Inteligencia Artificial

Práctica de Búsqueda Local

Documentación

Curso 2021/2022 1Q

Antonio Calvín

Daniel Muñoz

Cristian Sanchez

Grau en Enginyeria Informàtica - UPC

Departament de Ciències de la Computació

Identificación del problema

En una área geográfica de 100x100 km2, una compañía de distribución de carburantes tiene la tarea de: dado un número de ***n*** centros de distribución y de ***c*** camiones, atender las ***p*** peticiones de reabastecimiento de una cantidad de ***g*** gasolineras, con el objetivo de obtener el máximo beneficio posible.

Por un lado, cada gasolinera puede tener varias peticiones, y cada una de ellas se encuentra caracterizada por el número de días **d** que lleva sin ser atendida (factor que determinará crucialmente el precio de la misma), y en caso de serlo, sólo un camión puede ser el responsable de dar respuesta a la misma. Por otro lado, cada centro de distribución puede tener 1 o más camiones en activo, hecho que viene dado por la **multiplicidad** de los centros de distribución; es decir, la **multiplicidad** nos determinará cuántos camiones se tienen por centro de distribución.

Para determinar el precio de cada petición, se tienen en cuenta el número de días desde que la misma ha sido comunicada, y responde a la fórmula :

$$\%precio = (100 - 2^{días})\%$$

Cabe destacar la particularidad que, en caso de que la petición lleve 0 días sin ser atendida, el precio corresponde al 102% de las ganancias que se pueden generar atendiendola.

Vale la pena realizar una breve pausa para definir bien las diferencias entre precio, ganancia, gasto y beneficio:

1. El precio corresponde al porcentaje determinado por el número de días que una petición ha estado activa.
2. La ganancia es el producto del precio de la petición por el valor de un depósito de gasolina
3. El gasto es el producto de los kilómetros recorridos por el coste del kilómetro.
4. El beneficio es la diferencia entre la ganancia y el gasto.

El problema presenta una serie de restricciones:

- Los camiones viajan a 80km/h durante 8 horas al día, que supone un máximo de 640 km diarios.
- Una gasolinera puede tener máximo dos depósitos, con lo cual tiene un máximo de dos peticiones activas.
- Los camiones tienen una capacidad de carga de 2 depósitos por viaje.
- Un camión pertenece unívocamente a un centro de distribución.
- Un viaje corresponde a servir una o dos peticiones (que pueden ser de gasolineras diferentes) y volver a su centro de distribución.
- Un camión puede realizar un máximo de 5 viajes diarios.
- El valor del depósito es de 1000 unidades.
- Las características de los centros de distribución (ubicación y nº de camiones) y las gasolineras (ubicación, nº de peticiones y los días que llevan activas) dependen de unas semillas de generación (una para los centros y otra para las gasolineras) previamente indicadas.

En cuanto a la distancia entre centros y gasolineras, tenemos que cada uno de estos establecimientos tienen unas coordenadas **X** e **Y**, y la manera de determinar la distancia entre ellas se corresponde a aplicar la siguiente fórmula:

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

Asimismo, hay un coste implícito de 2 unidades por kilómetro recorrido, con lo cual implícitamente existe la posibilidad de generar pérdidas si no se eligen bien las peticiones a atender.

Con esto, podemos plantear el escenario básico del problema: dados **n** centros de distribución, con **c** camiones (donde **c = n*multiplicidad**), con **g** gasolineras y **p** peticiones por atender (que llevan **d** días en activo), queremos determinar una ruta óptima para atender las peticiones existentes, obteniendo así el máximo beneficio posible, minimizando a la vez la distancia total recorrida (gasto) y la ganancia de las peticiones que no se atienden. Respecto a esto último, se asume que dichas peticiones sin atender serán atendidas mañana, con lo cual se trata de equilibrar la proporción de peticiones recientes resueltas respecto las que llevan más tiempo en activo

Una vez tratado el estado de la cuestión, consideramos que se trata de un problema de búsqueda local. En primer lugar, nos encontramos tratando de maximizar y minimizar una serie de parámetros del problema (maximizar beneficios y minimizar gasto/coste), con lo cual podemos hacer uso de algoritmos que hayan sido diseñados para este mismo fin. En segundo lugar, partamos de la premisa que, efectivamente, se trate de un problema de búsqueda local: podemos convencernos que el espacio de posibles soluciones se encuentra formado por una secuencia de movimientos para cada camión, y que por lo tanto una solución será una asignación de un subconjunto **p'** de peticiones (donde **p'**$\subseteq$ **p**) para los **c** camiones. En base a esto, y partiendo del hecho que no hay restricciones respecto a lo que podría ser una solución (no hay indicaciones sobre un coste máximo, unas ganancias mínimas y viceversa), se pueden plantear estados solución tan sencillos como el estado vacío (donde, para los **c** camiones, no hay asignación alguna; en este caso, no tendríamos beneficios ni gastos, que de alguna manera se puede interpretar como una maximización y minimización, pero sí tendríamos bastantes peticiones sin atender y, por lo tanto, ganancias sin registrar; en definitiva, sería una mala solución inicial, puede que la peor, pero sería una válida dada la naturaleza del problema).

Entonces, partiendo de nuestro objetivo y de la capacidad de representar las condiciones que dan lugar a un estado solución, podemos afirmar que es posible realizar una representación de un estado solución inicial (con sus estructuras de datos y mecanismos pertinentes).. Además, en base a la posibilidad misma de representar toda esta información, también somos capaces de valorarla: dado el tipo de restricciones y sus evidentes efectos (buscamos maximizar valores numéricos, y como tal solo necesitamos hacer uso de las operaciones más básicas para manipularlos y poder sacar conclusiones en base a ellos), podemos realizar evaluaciones de la calidad de cada resultado.

Además, consideraremos como solución todo estado que contenga una ruta cronológica para cada camión, y donde dicha ruta respeta las restricciones del problema.

Considerando que somos capaces de representar el problema en un estado, que el estado formará parte del espacio de posibles soluciones, que tenemos mecanismos para valorar su calidad mediante una heurística, y que tratamos de maximizar y minimizar un conjunto de parámetros, afirmamos que se trata de un problema de búsqueda local.

Estado del problema y representación

Un estado debe concentrar la mínima cantidad de información necesaria para poder representar el problema. En nuestro caso, tenemos los **CentrosDistribucion** y **Gasolineras**, que representan el conjunto de centros de distribución y gasolineras presentes en el problema. Estos atributos se caracterizan de manera estática para que no se repliquen al crear otro estado y solo se guarda un puntero al objeto, ya que estos no se modificarán y nos permitirán poder saber en todo momento en que posición se encuentra cada gasolinera y cada centro de distribución. Además de estos atributos, también tenemos las variables estáticas **maxKm** (máximo de kilometros que puede recorrer cada camión en un día), **maxViajes** (máximo de viajes que puede recorrer un camión en un día), **precioDeposito** (valor de un depósito de gasolina) y **costeKm** (coste por kilómetro). Además, inicialmente les hemos dado los valores que se les asigna en el enunciado.

Para poder saber qué rutas hará cada camión en un día, tenemos el atributo **truckTrack**. Este atributo debe guardar el orden de las peticiones de las gasolineras a las que viaja, pero también debe indicar que vuelve a un centro cada ciertas peticiones servidas (entre 1 y 2). Para ello, hemos hecho uso de un objeto Pair, en el cual el segundo valor nos indica si es un centro (se representa con un -1) o, en caso contrario, una petición (indica los días que lleva sin ser servida). Además, el primer valor nos indica el índice del centro o gasolinera al que pertenece (este dependerá de lo que indique el segundo valor). Por lo tanto, este atributo será un Vector<Vector<Pair>>, en el cual el vector externo representará el camión i-ésimo y el interno el recorrido de dicho camión. Dado que esta estructura puede modificarse en cualquier parte del recorrido, y se consulta muy a menudo, hemos decidido usar estas estructuras de datos, dado que son de fácil acceso, fácil consulta y con mecanismos relativamente sencillos para realizar eliminaciones e inserciones, que serán las operaciones que más usaremos.

Para saber si se cumple la restricción de los viajes, tenemos **truckInfoViajes** que, para cada camión, indica el número de viajes que lleva. Asimismo, para saber si cumple la restricción de que no se pase del máximo de kilómetros, tenemos **truckInfoKm**, que es idéntica a la de los viajes y en la cual, para cada camión, guarda el número de kilómetros que lleva recorridos hasta el momento. Hacerlo de esta manera nos permite ir modificándolo dinámicamente, y además esto será más eficiente que calcular los viajes o kilómetras que lleva, ya que, para esta representación, el coste de consulta es constante, mientras que la alternativa sería ir recorriendo su **truckTrack**, con coste O(g*c) donde **g** es el número de gasolineras y **c** el número de camiones. La desventaja de hacer esto es que

gasta más memoria por cada estado. Además, hemos optado por usar un Array de Integers, ya que las consultas son constantes, su tamaño tiene coste en memoria O(c), y si se necesita hacer una modificación, también será constante.

Como para cumplir la restricción de que no pase los máximos kilómetros,se necesita saber la distancia de los centros a las gasolineras y desde las gasolineras a las gasolineras, hemos planteado dos matrices estáticas: **distCentro2Gas** y **distGas2Gas** respectivamente. En la primera, el coste en espacio de memoria sería de O(C*g) y en la segunda O(g^2), donde g son las gasolineras y C son los centros. Con esto, lo que conseguimos es una ganancia en memoria, ya que solo se tendrá una instancia común de dichas estructuras para cada objeto estado generado, y, además, las consultas de las distancias se realizan en tiempo constante.

Por otra parte, para generar los estados iniciales y para el heurístico, utilizamos una estructura de datos para almacenar peticiones (llamada **peticiones**), donde estan todas las peticiones no atendidas de las gasolineras, y donde cada elemento del vector tenemos un Pair en el que el primer valor seria el índice a la posición del vector gasolineras, mientras que el segundo será el nº de días que dicha petición lleva en activo.

Asimismo, para ahorrar tiempo de ejecución a la función heurística, tenemos 4 atributos más.

- **gananciasTotales** y **gastosTotales**: El primero son las ganancias totales registradas por el estado actual hasta el momento, y el segundo representan los gastos totales que lleva el estado actual hasta el momento. Al hacer esto, el coste de consulta es constante, con lo cual no hay que calcularlo en la función heurística, cálculo que tendría un coste de O(g*c), donde **g** serían las gasolineras y **c** los camiones. Además, los **gastosTotales** también se usan para que los operadores se ejecuten de manera más rápida, sobre todo en el caso particular del swap, ya que debe calcular las distancias nuevas y puede actualizarlos en tiempo constante.
- **gananciasParticulares** y **gastosParticulares**: Ambos son un Array de Integers, ya que siempre tendrán el mismo tamaño y de esta manera su consulta y modificación tiene coste constante. El primero representa las ganancias que lleva cada camión hasta el momento, y el segundo representa los gastos que lleva cada camión hasta el momento. En ambos casos, con el id del i-ésimo camión tenemos acceso a su secuencia de movimientos en el **truckTrack**, a sus **gananciasParticulares** y a sus **gastosParticulares**.

Consideramos que el tamaño del espacio de búsqueda es de O(c*p), donde c son los camiones y p son las peticiones de todas las gasolineras. Esto es así, dado que en el peor de

los casos, en cuanto a coste de la solución, tendremos que todos los camiones pueden atender todas las peticiones, con lo cual tenemos una casuística que abarca todo el espacio de posibles soluciones.

Representación y análisis de los operadores

Los operadores son las funciones que realizan cambios dentro del estado para generar un estado sucesor. Los operadores creados y utilizados en este problema son los siguientes:

- **Add petición**
    - Recibe como entrada un camión **c**, una petición desatendida **p'** y la posición pos del centro de distribución del camión.
    - El operador evaluará la posición anterior de **pos**, y si no ha tenido éxito, evaluará la posición posterior.
    - Se intenta insertar la petición **p'** detrás de la posición **pos** o delante dentro del truckTrack del camión.
    - La función devuelve verdadero o falso según si ha podido añadir la petición.
    - Si no hay peticiones desatendidas devuelve falso.
    - La posición **pos** puede no ser válida para la inserción de una petición, en este caso la función te devuelve falso.
    - Casos que se pueden dar en la posición **pos** dentro del truckTrack: (Siendo C un centro de distribución y P una petición/gasolinera servida).
        - **C**: No se le ha asignado ninguna petición al camión, por lo que la función intentará añadir la petición después del centro. En el caso de que se añada una petición, también se vuelve al centro, convirtiéndose en CPC.
        - CP**C**: El camión únicamente ha servido una petición y ha vuelto al centro. Se le puede añadir una petición antes y en el caso de que sea la última posición del trucktrack después.
        - CPP**C**: El camión ha hecho el viaje completo sirviendo dos peticiones y volviendo al centro. No se le puede añadir ninguna petición entre centros, así que se le asigna una petición al final. Este caso funciona igual que el primero.
    - Para poder aplicar el operador, se asegura de que sigue cumpliendo las diferentes restricciones del problema.
    - Si el operador puede añadir la petición, se calcula la distancia recorrida en ese trayecto, los beneficios obtenidos al servir la petición y los gastos generados.

- Siempre se va a poder aplicar este operador dentro de las restricciones ya que un camión siempre tiene asociado su centro al principio.
- En el peor caso, añadimos todas las peticiones al mismo camión, teniendo un coste de O(p). Las operaciones internas del operador tienen coste constante, ya que son accesos directos a las estructuras de datos definidas en el estado.
- El factor de ramificación del operador es de $r = c * p$

- **<u>Swap</u>**
  - Recibe como entrada dos camiones (C1 y C2) y dos peticiones (P1 y P2).
  - C1 ha servido a P1 y C2 ha servido a P2.
  - El operador realiza un intercambio entre las peticiones de los dos camiones, dando como resultado C1 sirviendo a P2 y C2 sirviendo a P1 y después evalúa si el cambio cumple las restricciones del problema.
  - C1 y C2 pueden ser el mismo camión, intentando intercambiar el orden de peticiones servidas en diferentes viajes.
  - Este operador solo se puede utilizar en un estado inicial **no** vacío y con un **mínimo** de dos peticiones.
  - Afecta a las ganancias de cada camión particular pero no de la ganancia global.
  - Afecta tanto a los gastos particulares como a los gastos globales.
  - Este operador intenta minimizar los gastos en el recorrido de servir peticiones. Intenta encontrar las peticiones que generan menos gasto para cada camión.
  - El coste del operador es constante ya que todas las operaciones intermedias acceden a estructuras de datos definidas en el estado directamente por el índice.
  - El factor de ramificación de este operador en el peor caso es de $r = C^2 x P^2$, ya que se pueden intercambiar todas las peticiones de un camión con todas las peticiones del resto de camiones.

- **Elección de los operadores**

    - La elección de los operadores venía directamente relacionada con los estados iniciales generados.
    - El estado inicial A intenta realizar 5 viajes únicamente sirviendo una petición.
        - Necesitábamos un operador para añadir peticiones a cada camión
        - Asegurar que el estado inicial no nos limitaba las peticiones iniciales las cuales servía.
        - Modificar la distribución de peticiones servidas en un inicio para encontrar el recorrido óptimo.
        - El operador add petición suple la primera limitación.
        - El operador swap petición suple la segunda y tercera limitación.
    - El estado inicial B intenta asignar la petición más cercana como primera petición atendida en cada camión.
        - Se asigna una segunda petición a cada camión que no necesariamente es la más óptima.
        - Únicamente se realiza un viaje por camión.
        - El operador swap suple la primera limitación.
        - El operador add suple la segunda limitación.
    - Los operadores actuales no permiten explorar todo el espacio de soluciones, ya que siempre un camión tendrá mínimo una petición asignada independientemente del estado inicial, por lo que la posibilidad de que un camión no tenga una petición asignada no se explora. Y además, las peticiones que se asignan en los estados iniciales nunca se van a poder dejar de servir.

Análisis de la función heurística

Tal y como se ha indicado en la identificación del problema, debemos maximizar las ganancias globales, minimizando a su vez la distancia recorrida y las ganancias de aquellas peticiones que no atendamos. Para la función heurística, utilizaremos los siguientes elementos:

- el atributo de gananciasTotales, para saber cuales son las ganancias registradas
- el vector peticiones, mediante el cual obtendremos las ganancias que no estamos registrando
- los gastos globales, que, indirectamente, nos dan la distancia global recorrida, y además nos ayudan a computar los beneficios generados.. Como añadido, tenemos otros elementos como **nPeticionesMax**, un parámetro que almacena el máximo número de peticiones desatendidas que hemos tenido, valor que nos resultará útil en caso de querer hacer ponderaciones posteriormente.

Con tal de obtener las mejores soluciones, hay que conseguir que la heurística apremie aquellas soluciones en las que el coste sea menor, lo que en nuestro caso se traduce en apremiar soluciones en las que el gasto y pérdidas por peticiones desatendidas es mínimo. Esto implica que las ganancias deberán ser máximas, y todo esto solo será posible si la heurística genera valores pequeños para las mejores soluciones. Con todo esto, afirmamos que el valor de la heurística tratará las ganancias como un valor negativo, mientras que el gasto y el valor de las peticiones sin atender como algo positivo.

En base a esto, se han planteado dos funciones heurísticas:

$$h(n) = gastos + "pérdidas" - ganancias$$

Esta primera función es la más simple, ya que cumple con los objetivos de maximización o minimización, pero no tiene en cuenta que algunos de los valores puedan tener más importancia que otros, como esel caso de nuestra segunda función heurística:

$$h(n) = -(ganancias - gastos) * (\frac{N_{viajes\ estado}}{N_{viajes\ totales}}) + "pérdidas" * (\frac{N_{peticiones\ estado}}{N_{peticiones\ máximas}})$$

Aquí hemos tratado de cuantificar el peso de cada elemento mediante una ponderación: por un lado, para los beneficios, trata de darles más peso a medida que se van realizando viajes, con lo cual cuantos más viajes hagamos, más peticiones atendemos y, por lo tanto, más ganancias registramos. En este sentido, $N_{viajes\ estado}$ es la variable que cuenta cuántos viajes se han realizado en total,, y $N_{viajes\ totales}$ es el nº de viajes realmente posibles

(el cual corresponde al producto del número de camiones por el número de viajes máximo por camión). Es así que, inicialmente, al tener menos viajes realizados y, por lo tanto, un margen de maniobra más grande para realizar viajes, se premia la realización de todos los movimientos posibles por cada camión. Por otro lado, tenemos la ponderación para las "pérdidas", que, de manera progresiva, le va restando peso, puesto que cada vez vamos teniendo menos peticiones desatendidas.En general, con estas ponderaciones lo que se procura es cambiar el peso de un elemento a otro según la situación del estado generado.

En definitiva, en cuanto a las funciones heurísticas descritas, se ha procurado plantear, por un lado, una función que fuera simple pero efectiva, mientras que, alternativamente, también se ha apostado por encontrar una expresión más refinada e igualmente razonable que conduzca a soluciones iguales o mejores. Para las dos expresiones, hemos procurado minimizar el gasto de cada estado, cosa que sólo conseguimos si premiamos la obtención de beneficios cada vez mayores, a la vez que asignamos peticiones sin atender. Además, para no plantear sólo una heurística sumamente simplista, hemos procurado incorporar elementos de ponderación que traten con los elementos que intervienen en la determinación del valor de un estado, de tal manera que pivotemos el peso de según qué elementos a otros, con tal de equilibrar y potenciar soluciones más adecuadas.

Elección y generación del estado inicial

Los estados iniciales que hemos propuesto para esta práctica han sido dos, a los cuales los hemos llamado **estado inicial A** y **estado inicial B**.

Para el **estado inicial A** , lo que hemos hecho ha sido asignar, para cada camión, una petición por viaje, siempre y cuando no supere el nº máximo de kilómetros a recorrer y de viajes a realizar, y además, no atenderá una petición si no obtiene beneficios o si tiene más pérdidas que beneficios. Por lo tanto, lo que queremos en esta solución es que se vayan rellenando camiones con 1 petición por viaje hasta que no haya más peticiones por atender, o que si las haya pero, en cambio, nos queden camiones con 0 viajes y sin peticiones porque las restantes no pueden ser servidas por los camiones.

Además, hemos asignado las peticiones solo si cumplen las restricciones, ya que si no fuera así, también estaríamos obteniendo soluciones peores que las que se podrían generar. Asimismo, se realiza la asignación de una petición por viaje debido a que, al no ponerle el máximo de peticiones por recorrido, le dejamos un margen al algoritmo para que, con los operadores add y swap que tenemos implementados, se pueda alcanzar una solución óptima del problema. Además, el añadir una petición en el truckTrack implica tener esa petición para siempre, ya que no se puede eliminar de la solución planteada. Esta era otra razón por la cual sólo hemos añadido una petición por viaje.

En otro orden de cosas, el hecho de no atender las peticiones si no hay beneficio se debe a que considerábamos que, al asignarle esa petición a otro camión, podemos pasar de no obtener beneficios a obtenerlos o sino, en el caso de que los días sea lo suficientemente altos, puede darse el caso de que obtengamos mayores beneficios obviando esta petición si al día siguiente algún camión puede atenderla.

Finalmente, el coste asintótico de hallar la solución inicial és de O(c*p) ya que, en el peor de los casos, sería uno en el cual ninguna de las peticiones se pueda servir y por lo tanto cada camión (c) ha de recorrer todas las peticiones (p) posibles.

Para el estado inicial B, lo que hemos hecho ha sido: por cada camión, ordenar las peticiones no servidas hasta el momento y asignarle la primera disponible. Si lo ha hecho, sigue mirando las peticiones restantes y trata de asignarle otra . A la hora de asignar una petición, solo se realizará si no sobrepasa el máximo de kilómetros recorridos (la restricción de no sobrepasar el máximo de viajes no es necesaria comprobarla ya que, como mucho, hará un viaje por camión). Con esto, lo que obtendremos será un recorrido de un único viaje, en el cual pueden haber 2, 1 o ninguna petición.

En la generación del **estado inicial B**, hemos asignado las peticiones solo si cumplen las restricciones, ya que de esta manera no generamos estados que no sean solución.. También hemos decidido ordenar todas las peticiones por distancia respecto de cada camión, ya que de esta manera estamos intentando minimizar la distancia que se tiene para llegar allí a cambio de generarla de manera más costosa. Como en este estado un viaje puede tener hasta 2 peticiones, hemos decidido hacer que cada camión, como mucho, haga un viaje para no restringir demasiado la exploración del espacio de posibles soluciones y, así, poder dejarle un margen al algoritmo para que, con los operadores add y swap, sea suficiente para que se pueda representar una solución óptima. De lo contrario, si estuvieran todos los viajes de los camiones completos, pero no estuvieran todas las peticiones servidas, no se estaría mirando de hacer un intercambio entre las posiciones servidas con las no servidas.

El coste asintótico del estado inicial B es: O(c*p) para ordenar las peticiones más O(c*p) para recorrerlas, así que el coste asintótico seria O(c*p) donde c son los camiones y p las peticiones.

Con esto podemos observar que el coste del estado inicial B es el mismo que el de generar el estado inicial A, a pesar de partir de premisas distintas. Además, los dos estados pueden arrojar resultados distintos, ya que con los operadores utilizados, podemos explorar parte del espacio de las posibles soluciones. Lo que de entrada podemos afirmar es que no se tiene seguridad sobre cuál de las dos estrategias generará mejores soluciones.

Experimentación

Experimento 1: Influencia del conjunto de operadores en la solución final

Para poder explorar adecuadamente el espacio de posibles soluciones, los estados solución iniciales requieren de un conjunto de operadores potentes, mediante los cuales puedan moverse por dicho espacio.

Para demostrar la importancia de este mismo elemento, planteamos aquí un experimento en el que pondremos a prueba 4 conjuntos de operadores distintos, comprobando, a su vez, no solo el valor de las soluciones obtenidas, sino también la importancia del orden en que se ejecutan. Los 4 conjuntos de operadores a tratar serán: **add**, **swap**, **add+swap** y **swap+add**.

Para realizar el experimento adecuadamente, fijaremos una estrategia de generación del estado inicial (en nuestro caso haremos uso del EstadoInicialA) y las características que dan a lugar al mismo, esto es: el número de centros de distribución y gasolineras, juntamente con la multiplicidad. Además, para este experimento realizaremos 10 pruebas distintas, es decir, realizaremos 10 pruebas con semillas distintas, y 1 ejecución por cada conjunto de operadores (lo cual implica que, para cada prueba, se harán 4 ejecuciones de dicha prueba, una con cada conjunto de operadores), lo cual nos permitirá comprobar la validez de los conjuntos y, además, corroborar que dicha validez no depende de la aleatoriedad propia de las semillas usadas.

Siendo conscientes del presente factor de ramificación de cada operador, consideramos conveniente almacenar, además del valor de la solución obtenida, el número de nodos expandidos y el tiempo que se ha tardado en obtener el resultado final. Con estos tres indicadores, se podrá valorar cuantitativamente el coste de espacio y tiempo de cada conjunto de operadores.

Podemos aventurar de antemano que, dadas las características del operador de **swap**, es altamente probable que sea el operador que, por sí solo, genere las peores soluciones. Aún así, es algo que dejamos en manos de los resultados de los experimentos.

Asimismo, para valorar cada estado solución, haremos uso de la función heurística más sofisticada, función que ya ha sido descrita previamente, y con la que trabajaremos siempre de ahora en adelante.

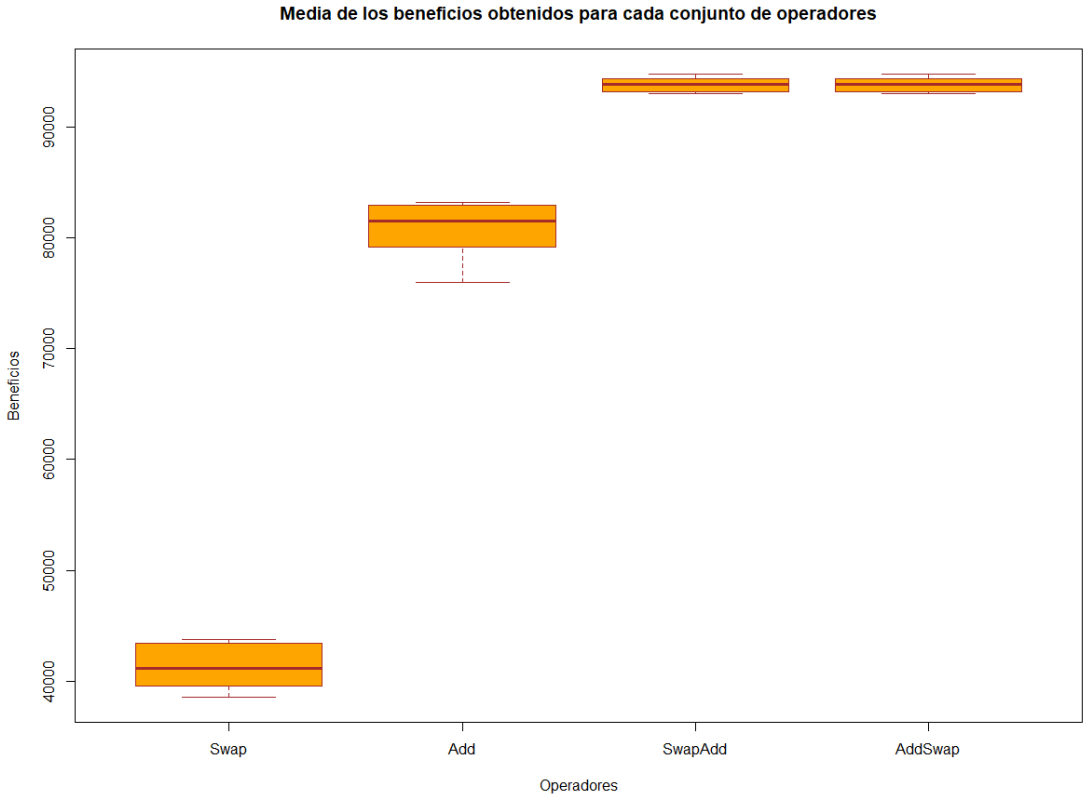
Resumimos las características del experimento en la siguiente tabla:

Observación	Puede haber conjuntos de operadores que, por su naturaleza, su factor de ramificación y su combinación, arrojen resultados distintos.
Planteamiento	Creamos diferentes conjuntos de operadores (y distintos órdenes entre ellos) y observamos las soluciones obtenidas de los mismos.
Hipótesis	Hay conjuntos de operadores mejores que otros
Método	● Elegiremos 20 seeds aleatoriamente. ● Para cada par de seeds, haremos una prueba, con una ejecución por cada conjunto de operadores. ● Fijaremos el EstadoInicialA como el generador del estado solución inicial. ● Establecemos 4 conjuntos de operadores: *add*, *swap*, *add* y *swap*, *swap* y *add*. ● Los problemas serán de 10 centros y camiones y 100 gasolineras, con una multiplicidad fija de 1. ● Usaremos el algoritmo de HillClimbing para atacar el problema. ● Mediremos el valor, el número de pasos ejecutados y el tiempo de ejecución de cada solución obtenida.

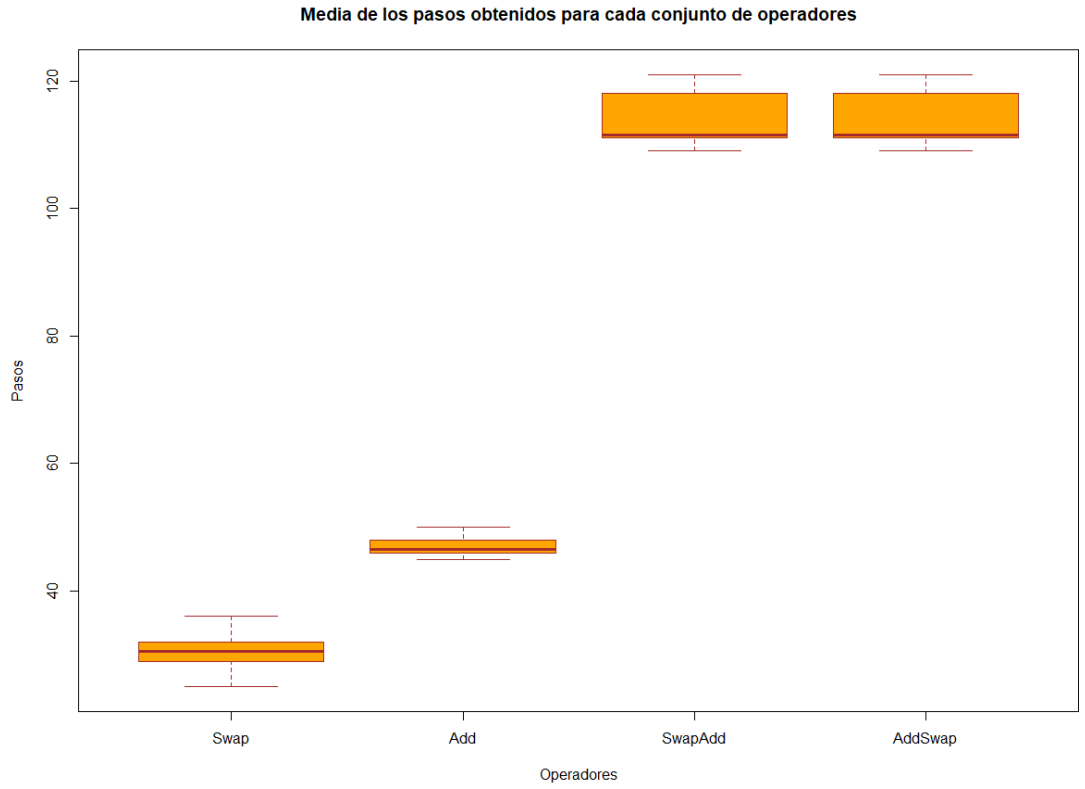
Resultados y gráficos del experimento:

	Swap	**Add**	**Add-Swap**	**Swap-Add**
Beneficios	41339,6	80710,4	93804,8	93804,8
#Pasos	30	47	113	113
Tiempo (ms)	99,523	230,894	1196,325	1593,761

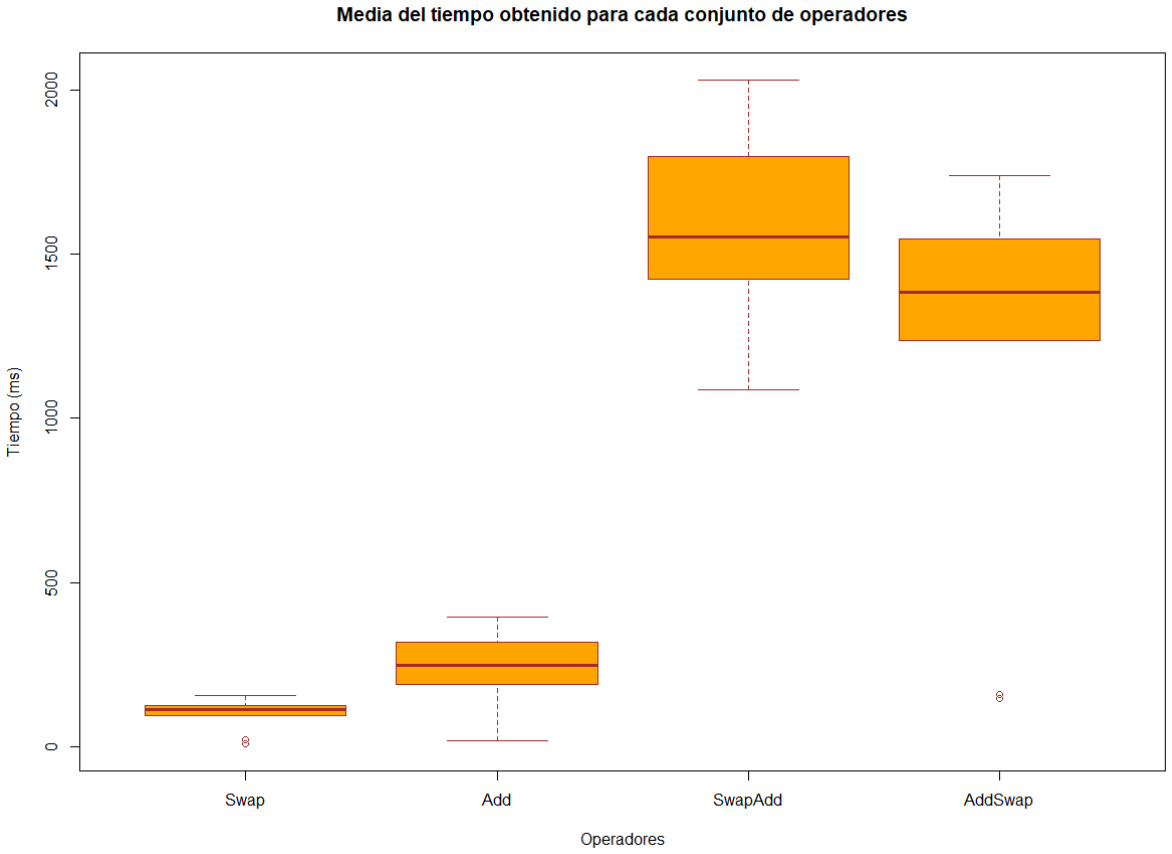
Tabla resumen de los resultados. (Los valores mostrados son la media de las diferentes pruebas)



Boxplots comparando los beneficios obtenidos por cada uno de los conjuntos de operadores.



Boxplots comparando los pasos realizados por cada uno de los conjuntos de operadores.



Boxplots comparando los pasos realizados por cada uno de los conjuntos de operadores.

Conclusiones

En primer lugar, tal y como podemos observar en los gráficos anteriores, el conjunto de operadores que peores resultados arroja es el operador de swap, ya que, tal y como se ha comentado, él por sí solo es incapaz de explorar todo el espacio de posibles soluciones, con lo cual hace pocos pasos en poco tiempo, lo que consecuentemente lleva a generar muy pocos beneficios. Para el caso en el que se usa únicamente el add, cómo puede explorar parte del espacio de posibles soluciones, en media hace más pasos y tarda más tiempo que el swap, lo que le lleva a generar un mayor beneficio. Finalmente tenemos el conjunto add y swap, conjunto que, tal y como se puede apreciar en las tablas y gráficas, es equivalente al conjunto de swap y add, pero este último, de media, es el que realiza más pasos, y además no es capaz de diferenciarse, en cuanto a beneficios y número de pasos se refiere, del conjunto de add y swap. De esta manera, se puede afirmar que el orden de ejecución de operadores add y swap es mejor, ya que por su propia naturaleza

llega antes al mejor estado solución alcanzable desde el estado inicial generado. Era de esperar, además, que el operador de swap fuera el peor entre todos, y que los conjuntos Swap-Add o Add-Swap, como se explicaron en los operadores, permitan explorar parte del espacio de posibles soluciones.

Finalmente, debido a la cantidad de beneficios registrados, y en relación con el tiempo de ejecución medio y número de pasos efectuados, escogeremos entre add-Swap o el Swap-Add, ya que para el problema es el conjunto que más capacidad y potencia tiene para explorar todas las posibilidades del espacio de posibles soluciones del problema. Para escoger de entre los dos, nos decantamos por el Add-Swap debido a que tarda de media 169,622 ms menos que el Swap-Add. De ahora en adelante, este será el conjunto de operadores con el que trabajaremos.

Experimento 2: Decidir la mejor estrategia de generación de la solución inicial

Una vez determinado el conjunto de operadores que nos genera mejores soluciones, decidiremos qué estrategia para la generación de la solución inicial arroja mejores resultados.

Para realizar este experimento, se partirá de las mismas condiciones que el experimento anterior, a la vez

Una vez determinada la función heurística que vamos a utilizar y el conjunto de operadores que nos generan mejores soluciones, decidiremos qué estrategia para la generación de la solución inicial nos proporciona mejor resultado.

Para este experimento utilizaremos los operadores Add y Swap en este orden y el algoritmo de búsqueda Hill Climbing.

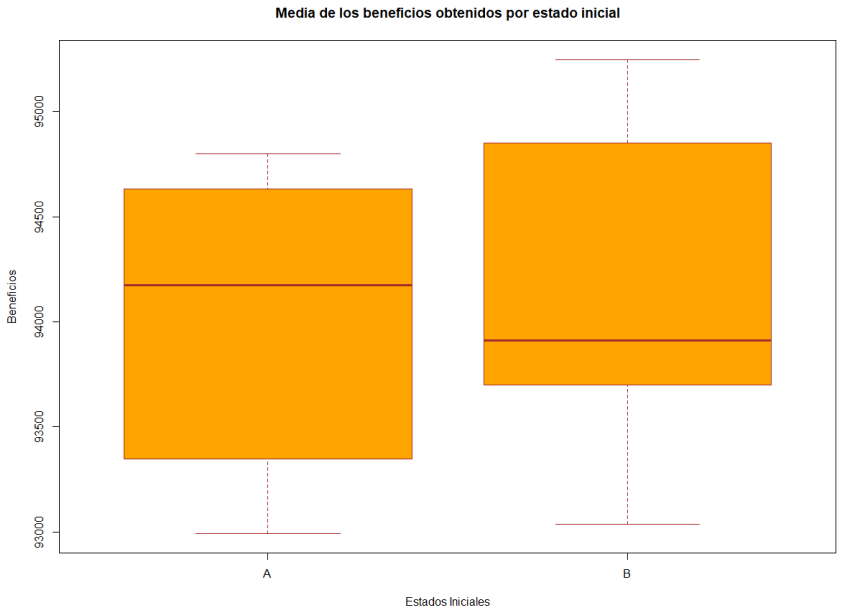
El método para valorar que una estrategia es mejor que otra será comprobar el beneficio total que nos genera cada una de las soluciones aplicando el mismo algoritmo y los mismos operadores .

Observación	Existe un método de inicialización mejor que otro.
Planteamiento	Escogemos los diferentes métodos de inicialización y observamos sus soluciones.
Hipótesis	El estado inicial A nos permite más libertad que el estado inicial B.
Método	● Elegiremos 20 seeds aleatorias. ● Para cada par de seed, haremos una prueba para cada estado inicial ● Los estados iniciales que utilizaremos serán Estado Inicial A y el Estado Inicial B ● Los problemas serán de 10 centros con multiplicidad 1 y 100 gasolineras. ● Usaremos el algoritmo de Hill Climbing. ● Obtendremos los beneficios totales de cada una de las soluciones. ● Mediremos el número de pasos ejecutados y el tiempo de ejecución de cada solución.

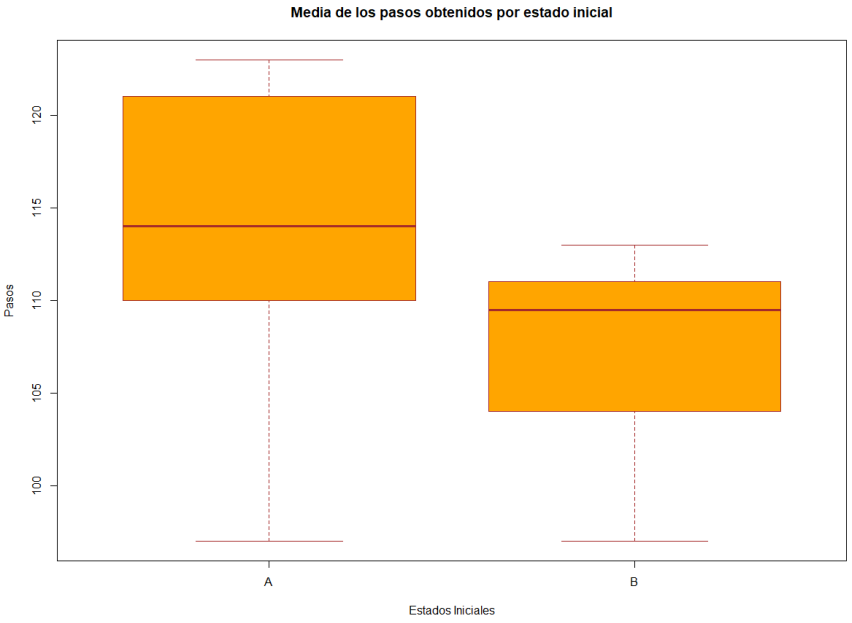
Resultados y gráficos del experimento:

	Estado Inicial A	Estado Inicial B
Beneficios	94051,2	94124,8
Número de pasos	114	107
Tiempo (ms)	1199,576	1102,723

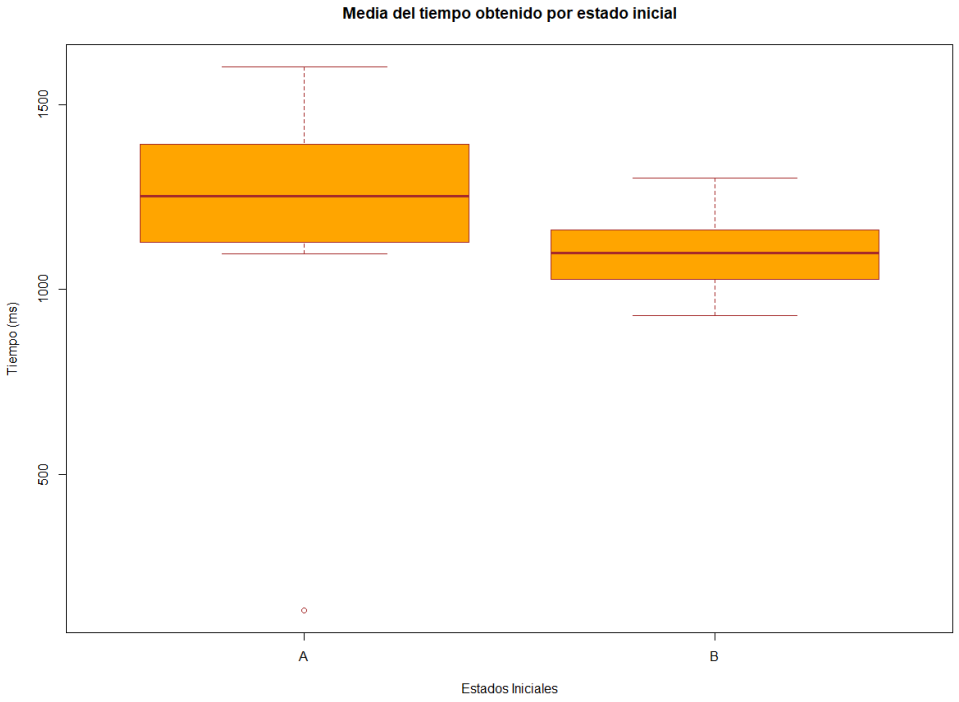
Tabla resumen de los resultados. (Los valores mostrados son la media de las diferentes pruebas)



Boxplot comparando los beneficios obtenidos entre los diferentes métodos de generación del estado inicial.



Boxplot comparando el número de pasos obtenidos por cada uno de los estados iniciales.



Boxplot comparativa entre la media del tiempo obtenido por cada uno de los estados iniciales.

Conclusiones

Como podemos observar en los gráficos anteriores, el rango de beneficios que nos proporciona el estado inicial B tiende al alza, esto es, se genera una cantidad de beneficios mayor a la mediana registrada. Esto, a su vez, viene acompañado del hecho que este mismo estado tiene una media de generación de beneficios que el estado inicial A, con lo cual podemos deducir que el estado inicial B apunta a mejores ganancias. Parte de este hecho es atribuible a que la estrategia seguida para generar este estado inicial procura realizar una primera asignación, en la cual cada camión atienda a las peticiones que se sabe con seguridad que le reportará más ganancias.

Respecto al tiempo, observamos que el estado inicial A tarda más tiempo de media que el estado inicial B y también realiza más pasos para encontrar la solución final. Esto puede ser debido a que la estrategia de generación del estado inicial A busca realizar, al menos, una asignación por cada viaje que pueda realizar un camión. En este sentido, es evidente que dicha estrategia acota mucho más el espacio de posibles soluciones alcanzables, la cual implica que, para llegar a una solución igual o mejor que la encontrada por el estado inicial B, necesitará más tiempo para (posiblemente) deshacer parte de las asignaciones inicialmente realizadas.

Finalmente, podemos afirmar que en este experimento, aunque no hay una gran diferencia en los resultados entre los estados, podemos ver como, en general, el estado inicial B arroja mejores resultados. Sin embargo, cabe realizar la siguiente aclaración:

Debido a un error en el cálculo de los valores en los estados iniciales el cual nos dimos cuenta cuando ya teníamos todo el trabajo hecho, en vez de escoger el estado inicial B que sería, experimentalmente el que mejores resultados nos ha dado en cuanto a beneficios, tiempo de ejecución y número de pasos, hemos escogido el estado inicial A, ya que era el que mejor resultado nos daba con los errores de cálculos. Este error ha sido porque en vez de calcular la media, hemos calculado la mediana, entonces los valores que se pueden observar en las gráficas, los cuales están hechos con la mediana, nos da mejores resultados el estado inicial A que el B. No decidimos cambiarlo ya que habría que rehacer todo el trabajo a partir del experimento 3 y además no sería un problema, ya que para los siguientes experimentos (modificar la multiplicidad, la entrada del problema, el coste del kilómetro o aumentar la distancia máxima recorrida por dia y camión), es cierto que alomejor maximizará más los beneficios , en principio tardará menos en ejecutarse y expandirá menos nodos, pero, al fin y al cabo lo que queremos estudiar en esos

experimentos es la influencia que tienen en la solución final. Esto último será independiente del estado inicial, ya que no influirá en el comportamiento del estudio.

Experimento 3: Influencia de los parámetros del SA en la solución final

Una vez determinada la estrategia de generación del estado solución inicial, procederemos a abordar la resolución del problema dado a través del algoritmo de Simulated Annealing. Este algoritmo, que consta de 4 parámetros (#iteraciones, stiter, k, λ), busca generar, de manera aleatoria, estados sucesores que sean candidatos a solución.

Es en base a esta premisa que, teniendo un enfoque sólido de cómo tratar el problema, debemos encontrar qué valores debemos darle a los parámetros del algoritmo para que este nos dé soluciones óptimas.

Para realizar la experimentación adecuadamente, fijamos, para cada prueba, dos seeds (una para los centros y otra para los camiones), de tal manera que podamos realizar comparaciones rigurosas entre los resultados de las mismas. Además, inicialmente procederemos a encontrar el #iteraciones óptimas para resolver el problema; después, buscaremos para qué stiter las ejecuciones se realizan de manera óptima; finalmente, buscaremos qué combinaciones de k y lambda funcionan mejor entre ellas.

La manera en que encontraremos cada parámetro será el siguiente:

- Para el #iteraciones, realizaremos 10 ejecuciones para cada valor del siguiente rango: {500, 1000, 2500, 5000, 10000}.
- Para el stiter, realizaremos 10 ejecuciones para cada valor del siguiente rango: {50, 100, 500, 2500}.[1]
- Para k y λ, realizaremos 10 ejecuciones por cada combinación de k con λ, teniendo k un rango: {16, 64, 96, 128}; y λ un rango: {0,02; 0,002; 0,0002}.

Antes de continuar, cabe destacar que, mientras que antes considerábamos el #pasos realizado, ahora lo ignoraremos por completo, dado que, para el Simulated Annealing, este será equivalente al #iteraciones elegido. Sin este elemento, trataremos de valorar la calidad de una solución en base a la cantidad de beneficios generada y al tiempo de ejecución de la misma.

[1] Cabe destacar que el stiter debe ser un múltiplo del #iteraciones. Además, hay que añadir que estos valores se determinaron a posteriori, una vez encontrado el #iteraciones con las que trabajar.

Las condiciones del experimento se resumen en la siguiente tabla:

Observación	Puede haber parámetros para el SA que arrojen resultados distintos.
Planteamiento	Elegimos diferentes valores para los parámetros y observamos sus soluciones
Hipótesis	Hay conjuntos de parámetros que arroja mejores resultados que otros
Método	● Elegiremos las seeds de manera aleatoria. ● Para cada parámetro a determinar, realizaremos 10 pruebas, y cada una de ellas con seeds distintas tanto para centros como para camiones. ● Para #iteraciones, tendremos valores de {500, 1000, 2500, 5000, 10000}. ● Para stiter, tendremos valores de {500,100,50} ● Para k, tendremos valores de {16, 64, 96, 128}; y para λ tendremos valores de {0,02; 0,002; 0,0002} ● Los problemas serán de 10 centros de multiplicidad 1 (1 camión por centro) y 100 gasolineras. ● Usaremos el algoritmo de Simulated Annealing. ● Mediremos los beneficios generados y el tiempo de ejecución de cada combinación de parámetros

Resultados y gráficos del experimento

En la tabla siguiente, en cuanto a la búsqueda del #iteraciones óptimo, hemos recopilado los resultados obtenidos con unos parámetros Stiter = 100, k = 4 y λ = 2. Los resultados de los beneficios y tiempo de ejecución para esta tabla y las siguientes son la media de las ejecuciones variando las semillas usadas para los centros y gasolineras del problema.

Pasos	Beneficio	Tiempo (ms)
500	86159,6	4165,140837
1000	89657,2	8837,191094
2500	91643,2	25164,78316
5000	92306,4	48273,36256
10000	92819,2	105736,3731

En la tabla siguiente, en cuanto a la búsqueda del stiter óptimo, hemos recopilado los resultados obtenidos con unos parámetros Steps = 2500, k = 4 y λ = 2:

Stiter	Beneficio	Tiempo en ms
50	91806,4	21468,43459
100	91643,2	1835,880667
500	92305,6	2427,110492
2500	91359,2	1863,278073

En la tabla siguiente, en cuanto a la búsqueda del k y λ óptimos, hemos recopilado los resultados obtenidos con los parámetros Steps = 2500 y stiter = 500:

K	λ	Beneficios	Tiempo (ms)
16	0,02	91100,4	23831,21121
16	0,002	91286,4	23866,45388
16	0,0002	92376,4	23743,94558
64	0,02	90264,4	23401,65801
64	0,002	90308,4	23344,40771
64	0,0002	89632,8	23226,20525
96	0,02	89278,8	23099,87074
96	0,002	89679,6	22960,2828
96	0,0002	89184,4	22925,54187
128	0,02	88606,8	22448,65466
128	0,002	88966,4	22689,67628
128	0,0002	88662	22575,37307

A continuación, presentamos los gráficos de tendencia y cuantificación de las medias obtenidas:

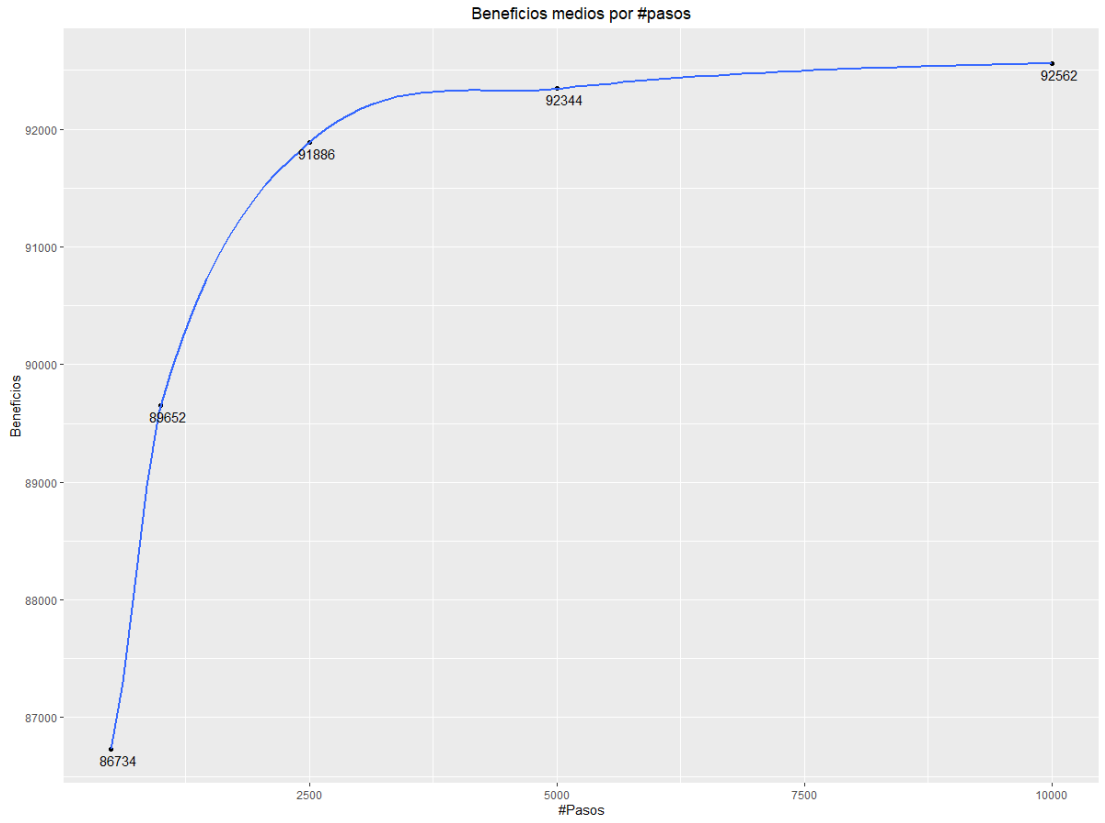
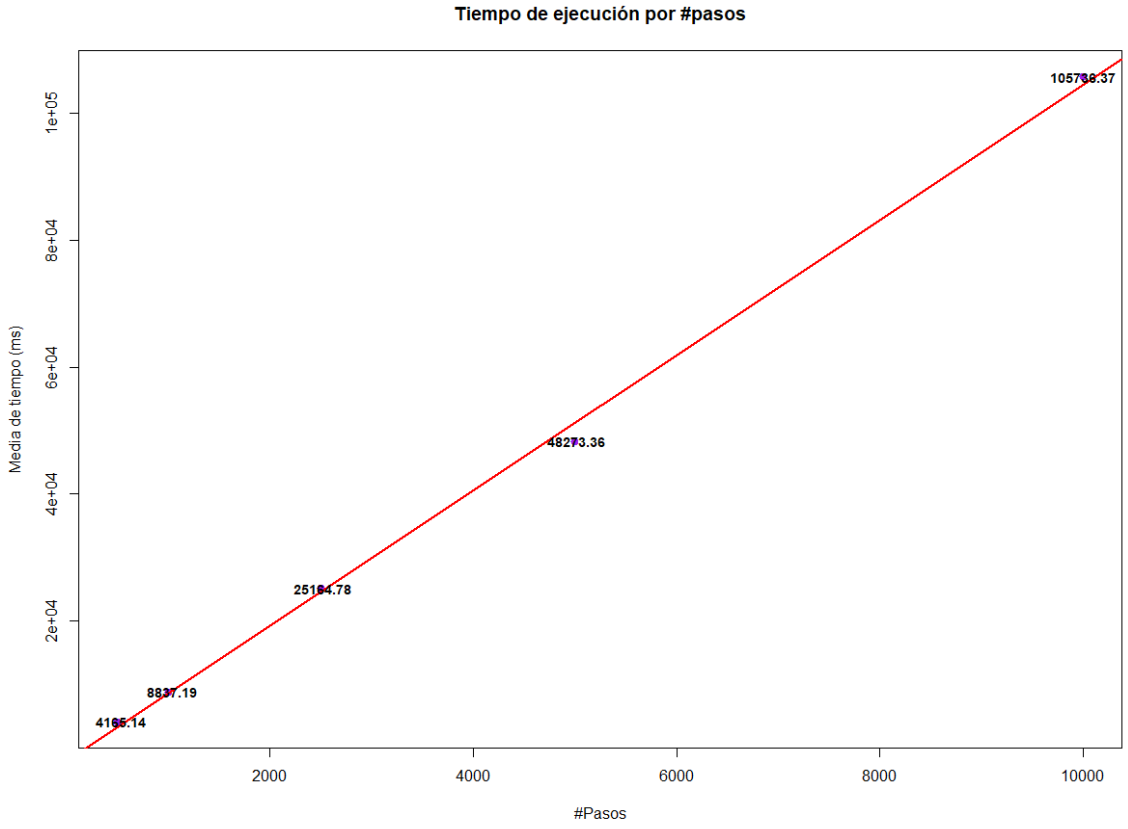
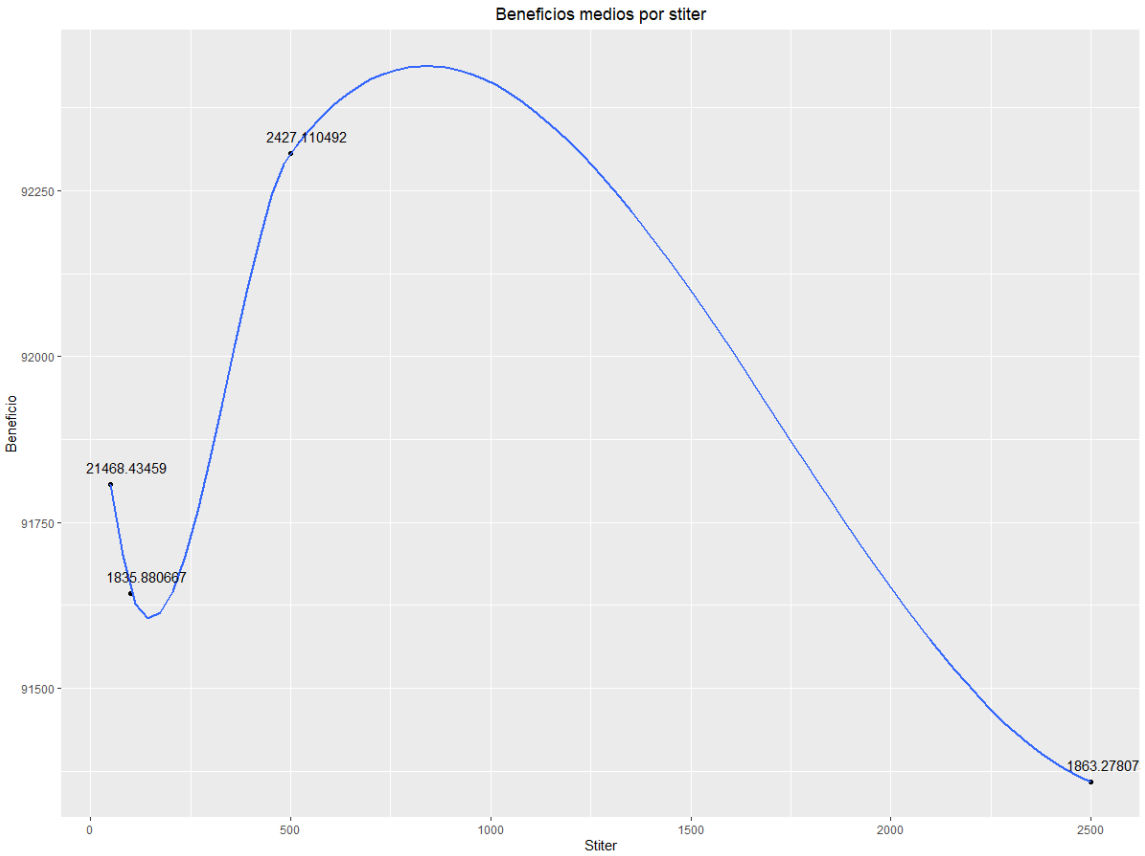


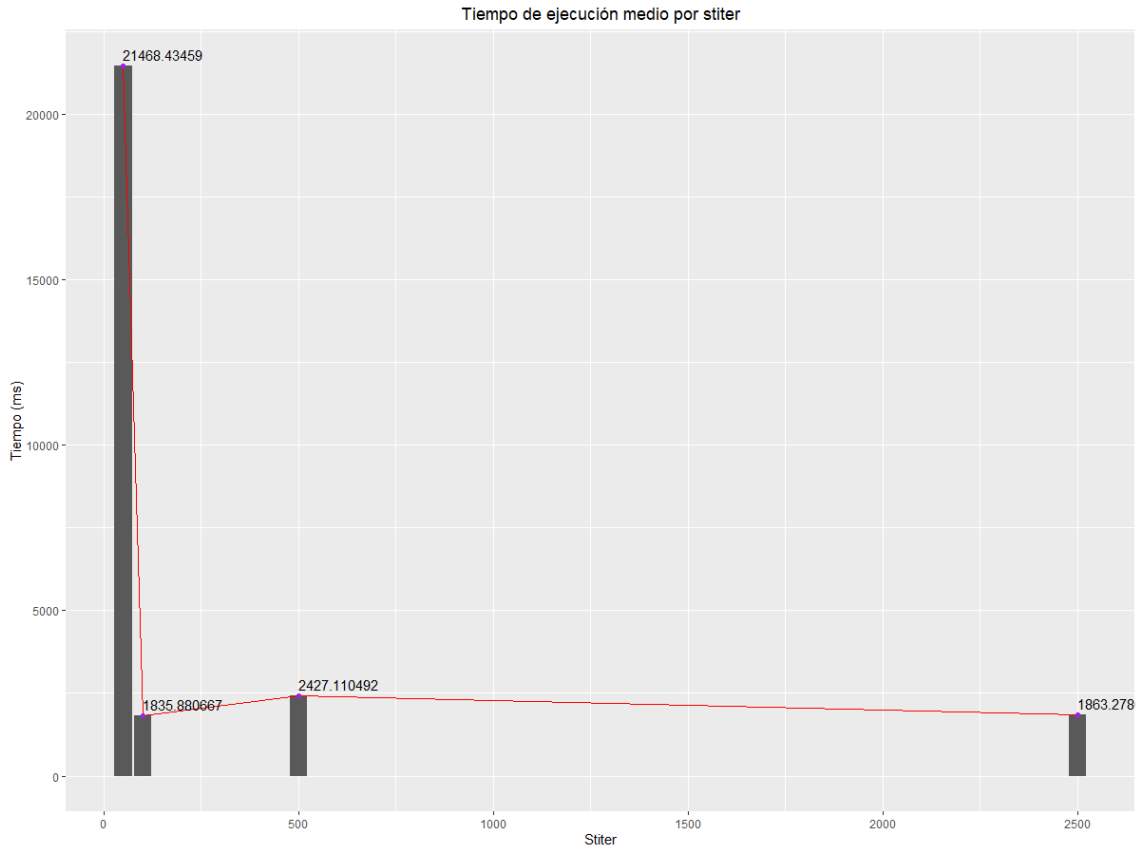
Gráfico del beneficio obtenido respecto al número de pasos realizados



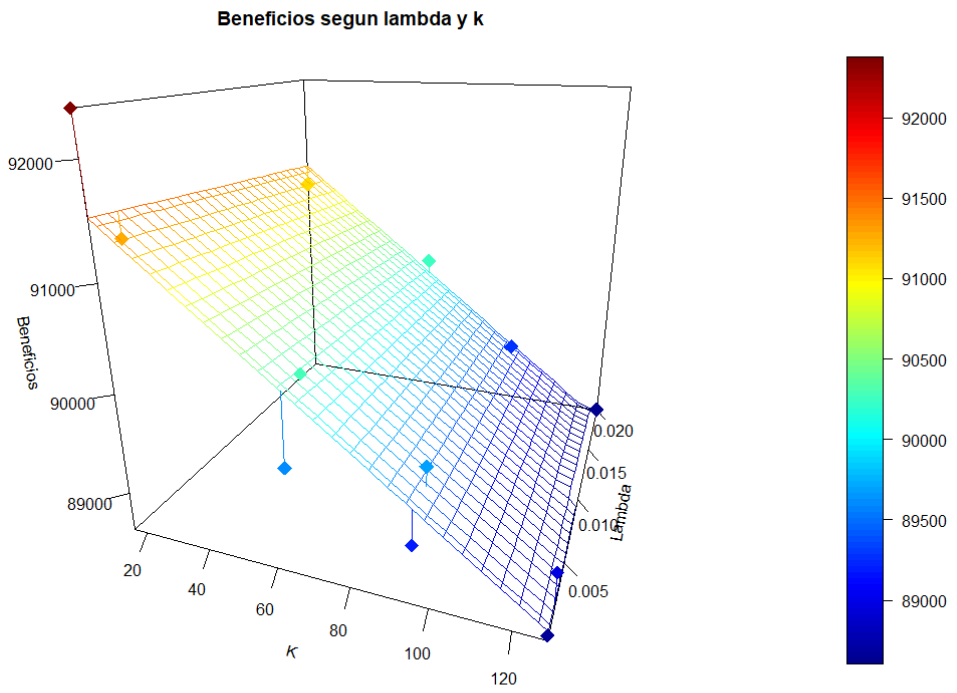
Regresión lineal entre el tiempo y el número de pasos realizados



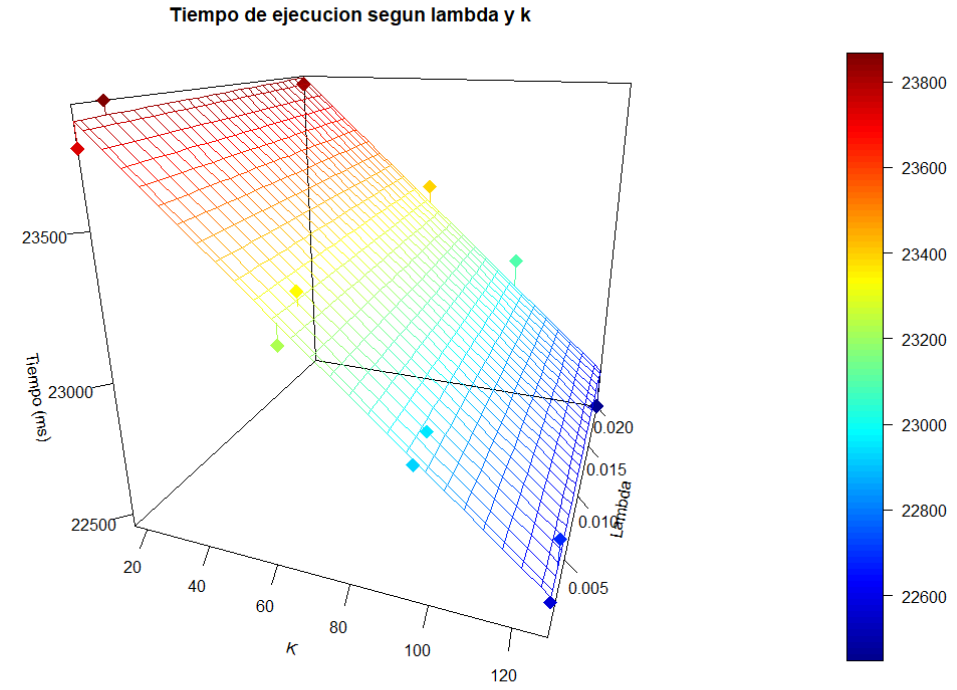
Gráfica del beneficio obtenido respecto al número de stiter



Gráfica entre el tiempo de ejecución en ms respecto al número de stiter



Gráfica en 3D en la cual se representa el parámetro λ (eje y) y k (eje x) respecto a los beneficios (eje z)



Gráfica en 3D en la cual se representa el parámetro λ (eje y) y k (eje x) respecto al tiempo de ejecución en(eje z)

Conclusiones

En primer lugar, en cuanto al número de iteraciones se refiere, a través de las gráficas aquí expuestas podemos inferir que, a partir de las 2500 iteraciones, los potenciales beneficios a registrar se estancan, con lo cual termina habiendo poca diferencia entre las 2500, las 5000 y las 10000 iteraciones. Sin embargo, en cuanto a coste temporal se refiere, podemos ver claramente que, a mayor nº de iteraciones, mayor es el coste temporal, y su relación es lineal. En base a estos datos, determinamos que fijar el número de iteraciones en 2500 es una buena decisión, ya que, por un lado, es a partir de este valor que los beneficios se estancan y, por otro lado, tenemos un coste temporal óptimo para llegar a una solución.

En segundo lugar, en cuanto al stiter se refiere, podemos ver que las estimaciones realizadas apuntan a que, entre las 500 y las 1500 iteraciones, el stiter arroja los mejores resultados posibles al problema estudiado. Asimismo, de acuerdo con el tiempo de ejecución medio por stiter, vemos que, más allá de que para un stiter pequeño se tarda muchísimo más en encontrar soluciones óptimas, el stiter de 500 apunta a que, para aquellos valores (asumimos que entre 500 y 1500) que arrojan las mejores soluciones, comportan un mayor coste temporal, coste que no implica un gran incremento en cuanto al tiempo de ejecución se refiere. Es así como determinamos que un stiter de 500 es la mejor alternativa: interviene positivamente en la generación de beneficios y no penaliza el tiempo de ejecución.

Finalmente, en cuanto a **k** y **λ** se refiere, podemos inferir lo siguiente: cuanto menor sea k, mayores beneficios podemos obtener y, como contrapartida, mayor será su tiempo de ejecución. Vale la pena, pero, realizar una consideración respecto al coste temporal: la diferencia entre ir a por los máximos beneficios (es decir, elegir una k pequeña) y modestamente conformarse con unos beneficios medianamente buenos es de 0,4-0,6 segundos, con lo cual no se considera que haya una severa penalización por tomar un valor pequeño de k. Además, en otro orden de cosas, sabemos que podemos elegir cualquier valor de λ, ya que hemos podido constatar que dependerá (en su mayor parte) de k que lleguemos a una solución óptima. Aún así, cabe destacar que, para una λ pequeña, podremos alcanzar el mayor beneficio posible (pero eso sí, con un *elevado* coste temporal).

Con todo lo dicho, procurando equilibrar la optimalidad del tiempo de ejecución con la búsqueda del máximo beneficio generado, decidimos fijar los parámetros del Simulated Annealing a: #iteraciones = 2500; stiter = 500; k = 16; λ = 0,02. De ahora en adelante, siempre usaremos estos valores para las ejecuciones del algoritmo aquí estudiado.

Experimento 4: Influencia el aumento de la entrada del problema en la solución final

En este experimento, lo que queremos hacer es estudiar el comportamiento que tendrá en el Hill Climbing y el Simulated Annealing (con los parámetros anteriormente fijados) ante la modificación de la entrada que teníamos definida anteriormente (10 centros de multiplicidad 1 y 100 gasolineras, es decir, de escala 10:100), esto es, ante un incremento los centros de 10 en 10 y las gasolineras de 100 en 100.

Para ello, lo que haremos será escoger 2 semillas aleatorias (una para los centros de distribución y otra para las gasolineras), y para cada una de ellas ejecutaremos los diferentes valores para las entradas. Estos valores serían 10, 20 y 30 para los centros y 100, 200 y 300 para las gasolineras.

Esto lo haremos para los dos algoritmos, de manera que se usen las mismas semillas para ambos.

Observación	El aumento de la entrada del problema puede afectar al rendimiento del algoritmo usado para la resolución del mismo.
Planteamiento	Aumentamos dinámicamente el tamaño de la entrada y evaluamos el rendimiento de los algoritmos de búsqueda local utilizados para resolver el problema.
Hipótesis	Las soluciones pueden oscilar debido a la distribución de las gasolineras y los centros en el mapa de 100*100 km².
Método	● Elegiremos las seeds aleatoriamente. ● Se parte de las mismas condiciones que en los experimentos anteriores, manteniendo los valores hasta ahora fijados. ● Se realizarán 4 secuenciaciones: inicialmente empezaremos con 10 centros y 100 gasolineras, y a medida que hagamos ejecuciones, iremos aumentando el tamaño del problema de 10 en 10 para los centros y de 100 en 100 para las gasolineras hasta llegar a 30, punto desde el cual volvemos a empezar. Procederemos así hasta terminar las 4 secuencias. ● Usaremos el algoritmo de Hill Climbing y Simulated Annealing. ● Mediremos el número de pasos ejecutados, los beneficios y el tiempo de ejecución de cada algoritmo.

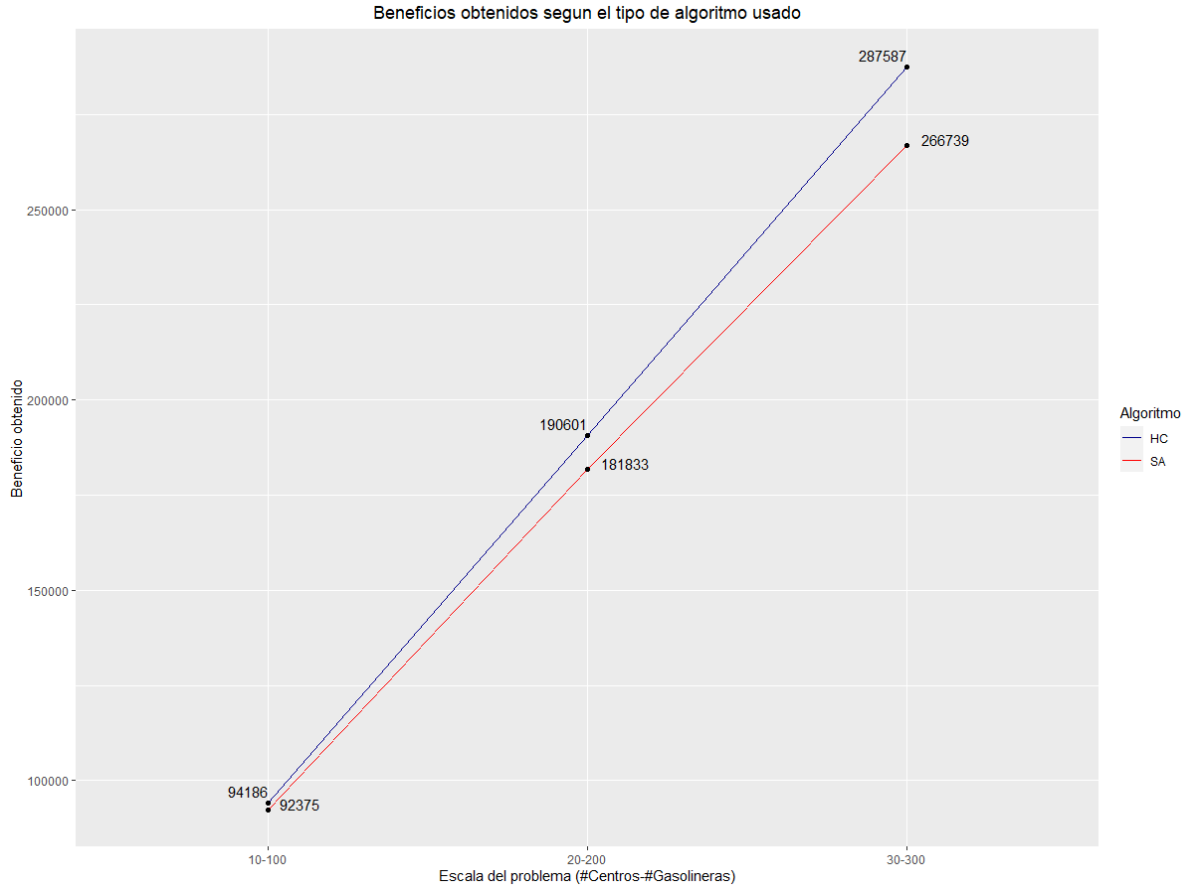
Resultados y gráficos del experimento:

Para el Hill Climbing (de ahora en adelante referenciado como HC), hemos obtenido la tabla siguiente con la media de las ejecuciones realizadas:

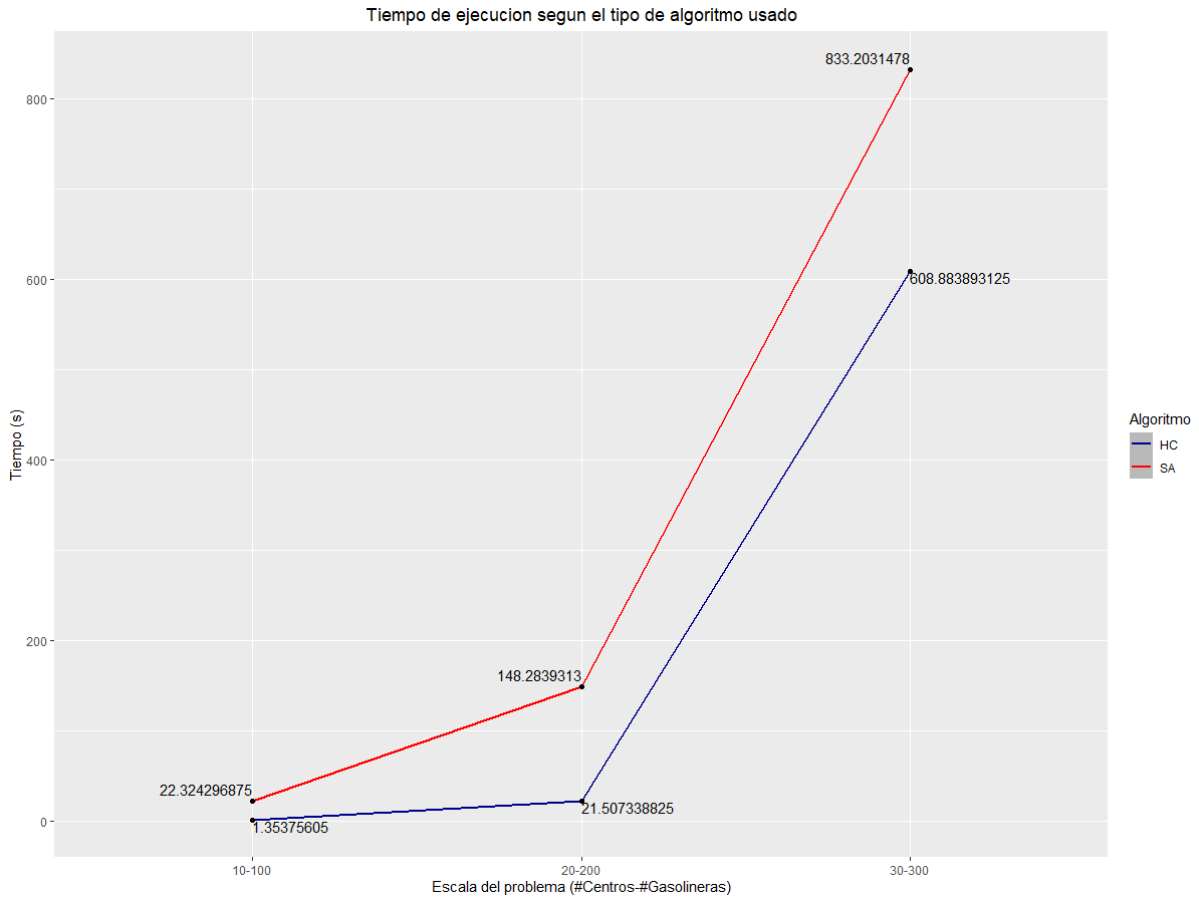
Escala	Beneficios	Pasos	Tiempo (s)
10-100	94186	115	1,35375605
20-200	190601	117	21,50733883
30-300	287587	119	608,8838931

Para el Simulated Annealing (de ahora en adelante referenciado como SA), hemos obtenido la tabla siguiente con la media de las ejecuciones realizadas:

Escala	Beneficios	Pasos	Tiempo (s)
10-100	92375	2501	22,32429688
20-200	181833	2501	148,2839313
30-300	266739	2501	833,2031478



Gráfica del beneficio obtenido respecto a la escala del problema. La escala estaría representada por el formato (número de camiones)-(número de gasolineras)



Gráfica del tiempo de ejecución obtenido respecto a la escala del problema. La escala estaría representada por el formato (número de camiones)-(número de gasolineras)

Conclusiones

En primer lugar, en cuanto a beneficios se refiere, podemos constatar la diferencia entre las tendencias del HC y del SA: en base a las medias representadas, podemos ver cómo el HC siempre lleva ventaja. Podemos inferir que, para tamaños pequeños, SA se desempeña mejor que HC, pero para tamaños grandes la tendencia juega más a favor del HC. Dicho fenómeno puede atribuirse, o bien a que los parámetros del SA no encajan correctamente con tamaños de la entrada muy grandes, o bien a que el SA sea, por su propia naturaleza, peor ante el hecho de tener que encontrar soluciones mejores en un espacio de posibles soluciones muy grande. En cualquiera de los casos, podemos constatar que HC genera más beneficios que SA, y la tendencia apunta a que siempre será, de media, así.

En cuanto al tiempo de ejecución se refiere, podemos destacar la clara diferencia entre los dos algoritmos: los dos, ante un aumento de la entrada, ven sus costes temporales disparados, pero el SA se ve mucho más afectado por ello. Varios factores pueden jugar a favor y en contra de los dos algoritmos: HC genera todos los posibles sucesores que un

estado puede llegar a tener y, ante un espacio de posibles soluciones muy grande, su coste aumenta debidamente, aunque no deberíamos saltar a conclusiones precipitadas, puesto que HC, por un lado, no se encuentra atado a unas restricciones de generación de sucesores (tales como el #iteraciones a realizar, por ejemplo), y por otro lado una buena heurística puede llevarnos por buen camino, aunque sea un mínimo local. En otro orden de cosas, SA, al ser estocástico, tiene más libertad para explorar toda la casuística del espacio de posibles soluciones, pero al encontrarse atado a sus parámetros, puede llegarse a ver obligado a explorar regiones que no terminen siendo del todo fructíferas.

Debido a estos hechos, nuestro experimento indica que HC le lleva la delantera al SA y que, además, los parámetros encontrados en el experimento anterior pueden llevar a contrariedades curiosas: ¿puede ser que, debido a la elección de los mismos, el SA genere soluciones mucho peores de las que podría alcanzar? Dado que no hemos realizado una elección acorde con un criterio razonable, consideramos que la inferioridad del SA ante el HC no es debido a la elección de los parámetros, y que dicha elección consigue mantener relativamente a la par los dos algoritmos.

Experimento 5: Influencia de la multiplicidad de los centros en la solución final

Para este experimento, procederemos a cambiar una de las características hasta ahora fijadas: la multiplicidad del problema. La multiplicidad es el elemento que determina cuántos camiones tenemos por centro de distribución: multiplicidad=1 significa 1 camión por centro, multiplicidad=2 significa 2 camiones por centro, y viceversa.

A pesar de desfijar este parámetro, mantendremos toda la información hasta ahora adquirida: mantenemos la función heurística, la estrategia de generación del estado inicial, el conjunto de operadores usados, etc.

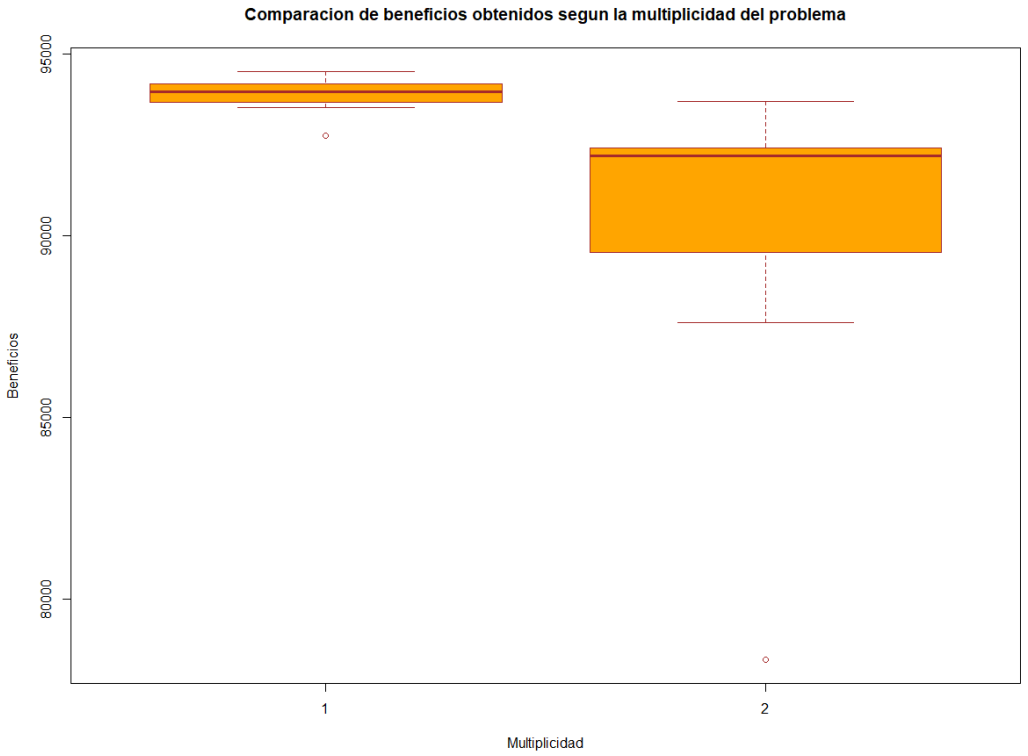
En cuanto a la naturaleza del experimento que se realiza, nos planteamos dos situaciones: o bien trabajamos con 10 centros de distribución y 1 camión por centro (nos referiremos a esta situación, de ahora en adelante, como aquella donde la multiplicidad=1), o bien trabajamos con 5 centros de distribución y 2 camiones por centro (nos referiremos a esta situación, de ahora en adelante, como aquella donde la multiplicidad=2). La intención es poder comparar qué tipo de soluciones emanan de estas 2 premisas. Además, para este experimento, mantendremos el criterio de valoración hasta ahora usado.

Observación	Si, en lugar de tener 10 centros (con 1 camión por centro), tenemos 5 centros (con 2 camiones por centro), el espacio de posibles soluciones se verá reducido debido a la probabilidad que viene dada por la aleatoriedad de la generación de los centros de distribución.
Planteamiento	Comparamos cuál sería la solución propuesta por un algoritmo de búsqueda local, dadas las dos situaciones planteadas en el experimento.
Hipótesis	Los resultados serán peores en el caso de tener multiplicidad=2 (5 centros de distribución, 2 camiones por centro).
Método	● Elegiremos las seeds aleatoriamente. ● Realizaremos 10 pruebas, cada una de las pruebas tiene unas seeds aleatorias. Se ejecuta este contexto tanto en la solución con multiplicidad 1 como en la solución de multiplicidad 2. Se observará el beneficio generado por cada solución. ● Usaremos el algoritmo de Hill Climbing. ● Mediremos el número de pasos ejecutados, los beneficios y el tiempo de ejecución para los dos casos.

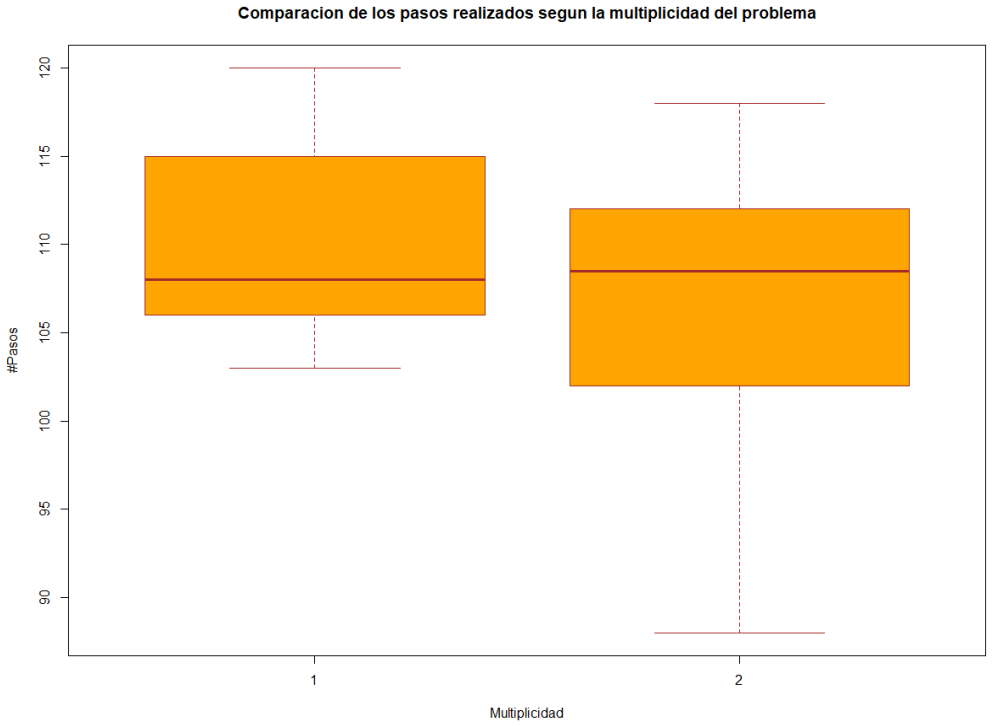
Resultados y gráficos del experimento:

En base a las ejecuciones realizadas, y una vez computadas las medias de cada secuencia de ejecuciones, tenemos la siguiente tabla y los siguientes gráficos:

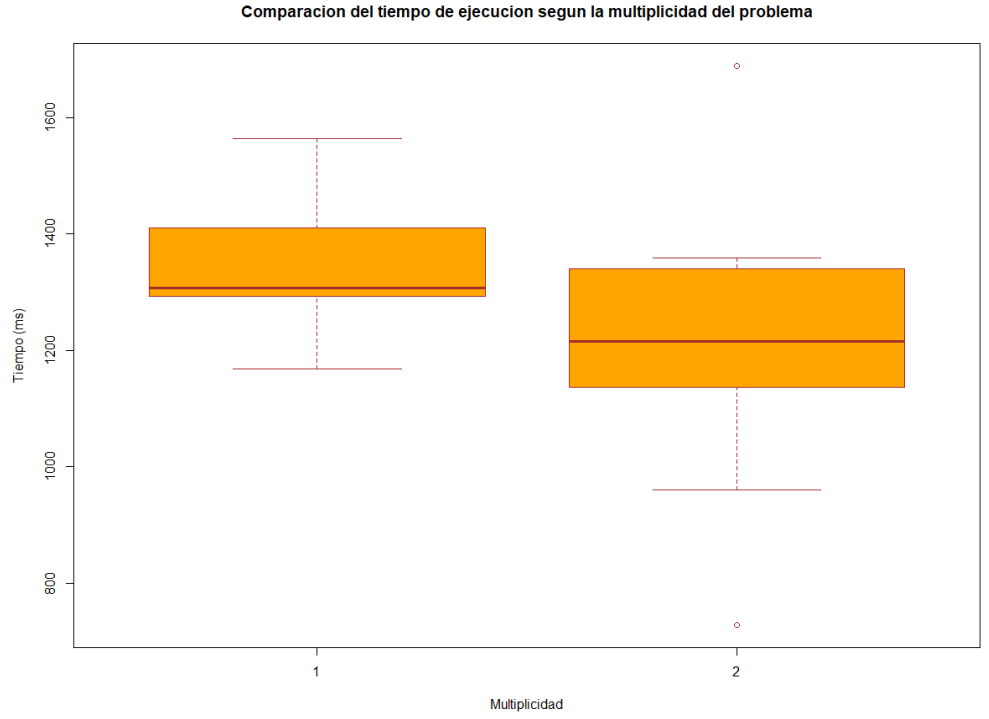
Multiplicidad	Beneficio	Nodos expandidos	T eje (s)	Km recorridos
1	93864,4	110,3	11,0388928	3169,8
2	90276,4	106,7	8,6304949	4100,8



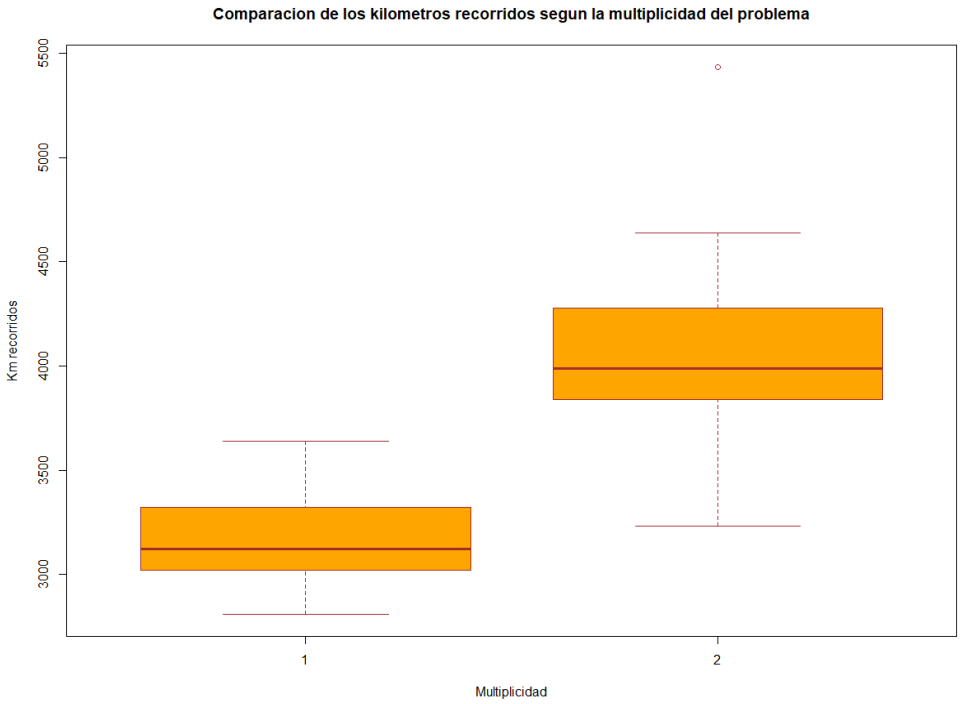
Boxplots comparando los beneficios obtenidos según la multiplicidad de los centros.



Boxplots comparando los pasos realizados según la multiplicidad de los centros.



Boxplots comparando los tiempos de ejecución obtenidos según la multiplicidad de los centros.



Boxplots comparando los kilómetros recorridos según la multiplicidad de los centros.

Conclusiones

En primer lugar, trataremos los beneficios: por un lado, es observable que, con multiplicidad=1, el rango de beneficios obtenibles es mucho más acotado que en el otro caso, pero esto, a su vez, implica que, con toda seguridad, si se obtienen beneficios siempre serán altos (eso sí, dado el contexto actual del problema). En el caso de multiplicidad=2, podemos detectar claramente más disparidad entre las muestras recogidas, lo cual termina provocando que su media sea inferior a la de su contrincante. Que los beneficios registrados dependen de la distancia recorrida y las peticiones desatendidas es un hecho conocido por todos, pero para determinar si verdaderamente estos datos son una consecuencia de ello, deberemos analizar los otros elementos aquí expuestos.

En segundo lugar, respecto al número de pasos, en el caso de multiplicidad=1, se tiene más margen de maniobra y más capacidad de exploración del estado de posibles soluciones. Dicha mayor capacidad podría venir dada por la multiplicidad del problema, suposición que se refuerza más ante los datos referentes al caso de multiplicidad=2: la media de pasos es menor, pero incluso cabe la posibilidad de realizar bastantes menos pasos que su adversario. En este sentido, podemos empezar a intuir que, efectivamente, la multiplicidad es la culpable de dicha diferencia.

En tercer lugar, en lo que atañe al tiempo de ejecución, podemos constatar que el caso de multiplicidad=1 es el más costoso en términos de temporalidad: mayor #pasos y

mayores beneficios parece implicar una mayor inversión de tiempo en la exploración de posibles estados solución. Por su parte, el caso de multiplicidad=2 presenta, de media, menos tiempo de ejecución, con lo cual tarda menos en presentar una solución óptima al problema, lo que, a su vez, también implica (de manera probable) una menor exploración del espacio de soluciones.

En cuarto y último lugar, tenemos la cantidad de kilómetros recorridos por cada una de las situaciones aquí tratadas: en cuanto al caso de multiplicidad=1, vemos cómo, de manera universal, se recorren alrededor de 1000 km menos, de media, que en el caso de multiplicidad=2. ¿A qué podría deberse esto? Pareciera que es más útil tener muchos centros de distribución con pocos camiones que al inrevés, y verdaderamente, para las características del problema aquí tratado, es así: cuantos más centros de distribución se tengan, más probabilidades tendremos de que estos se encuentren dispersos entre sí y, por lo tanto, que desde cada uno de ellos tengamos acceso a más peticiones (y, evidentemente, con una menor distancia por recorrer). Dadas estas circunstancias, parece razonable que, si reducimos el número de centros y aumentamos la multiplicidad, estamos reduciendo el tamaño del espacio de posibles soluciones, mientras que aumentar el número de centros y disminuir la multiplicidad (muy probablemente) la aumentaría.

En definitiva, a menor número de centros y más multiplicidad, en el contexto de este experimento, implica recorrer mayores distancias y, por lo tanto, registrar menores beneficios, mientras que hacer lo contrario (en este caso mantener un mayor número de centros y menor multiplicidad) implica recorrer menos distancia y, por ende, registrar más beneficios.

Experimento 6: Influencia del coste por kilómetro recorrido en el número de peticiones servidas de la solución final

Otro de los parámetros fijos que teníamos hasta ahora era el coste por kilómetro recorrido (2). Para comprobar cómo afecta este valor a la solución final del problema, vamos a ir aumentando el coste del kilómetro para ver el comportamiento del algoritmo elegido a la hora de generar la solución inicial y a la hora de aplicar los operadores.

El resto de variables que teníamos en el problema seguirán constantes para poder únicamente ver cómo afecta el cambio en el coste.

Para este experimento, ampliaremos el criterio de valoración de la solución obtenida, incorporando una métrica de *proporción de peticiones sin atender* por muestra obtenida.

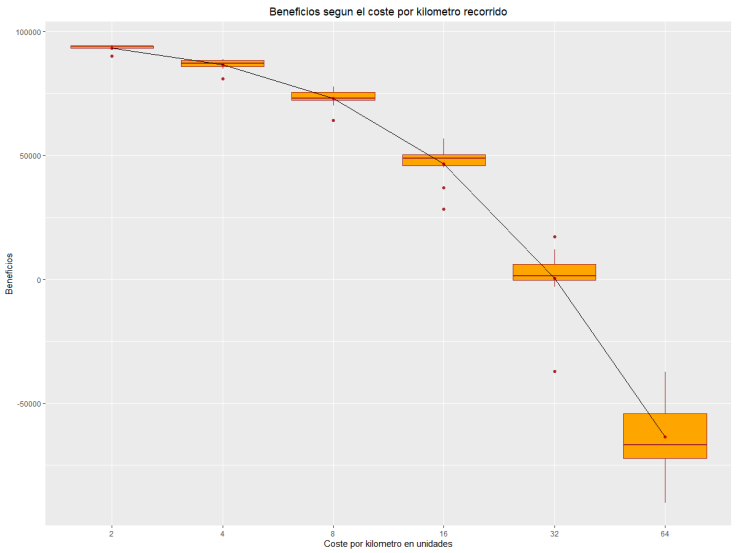
Para cada réplica, doblaremos el coste del kilómetro 5 veces y comprobaremos los resultados obtenidos en cada iteración.

Observación	El coste por kilómetro recorrido puede afectar al número de peticiones servidas y los beneficios obtenidos.
Planteamiento	Vamos doblando el coste por kilómetro con la misma entrada (10 centros con multiplicidad 1, con 1 camión por centro, es decir, 10 camiones y 100 gasolineras.
Hipótesis	Cuanto mayor coste se tenga por kilómetro, peores soluciones obtendremos.
Método	● Elegiremos seeds aleatorias. ● Para cada valor del máximo de kilómetros recorridos, realizaremos 10 pruebas para una entrada de 10 centros de distribución, 1 camión por centro, y 100 gasolineras. ● Los valores para la restricción tratada son: 2, 4, 8, 16, 32, 64. ● Usaremos el algoritmo de Hill Climbing. ● Mediremos el número de pasos ejecutados, peticiones servidas, el tiempo de ejecución y el beneficio generado.

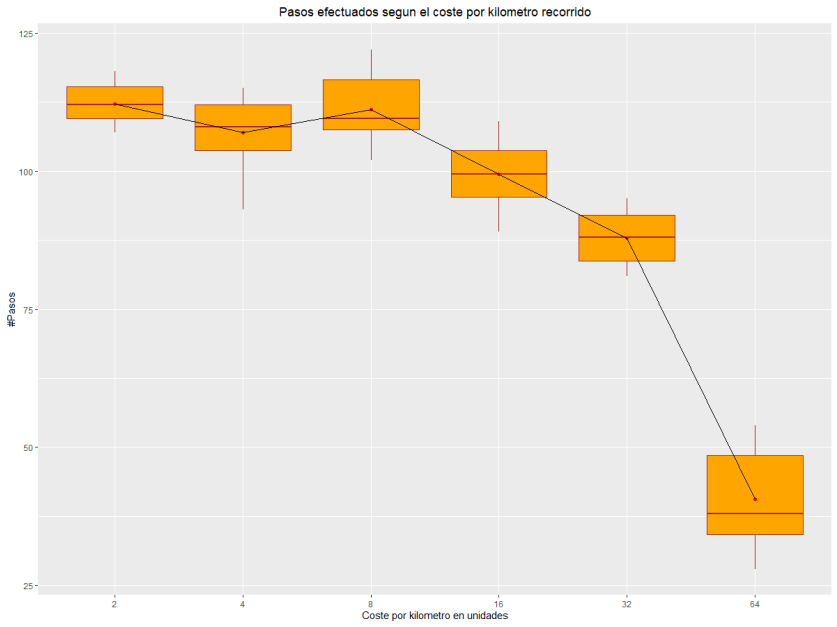
Resultados y gráficos del experimento:

Coste Km	Beneficio	Pasos	T eje (ms)	Proporción de peticiones sin atender
2	93479,6	112,2	1438,05419	0,2060159063
4	86628	107	1285,40139	0,2060159063
8	73214,8	106,9	1407,48126	0,2052466755
16	46608,8	111,1	1374,03369	0,2085800089
32	323,2	87,9	1107,11538	0,2707190197
64	-63707,6	40,6	295,44668	0,6146751452

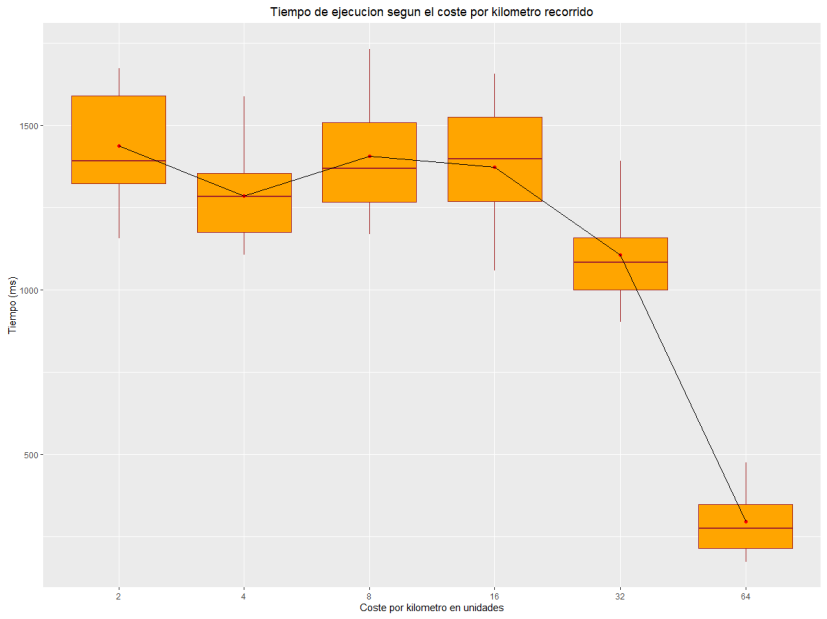
Tabla resumen de los resultados. (Los valores mostrados son la media de las diferentes pruebas)



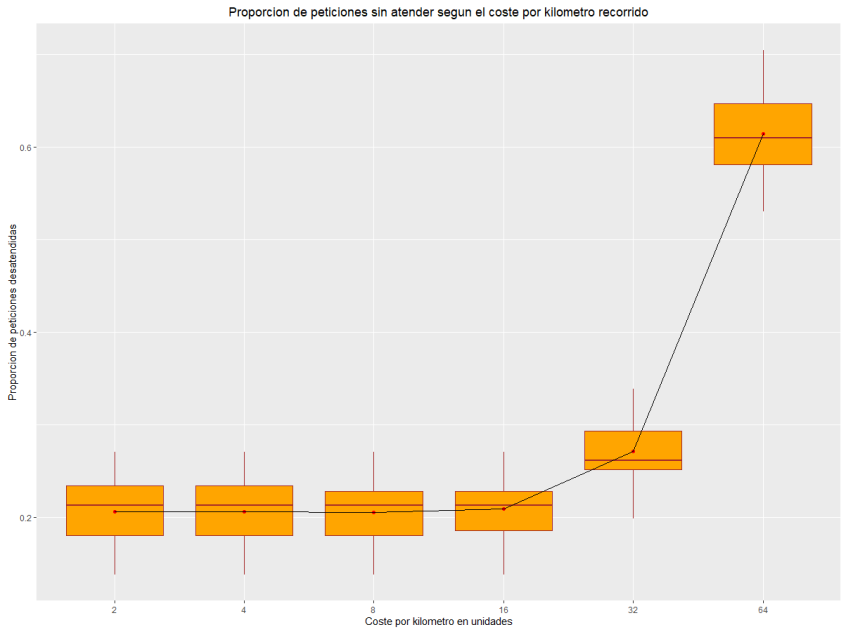
Boxplot representando la tendencia y disparidad de los beneficios obtenidos según el coste (en unidades) por kilómetro recorrido



Boxplot representando la tendencia y disparidad de los pasos realizados según el coste (en unidades) por kilómetro recorrido



Boxplot representando la tendencia y disparidad de los tiempos de ejecución registrados según el coste (en unidades) por kilómetro recorrido



Boxplot representando la tendencia y disparidad de la proporción de peticiones sin atender según el coste (en unidades) por kilómetro recorrido

Conclusiones

Los resultados obtenidos en este experimento nos proporcionan, en general, una visión clara sobre las implicaciones que tiene aumentar el coste del kilómetro.

En cuanto a los boxplots se refiere, podemos inferir lo siguiente: en el primer box plot, podemos observar como, en cada uno de los saltos, decrece el beneficio de forma exponencial; en el segundo y tercer box plot, observamos que el tiempo de ejecución y el número de pasos realizados por cada solución va disminuyendo de manera muy parecida a los beneficios generados; en el cuarto boxplot, se observa la situación inversa: para un coste por kilómetro pequeño, quedarán menos peticiones sin atender, pero para un coste muy elevado, tendremos una gran cantidad de las mismas..

Esto se explica ya que cada paso implica recorrer más kilómetros, y como la heurística utilizada prima los beneficios, si el coste del kilómetro aumenta, a más pasos más gastos, por lo que la heurística decide que es mejor realizar pocos pasos y servir pocas peticiones.

En definitiva, un mayor gasto implica mayores dificultades para generar beneficios, y si la heurística determina que los pasos posibles a realizar, en general, no reportan ganancias, realizará menos pasos, ello implica reducir el tiempo de ejecución y, por lo tanto, tener menos beneficios.

Experimento 7: Influencia del máximo de kilómetros recorridos en la solución final

Para este experimento, procederemos a cambiar una de las características hasta ahora fijadas: el máximo de kilómetros recorridos. Esto que modificaremos es la restricción que decía que no se puede sobrepasar el máximo de kilómetros recorridos por camión y día.

A pesar de desfijar este parámetro, mantendremos toda la información hasta ahora adquirida: mantenemos la función heurística, la estrategia de generación del estado inicial, el conjunto de operadores usados, etc.

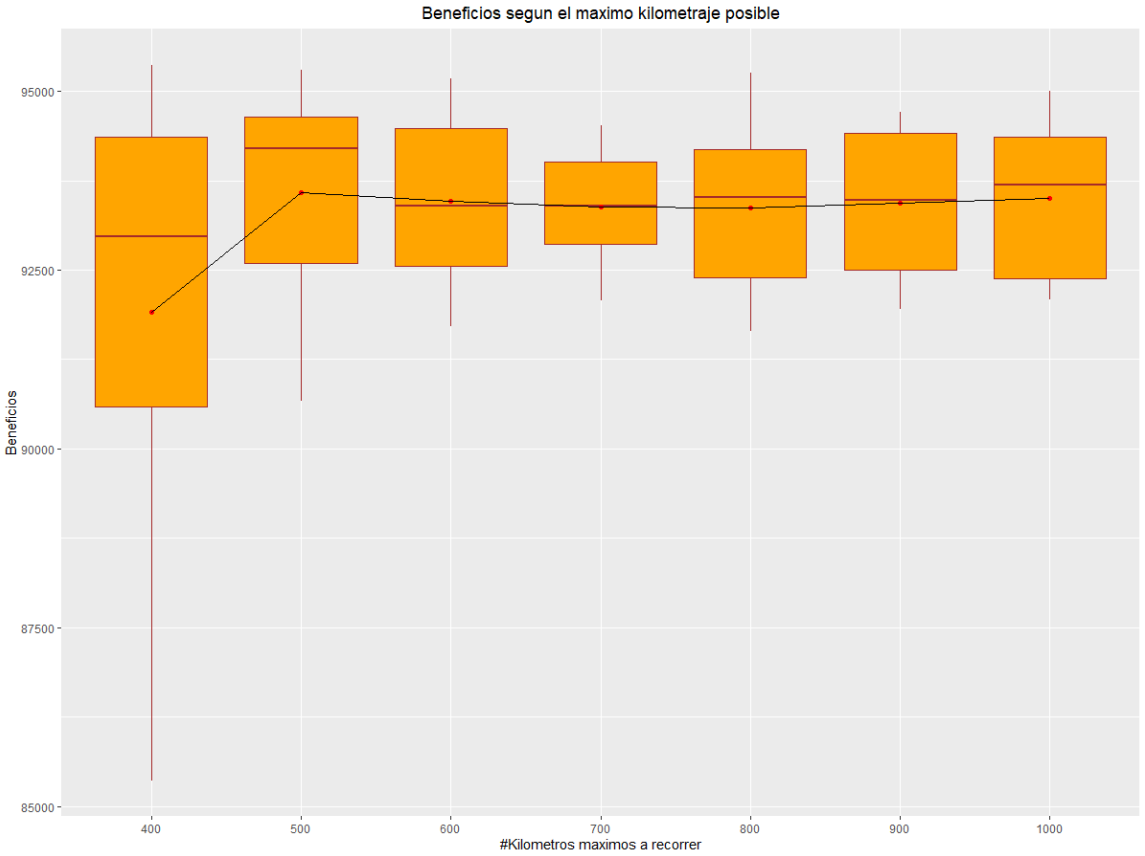
Los valores que le daremos irán desde los 400 hasta 1000 y se irán incrementando de 100 en 100.

Observación	Modificar el máximo de kilómetros para recorrer puede afectar a los beneficios.
Planteamiento	Comenzamos con un máximo de kilómetros bajo, y lo vamos aumentando.
Hipótesis	La modificación del máximo kilómetros por recorrer no tiene efectos significativos en los beneficios.
Método	● Elegiremos seeds aleatorias. ● Para cada valor, ejecutaremos 10 veces con la entrada del problema 5 centros de multiplicidad 2 y 100 gasolineras y diferente seed. ● Empezaremos con un valor inicial de 400 y a medida que vayamos haciendo las ejecuciones, lo iremos modificando con los siguientes valores: 500, 600, 700, 900 y 1000. ● Usaremos el algoritmo de Hill Climbing. ● Mediremos el número de pasos ejecutados, peticiones servidas y el tiempo de ejecución.

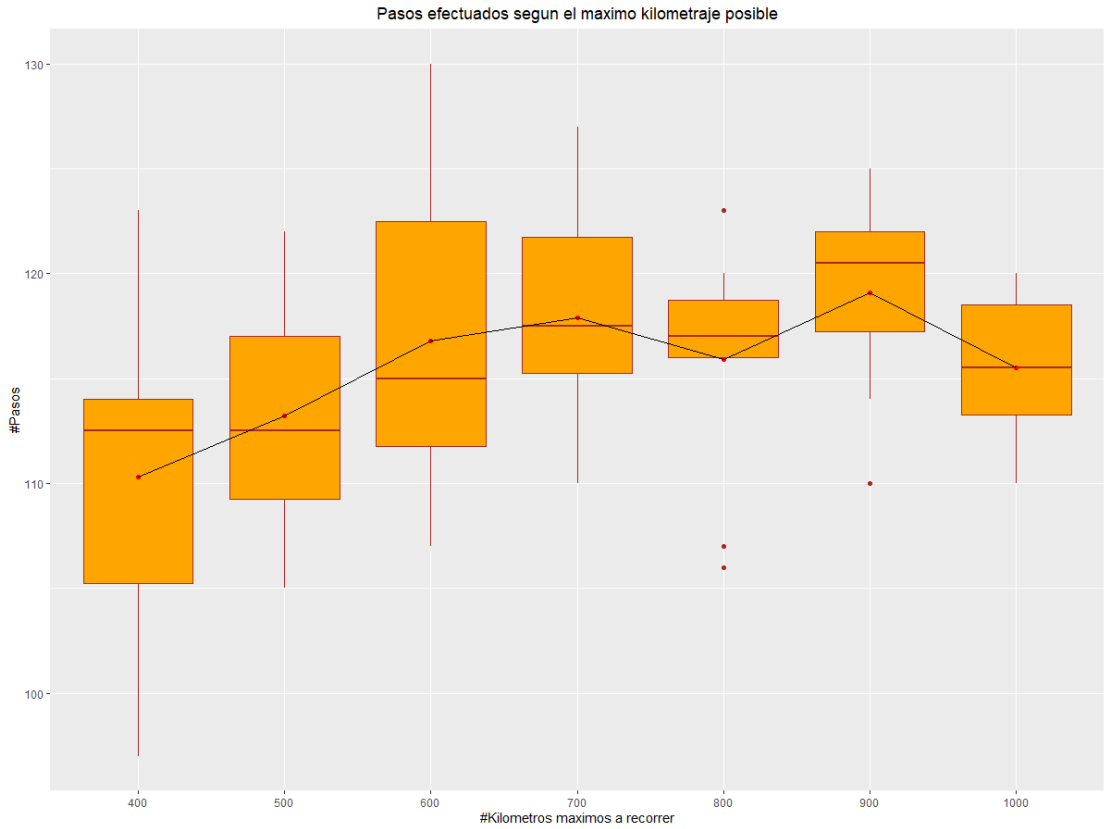
Resultados y gráficos del experimento:

Kilómetros máximos a recorrer	Beneficio	Nodos expandidos	T eje (s)
400	91906	110,3	972,7500929
500	93585,2	113,2	1156,518774
600	93468,8	116,8	1400,247421
700	93388	117,9	1540,509919
800	93373,2	115,9	1772,586354
900	93434,8	119,1	175097602,8
1000	93500	115,5	1937,897733

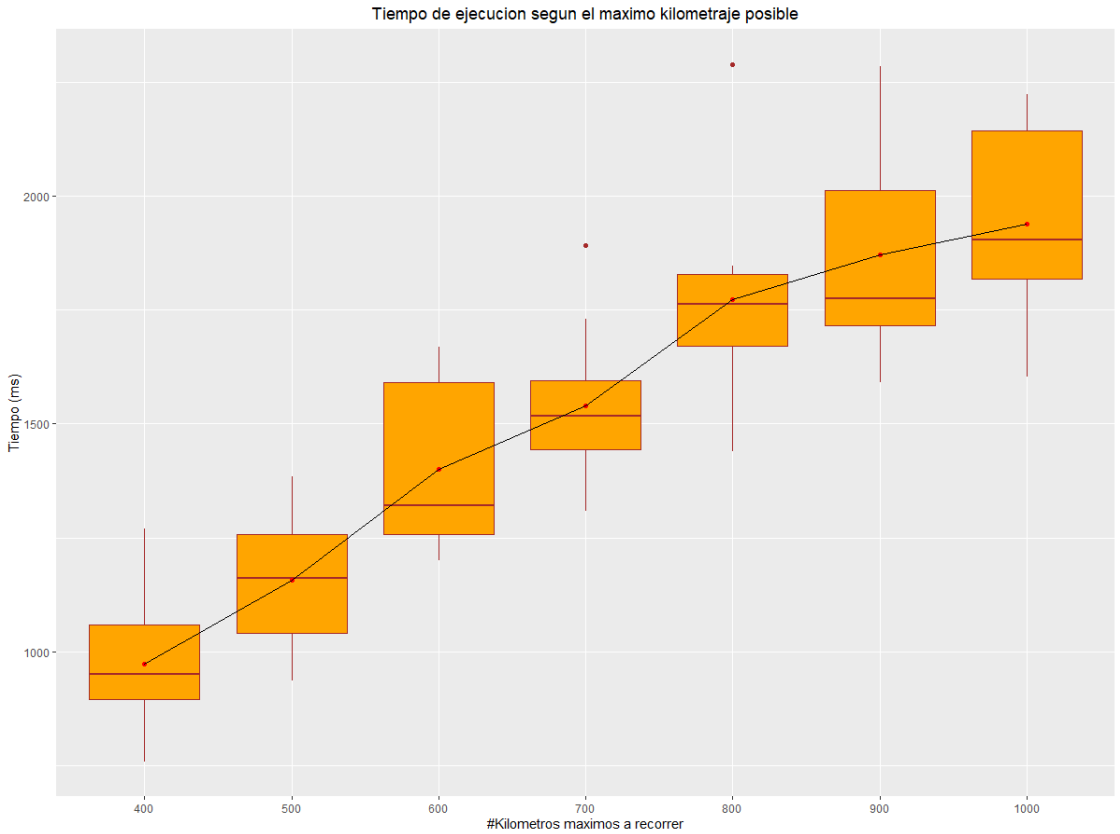
Tabla resumen de los resultados. (Los valores mostrados son la media de las diferentes pruebas)



Boxplot representando la tendencia y disparidad de los beneficios obtenidos según el máximo kilometraje posible



Boxplot representando la tendencia y disparidad de los pasos efectuados según el máximo kilometraje posible



Boxplot representando la tendencia y disparidad de los tiempos de ejecución registrados según el máximo kilometraje posible

Conclusiones

En primer lugar, trataremos los beneficios. Por un lado, podemos observar en la primera gráfica, que a medida que vamos aumentando los kilómetros, llega un momento en que se estabilizan y no varían apenas (a partir de los 500 kilómetros recorridos). También podemos observar que si vamos disminuyendo de 500 a 400 en adelante, la tendencia será a ir disminuyendo los beneficios, ya que cada vez pueden llegar a menos gasolineras, por lo tanto podrán servir menos peticiones y, en consecuencia, los beneficios disminuyen. Aunque no se vea en el gráfico, también podemos intuir que irán bajando los beneficios, como acabamos de explicar, hasta que llegue un punto en el cual los kilómetros recorridos sean lo suficientemente pequeños como para que ningún camión pueda ir y volver de la gasolinera más cercana, por lo tanto no harán ningún recorrido y entonces los beneficios serán de 0.

En segundo lugar, en lo que al número pasos respecta en la segunda gráfica, podemos ver cómo, a medida que vamos aumentando los kilómetros, también aumenta el número de pasos. Esto tiene sentido, ya que a medida que voy aumentando el número de pasos, nos permite llegar a más peticiones y por lo tanto el número de pasos aumentará debido a que hay más combinaciones para computar y se llegará a una profundidad mayor.
Al ser esta la tendencia esperada, puede ser que el antepenúltimo boxplots no representa lo que sucede ya que al fin y al cabo hemos hecho la media con 10 ejecuciones, pero puede ser que las semillas no hayan favorecido mucho a lo que suele pasar, ya que podemos ver que hay 3 outliers. Por lo tanto, se puede ver que la tendencia es a crecer hasta los 700 a 900 aproximadamente y luego estancarse.

En tercer lugar, en lo que atañe al tiempo de ejecución, podemos constatar que a medida que se aumenta el número de kilómetros, el tiempo aumenta. Esto tiene sentido ya que al aumentar el número de kilómetros, lo que estamos haciendo es dar más posibilidades a los camiones para llegar a más gasolineras, es decir, pueden servir más peticiones, con lo cual en número de nodos hijos también aumenta ya que hay más combinaciones que probar con los operadores y por esta razón, el tiempo de ejecución aumenta, debido al hecho de que hay que computar más nodos hijos en cada paso.