto el obtener la solución óptima sino cuánto se tarda en generar, y en eso los algoritmos de búsqueda local sobresalen respecto a los algoritmos de búsqueda heurística o búsqueda ciega (si en un momento dado decidimos que no podemos dar más tiempo al algoritmo y detenemos su ejecución en el caso de los algoritmos de búsqueda local obtendremos una solución, mientras que usando algoritmos de búsqueda heurística o ciega probablemente no). Es por ello que nosotros usaremos algoritmos de búsqueda local para resolver este problema.

Dicho esto, es el momento de entrar en detalle en el problema.

Nuestro problema parte de una distribución geográfica de **a x a** km^2 , en la que tenemos un número **n** de centros de distribución y un número **m** de gasolineras.

Sobre los centros de distribución podemos afirmar que un centro de distribución tiene a su disposición uno o más camiones de reparto (notar que todos los centros de distribución disponen del mismo número de camiones de reparto, sea cual sea este número). Cada camión, a su vez, tiene una cisterna con capacidad para atender a dos peticiones recibidas por parte de las gasolineras.

Además de lo comentado anteriormente, los camiones tienen una distancia máxima que pueden recorrer **v** y un número máximo de viajes por día **m**. Cada kilómetro recorrido tiene un coste **x** para la empresa, lo que implica que el coste de los viajes es lineal respecto a la distancia de los mismos (a la empresa le interesa reducir la distancia recorrida por los camiones). A efectos de nuestro problema consideraremos que los camiones se mueven a 80 km/h de forma constante (no tenemos en cuenta el tiempo de aceleración y deceleración del camión, y consideraremos que los camiones cargan y descargan instantániamente), y trabajan durante 8h diarias. Esto, en conjunto, nos permite deducir que cada camión puede recorrer **640 km al día como máximo**. Otros parámetros relevantes de los camiones para el problema serán que un camión puede realizar como máximo **5 viajes al día** y que el **coste por kilómetro de un camión es de 2**.

Las gasolineras tienen a su disposición un conjunto de depósitos, que usan para atender a los clientes que van recibiendo. Supondremos que una gasolinera utiliza un único depósito cada vez, y que no empiezan a usar un depósito lleno hasta que no vacían del todo el depósito anterior.

Lo siguiente que hay que saber es que en momento en que una gasolinera vacía un depósito, ésta realiza una petición a la compañía que gestiona los centros de distribución a efectos de que sea rellenado. Las peticiones no tienen porque ser atendidas inmediatamente, así que una gasolinera puede tener vacíos varios depósitos y por tanto tener varias peticiones pendientes por atender. Las peticiones, por tanto, pueden llevar días sin atender.

La forma de proceder de los repartos es la siguiente: cada mañana le genera una lista con las peticiones a atender ese día, y estas peticiones son asignadas a camiones para que hagan el reparto. Los camiones, como ya hemos comentado anteriormente, pueden atender dos peticiones cada vez como máximo (nada impide que se atienda una única petición) antes de volver al centro de distribución. Los camiones siempre vuelven al centro de distribución del que salen.

Dependiendo del número de días que lleve una petición pendiente, el beneficio obtenido por atenderla disminuye. Al atender la petición el mismo día que se realiza se cobra un 102% del precio de esta. A partir de ahí, el precio sigue la siguiente fórmula:

$$\%precio = (100 - 2^{dias})\,\%$$

Es por tanto evidente que a la compañía que gestiona los centros de distribución le interesa atender las peticiones cuanto antes. Comentar que de base el **coste de rellenar un depósito es de 1000.**

Sobre la distribución geográfica que abarcaremos en nuestro problema, consideraremos que nos movemos por una **área de 100 x 100 km^2,** en la que conoceremos la ubicación tanto de las gasolineras como de los centros de distribución (en valores enteros). La forma de calcular la distancia entre las gasolineras y los centros de distribución será la siguiente:

$$d(D,G) = |D_x - G_x| + |D_y - G_y|$$

donde D_x y D_y son las coordenadas x e y del centro de distribución y G_x y G_y son las coordenadas x e y de la gasolinera (en valor absoluto, indicado por los | |).

Las características del problema, analizando lo expuesto anteriormente, serán las siguientes:

- **Número de gasolineras a abastecer:** Las gasolineras son las que realizan las peticiones, así que será necesario tener en cuenta el número de gasolineras que están realizando peticiones a efectos de priorizar las que nos den más beneficios.
- **Coordenadas de cada gasolinera:** Para poder calcular el coste que nos supondrá que un camión abastezca una gasolinera (coste por km) será necesario saber dónde se ubican las gasolineras.
- **Peticiones a abastecer de cada gasolinera (y días que llevan pendientes):** Probablemente la característica más importante del problema. A la hora de asignar los viajes a camiones necesitaremos saber que gasolineras tienen peticiones asignadas, y sobretodo cuantos días llevan pendientes esas peticiones (ya que el beneficio por atender una petición decrece exponencialmente en función de los días pendientes de la petición).
- **Número de centros de distribución de los que disponemos:** Los centros de distribución son los encargados del reparto de gasolina por parte de los camiones. A más centros de distribución más peticiones podremos atender.
- **Coordenadas de los centros de distribución:** Al igual que con las coordenadas de las gasolineras, estas coordenadas son necesarias para calcular el coste por km que nos supondrá atender una petición
- **Número de km que puede recorrer un camión cisterna por día:** Característica que limita la distancia máxima que puede recorrer un camión en un día. Ésto limitará el número de peticiones que podrá atender un camión en un día.
- **Viajes máximos de un camión en un día:** Relevante debido a que esta limitación supondrá que solo podremos atender un número limitado de peticiones por camión

No obstante, muchos de estos elementos no son necesarios para describir el estado del problema, ya que simplemente son restricciones sobre las soluciones a generar (nos limitarán el espacio de soluciones sobre el que navegaremos). El conjunto de elementos, pues, que nos representarán el estado del problema serán los siguientes:

- **Camiones de reparto:** Encargados de realizar los repartos. Vienen definidos por el centro de distribución del que provienen.
- **Viajes a realizar:** Pares de peticiones a atender por un camión en un viaje determinado. Un viaje se compone por el viaje del camión desde el centro de distribución a la primera gasolinera, el viaje de éste a la segunda gasolinera (si se requiriera) y el regreso al mismo centro de distribución.

Representación del estado

Para solucionar el problema de la planificación de rutas de abastecimiento de las gasolineras mediante algoritmos de búsqueda local, el primer paso que debemos realizar es la definición de la representación del estado que será utilizada. Esto implica analizar los elementos que influyen en la solución del problema. Contamos con un conjunto de camiones, que tienen que atender ciertos pedidos pertenecientes a gasolineras, y debemos asignar pedidos a los camiones, cumpliendo unos ciertos criterios, de forma que se optimice el beneficio obtenido. Es decir, los elementos básicos que intervienen en el problema son los camiones, y los pedidos.

Un camión se puede identificar mediante un entero: su posición en la clase CentrosDistribucion. Sin embargo, un pedido necesita dos enteros para ser identificado: su posición en la clase Gasolineras, y dentro de esa instancia de Gasolinera, su posición en el vector de peticiones.
Por lo tanto, hemos decidido implementar una clase Order que representa una petición, con sus correspondientes atributos.

Como posible representaciones del estado proponemos las siguientes:

- Un diccionario, de la forma <Order, int>, de forma que a cada orden le corresponde un camión. Las ordenes no asignadas quedarían representadas por un value **null.**
- Un vector, donde a cada posición le corresponde otro vector de tipo Order. Cada fila del vector representa un camión, y el vector que contiene son las Ordenes que tiene asignadas. Dado un número de camiones C, el vector tendría C+1 vectores de órdenes, donde el último vector son las órdenes no asignadas. Como los camiones pueden realizar dos pedidos en un solo viaje, decidimos implementar una clase Trip que agrupa dos pedidos.

Dadas estas dos representaciones, vamos a pensar en las implicaciones temporales que tiene cada una de ellas. Al realizar cualquier operación posible sobre los peticiones y los camiones (asignar una petición no asignada a un camión, intercambiar una petición entre camiones, o mover una petición de un camión a otro), siempre hemos de comprobar que no nos salimos del espacio de soluciones, comprobando las restricciones sobre el número de viajes y la distancia recorrida. En la segunda representación, dado que podemos acceder directamente a todas las peticiones asignadas a un camión, podemos hacer este cálculo de forma constante recorriendo sus peticiones. Sin embargo, en la representación en forma de diccionario deberíamos recorrer todas las peticiones para saber cuales pertenecen al camión que alteramos, por lo que tendríamos una complejidad temporal T= O(|peticiones|). La aplicación de los operadores es constante en cualquier caso, ya que el acceso al pedido es constante.

Otro problema de la representación en forma de diccionario es que mantiene información sobre a qué camión está asignada cada petición, pero no en qué orden se realizan esas peticiones. Dado que en un viaje se pueden hacer dos peticiones, la forma de agruparlas es importante, porque afecta a la distancia recorrida y por tanto al cálculo del beneficio.
Por tanto, elegimos la representación del estado en forma de vector de vectores de Trip, que se implementa en Java con la clase ArrayList < ArrayList < Trip > >. Llamaremos a esta estructura trips. Cada vector tendrá un número fijo de viajes, pero podrá tener o no ordenes asignadas. En caso de no tener una orden asignada, el valor de la orden dentro del viaje será null. El camión fantasma tendrá |peticiones|/2 viajes, de forma que podremos tener todas las ordenes sin asignar. Cuando una orden se asigne, el espacio que ocupaba en el camión fantasma pasará a tener valor null.
Hemos decidido utilizar esta representación porque nos permite iterar sobre una misma estructura de datos para realizar todas las operaciones.

Para acceder a un pedido se hará mediante tres variables, i, j, y k. La variable i representará el camión al que accedemos, la variable j representa el número de viaje y la variable k representa el número de pedido dentro de ese viaje.

Generación de la solución inicial

Como posibles soluciones iniciales a estudiar tenemos 4 variantes:

- **Solución vacía:** Esta propuesta generará un estado inicial donde todos los vectores de Trips de los camiones que no son el fantasma tendrán órdenes nulas (no atienden ninguna petición), y el camión fantasma contendrá todas las órdenes.
  El algoritmo itera sobre el número de peticiones, lo que implica que su coste es O(|peticiones|). Permite que el algoritmo trabaje más con operadores de adición.

- **Solución llena, aleatoria:** Esta propuesta generará un estado inicial donde las peticiones se introducen en un viaje (con órdenes nulas) de un camión aleatorio (el camión fantasma incluido), comprobando siempre la restricción del máximo de distancia permitido.
  El algoritmo itera también sobre el número de peticiones y en cada iteración genera índices aleatorios para el camión, viaje y orden del viaje que fijan la posición y en caso que se cumplan las restricciones añade la orden, en caso contrario genera unos nuevos índices. En el caso peor el coste es O(|peticiones|*|camiones|), siendo esta una cota bastante superior, en media tiende a O(|peticiones|).

- **Solución llena, ordenada:** Esta propuesta generará un estado inicial donde las peticiones se introducen en una orden nula de un viaje de un camión. En este caso se escoge siempre el primer camión que cumple las restricciones (recorriendo los camiones según su índice). Las peticiones restantes pasan al camión fantasma.
  El algoritmo itera también sobre el número de peticiones y en cada iteración se recorre los camiones por orden. En el caso peor el coste es O(|peticiones|*|camiones|).

- **Solución llena, greedy:** Esta propuesta generará un estado inicial donde las peticiones se introducen en cada camión siguiendo el criterio de introducir las órdenes cuya distancia centro de distribución-gasolinera sea mínima (las órdenes más cercanas). Tendremos que ordenar las órdenes no asignadas para cada camión que empezamos a llenar. Se itera sobre los camiones y se ordena en cada iteración por lo que el coste es O(|camiones| * (|peticiones|*log(|peticiones|))).

Todas las posibles soluciones iniciales tienen un coste O(|camiones|), paralelo previo a su ejecución, ya que se inicializa la representación del estado a nulo.

Generadora de sucesores

La generación de los estados sucesores es uno de los elementos más importantes a fijar, ya que marcan los posibles caminos que puede seguir un estado inicial para convertirse en uno con características más favorables.
La generadora de sucesores se basa en la aplicación de unos operadores en el estado actual. Para estos operadores hay que considerar diversos factores:

- **El factor de ramificación:** Los operadores se aplicarán de todos los modos posibles en el estado actual, en el caso de Hill-Climbing, y es el valor del heurístico el que decide cual es el mejor estado sucesor, y en Simulated Annealing se aplicará un operador al azar y se decide si es suficientemente bueno. Por ello el número de combinaciones posibles que puede generar el operador es relevante, ya que se estarán generando todos los estados posibles cada vez que se fije un nuevo estado actual y esto afecta en el tiempo de ejecución.

- **La posibilidad de explorar todo el espacio de soluciones:** Si los operadores no permiten llegar a ciertos estados, podríamos impedir que se llegue a ciertos estados con soluciones óptimas. Al contrario si los operadores generan estados repetidos estaríamos afectando el tiempo de ejecución.

Por tanto debemos escoger unos operadores que nos permitan explorar el espacio de estados, pero suficientemente restringidos para evitar estados repetidos, o no accesibles (no pueden mejorar la solución propuesta).

Considerando la naturaleza de nuestro problema y considerando la representación del estado escogida anteriormente, planteamos diversas alternativas:

(A) Operador intercambiar orden:

Esta opción plantea un solo operador encargado de realizar intercambios de órdenes entre cualquier camión y viaje, incluido el camión contenedor de órdenes no atendidas (camión fantasma) y las órdenes vacías. Este operador permite generar todos los espacios de solución posibles ya que permite permutar cualquier petición asignada o no asignada. Por otra parte este operador genera una gran cantidad de estados idénticos en cada generación de estados.

Las condiciones de aplicabilidad son:

- El camión A tenga identificador menor o igual que el camión B (para no generar 2 intercambios idénticos).

- Si los identificadores de camiones son iguales no sean el camión fantasma y tampoco sean el mismo viaje (para evitar intercambios sin efecto en el heurístico, es decir estados idénticos).
- La distancia recorrida por los camiones tras el intercambio no supere la distancia máxima permitida (640 km).

El factor de ramificación es O(|Peticiones|*|Peticiones|) ya que permite intercambiar todas las peticiones con todas.

(B) Operador añadir orden:

Esta opción plantea un solo operador que se encarga de insertar órdenes del camión fantasma a otro camión. Este operador limita el espacio de estados ya que fija las órdenes asignadas.

Las condiciones de aplicabilidad son:

- El camión A tenga identificador del camión fantasma y el camión B diferente al camión fantasma.
- Que la orden del camión A no sea nula y que el camión B tenga algún espacio nulo.
- La distancia recorrida por el camión B tras el movimiento no supere la distancia máxima permitida (640 km).

El factor de ramificación es O(|Ordenes|*|Camiones|) ya que permite añadir cualquier orden del camión fantasma a todos los camiones.

(C) Operadores intercambiar orden y añadir orden:

Esta opción plantea trabajar con dos operadores con aplicaciones más controladas. Intercambiar orden permite permutar dos órdenes asignadas (no nulas) pertenezcan al camión fantasma o no, añadir orden permite mover una orden no atendida (en el camión fantasma) a un espacio de orden no asignada (orden nula en un camión). Estos operadores amplían el espacio de estados y minimizan la generación de estados idénticos ya que no se intercambian peticiones no asignadas (nulas).

Las condiciones de aplicabilidad de intercambiar son:

- El camión A y el camión B tengan peticiones asignadas (no nulas).
- El camión A tenga identificador menor o igual que el camión B (para no generar 2 intercambios idénticos).
- Si los identificadores de camiones son iguales no sean el camión fantasma y tampoco sean el mismo viaje (para evitar intercambios sin efecto en el heurístico, es decir estados idénticos).
- La distancia recorrida por los camiones tras el intercambio no supere la distancia máxima permitida (640 km).

El factor de ramificación es O(|Ordenes|*|Ordenes|) ya que permite intercambiar todas las ordenes con todas en caso peor.
Las condiciones de aplicabilidad y el factor de ramificación de añadir orden son los descritas anteriormente.

(D) Operadores intercambiar orden y añadir orden (alternativa):

Esta opción es similar a la descrita anteriormente, pero añade una restricción adicional. No se permite el intercambio de una orden asignada con una orden del camión fantasma. Esto reduce el espacio de estados y puede afectar en el estado final a cambio mejora el tiempo de ejecución.

Las condiciones de aplicabilidad de intercambiar son:

- El camión A y el camión B tengan peticiones asignadas (no nulas).
- El camion A y el camion B no sean el camion fantasma.
- El camión A tenga identificador menor o igual que el camión B (para no generar 2 intercambios idénticos).
- Si los identificadores de camiones son iguales no sean el camión fantasma y tampoco sean el mismo viaje (para evitar intercambios sin efecto en el heurístico, es decir estados idénticos).
- La distancia recorrida por los camiones tras el intercambio no supere la distancia máxima permitida (640 km).

El factor de ramificación es O(|Ordenes|*|Ordenes|) ya que permite intercambiar todas las ordenes con todas en caso peor.

Las condiciones de aplicabilidad y el factor de ramificación de añadir orden son los descritas anteriormente.

(E) Operadores intercambiar orden y añadir orden (alternativa 2):

Esta opción presenta otra alternativa sobre las condiciones de aplicabilidad del operador intercambiar. En este caso, se permite el intercambio con el camión fantasma y dentro de un mismo camión. Sin embargo, impide los cambios con el resto de camiones. Esto reduce el espacio de estados y puede afectar en el estado final a cambio mejora el tiempo de ejecución.

Las condiciones de aplicabilidad de intercambiar son:

- El camión A y el camión B tengan peticiones asignadas (no nulas).
- El camion A sea el camión fantasma, o el camión A y B sean el mismo.
- La distancia recorrida por los camiones tras el intercambio no supere la distancia máxima permitida (640 km).

- Si los identificadores de camiones son iguales no sean el camión fantasma y tampoco sean el mismo viaje (para evitar intercambios sin efecto en el heurístico, es decir estados idénticos).

El factor de ramificación es O(|Ordenes|*|Ordenes|) ya que permite intercambiar todas las ordenes con todas en caso peor.

Las condiciones de aplicabilidad y el factor de ramificación de añadir orden son los descritas anteriormente.

Heurísticos

En toda empresa lo que se busca siempre es maximizar los beneficios y minimizar los costes, a efectos de obtener el mayor beneficio neto posible. Basándonos en esto, en el caso del problema de abastecimiento a tratar nos encontramos con tres factores a tener en cuenta a la hora para calcular cuán buena es una solución:

- Los beneficios obtenidos por atender las peticiones.
- Los costes de hacer el reparto desde un centro de distribución a las gasolineras (los km que recorre el camión multiplicados por el coste de recorrer cada kilómetro).
- La cantidad de dinero que dejamos de ganar por no atender un pedido un día y atenderla al dia siguiente.

A partir de aquí, podemos inferir que un buen heurístico será aquel que intenten cumplir una o varias de las condiciones anteriormente mencionadas. En esta práctica se han planteado los siguientes heurísticos:

- **Número de órdenes a atender:** Heurístico que orienta la búsqueda a atender el máximo número de órdenes posibles. Al no tener en cuenta los beneficios ni los costes este heurístico está lleno de mesetas (multitud de soluciones distintas darían el mismo valor de heurístico). Pese a ser un heurístico muy poco costoso de calcular en tiempo, el gran conjunto de mesetas que genera lo convierten en un mal heurístico, lo que provoca que no lo hayamos tenido en cuenta para la experimentación.
- **Beneficios obtenidos por las peticiones servidas en un día menos el coste de hacer el reparto:** Este heurístico orienta la búsqueda a obtener el máximo beneficio posible, ignorando el hecho de que las peticiones no atendidas comportarán una pérdida económica. Ésto implica que la solución generada contendrá las peticiones más recientes (son las que mayor beneficio tienen) y las más cercanas (minimizando los costes). Éste heurístico tiene un coste en tiempo mucho mayor al anterior, pero el hecho de no tener mesetas lo convierte en un heurístico apto para la búsqueda a realizar.
- **Beneficio obtenido por las peticiones servidas en un día menos las pérdidas por hacer el reparto y el beneficio no ganado por las peticiones no atendidas ese dia:** Este heurístico orienta la búsqueda a, además de obtener el máximo beneficio posible, evitar pérdidas por las peticiones no atendidas. Ésto significa que, además de intentar tratar las peticiones más cercanas y con menos días pendientes, también se atenderán las peticiones más antiguas, ya que las pérdidas aumentan de forma exponencial respecto al número de días que llevan pendientes las peticiones. Este heurístico es el más costoso de los propuestos en cuanto a tiempo de cálculo, pero también es el que da una solución más certera respecto a lo que queremos maximizar.

En todos los casos las ponderaciones usadas son una ponderación de 1 para cada componente del heurístico, debido a que todos los componentes son igual de relevantes para maximizar al final los beneficios (dar más peso a los beneficios orienta la búsqueda a las peticiones recientes, aumentando las pérdidas, dárselo a los km recorridos orienta la búsqueda a atender a las gasolineras más cercanas ignorando los beneficios y las pérdidas y dárselo a las pérdidas orienta la búsqueda a atender las peticiones más antiguas, ignorando maximizar los beneficios). Experimentalmente se ha encontrado que se cumple lo comentado anteriormente, descendiendo los beneficios al cambiar las ponderaciones (sin un aumento significativo del resto de parámetros a tener en cuenta, como el tiempo de ejecución).

En conclusión, comentar que el heurístico escogido para este problema es el segundo heurístico (beneficios menos costes de viaje). Los motivos son que pese a que este heurístico genera menos beneficios que el tercer heurístico expuesto (aunque ésta diferencia es despreciable), los tiempos de ejecución se reducen sensiblemente al usar el segundo heurístico respecto al tercero. Ésto provoca que el segundo heurístico sea el más idóneo para el tipo de problema que estamos tratando, ya que los parámetros más importantes para el problema a tratar son beneficios y tiempo de ejecución (de nada nos sirve un algoritmo de generación de rutas de abastecimiento si éstas rutas se generan cuándo ya no sirven de nada).

Experimentación

Experimento 1

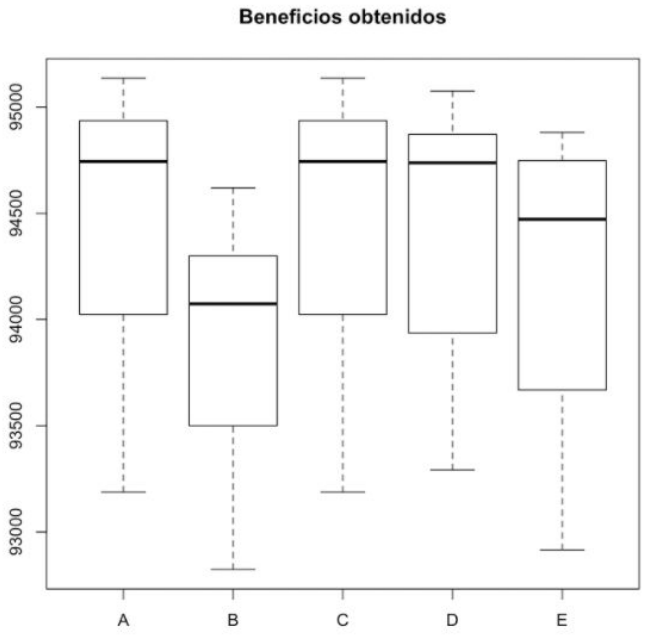
En este experimento queremos comparar los distintos conjuntos de operadores propuestos en función de la calidad de la solución obtenida con cada uno de ellos.
Para comparar soluciones, lo haremos basándonos en distintos parámetros: el tiempo de ejecución del algoritmo T, la puntuación obtenida dados los criterios que se describen P, y la relación que hay entre ellos.

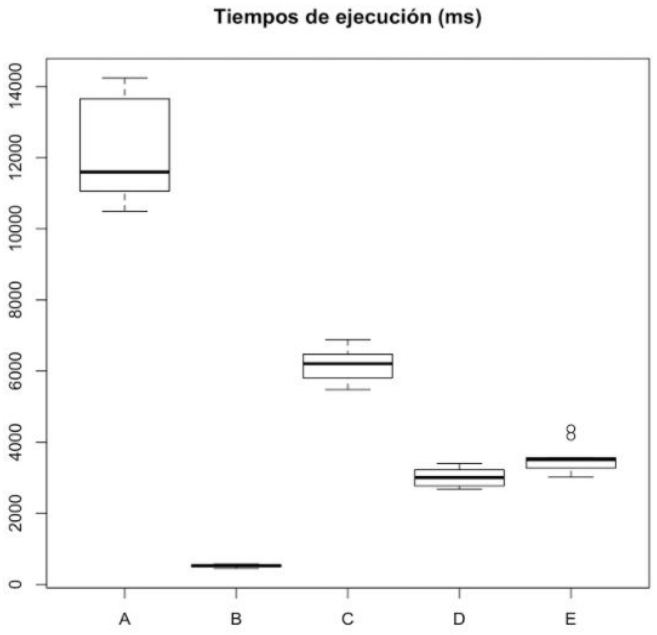
La puntuación P la calcularemos como la suma de los beneficios totales obtenidos por los pedidos atendidos (siguiendo la fórmula especificada), menos los costes de transporte de los camiones y la diferencia de ganancias que supone no atender las ordenes restantes.
Nuestra hipótesis nula es que no todos los conjuntos de operadores se comportan igual, creemos que el conjunto D será el mejor, ya que reduce lo suficiente el espacio de estados para obtener buenos tiempos de ejecución, pero una vez realizada la asignación de los pedidos permite reordenarlos de manera que se minimice el recorrido.

Para realizar el experimento, partiremos de una solución vacía en todos los casos. La realización del experimento será la siguiente: se escogerán 10 semillas de forma aleatoria. Para cada una de estas semillas, se hará una única ejecución con cada conjunto de operadores, dado que el algoritmo de Hill Climbing no es estocástico y por tanto obtendremos siempre los mismos resultados.
Después, compararemos los diferentes resultados obtenidos según los siguientes criterios:

- Una búsqueda es buena cuanto mejor es su puntuación
- Una búsqueda es buena cuando menor sea el tiempo de ejecución
- Una búsqueda es buena cuanto mayor sea la relación P/T

Después de realizar las ejecuciones, obtenemos los resultados mostrados en las siguientes gráficas para los parámetros descritos:





De las anteriores dos gráficas, podemos descartar varias de nuestras propuestas iniciales.

- La propuesta A resulta demasiado costosa en tiempo, así que decidimos no tenerla en cuenta. Además, podemos observar que los beneficios obtenidos son equivalentes a los de la propuesta C. Esto es debido a que la única diferencia en los estados generados por A es que permite eliminar pedidos asignados a un camión, cosa que el heurístico impide siempre, por lo que en definitiva nos lleva a los mismos estados. Podemos comprobar esta afirmación con el siguiente test de hipótesis:

```
data:  test1$BA and test1$BC
t = 0, df = 18, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -697.7301  697.7301
sample estimates:
mean of x mean of y
  94253.2   94253.2
```

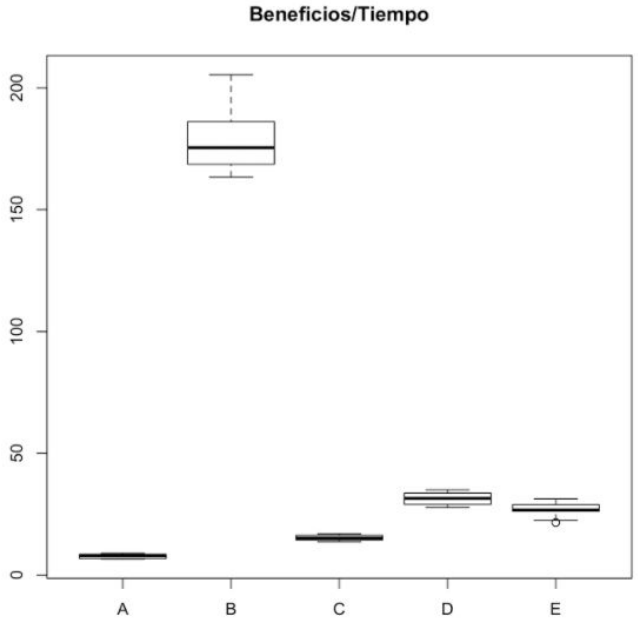
- La propuesta B es la que peores resultados nos ofrece, aunque con un coste temporal muy pequeño. No nos interesa este tipo de soluciones, ya que no dejamos que el algoritmo realice una búsqueda suficiente sobre el espacio.
- La propuesta C, pese a dar mejores resultados que D y E, sigue siendo demasiado costosa en el tiempo, puesto que contempla un espacio de búsqueda mucho mayor. Por tanto, decidimos descartarla también.
- Por último, entre las propuestas D y E está claro también cual nos resulta más conveniente, puesto que D obtiene mejores resultados tanto en puntuación como en

tiempo. Comprobaremos en los resultados de las pruebas de hipótesis que D nos da mejores beneficios que E, en un tiempo menor.

```
data:  test1$TD and test1$TE
t = -1.9199, df = 15.23, p-value = 0.0369
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
     -Inf -32.6936
sample estimates:
mean of x mean of y
   2900.9    3273.3

data:  test1$BD and test1$BE
t = 1.0575, df = 16.788, p-value = 0.1526
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -272.4527       Inf
sample estimates:
mean of x mean of y
  94240.4   93818.8
```

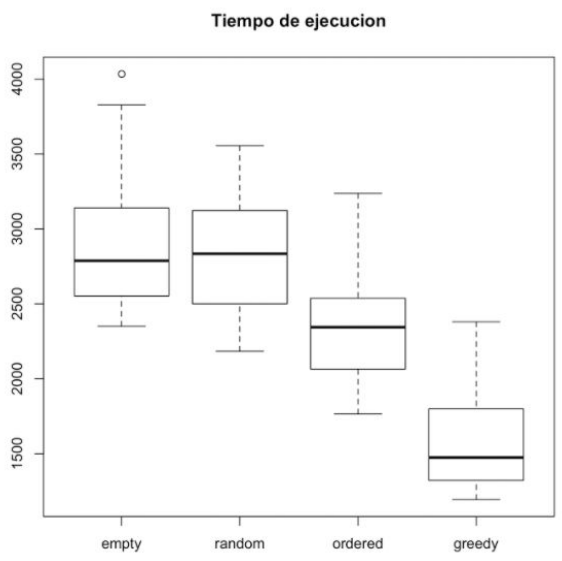
Para terminar de decidir, observaremos la siguiente gráfica:

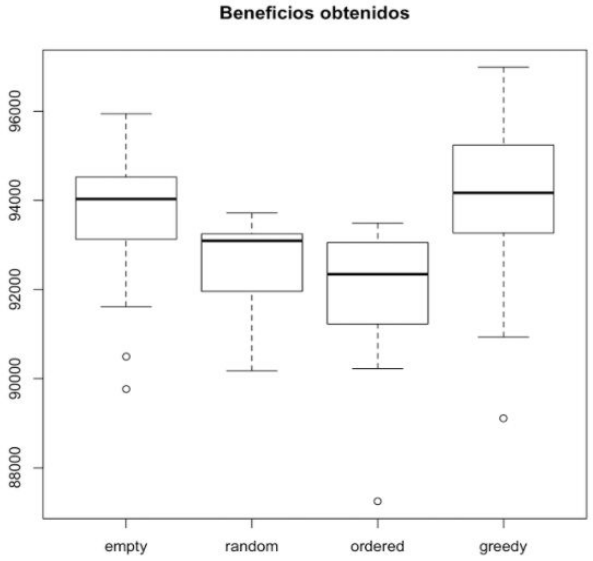


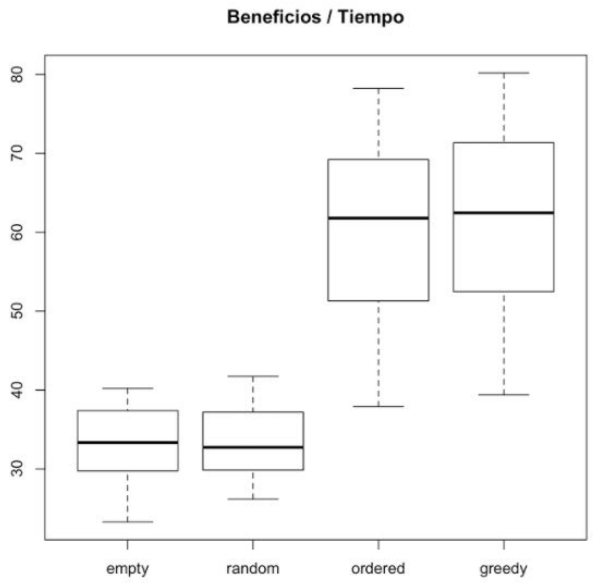
Podemos ver que el conjunto de operadores que ofrece mejores resultados es B, pero esto es debido a su excesivamente corto tiempo de ejecución, cosa que nos permite rechazarla en primer lugar. Después de esto, la opción que mejor resultado da en función de su tiempo de ejecución es D, por lo que nuestra decisión queda respaldada y decidimos quedarnos con este conjunto de operadores.

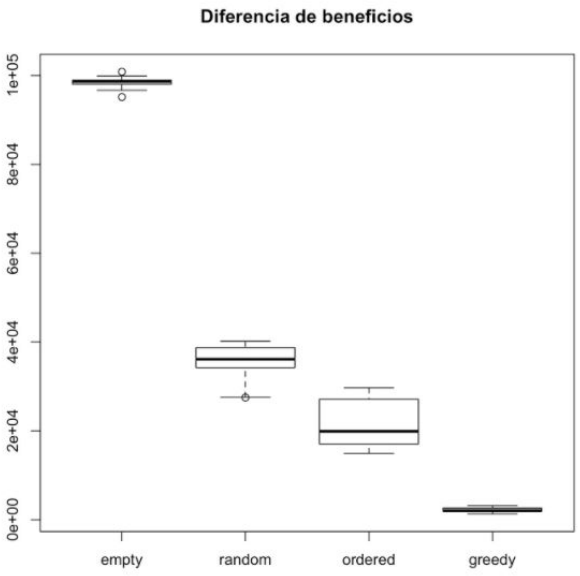
Experimento 2

En este experimento vamos a decidir cual de las posibles generaciones de solución inicial es la más correcta para el problema que tratamos, teniendo en cuenta su afectación en el beneficio final obtenido y el tiempo de ejecución del algoritmo.
Partiremos del escenario presentado en el experimento anterior, utilizando la misma función heurística y el conjunto de operadores D.
La hipótesis nula para este experimento es que la solución inicial vacía dará los mejores resultados, ya que, como no se permite el cambio con pedidos no asignados, introducir un pedido malo hará que no se pueda llegar a la ordenación óptima.
Procederemos de forma parecida al experimento anterior, también. Realizaremos 10 pruebas con 10 semillas generadas de forma aleatoria, y lanzaremos el algoritmo una vez para cada estrategia de generación y cada semilla.
Los resultados del experimento son los siguientes:









Podemos observar, de los datos obtenidos, que la solución inicial greedy es la que en teoría obtiene unos mejores resultados. Sin embargo, el trabajo realizado por el algoritmo es casi nulo, como podemos observar en la gráfica de Diferencia de beneficios. Este tipo de solución no nos interesa, ya que queremos que el algoritmo recorra el espacio de búsqueda.
El siguiente que obtiene mejores resultados de Beneficio en función del tiempo es el que empieza con una solución ordenada. No tarda tan poco tiempo como el greedy, pero aún así tarda bastante poco. Sin embargo, las soluciones son bastante peores que las obtenidas mediante la inicialización greedy o vacía.
Después tenemos la inicialización random y la vacía, aproximadamente empatadas en la relación beneficios/tiempo, pero podemos observar que, en relación con la solución vacía, obtenemos peores resultados en media.
Por tanto vamos a optar por la solución vacía, que ofrece en media mejores resultados que el resto de opciones excepto greedy, pero ya lo hemos descartado.

Como hemos podido observar, se cumple nuestra intuición de la hipótesis nula.

Experimento 3

En este experimento trataremos de discernir qué parámetros podemos usar para que al usar el algoritmo “Simulated Annealing” obtengamos el mejor resultado posible en relación al beneficio y al tiempo de ejecución (tratando de maximizar el primero y minimizar el segundo en la medida de lo posible).

El algoritmo de Simulated Annealing tiene 4 parámetros a considerar para su correcto funcionamiento, que será sobre los que trabajaremos:

- **Iteraciones:** Número de iteraciones que va a ejecutar el algoritmo.

- **Iteraciones por cada cambio de temperatura:** Número de iteraciones en los que mantendremos constante la probabilidad de saltar a un sucesor peor. Cada vez que se cumple este número de iteraciones la probabilidad de aceptar un sucesor peor decrece.

- **k:** Parámetro que afecta a la probabilidad de aceptar un sucesor peor. A mayor k, más tardará en decrecer la probabilidad de aceptar un sucesor peor.

- **Lambda:** Parámetro que afecta a la probabilidad de aceptar un sucesor peor. A mayor lambda, menos tardará en decrecer la probabilidad de aceptar un sucesor peor.

De partida, es complicado saber cuáles serán los parámetros que más beneficiarán a la búsqueda, por lo que no daremos una estimación inicial sobre qué parámetros funcionarán mejor.

Dicho lo anterior, el análisis consistirá en realizar búsquedas con Simulated Annealing sobre los mismos centros de distribución y gasolineras, mientras cambiamos gradualmente los valores referentes a k y lambda. Sobre el número de iteraciones, lo que haremos será dar un valor lo bastante grande como para asegurarnos que la probabilidad de aceptación de un estado peor cae a 0 antes de terminarse la ejecución del algoritmo. Finalmente reduciremos este número de iteraciones al que sea necesario. Tener en cuenta que debido a la aleatoriedad del Simulated Annealing realizaremos las búsquedas varias veces con los mismos parámetros (10 veces), y después haremos la media de los resultados.

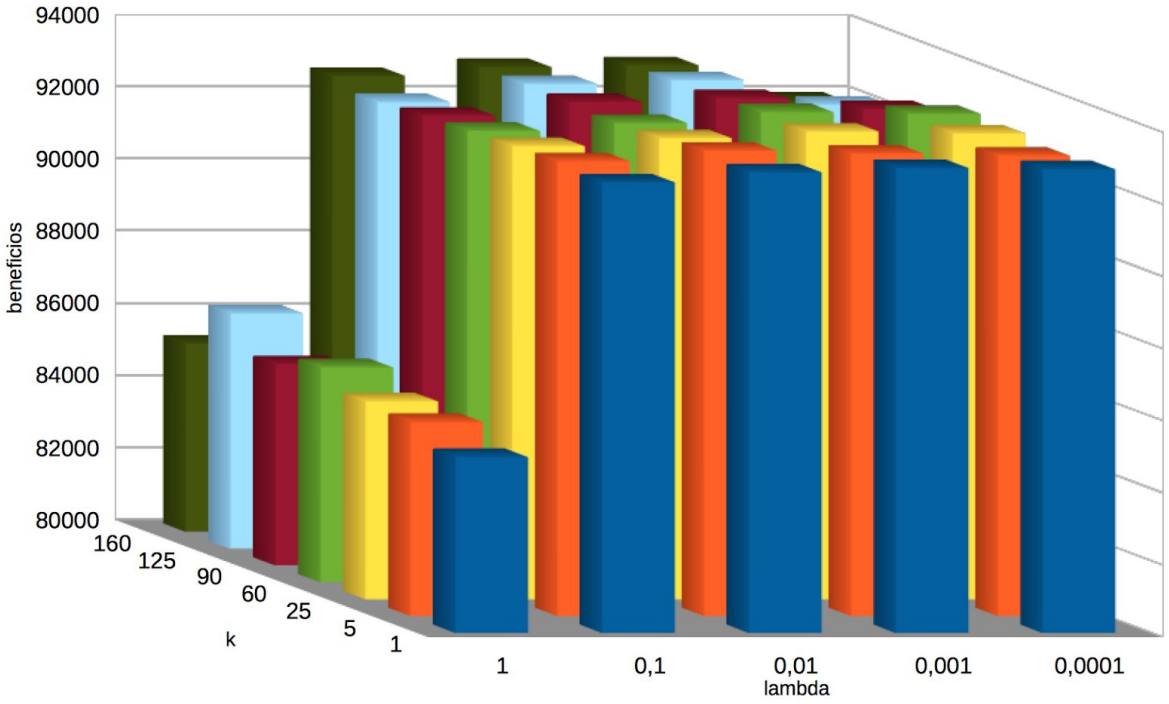
Las características del problema sobre el que experimentaremos serán las mismas que las del experimento 8:

Número de centros de distribución	10
Multiplicidad de centros de distribución	1
Número de gasolineras	100
Número de repeticiones por grupo de valores	10

- Seed aleatoria para cada repetición de las búsquedas (dentro de una misma repetición la seed se mantiene)

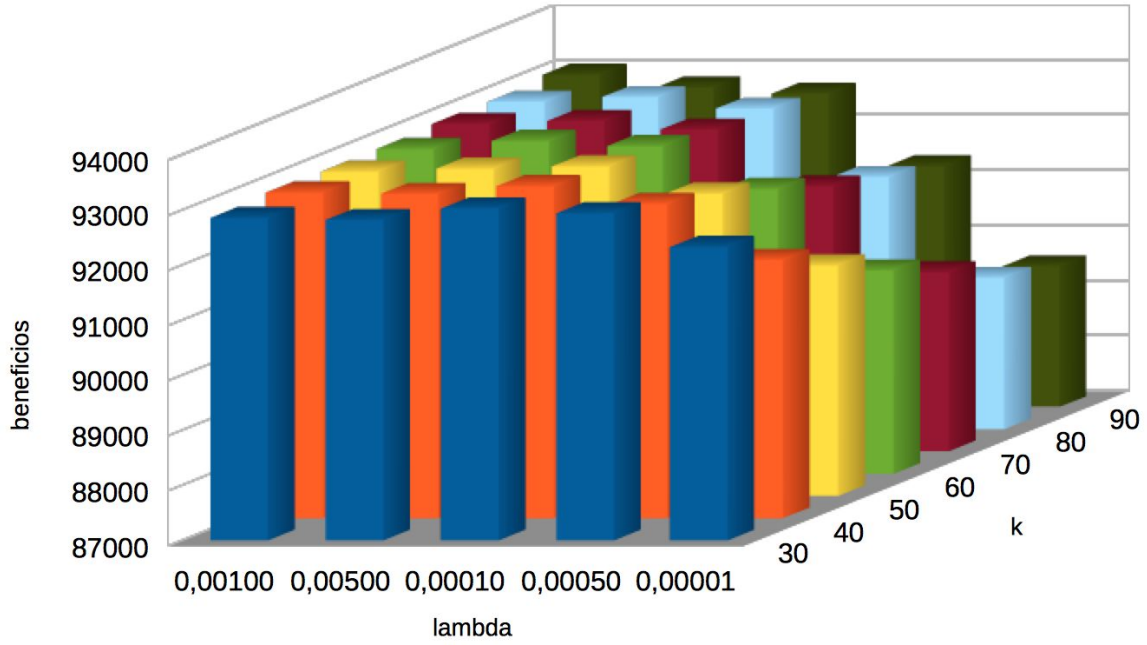
En cuanto a la función heurística, sucesores y distribución inicial usaremos las mismas que las expuestas en los experimentos anteriores.

Lo primero que haremos establecer un rango de valores para k y lambda lo bastante amplio como para poder entrever cuales dan las mejores soluciones. Los rangos fijados son k [1 - 160] y lambda [1 - 0,0001] (no es un rango continuo, sino que se usan variables discretas). Los resultados son los siguientes:



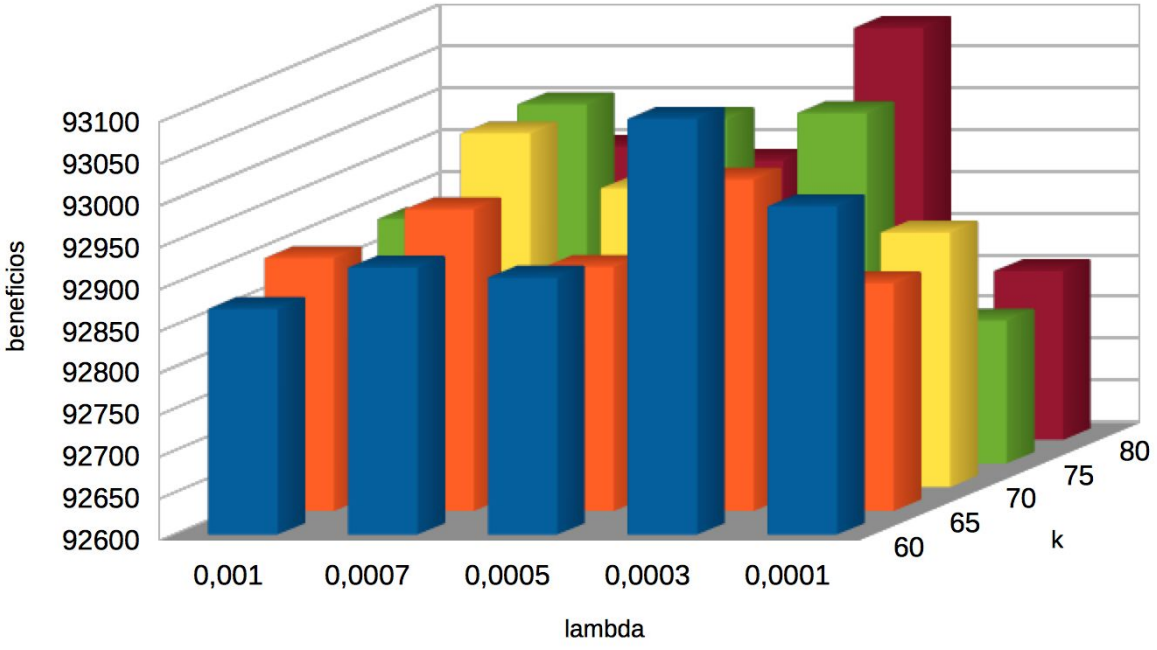
Se observa que los valores máximos surgen de unos valores de lambda de entre 0,01 y 0,001 y unos valores de k de entre 30 y 90.

Ahora, pues, pasaremos a realizar el mismo experimento para los rangos de lambda y k dichos en el párrafo anterior. Los resultados son los siguientes:



Se observa que los valores máximos surgen de unos valores de lambda de entre 0,001 y 0,0001 y unos valores de k de entre 60 y 90.

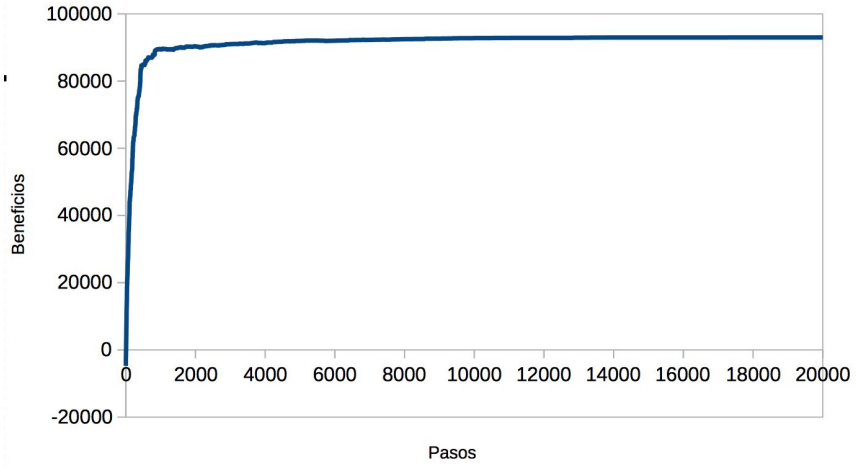
Finalmente, volveremos a realizar el mismo experimento para los rangos anteriores, y nos quedaremos los valores que nos den unos beneficios máximos. Los resultados són los siguientes:



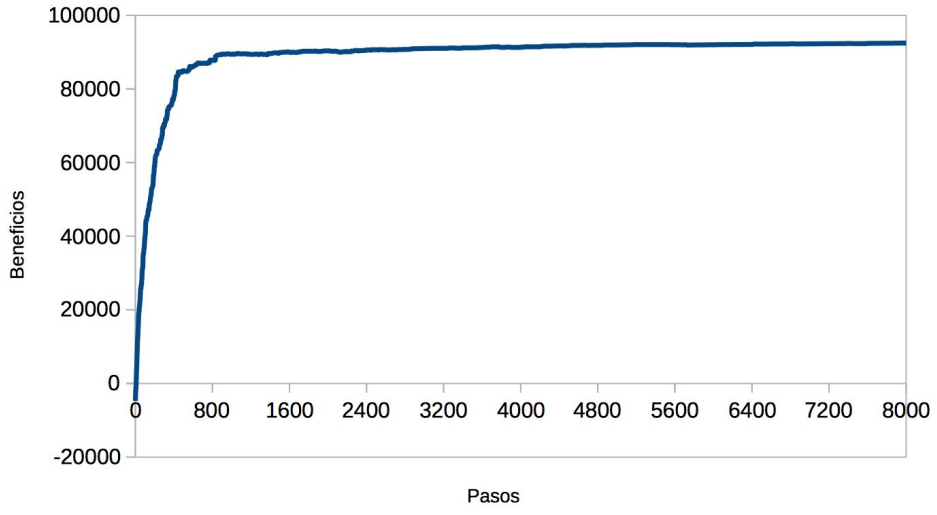
Se observa que los parámetros que dan mejores resultados son k = 60 y lambda = 0,0003. Hay que hacer énfasis para éstos valores se obtienen unos beneficios de 93000 aproximadamente, lo cuál es menos que con Hill Climbing. Sin embargo, el tiempo de ejecución es entre 5 y 7 veces menor. Tener en cuenta también que los tiempos de ejecución no se mencionan a la hora de discernir que valores de k y lambda son mejores debido a que los tiempos de ejecución obtenidos tenían diferencias despreciables.

Dicho esto, pasaremos a deducir cuál es el número de iteraciones necesario para que el algoritmo funcione correctamente sin comprometer el tiempo de ejecución.

Primeramente, si observamos los valores de k y lambda obtenidos anteriormente con un número de iteraciones mucho más grande del punto donde se dejan de aceptar sucesores peores obtenemos la siguiente gráfica:



Se observa que gran parte del tiempo de ejecución usado se está desperdiciando, ya que no se está mejorando sustancialmente el resultado. Para mejorar el tiempo de ejecución lo que podemos hacer es reducir el número de iteraciones. Reduciendo el número de iteraciones en un factor de 2.5 (obtenido a partir de la gráfica anterior) obtenemos lo siguiente:



En esta gráfica se observa que damos muchas iteraciones para que acepte soluciones mejores, mejorando sensiblemente el resultado hasta llegar a ser prácticamente constante. Es en éste punto en el que cortamos la ejecución del Simulated Annealing, ya que a partir de aquí seguir ejecutando iteraciones es gastar tiempo de ejecución sin obtener un resultado mejor.

Así que en conclusión, los parámetros que usaremos para Simulated Annealing serán los siguientes:

Número de iteraciones	8000
Iteraciones para cambio de temperatura	400
k	30
Lambda	0,0003

Experimento 4

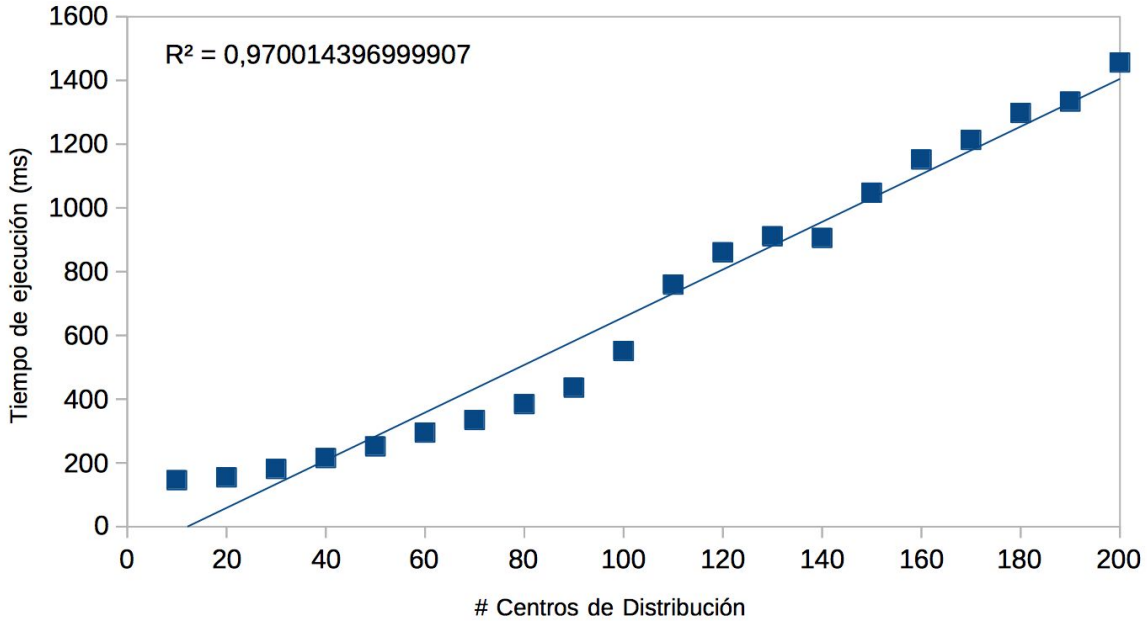
En este experimento se nos pide que observemos la tendencia en tiempo de ejecución de Hill Climbing y Simulated Annealing al aumentar el número de centros de distribución y gasolineras. Para ello, se van a ejecutar diferentes veces los dos algoritmos con seeds distintas por cada variación del número de centros de distribución y gasolineras, y después se hará la media de los valores obtenidos antes de analizar los resultados.

Más concretamente, empezaremos lanzando los algoritmos con 10 centros de distribución y 100 gasolineras, e iremos incrementando estos valores de 10 en 10 en cuanto a centros de distribución pero manteniendo una proporción de 10:100 en cuanto a centros de distribución y gasolineras. Tras cada aumento, lanzaremos 10 veces cada algoritmo, y trataremos la media generada a partir de cada una de las 10 iteraciones con el mismo número de gasolineras y centros de distribución.

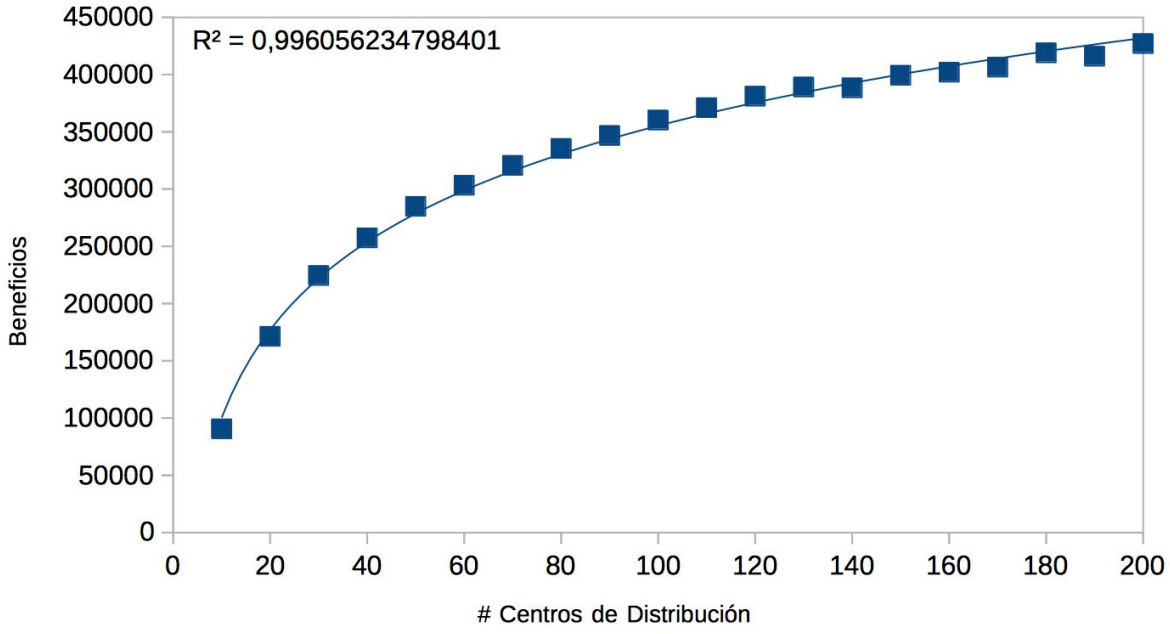
El resultado esperable para este experimento es que de media el tiempo de ejecución para Simulated Annealing crezca de forma lineal, ya que a cada iteración solo coge un sucesor, y el número de iteraciones es fijo. Ésto hará que lo único que influya en el tiempo de ejecución de la solución es el incremento del tiempo que tardan en generarse la solución inicial y los operadores. En cambio, para Hill Climbing es esperable que el tiempo de ejecución crezca de forma exponencial, ya que al aumentar el número de gasolineras y de centros de distribución se aumenta el factor de ramificación.

Tras lanzar los algoritmos, los resultados obtenidos son los siguientes:

Para Simulated Annealing:

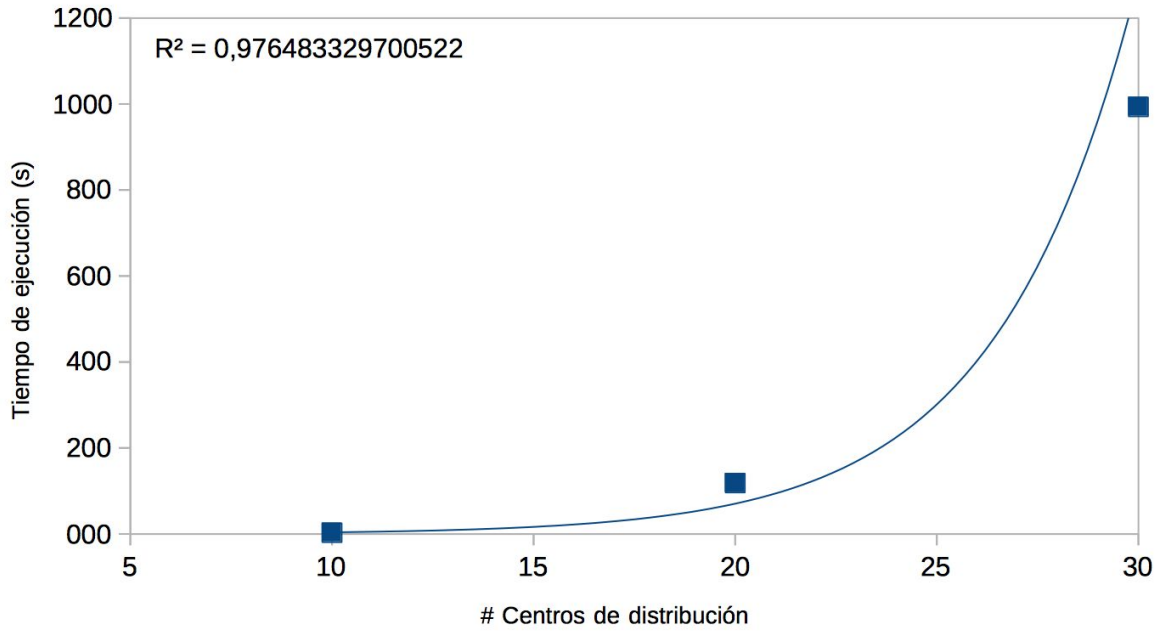


El coeficiente R^2 de 0,97 nos indica que el tiempo de ejecución aumenta de forma lineal respecto al número de centros de distribución (los puntos se adaptan a una recta de regresión lineal), tal y como apuntábamos en el inicio del experimento.
Además, cabe recalcar que a pesar de que el tiempo de ejecución aumente de forma lineal, los beneficios no lo hacen de la misma forma (aumentan de forma logarítmica), tal y como se observa en la siguiente gráfica:



Ésto creemos que es debido a que aunque estamos aumentando el número de gasolineras y centros de distribución no estamos aumentando el número de iteraciones de Simulated Annealing (las iteraciones han quedado fijadas en el experimento anterior) con lo cual no le estamos dando tiempo al algoritmo para que llegue a la solución más óptima.

Para Hill Climbing:



El coeficiente R^2 de 0,97 nos indica que el tiempo de ejecución aumenta de forma exponencial respecto al número de centros de distribución (el conjunto de puntos queda representado bajo una recta de regresión exponencial), tal y como apuntábamos en el inicio del experimento.

En el caso de Hill Climbing hay que recalcar que el hecho de que sólo haya valores para hasta 30 centros de distribución es debido a que a partir de 40 centros de distribución el número de sucesores que genera Hill Climbing es demasiado grande como para ser tratado por cualquiera de los ordenadores de los que disponíamos (en el caso de este experimento se intentó realizar el propio experimento en 3 ordenadores personales y un ordenador de la FIB, sin resultados satisfactorios).

En conclusión, comentar que a pesar de que en media Hill Climbing nos ha dado resultados sensiblemente mejores en cuanto a beneficios (que no en cuanto a tiempos de ejecución) para valores pequeños de gasolineras y centros de distribución, queda claro que Simulated Annealing es mejor que Hill Climbing conforme aumentamos el número de elementos del problema (Simulated Annealing crece de forma lineal en cuanto a tiempo de ejecución mientras que Hill Climbing lo hace de forma exponencial).

Experimento 5

En éste experimento se nos plantea un nuevo posible escenario, donde el número de centros de distribución se ha reducido a la mitad. Se quiere mantener la cantidad de camiones de reparto por lo que cada centro de distribución tendrá dos camiones (multiplicidad 2). El objetivo es estudiar la influencia de este cambio en los beneficios obtenidos y en el número de kilómetros adicionales que se recorren en total. Adicionalmente estudiaremos si los cambios de ambos valores están directamente relacionados o hay otros factores que cambian.

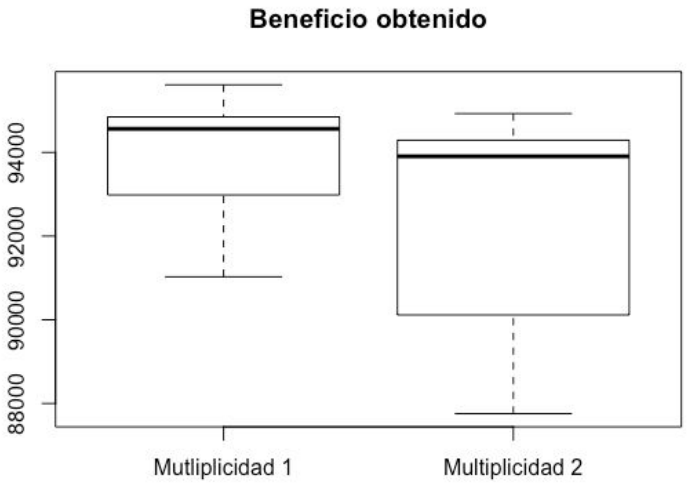
Para realizar el estudio se realizaremos diversas repeticiones del experimento (10 repeticiones). En cada repetición se ejecutará una búsqueda local, con el algoritmo de Hill Climbing y el heurístico, estado inicial y operadores fijados en los experimentos 1 y 2. La búsqueda se lanzará 2 veces: la primera sobre estado generado con multiplicidad de camiones 1, 10 centros de distribución y 100 gasolineras; la segunda sobre el estado generado con multiplicidad de camiones 2, 5 centros de distribución y 100 gasolineras. El valor de la semilla será el mismo para las 2 búsquedas de una repetición y se generará aleatoriamente para cada repetición.

Los resultados del experimento muestran una disminución, en media, de los beneficios obtenidos cuando reducimos los centros de distribución a la mitad (multiplicidad de camiones 2). También observamos un augmento de la varianza de los resultados.

En concreto:

Cuando la multiplicidad es 1	Beneficios medios = 93970 y desviación estandard = **1468,266**
Cuando la multiplicidad es 2	Beneficios medios = 92650 y desviación estandard = **2491,585**

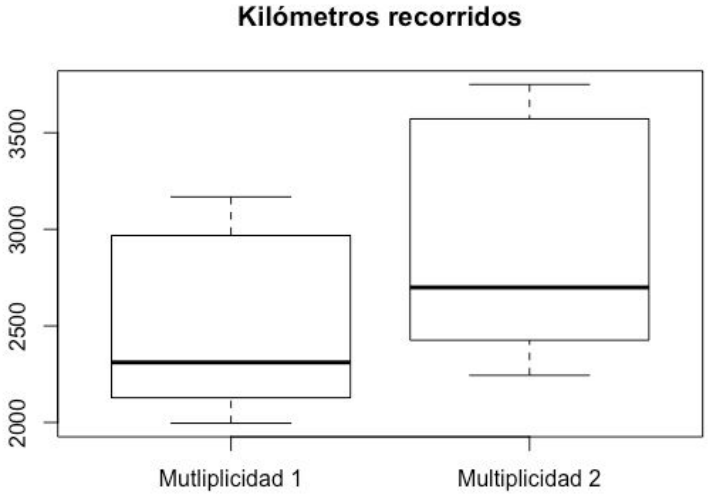
Estos resultados se pueden observar gráficamente a continuación.



Paralelamente se observa, en media, un augmento del número de kilometros recorridos al reducir el número de centros de distribución.

Cuando la multiplicidad es 1	Kilometros totales recorridos medios = 2494 y desviación estandard = **449,3205**
Cuando la multiplicidad es 2	Kilometros totales recorridos medios = 2898 y desviación estandard = **578,7684**

Como podemos observar a continuación.



Al plantear el experimento nuestra hipótesis era que el beneficio disminuiría el doble de lo que incrementara el número de kilómetros recorridos en total, ya que cada kilómetro adicional tiene un coste de 2 (por lo tanto disminuye en 2 los beneficios). Además el incremento de kilómetros también era esperable, debido a que las gasolineras son las mismas y disminuir los centros de distribución implica alejarse de alguna de las gasolineras.

Los resultados muestra la tendencia que esperábamos, pero la relación entre el aumento de kilometraje y el doble de disminución del beneficio no es exacta. La diferencia de medias del beneficio es de 1320 mientras que la de distancia recorrida es 404. Pero el doble de esta (valor 808) es más pequeño que el cambio de beneficios.
Esto ligado con el aumento de la variabilidad nos indica que hay peticiones, que debido a la disminución del número de centros de distribución, en algunos casos (en algunas de las repeticiones) no pueden ser atendidas provocando una mayor diferencia entre los beneficios.

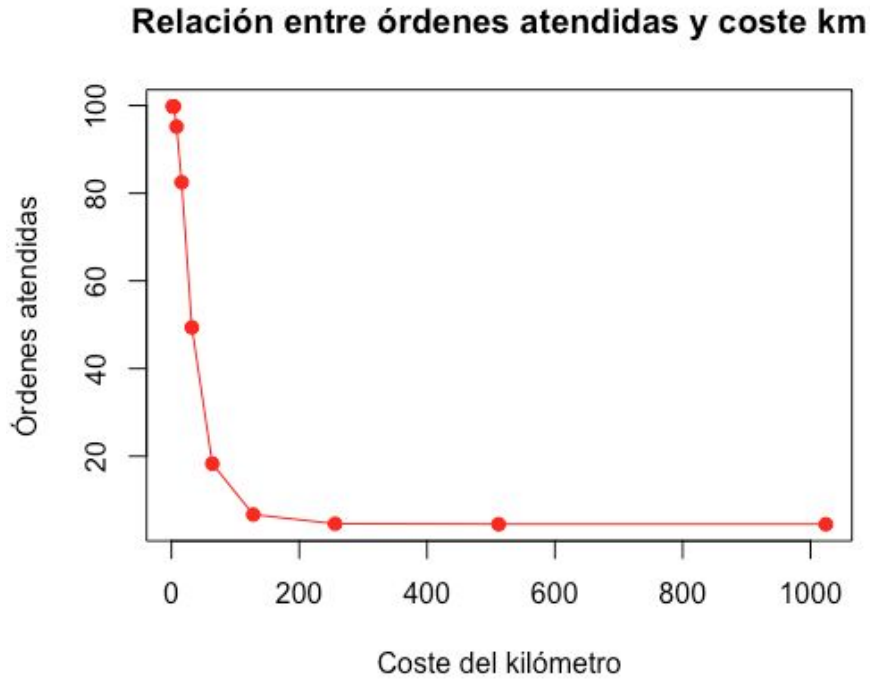
Hay que considerar también que probablemente el coste real de mantener un centro de distribución abierto es más elevado que las pérdidas que te pueda reportar no tenerlo, ya que respecto a los beneficios totales la diferencia detectada no es tan grande.

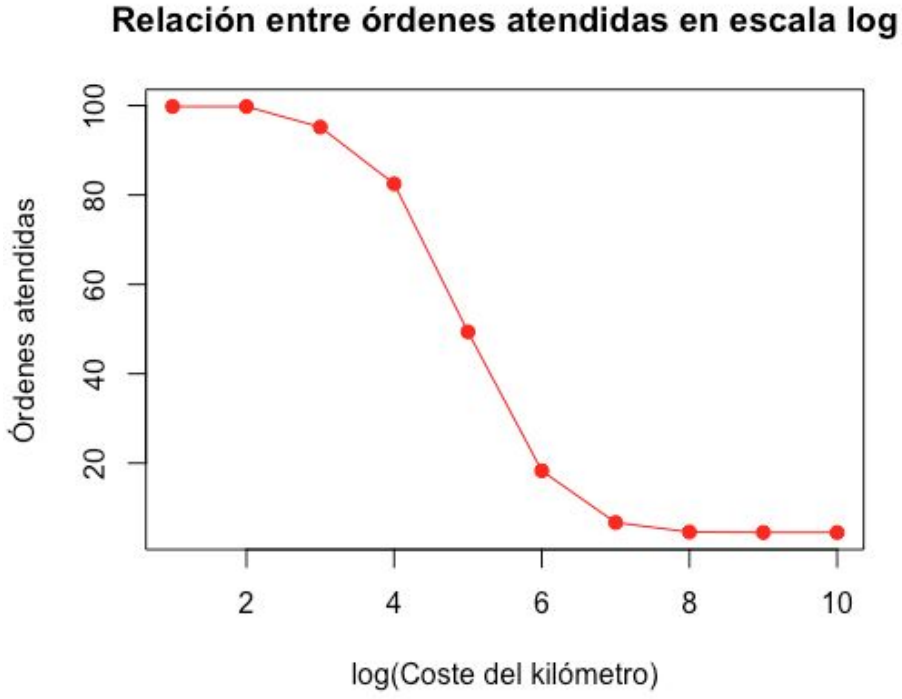
Experimento 6

El objetivo del experimento es estudiar la influencia del coste de recorrer un kilómetro, en el número de peticiones atendidas. Adicionalmente se pide observar si la proporción de peticiones atendidas según los días que llevan pendientes se ve afectada al modificar el coste de los kilómetros.

Para analizar estos factores vamos a plantear un experimento que se repetirá 10 veces y se trabajarán los datos a partir de las medias. En cada repetición se fijará un escenario a partir de una semilla aleatoria, y se ejecutará la búsqueda local en 10 ocasiones, en cada ejecución se dobla el coste del kilómetro (entre 2 y 1024). El heurístico, operadores y estado inicial son los que se fijaron en los experimentos 1 y 2. Tras cada búsqueda local, se almacenarán los datos de el número de peticiones atendidas totales, y la proporción (órdenes atendidas el día “j” / total de órdenes del dia “j”), consideramos los días [0..3]. Es decir, para cada precio del kilómetro almacenamos 5 valores (un valor total y cuatro proporciones, una por día). Se calcularán las medias de los resultados obtenidos para cada repetición y se analizarán los datos a partir de las medias.

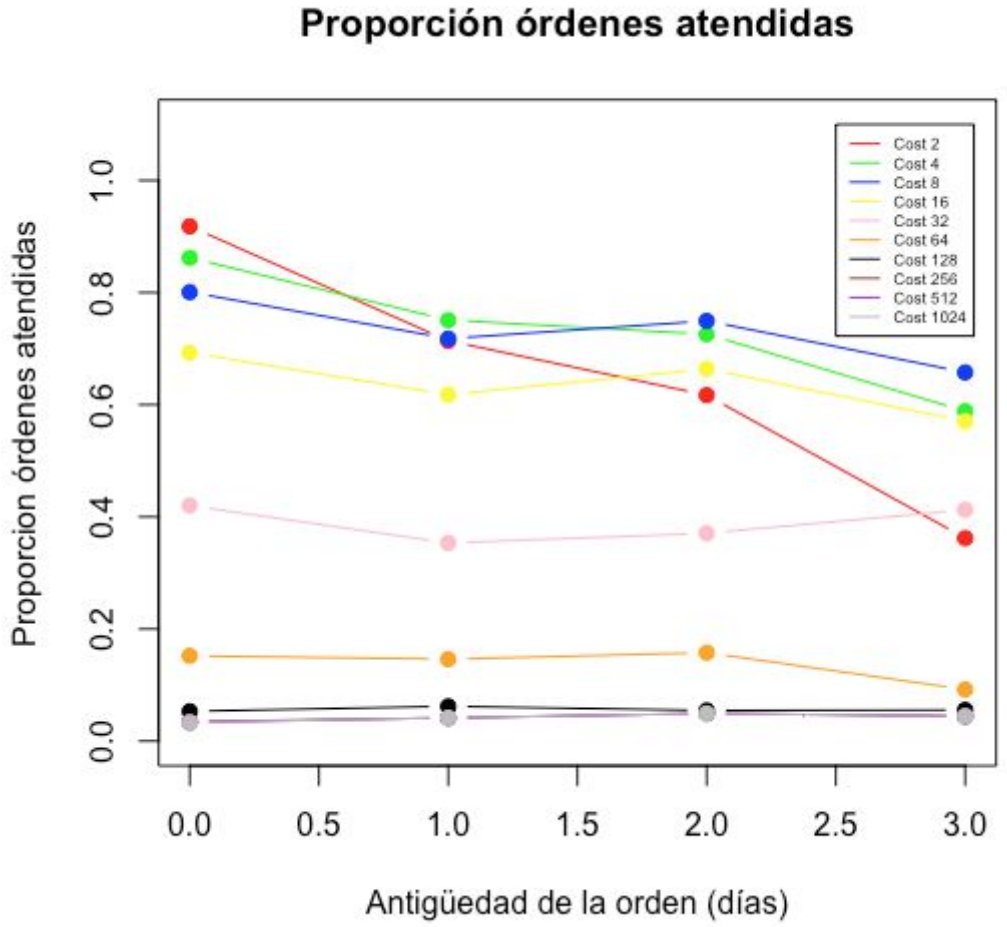
Por una parte, podemos observar que el número de órdenes atendidas se ve reducido a medida que doblamos el coste de los kilómetros recorridos. Esta disminución no es lineal, sino que para los valores en que el coste es bajo se mantiene constante, a continuación hay un decrecimiento exponencial, hasta un punto en el que el número de órdenes atendidas pasa a ser constante de nuevo y prácticamente vacío. Podemos observar este patrón en los gráficos siguientes, en el primero se muestra la tendencia según el coste, y en el segundo se ha expresado el eje x en escala logarítmica para ver si en algún tramo se sigue una disminución lineal que corrobora el comportamiento exponencial.





Por otra parte, podemos observar que la proporción de peticiones atendidas según la antigüedad de la petición, no se comportan de la misma manera para costes bajos que altos. Cuando los costes de kilómetros son bajos, se atiende con más frecuencia peticiones más recientes, mientras que para para costes de kilómetros altos esta proporción muestra equiprobabilidad. Esto se puede ver gráficamente a continuación, en el gráfico también

observamos que las proporciones entre costes vans disminuyendo, consecuencia lógica de que el número total de peticiones atendidas disminuya.



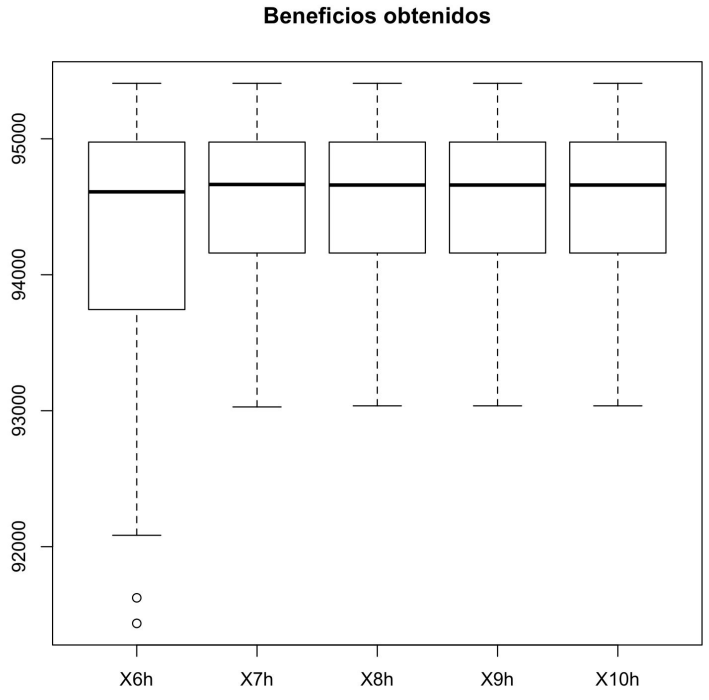
Nuestra hipótesis, antes de realizar el experimento, era que efectivamente el número de peticiones atendidas disminuiría si aumentamos los costes de desplazamiento, ya que si el coste aumenta se reduce el beneficio y hay peticiones que no compensa atenderlas (solo aportan más pérdidas). Que este cambio se produzca de forma exponencial es también bastante lógico porque a partir de un cierto coste de kilometraje es donde ese valor superara los potenciales beneficios de atender la petición (para la distancia media recorrida).

Para el segundo objetivo de estudiar la influencia del coste en la proporción de órdenes atendidas según su antigüedad, no esperábamos una relación entre los aspectos. Los resultados, en cambio, muestran que la proporción de órdenes atendidas para costes bajos es linealmente decreciente según la antigüedad, mientras que para costes de kilometraje altos la proporción es constante, es decir la antigüedad de la petición no influye en la probabilidad de ser atendida. Una vez analizados los resultados y razonado sobre ellos, entendemos que la influencia que hay del coste del kilometraje es cada vez mayor.

Para costes bajos, la distancia no es tan influyente en el beneficio final, como atender primero una petición más reciente; ya que una petición reciente (antigüedad 0) aporta más beneficios (estos disminuyen con el aumento de días).
Para costes altos en cambio, la influencia de recorrer más kilómetros tiene un impacto negativo mayor en los beneficios de la petición, que el impacto que pueda generar que la petición sea más antigua. Es por este motivo que observamos un cambio de comportamiento al aumentar el precio de recorrer un kilómetro.

Experimento 7

En este experimento se nos pide comprobar qué efecto tendría aumentar o disminuir el horario de trabajo de los camiones. Nuestra intuición nos dice que aumentar el número de viajes no va a afectar en el beneficio obtenido si no aumentamos también el número de viajes máximo, dado que para maximizar el beneficio queremos recorrer el mínimo número de Km posible, por lo que la capacidad de recorrer más es inútil.
Sin embargo, creemos que al disminuir el horario de trabajo afectaremos negativamente al beneficio, ya que es posible que algún viaje que antes se realizaba se quede sin atender.
Para proceder con el experimento, realizaremos 10 repeticiones con distintas seeds generadas de forma aleatoria. En cada repetición lanzaremos una ejecución con los distintos horarios que vamos a estudiar, desde 6h hasta 10h,es decir, aumentamos y disminuimos en una y dos horas el tiempo de trabajo.
De cada ejecución hemos recogido los datos de los beneficios obtenidos, y los resultados son los siguientes:



Como podemos observar, los resultados son tal y como esperábamos. Al aumentar el tiempo de trabajo, ya sea un una o dos horas, no obtenemos ninguna diferencia en el beneficio obtenido. Podemos comprobarlo para el aumento de una hora con el segundo test

de hipótesis presentado a continuación, en que el p-valor es de 0.99, así que los resultados son casi idénticos.

Para nuestra sorpresa, como podemos ver en el primer test, el p-valor del test al disminuir una hora es algo menor que al aumentarla, es decir, que hay una ligera diferencia pero es despreciable, y no tenemos por qué dudar en que las medias son iguales.

Creemos que esto es debido a que, como nuestro heurístico hace que siempre minimicemos el recorrido, en media desaprovechamos tiempo de trabajo, es decir, no llegamos a recorrer el máximo que tenemos con las 8h de las que partíamos, así que al disminuirlo ligeramente podemos seguir realizando casi los mismos viajes.

Sin embargo, podemos observar una diferencia notable al disminuir el tiempo en dos horas, por las razones que comentábamos anteriormente: no seremos capaces de introducir todos los viajes en el período de trabajo.
Por lo tanto, es razonable pensar que la jornada de trabajo óptima sería la de 7h, ya que la disminución del beneficio es despreciable, pero seguramente el sueldo que se tenga que pagar

```
data:  test7$X7h and test7$X8h
t = -0.15786, df = 57.634, p-value = 0.8751
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -335.6729  286.6062
sample estimates:
mean of x mean of y
 94500.53  94525.07
```

```
data:  test7$X9h and test7$X8h
t = -0.011632, df = 58, p-value = 0.9908
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -300.0247  296.5580
sample estimates:
mean of x mean of y
 94523.33  94525.07
```

Conclusiones

Un vez finalizada la experimentación, podemos concluir que el trabajo ha sido de una gran utilidad. Nos ha permitido transformar todo el conocimiento teórico en un ejemplo práctico, clarificando conceptos y poniendo de manifiesto por ejemplo las diferencias entre los algoritmos de Hill Climbing y Simulated Annealing.

Además, nosa permitido clarificar cómo trabajan estos algoritmos y cómo aplicarlos en el mundo real anti situaciones reales, y ser conscientes de las limitaciones que tienen por complejidad espacial y temporal.

Finalmente, la experimentación nos ha permitido razonar sobre resultados y hemos aprendido a deducir y analizar la información que nos aporta un resultado, para detectar tendencias, comportamientos variables según el tamaño de las muestras… nada más lejos de la realidad de un problema complejo que en un futuro quizá tenemos que afrontar.

Por todo ello nos ha parecido una práctica muy enriquecedora y valoramos positivamente la realización de esta.