Lunes 1 Noviembre 2021

BÚSQUEDA LOCAL
ABASTECIMIENTO DE GASOLINERAS

Inteligencia Artificial
Universitat Politècnica de Catalunya
2021-2022 Q1

AUTORAS:
CARLA CAMPÀS GENÉ
BEATRIZ GOMES DA COSTA
LAIA VALLÈS GUILERA

1 INTRODUCCIÓN

Esta práctica tiene como objetivo experimentar con diferentes algoritmos de búsqueda local para encontrar una permutación eficaz de las rutas de abastecimiento. En específico, cómo optimizar las rutas que hacen diferentes camiones para distribuir sus recursos a diferentes localizaciones para

Los algoritmos usados para este experimento serán Hill Climbing y Simulated Annealing. Para realizar esta práctica se ha tenido que realizar un estudio de funcionamiento de los dos algoritmos previamente indicados así como del funcionamiento de las clases AIMA en java. Para completar el estudio, se han planteado diferentes soluciones al estado del problema y las funciones heurísticas y experimentado con estos para poder encontrar una solución optimizada.

2 DESCRIPCIÓN DEL PROBLEMA

Se plantea una compañía de distribución con diferentes centros logísticos, estos se encuentran en un mapa de $100\,x\,100\,km^2$. Cada uno de estos tiene un camión cisterna que lleva gasolina a las diferentes gasolineras. Las restricciones y atributos de cada elemento serán explicados posteriormente. El objetivo es encontrar la asignación de peticiones a centros y rutas que tienen que hacer los camiones para obtener el mayor beneficio.

2.1 Identificación del Problema

En este problema tenemos N centros logísticos y camiones cisterna (uno por centro logístico) estos tienen que distribuir gasolina a M gasolineras. Las gasolineras tienen depósitos que estos camiones han de llenar. Como cada gasolinera puede tener más de un depósito diremos que tenemos D depósitos a llenar.

Un punto en el espacio geográfico será definido como un centro de distribución o una gasolinera. El camión podrá ir de un centro de distribución a una gasolinera, de una gasolinera a otra o de una gasolinera a un centro de distribución. El cálculo de la distancia entre un camión y cualquier otro punto se hará usando la siguiente fórmula:

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

No todas las peticiones tienen que ser respondidas en un día, para cada petición que no sea respondida en el primer día. Si una petición se responde el mismo día que es recibida, el centro cobrará 102% del precio. Si pasa más de un dia la siguiente fórmula se usará para calcular el precio que cobrará un centro:

$$\%precio = (100 - 2^{días})\%$$

2.1.1 Elementos del Problema

- **Centros Logísticos:**
  Cada centro tiene un camión cisterna asignado, que deberá regresar a su respectivo centro de distribución una vez atendida la petición que se le haya asignado.

- **Camiones Cisterna:**
  La capacidad de las cisternas es de dos depósitos. Por tanto, en cada salida realizada por un camión podrán ser rellenados como mínimo un depósito de una gasolinera concreta, y como máximo dos depósitos (que pueden ser de la misma gasolinera, o de dos gasolineras distintas).

- **Gasolineras:**
  Cada una de las gasolineras dispone de dos depósitos. Cuando éstos se vacían, envían una solicitud para que sean rellenados de nuevo. Una gasolinera puede entonces, tener entre cero y dos peticiones de relleno de depósito. De éstas, se conocerá además el momento en que se hace la solicitud, pues este dato es necesario para el cálculo del precio que se pagará por esa gasolina.

2.2 Restricciones de la Solución y Análisis

El estado final o solución, debe cumplir con ciertos requisitos, que vienen dados por las restricciones del problema. Así que cualquier solución que no cumpla con las restricciones jamás llegará a explorarse y por lo tanto será descartada. Las restricciones en cuestión son las siguientes:

- **Acerca de los camiones:**
  Viajan a 80 Km/h, velocidad que consideraremos constante durante todo el trayecto. Están operativos durante un máximo de 8 horas, por lo que como máximo podrán recorrer una distancia de 640 Km diarios. Por simplificar, el tiempo de carga y descarga no será tenido en cuenta (asumimos que es instantáneo). Por otra parte, cinco es el máximo número de viajes permitidos por cada cisterna, aunque le queden suficientes kilómetros para atender otras peticiones.

- **Optimización:**
  El objetivo es maximizar el beneficio económico de las distribuidoras. Como hemos mencionado anteriormente el precio por rellenar un depósito disminuye en función de los días que lleva la petición sin ser atendida. Queremos asegurar que la solución encontrada maximice dentro de lo posible el beneficio de la distribuidora. El precio máximo por el relleno de un depósito (es decir, por rellenar el depósito el mismo día que se hace la petición), es de 1000.

  Por último, tenemos en cuenta la existencia de un coste fijo por kilómetro recorrido. Este coste es de 2 por kilómetro, de modo que la solución garantiza también la minimización de la distancia recorrida por cada uno de los camiones cisterna.

2.3 Criterios de Evaluación de la Solución y Análisis

El enunciado define el objetivo principal como la maximización del beneficio de todos los centros. Aparte de esto también define otros criterios de éxito:

- **Minimizar lo que perdemos con las peticiones no atendidas.**
  Estas pueden ser tanto peticiones que nos llegan el día que analizamos, o peticiones que nos llegan con días de retraso acumulados. Para esto deberíamos comparar la prioridad de una petición con los días que lleva sin atenderse, y adecuar las prioridades.
- **Minimizar la distancia total recorrida.**
  Esta debería ser considerada para cada camión cisterna. También tenemos en consideración que si cada camión cisterna recorre menor distancia, se puede maximizar el número de gasolineras al que un camión cisterna puede llegar en un día. Tal y como en el anterior, debemos adecuar las prioridades teniendo esto en consideración.

Para incluir estos en nuestros experimentos habrá que dar a estos atributos cierta ponderación para elegir la solución más adecuada.

2.4 Justificación del uso de Búsqueda Local

Nuestro objetivo es encontrar una solución entre las mejores posibles que se encuentren a nuestra disposición dentro de un tiempo razonable. El camino que nos lleve hasta dicha solución relativamente óptima, no tiene un papel significativo en este problema. Estos dos factores comienzan a justificar la prevalencia de la búsqueda local sobre la heurística.

Además, debemos tener en cuenta que el tamaño del espacio de búsqueda es muy amplio; con un elevado número de camiones y gasolineras, la cantidad de estados explorables crece rápidamente, lo que imposibilita la exploración del espacio al completo en un tiempo aceptable. Veremos más adelante, cuando discutamos los operadores que hemos seleccionado para resolver este problema, que todos tienen un factor de ramificación considerable. Es decir, que la aplicación de cada uno de ellos puede dar paso a numerosos estados sucesores, que aumentan con el tamaño del problema planteado.

Por lo tanto, la búsqueda local es nuestra mejor opción para tratar de dar una solución relativamente óptima a este problema en un tiempo razonable. Partiremos pues, de una solución inicial que nos acercará a esa solución parcialmente óptima, aplicando operadores sobre nuestro estado-solución inicial.

3 IMPLEMENTACIÓN DEL PROYECTO

Para poder implementar y experimentar este escenario con los algoritmos en mente tenemos que tomar decisiones informadas para poder implementar estos algoritmos de forma que optimizaremos la solución y como encontramos esta. Para esto, primero hemos estudiado los dos algoritmos y la implementación de los dos y posteriormente hemos visto diferentes posibilidades del estado del problema y sus posibles representaciones, los operadores que podríamos implementar, las funciones heurísticas y el estado inicial. Estos son detallados a continuación.

3.1 Hill Climbing

El algoritmo de hill climbing nos intenta encontrar un máximo local, en nuestro caso, intentando encontrar el máximo beneficio posible. Este algoritmo selecciona el vecino que proporcione el incremento más grande, cuando no se encuentre vecino con valor más grande, determina que es un máximo local y devuelve este. A continuación enseñamos el algoritmo en pseudocódigo.

```
function HILL-CLIMBING (problem) returns a state that is a local maximum
    current <- problem.INITIAL
    while true do
        neighbour <- a highest-valued successor state of current
        if VALUE(neighbour) ≤ VALUE(current) then return current
        current <- neighbour
```

Hill Climbing no mira más allá de sus vecinos directos. A veces este se llama greedy local search porque coge el mejor vecino sin pensar de dónde va este. El problema con hill climbing es que el algoritmo se puede quedar atascado de las siguientes maneras:

- **Máximo Local**: si el máximo local que encuentra primero no es el máximo global, el algoritmo va devolver este como el máximo, por lo tanto hay casos que no devuelve la solución óptima.
- **Crestas**: en el caso de las crestas el hill climbing tiene mucha dificultad para encontrar el máximo local

- **Mesetas:** el algoritmo se puede perder en una meseta, y por lo tanto sería ineficiente.

3.2 Simulated Annealing

Simulated Annealing mezcla un algoritmo de randomización completa con el algoritmo de hill climbing. Hacemos un cambio de perspectiva de hill climbing a gradient descent, minimizando el coste. En el caso de nuestro problema esto sería minimizar la distancia recorrida por los camiones.

function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    current <- problem.INITIAL
    for t = 1 to infinity do
        T <- schedule(t)
        if T=0 then return current
        next <- randomly selected successor of *current*
        if ΔE > 0 then *current* <- next
        else current <- next with probability $e^{\Delta E/T}$

La diferencia más grande entre los algoritmos Hill Climbing y Simulated Annealing es que Hill Climbing usa el mejor vecino inmediato para determinar el siguiente movimiento pero Simulated Annealing determina el siguiente movimiento usando un movimiento randomizado.

3.3 Estado del problema y representación

Para representar el problema, definiremos primero lo que entendemos en éste como estado. Denominaremos estado a una asignación de rutas de satisfacción de peticiones a cada uno de los camiones cisterna de los que disponemos.

Una ruta, será una serie de viajes que hará una cisterna para rellenar uno o dos depósitos de la misma gasolinera o bien de dos distintas. Por lo tanto, para un camión cisterna existirán dos tipos de rutas:

- La **ruta vacía**, o aquella en la que al camión no se le ha asignado ninguna petición.
- Una **ruta con una o más peticiones** de relleno de depósitos. A lo largo del día, la cisterna deberá hacer sus respectivos viajes a las gasolineras y de vuelta a su distribuidora, en el orden que su ruta le indique. Por supuesto esta ruta debe tener en cuenta las restricciones del problema que han sido mencionadas anteriormente (apartado 2.2). Por otra parte, el estado considerado solución también deberá asegurar que se cumplan los criterios de optimización enumerados en el apartado 2.3.

Para la representación de este problema, usaremos las siguientes estructuras de datos:

- Una ArrayList de ArrayLists de peticiones en la que almacenaremos, para cada camión cisterna, qué peticiones le corresponde atender. Por lo tanto, en la posición i de la estructura, encontraremos una ArrayList con las Peticiones (implementadas como un Pair <Integer, Integer> donde el primer elemento indica la gasolinera y el segundo la petición dentro de ésta) que serán rellenadas por el camión i.
- Una segunda ArrayList en la que mantendremos actualizada la distancia que le queda por recorrer a cada camión. Es decir que en la posición i de esta ArrayList de enteros, tendremos el número de kilómetros que le quedan disponibles al camión por recorrer ese día. Esto significa también, que como deben respetarse las restricciones, nunca asignaremos a un camión una petición si ya ha alcanzado el máximo de peticiones, aunque le queden suficientes kilómetros por recorrer.

- Por último, contamos con una tercera estructura de datos en la que almacenaremos las peticiones que aún siguen desatendidas. La estructura escogida es un set de Strings, donde cada elemento es una petición representada como un String.

Las tres estructuras de datos escogidas nos parecen las más prácticas tanto para conocer cuál es el estado actual (consulta de las Arrays y el Set) como para explorar los estados vecinos (mediante la modificación de las estructuras).

Por lo que a las estructuras para asignaciones y distancias respecta, una ArrayList era la mejor candidata. Esta estructura de datos nos permite tanto consultar, como añadir y quitar Peticiones de cualquier índice, manteniendo siempre el orden en que han sido añadidos los elementos (en lugar de tenerlos ordenados según algún criterio), ya que para nuestro problema éste es muy importante.
Por otro lado, nos hemos decantado por el Set de Strings para almacenar las peticiones no asignadas. Si bien es cierto que ya disponíamos de esta información gracias a la clase gasolinera, tenerla en un Set que vamos modificando a medida que se asignan peticiones nos resulta mucho más práctico en varias ocasiones. El Set nos facilita, por ejemplo, la búsqueda así como la comprobación de peticiones que no han sido asignadas (no hay que iterar por cada una de las gasolineras y comprobar para cada petición si está asignada o no en alguna de las ArrayLists de asignaciones).

Finalmente, cabe añadir que nos decantamos por guardar las peticiones no asignadas como Strings y no Peticiones ya que nos facilita la comparación. La igualdad entre Strings es más sencilla gracias a la función String.equals(), que entre otros objetos que no implementan su versión. Y es que el uso del '==' en Java da falso puesto que para objetos que no tengan implementada una función para determinar igualdad, se comparan las respectivas direcciones de memoria y no la similitud entre los atributos de los dos objetos comparados.

3.4 Representación y Análisis de los Operadores

Como terminamos de ver, lo que determina el paso de un estado a otro es el cambio en la asignación de peticiones. De modo que eso es exactamente lo que harán nuestros operadores.

Definimos, por un lado el operador ***asignaPeticon*.** Como su nombre indica, este operador modificaría el estado, asignándole un camión cisterna a una petición de una gasolinera. Así que al terminar la aplicación de este operador, una petición previamente sin asignar, pasa a pertenecer a la ruta de una cisterna. Por supuesto, para que este operador pueda ser aplicado sobre un estado, la petición sobre la cual se aplique debe estar sin asignar a ningún camión, ni tampoco puede haber sido ya satisfecha.

En segundo lugar, definimos el operador ***intercambiaPeticiones*,** que dadas dos peticiones que han sido asignadas a dos cisternas distintas, intercambia sus camiones. De manera que después de aplicar este operador sobre el estado, si la petición p1 tenía asignado el camión c1 y la petición p2 tenía asignado el camión cisterna c2, p1 pasará a formar parte de la ruta de c2 y p2 será satisfecha por c1.

El siguiente operador, ***intercambiaOrden,*** que dadas dos peticiones que ya estén asignadas al mismo camión, les intercambie el orden. Por lo tanto, tras la aplicación de este operador, pese a que ambas peticiones serían atendidas por el mismo camión cisterna, la ruta de éste se vería modificada.

Después tenemos el operador ***cambiaPeticion,*** que dada una petición p1 y dos cisternas c1 y c2, si antes de aplicar el operador p1 estaba asignada a c1, ahora pasa a formar parte de la ruta de c2 y ya no pertenece a la de c1.

Finalmente el operador ***cambiaPeticionNoAsig,*** recibe como parámetros un camión c, una petición p perteneciente a este camión y finalmente una petición p1 que no ha estado asignada a ningún camión todavía. Tras la aplicación de este operador, el camión c pasa a tener asignada la petición p1 en la misma posición en la que estaba p, que ahora pasa a formar parte de las peticiones que no están asignadas a ningún camión. Sólo podemos aplicar este operador sobre una p que pertenezca al camión c, con una p1 que no esté asignada, y teniendo en cuenta que al hacer este cambio se sigan respetando los kilómetros máximos que tiene permitidos recorrer dicho camión en un solo día.

Con estos cuatro operadores podremos explorar con facilidad el espacio de búsqueda, puesto que nos permite modificar las rutas de relleno de las cisternas. El segundo operador definido es una versión más eficaz de una combinación del primer y el tercer operador, lo cual nos simplifica el paso de estado a estado sin tener que pasar por estados parciales que nos puedan ser irrelevantes. Descartamos otros tipos de operadores, como podrían serlo *añadir/quitar Cisterna*, *acercar/alejar Gasolinera*, *vaciarDepósito*, etc. pues son o bien física o bien logísticamente inaplicables a nuestro problema.

3.4.1 Análisis de Factor de Ramificación de los Operadores

El factor de ramificación determina el número máximo de nodos que podría expandir nuestro programa usando un cierto operador. Esto está bien analizarlo para el hill climbing, pero no nos será de mucha utilidad en el simulated annealing ya que vamos a coger un sucesor al azar. Con tal de simplificar el estudio del factor de ramificación vamos a determinar unos acrónimos:

- Nos referiremos al número de camiones como C
- Nos referiremos al número de peticiones como P
- Nos referiremos al número de peticiones no asignadas como NAP
- Nos referiremos al número máximo de peticiones en un camión como NPC

Asigna petición puede asignar un petición que aún no está asignada a ningún camión a cualquier camión que tenga kilómetros y viajes suficientes para atender esa petición. En el caso peor donde todas las peticiones están libres y todos los camiones tienen capacidad de atender esa petición tendremos un factor de ramificación de asigna petición de P*C. En el caso general tendremos que P en realidad es el número de peticiones no asignadas, por lo tanto nuestro factor de ramificación sería NAP*C.

Intercambia petición coge dos peticiones ya asignadas de dos camiones diferentes y las intercambia entre ellos. En el peor caso tendremos todas las peticiones asignadas entre todos los camiones, por lo tanto podremos escoger entre todas las peticiones. Por lo tanto tendríamos $\frac{P*(P-1)}{2}$. De tal manera que la primera petición que vamos a elegir puede ser entre todas las peticiones, pero la segunda no vamos a poder elegir la misma otra vez.

Intercambia orden puede elegir de dos peticiones asignadas al mismo camión. Por lo tanto su factor de ramificación seria $\frac{NPC*(NPC-1)}{2}$. De forma predeterminada en esta práctica nos viene dado que NPC = 10, por lo tanto nuestro factor de ramificación sería de 45 nodos.

Cambia petición nos coje una petición de un camión y la meterá en otro camión que tenga los recursos necesarios para satisfacer esta petición. Por lo tanto en el peor caso esta podría elegir una de las peticiones y asignarla a otro camión, dando un factor de ramificación de P*(C-1).

Cambia petición no asignada, nos coge una petición y la cambia con una petición ya asignada. Por lo tanto nos da factor de ramificación de NAP*(P - NAP).

3.5 Análisis de la Función Heurística

Nos encontramos ante el clásico problema de maximización de un beneficio, ya que es de interés del centro de distribución maximizar las ganancias y minimizar los costes. Por esta razón podemos usar una heurística que nos calcule el beneficio hipotético resultante de una solución, y le aplique una penalización que se describe a continuación. Los tres componentes de la heurística serán entonces los siguientes:

- Los **ingresos**, que vendrán dados por la suma de los precios de cada una de las peticiones atendidas, teniendo en cuenta que el precio de un servicio variará en función de la cantidad de días que un pedido haya estado en espera de ser atendido. Asumiendo que el valor de un depósito es de 1000, en caso de que la petición no lleve ningún día pendiente se cobrará el 102% del precio, eso es, 1020. En caso que la petición lleve uno o más días pendientes el precio *P* de una petición vendrá determinado por la fórmula

  $$P = 1000 * \frac{100 - 2\text{^}días}{100}$$

  dónde *días* es el número de días pendientes.. Mediante esta fórmula podemos observar que el precio de un servicio bajará de forma exponencial por cada día de más que tarde una petición a ser atendida, por lo que nuestra heurística también estará teniendo en cuenta el coste del retraso con una disminución de los ingresos, y por lo tanto, con una disminución de los beneficios, que a su vez implican un peor valor heurístico.
  La cantidad total de ingresos *I* será la suma de todos los precios *P* de las peticiones atendidas.

- Los **costes**, que vendrán dados por la distancia total recorrida entre todos los camiones multiplicada por el coste fijo de recorrer un kilómetro. Asumiendo que este coste es de 2, los costes de la operación vendrán dados por $C = 2 * distanciaTotal$

- Las **penalizaciones.** Aunque el experimento dure solamente un día, hay que tener en cuenta que las peticiones que se queden sin atender tendrán que ser atendidas en el futuro a un menor precio. Esto implica una pérdida de beneficio en el futuro, y por eso, una solución que deja muchas peticiones desatendidas, por alto que sea el beneficio para ese día, no puede considerarse óptima de forma incondicional. Para tener en cuenta este factor, se calcula la penalización como la suma de una constante *k* elevada a los días sin atender que lleva cada una de las peticiones. Esta constante *k* es la diferencia entre las dos heurísticas desarrolladas. En la primera heurística tenemos que $k = 2$; de forma que la penalización aplicada se corresponda con el coste de dejar sin atender el conjunto de peticiones no atendidas. En la segunda heurística $k = 4$, lo que establecerá una estrategia que tenderá aún más a no dejar peticiones pendientes.

Así, el beneficio vendrá dado por la cantidad de ingresos obtenidos sustrayendo los costes incurridos. Eso es, $Beneficio = Ingresos - Costes$, y de esa misma forma nuestras funciones heurísticas harán el cómputo de su valor restando la penalización a los beneficios.

3.6 Elección y Generación del Estado Inicial

El estado inicial de la búsqueda local tiene que representar una solución a nuestro problema, respetando todos sus limitaciones. Para generar esta solución exploramos varias posibilidades que esta solución inicial podría tomar y exploramos estas sus limitaciones y sus beneficios con tal de escoger las dos que mejor nos iban para nuestros experimentos. Las siguientes son las soluciones iniciales que habíamos escogido.

1. Asignar aleatoriamente las peticiones a los camiones hasta el máximo de kilómetros o viajes que cada camión podría recorrer.
2. Asignar aleatoriamente las peticiones a los camiones hasta que cada camión tenga el máximo de kilómetros, viajes o un número de peticiones igual a #peticiones/#camiones, o un reparto de peticiones igual entre cada camión.
3. Ordenar los peticiones por prioridad y asignar a los camiones hasta llegar al máximo de kilómetros o viajes que este puede hacer. Exploramos que esta prioridad se podría hacer de varias maneras:
    - Distancia a recorrer para recoger una petición
    - Peticiones más antiguas
    - Beneficio de recoger una petición (ponderación de la antigüedad de los peticiones y la distancia recorrida para llegar a esta)
    - Una ponderación de los tres componentes previamente descritos.

Dadas estas opciones la 1 y 2 son muy similares. Por lo tanto exploramos la diferencia entre estas. La primera no expresa dos limitaciones (dadas por el enunciado) que se respeten los kilómetros y el máximo de viajes. Mientras que la 2 nos introduce una nueva restricción, no podemos añadir más peticiones que #peticiones/#camiones. Vemos dos casos para esta restricción, el caso donde esta restricción es la que se impone, y el caso en el que no es así.

En el caso que nos suponga una restricción, por lo tanto no podemos añadir más peticiones a un camión porqué el número de peticiones, no estamos maximizando el beneficio que este nos podría aportar, por lo tanto en este caso no nos interesa utilizar esta restricción. En el caso que no nos suponga una restricción, se aplique una de las otras dos restricciones, no nos supondrá ni beneficioso ni un inconveniente. Dado que en un caso nos supone un inconveniente, y en el otro tampoco nos interesa vemos que la versión 1 nos aportará mejor solución inicial que la versión 2.

Mientras que en la tercera vemos diferentes variaciones que puede tomar el mismo algoritmo. La diferencia más importante entre estas es el hecho de que si ponderamos por peticiones más antiguas exclusivamente esto se puede hacer antes de asignar las peticiones a los camiones, por lo tanto optimizando el tiempo. Para las otras tres opciones vemos que tenemos que recorrer las peticiones para cada camión (o cada posición que estos representan). Vemos que en la asignación directa de peticiones más antiguas ganaríamos tiempo, peró el algoritmo nos daría una solución inicial con beneficio pésimo. Por lo tanto, podemos considerar que lo que ganamos en tiempo no nos compensa lo que perdemos en beneficio. Específicamente dado que este algoritmo solo se ejecutará una vez por búsqueda, nos podemos permitir aumentar este coste.

A continuación vemos estos algoritmos en más detalle.

3.6.1 Explicación y Justificación de Solución Inicial Aleatoria

La solución aleatoria que planteamos recorre todas las peticiones de las gasolineras y para cada una de estas la intenta asignar a un camión. Aunque a simple vista esta solución parece de muy baja complejidad cuando analizamos esta vemos que las peticiones que no pueden ser asignadas al primer camión que hemos conseguido mediante randomización vamos a perder la posibilidad de asignar esta petición. Lo cual nos podría resultar peligroso en el caso de que a algunos camiones les queden pocos kilómetros para recorrer y a otros camiones les queden muchos. Por lo tanto decidimos poner esto en un loop donde mientras no haya repetido el camión al que queremos acceder si no se puede asignar la petición continuaremos randomizado el número correspondiente al camión.

Para poder usar esta solución inicial tenemos que comprobar que está es realmente una solución, por lo tanto cumple con todas las restricciones. Estas son, que no hagan más de los kilómetros que se les indica como máximo y que no hagan más de 5 viajes. Estas dos cosas se comprueban al asignar la petición por lo tanto respetamos estas restricciones y nuestro estado inicial se puede considerar una solución del estado.

El coste de implementar esta solución será en caso peor de $O(P * C)$, donde P es el cardinal de las peticiones, el total de peticiones que hay, y C es el cardinal de los camiones. En cuanto a espacio esta solución usa O(C), ya que nos guardará un vector de booleanos para la comprobación de los camiones que han sido accedidos. Además de ser una solución inicial con un coste no muy alto, vemos que es una solución bastante interesante ya que nos dará diferentes resultados cada vez que ejecutemos el programa. Esto nos enseña dos cosas, la optimalidad de nuestros algoritmos en diferentes situaciones, y diferentes resultados para el mismo problema, que podría ser beneficioso.

3.6.2 Explicación y Justificación de Solución Inicial Ordenados por Prioridad

La solución inicial con las peticiones ordenadas nos da una estructura de ordenación de peticiones muy similar a la que estamos buscando con nuestra heurística. En este, para cada camión calculamos el orden de prioridad en la que deberíamos poner las peticiones que aún tenemos disponibles, y estas se asignan al camión correspondiente. De esta manera un camión nos cogerá las mejores peticiones que sigan activas para crear sus rutas.

Un problema que habíamos visto en este algoritmo era que, en cuanto cambiamos la multiplicidad nos daba soluciones exponencialmente peores para los camiones sucesivos en la misma posición. Esto hacía que nuestra solución inicial fuese muy ineficiente en casos de multiplicidad mayor que uno. Para arreglar esto, para cada coordenada distinta cogeremos todas las posiciones de camiones con las mismas coordenadas, y distribuiremos estas de forma que asigna las peticiones del primer camión que hemos encontrado al último, y después desde el último hasta el primero. De esta manera conseguimos una distribución más ecuánime de las peticiones en los camiones con coordenadas repetidas. Una vez visitada una coordenada, ya no tenemos que volver a visitarla.

En cuanto a complejidad temporal nuestro algoritmo, itera sobre todos los camiones, y en el peor caso (multiplicidad 1), deberá calcular para todos los camiones las peticiones más adecuadas. En este caso, calcularemos la ponderación de todas las peticiones para la que visitaremos todas las peticiones de todas las gasolineras. En total estos pasos serían $O(P * C)$.A continuación miramos todos los camiones que nos quedan para encontrar esos con la misma posición $O(C^2)$. Y volvemos a mirar todas las peticiones para asignarlas a nuestros camiones $O(P * C)$. En total tendríamos $O(2 * (P * C) + C^2)$. En cuanto a complejidad espacial vemos que guardamos todas las

coordenadas en el caso peor (multiplicidad 1) esto serán coordenadas diferentes para cada camión. Además guardamos todas las peticiones, esto no da un coste espacial total de $O(P + C)$.

Como se ha explicado anteriormente, esta solución respeta todas las restricciones impuestas por el problema, ya que antes de asignar las peticiones se va a comprobar que tanto los kilómetros como los viajes que pueden hacer los camiones se cumplan. Por lo tanto la consideramos una solución al problema en mano. Este estado inicial nos crea una solución adecuada al problema en mano ya que pondera las peticiones de manera muy similar a la que utiliza la heurística, por lo tanto nos dará una solución más cercana a nuestra solución final.

4 EXPERIMENTACIÓN

4.1 Influencia sobre los Operadores

Observación	Pueden haber operadores que obtienen mejores soluciones
Planteamiento	Escoger distintas combinaciones de operadores y observar las soluciones a las que nos llevan
Hipótesis	Cualquier combinación de operadores lleva a la misma solución (H_0) o existe una combinación de operadores que lleva a soluciones mejores (H_1)
Método	● Generar 10 semillas aleatorias. ● Ejecutar 1 experimento para cada semilla con cada combinación de operadores. ● Experimentar con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras ● Usar el algoritmo de Hill Climbing y la función heurística 1 ● Medir diferentes parámetros para realizar la comparación

Plantemos cinco operadores iniciales, a continuación los volvemos a exponer:

- Asignar petición a un camión (nos vamos a referir a este como AP de aquí en adelante)
- Intercambiar dos peticiones entre dos camiones distintos (nos vamos a referir a este como IP de aquí en adelante)
- Intercambiar el orden de peticiones dentro de los viajes de un camión (nos vamos a referir a este como IO de aquí en adelante)
- Cambiar petición de un camión a otro (nos vamos a referir a este como CPO de aquí en adelante)
- Cambiar petición con petición no asignada (nos vamos a referir a este como CPNA de aquí en adelante)

Inicialmente, planteamos hacer el experimento propuesto con los operadores individualmente para ver qué optimización nos daba cada uno. De estos experimentos obtenemos las siguientes medidas.

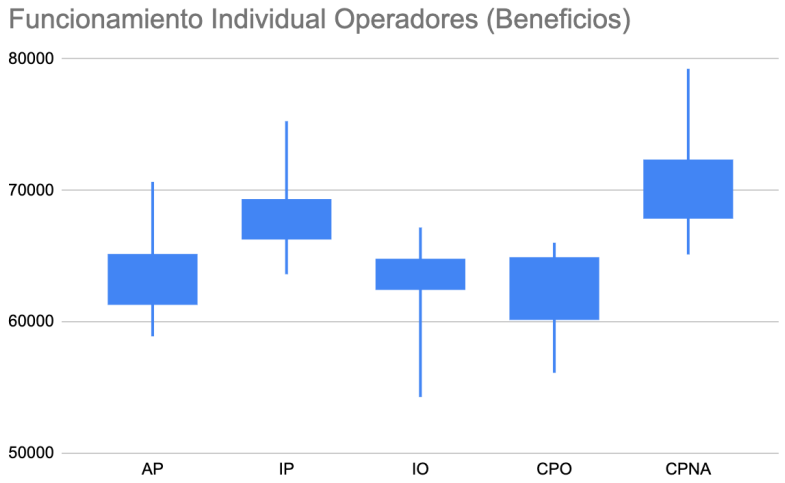
Para la realización de este experimento hemos decidido coger la solución inicial randomizada ya que nos proporciona una visión más amplia de cómo funciona nuestro algoritmo. También vemos que esta manera de inicialización nos puede dar diferentes soluciones para diferentes algoritmos, para mitigar esta diferencia vamos a ejecutar las pruebas cinco veces por operador o combinación de operadores para así tener resultados más normalizados.

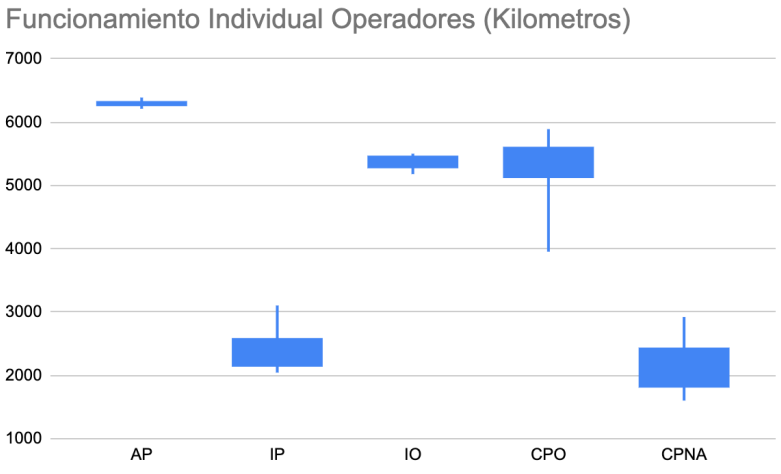
A continuación veremos la media de los resultados obtenidos. En esta exploración inicial nos basaremos en los kilómetros que recorre un camión, el beneficio que conseguimos con cada camión y el tiempo que nos tarda en ejecutar el algoritmo.

Operador	Beneficio	Kilometros	Tiempo de Ejecución
AP	63659,00 €	6299,50 km	47,4 ms
IP	68256,30 €	2409,8 km	6741,0 ms
IO	62921,10 €	5368,4 km	300,8 ms
CPO	62097,20 €	5195,4 km	17 ms
CPNA	70332,70 €	2193,6 km	5265,0 ms

Para esta parte del experimento hemos considerado que el tiempo no era relevante ya que no tiene ningún efecto en los resultados que podemos conseguir a continuación. Por lo tanto, notamos que los operadores de intercambio de peticiones nos dan un coste temporal más alto pero también tienen una repercusión alta en la bajada de kilómetros.

A continuación representamos los datos obtenidos de los operadores individuales en forma más visual, esto nos ayudará a sacar conclusiones más adecuadas.





Después de estas pruebas iniciales podemos hacer las siguientes observaciones, durante las diferentes pruebas el beneficio obtenido por cada operador se mantiene en un rango muy similar. De aquí podemos sacar varias conclusiones. Los operadores que minimizan más los kilómetros son IP y CPNA que también són los que más maximizan el beneficio del experimento.

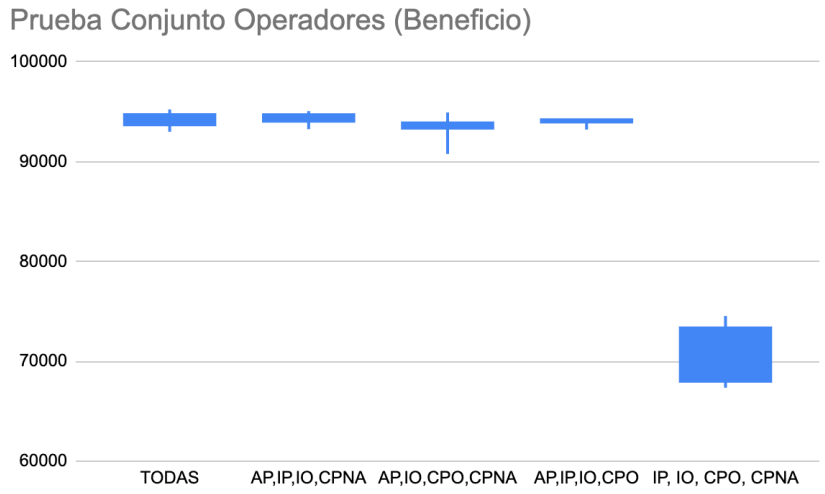
En cuanto a IP y CPO vemos que son operadores muy similares. Fundamentalmente, un intercambio de petición son dos cambios de peticiones a otro camión. Por esta razón, podemos suponer que la eliminación de uno de estos operadores no causaría grandes cambios en la ejecución de nuestro programa. También vemos esta similitud con los operadores de AP y CPNA. Esta tiene una pequeña diferencia, estas dos se usan en casos conversos. Si un camión ya tiene sus peticiones llenas, no podremos usar AP peró CPNA va a seguir siendo aplicable. Mientras que si un camión sigue teniendo espacio libre, CPNA no puede asignar a estos espacios libres, así que dejara trayectos sin ejecutar.

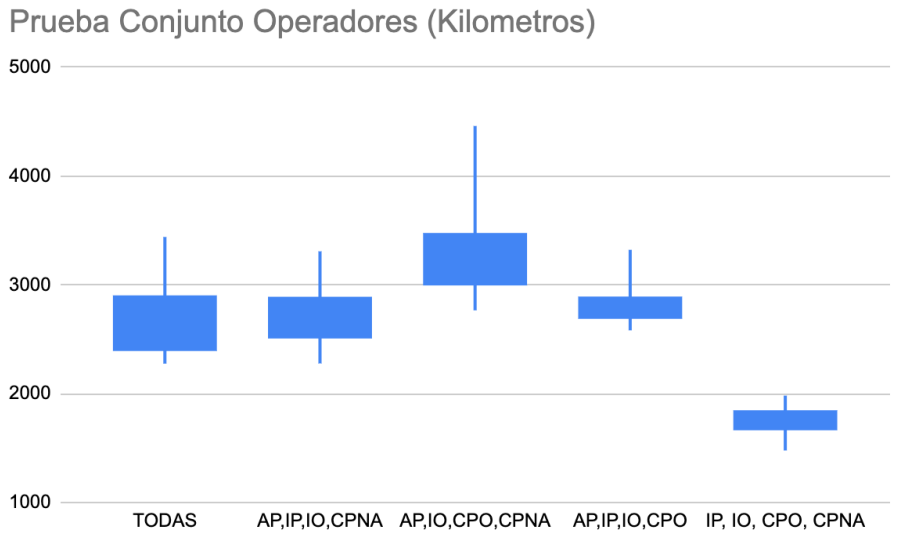
Para empezar la siguiente fase del estudio de operadores, primero miraremos el beneficio y los kilómetros que nos generaría una solución con todos los operadores activos, de esta manera podremos comparar con más facilidad los resultados siguientes. Con esto en mente, para realizar este experimento vamos a usar los siguientes conjuntos de operadores.

- AP, IP, IO, CPNA
- AP, IO, CPO, CPNA
- AP, IP, IO, CPO
- IP, IO, CPO, CNA

A continuación exponemos los datos extraídos de ejecutar con los diferentes operadores.

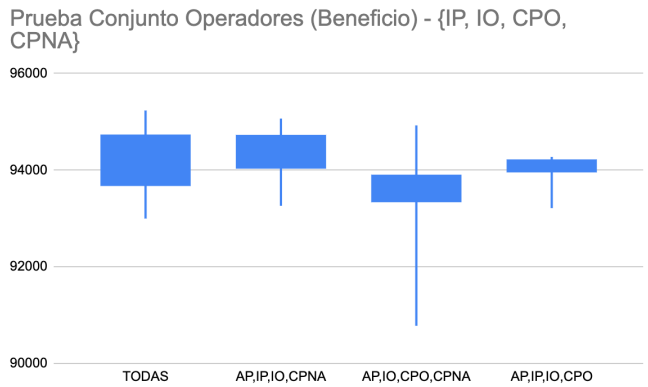
Operador	Beneficio	Kilometros	Tiempo de Ejecución
TODAS	94272,5 €	2695,4 km	14547,0 ms
AP,IP,IO,CPNA	94325,9 €	2704,6 km	14075,0 ms
AP,IO,CPO,CPNA	93379,8 €	3319,6 km	15870,0 ms
AP,IP,IO,CPO	93976,3 €	2838,4 km	12580,3 ms
IP, IO, CPO, CPNA	70503,9 €	1747 km	14934,6 ms

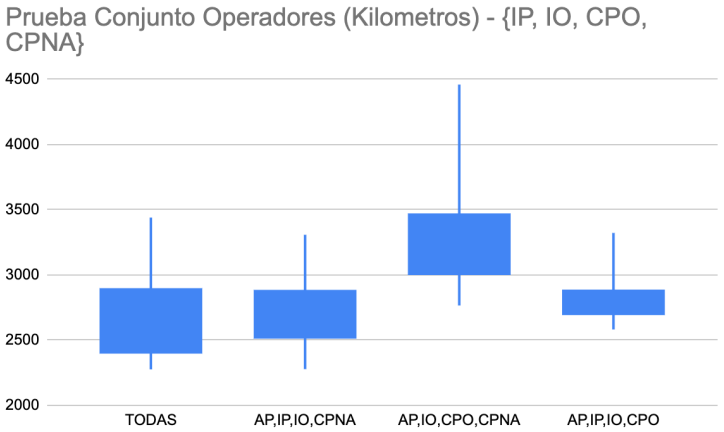




Vemos que si excluimos el parámetro de asigna petición nos da un beneficio mucho más bajo, esto también repercute en los kilómetros que hace el camión, estos se reducen significativamente. Esto es dado a que no asignamos las máximas peticiones posibles a cada camión, cuando las soluciones usando los otros conjuntos de operadores nos atenderán en la gran mayoría de casos el máximo número de peticiones posible. En el caso de nuestros experimentos, 100 peticiones. Dado que no hay ningún otro operador que pueda meter peticiones en camiones con espacio en sus viajes para cumplir más peticiones.

A continuación para ver los gráficos más claros y poder extraer más información presentamos la información sin la combinación de operadores {IP, IO, CPO, CPNA}.





Aquí vemos que las que mejor nos dan en beneficio y kilómetros está entre la combinación de todos los operadores o {AP, IP, IO, CPNA}. Analizando los datos vemos que todos los operadores juntos nos dan un beneficio más bajo y una variabilidad más alta. Aunque vemos que en kilómetros, hay veces que todos nos dan menos que la otra opción. También hay veces que nos dará más. Considerando la variabilidad de los datos, y el beneficio obtenido por {AP, IP, IO, CPNA}, vemos que en muchos casos ganamos en kilómetros y beneficio al conjunto de todos los operadores.

Como hemos dicho antes, CPO era muy similar a IP, por lo tanto quitar este operador no nos supone una restricción de las soluciones a las que podemos llegar. Si no una reducción en las formas que podemos llegar al mismo estado. Por lo tanto, podemos concluir que estos son los que mejor nos irán.

4.2 Influencia sobre la Solución Inicial

Observación	Pueden haber métodos de inicialización que obtienen mejores soluciones
Planteamiento	Escoger dos métodos de inicialización y observar sus soluciones
Hipótesis	Los dos métodos de inicialización son iguales (H_0) o un método es mejor que otro (H_1)
Método	• Generar 10 semillas aleatorias • Ejecutar 5 experimentos para cada semilla para la inicialización 1 (aleatoria) y calcular la media de los resultados • Ejecutar 1 experimento para cada semilla para la inicialización 2 • Experimentar con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras • Usar el algoritmo de Hill Climbing, la función heurística 1 y los operadores que han dado mejor resultado en el experimento anterior • Medir diferentes parámetros para realizar la comparación

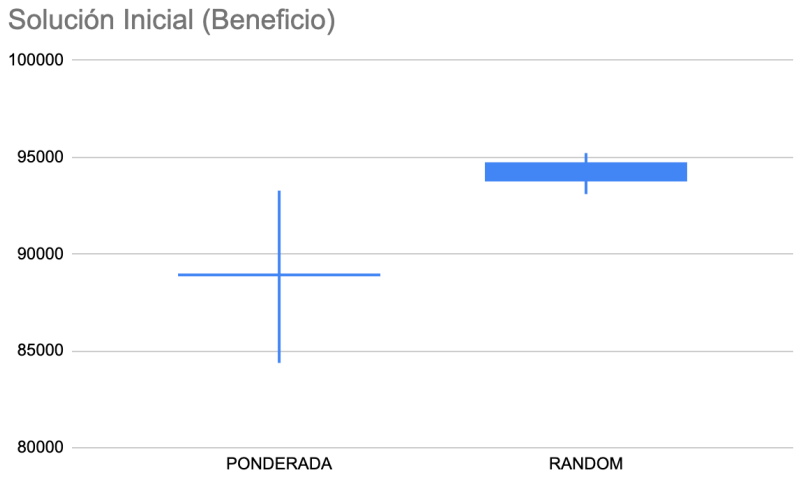
Hemos optado por crear dos soluciones iniciales, tal y como explicamos en el apartado 3.6. Para hacer estos experimentos vamos a usar los operadores que hemos decidido en el experimento visto en el 4.1.

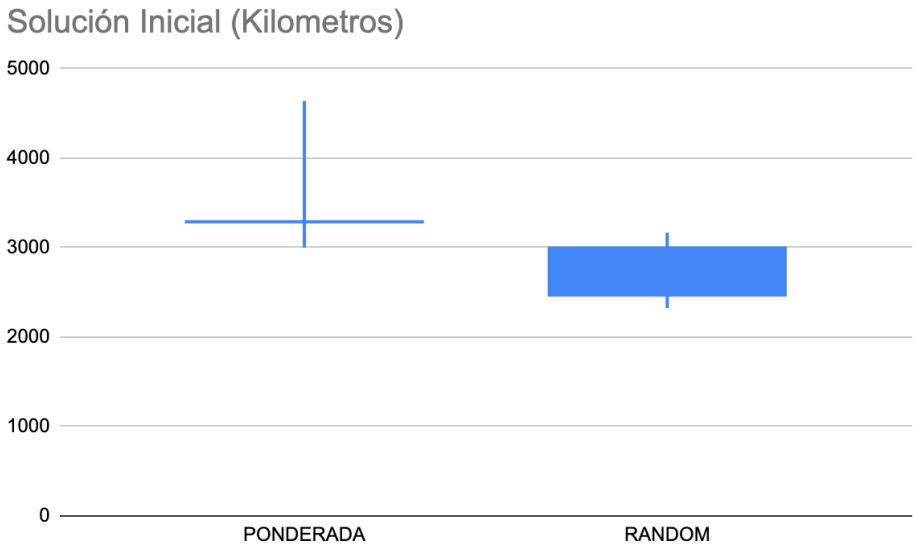
La primera solución es aleatoria por lo tanto nos dará resultados diferentes para cada ejecución. Por lo tanto hemos decidido hacer 5 ejecuciones por cada seed para mitigar la diferencia que nos puede dar cada ejecución del algoritmo. En cuanto a la solución ponderada nos dará siempre el mismo resultado por cada seed que utilizaremos, por lo tanto solo hace falta una ejecución por seed.

A continuación vemos los resultados obtenidos de la ejecución de este experimento.

Operador	**Beneficio**	**Kilometros**
ALEATORIA	94230,02 €	2730,54 km
PONDERADA	90161,10 €	3694,00 km

En este caso no nos interesa el tiempo ya que no contamos la inicialización del estado en el tiempo, por lo tanto no nos hace falta mirar este.





Podemos determinar que la solución aleatoria nos da mejores resultados. Esta nos da menos kilómetros y más beneficio que la que pondera el beneficio de las peticiones. También podemos ver que minimiza la variabilidad de los resultados en los diferentes seeds. Por lo tanto, consideraremos que la mejor solución inicial que tenemos es la random.

4.3 Influencia de los parámetros del algoritmo Simulated Annealing

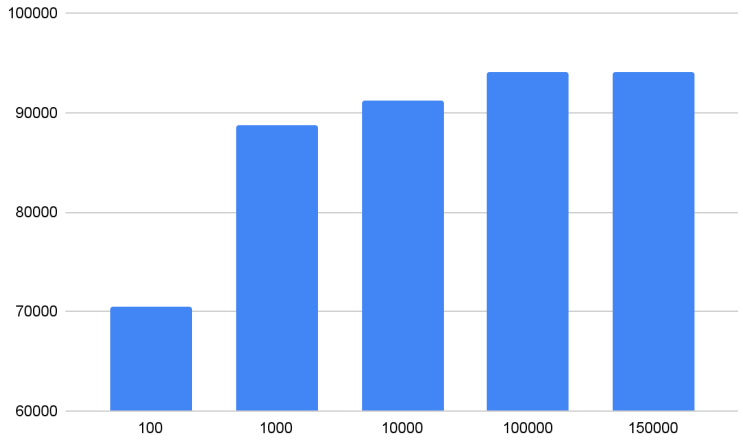
Observación	Pueden haber parámetros k y λ que obtengan mejores soluciones
Planteamiento	Escoger distintas combinaciones de k y λ y observar sus soluciones
Hipótesis	Cualquier combinación de k y λ lleva a la misma solución (H_0) o existe una combinación de k y λ que lleva a soluciones mejores (H_1)
Método	● Generar 10 semillas aleatorias. ● Ejecutar 1 experimento para cada semilla con cada combinación de k y λ ● Experimentar con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras ● Usar el algoritmo de Simulated Annealing, la función heurística 1, los operadores y el método de inicialización que han dado mejor resultado ● Medir diferentes parámetros para realizar la comparación

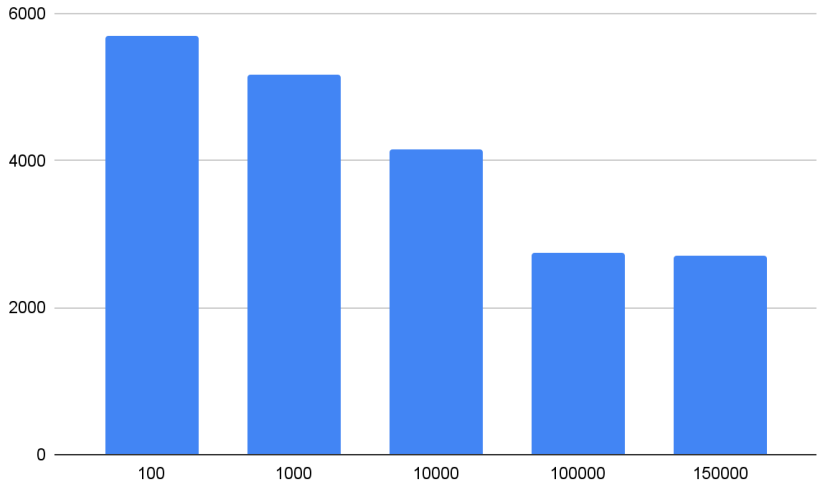
Una vez fijamos los operadores, la función heurística y el método para obtener la solución inicial, procedemos a calibrar los parámetros del Simulated Annealing. Estos son:

- **Steps:** que se encarga de poner un límite a la ejecución.
- **Stiter:** corresponde al número de iteraciones por cada paso de temperatura. No lo modificaremos para la realización de este experimento, y lo fijamos a 10.
- **k:** parámetro que influye en la función de aceptación de estados sucesores en el algoritmo.
- **λ:** parámetro que influye en la función de aceptación de estados sucesores en el algoritmo.

El procedimiento que seguiremos para determinar qué combinación de estos tres parámetros nos ofrece mejores resultados (si la hay), es el siguiente: fijamos k y λ a valores arbitrarios momentáneamente, y vamos variando únicamente los steps. Una vez encontremos el valor óptimo de iteraciones, lo mantenemos fijo para el resto de pruebas, y repetimos el proceso primero con el parámetro k, y finalmente con el parámetro λ. En el caso de existir una variación en los resultados, la combinación obtenida al final será la que nos habrá ofrecido los mejores.

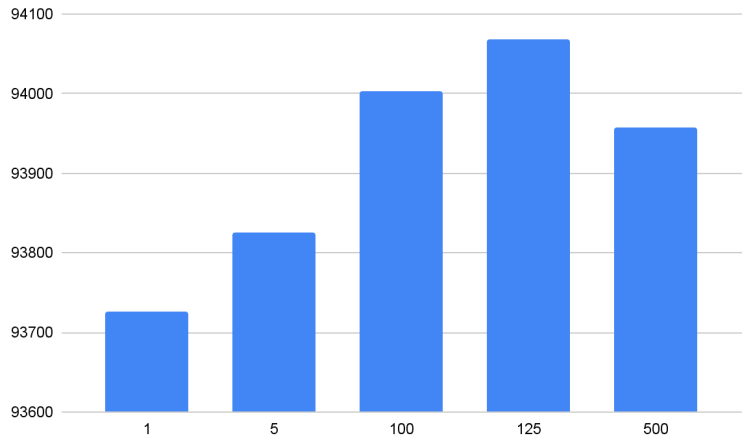
Variación en el número de steps o límite de iteraciones:
Probamos a ir aumentando los steps gradualmente hasta que vemos que el resultado ya no mejora. Los valores usados para los steps son 100, 1000, 10000, 100000 y finalmente, 150000. A partir del último valor vemos que los resultados se mantienen consistentes. Las siguientes tablas muestran la evolución de la media de los resultados del experimento (km recorridos y beneficio total obtenido) para las diez semillas probadas. Observamos que el rápido descenso de los kilómetros y el ascenso del beneficio se suavizan notablemente a partir de los 100000 steps. Escogemos 150000 dado que aunque de manera ínfima, la media de resultados para las diez semillas son ligeramente mejores.





Variación del parámetro k

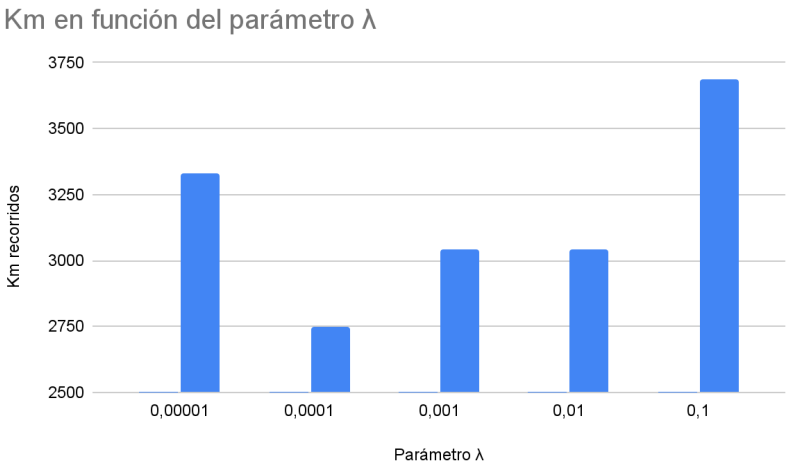
A continuación, habiendo fijado los steps a 150000, modificamos la k. El valor arbitrario que habíamos escogido inicialmente era bastante alto, de 125. Probamos ahora con los valores 1, 5, 100, 125 (ya realizado) 150, y 500. Nos damos cuenta de que es precisamente con la k de 125 que obtenemos el mejor rendimiento, pues al pasar a valores mayores, el resultado vuelve a empeorar notablemente. Los resultados obtenidos se describen en los siguientes gráficos.

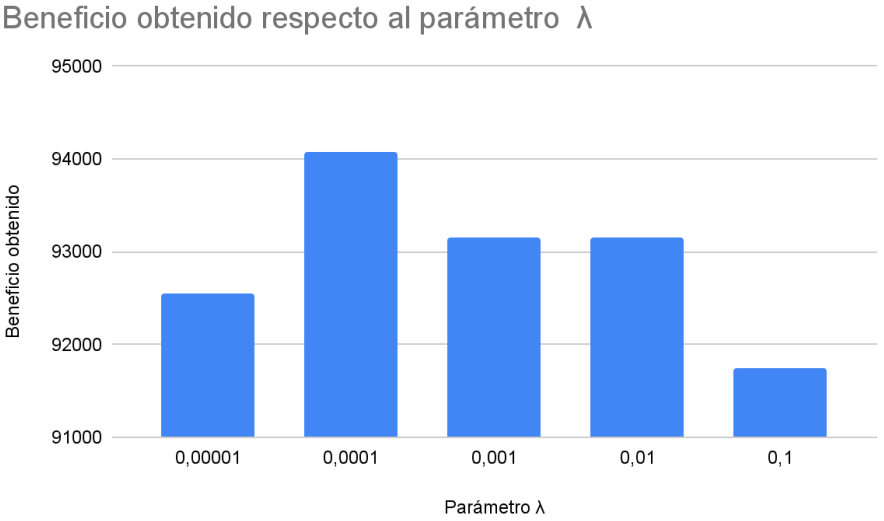


Variación del parámetro λ

Finalmente, repetimos todo el proceso con el parámetro λ. Probamos con los valores 0.000001, 0.00001, 0.0001, 0.001, 0.01, y 0.1. Después de ejecutar Simulated Annealing para nuestras 10 seeds, observamos que es con una λ de 0.0001 que obtenemos el mayor rendimiento. Mostramos el resultado de estas pruebas en los dos siguientes gráficos.

En primer lugar debemos determinar los steps





Por lo tanto, que los parámetros más adecuados para el Simulated Annealing son:

Steps: 150000, **k**: 125 y **λ**: 0.0001.

Valores más bajos para el máximo de iteraciones o steps nos ofrecen resultados peores, y con valores más altos, el rendimiento se mantiene estable. Y de una manera similar, tanto para valores más elevados como más bajos de los parámetros k y λ escogidos, observamos que el rendimiento es siempre peor (aumentan los kilómetros recorridos, disminuye el beneficio, etc.).

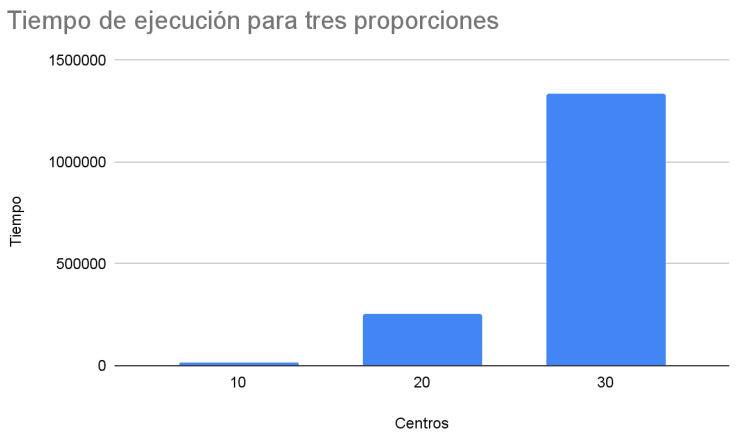
Finalmente, respecto a nuestras hipótesis, comprobamos que la primera no se cumple mientras que la segunda si. Nuestros gráficos ilustran cómo, efectivamente, obtenemos soluciones de calidades muy distintas en función de los parámetros que utilicemos. Éstos serán pues, los parámetros que usaremos para el simulated Annealing a partir de ahora.

4.4 Influencia del tamaño del problema

Observación	El tiempo sigue una función creciente respecto al tamaño del problema
Planteamiento	Observar los tiempos de ejecución para distintos tamaños del problema
Hipótesis	La función es al menos cuadrática respecto al tamaño del problema
Método	• Generar 10 semillas aleatorias. • Ejecutar un experimento para cada semilla con cada tamaño diferente del problema • Experimentar inicialmente con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras e incrementar de 10 en 10 el número de centros de distribución hasta que se vea la tendencia • Usar el algoritmo de Hill Climbing y Simulated Annealing con la función heurística anterior • Medir los tiempos de cómputo • Comprobar que los parámetros k y λ hallados en el experimento anterior siguen dando buenos resultados

Influencia del tamaño del problema para Hill Climbing

Con las 10 semillas arbitrarias que hemos usado, hemos resuelto el problema para las proporciones *nCentros:nGasolineras* 10:100, 20:200 y 3:300. Con tan sólo estas tres ejecuciones para las 10 semillas ya podemos observar la tendencia ascendente del tiempo. Después de resolver el problema para cada proporción, recogemos el tiempo (en ms) que ha tardado en encontrar dicha solución y hacemos la media. El gráfico siguiente muestra estos valores de media del tiempo para las tres proporciones observadas. La tabla muestra los valores medios exactos del tiempo, así como de los km y el beneficio, aunque no son relevantes para este experimento.

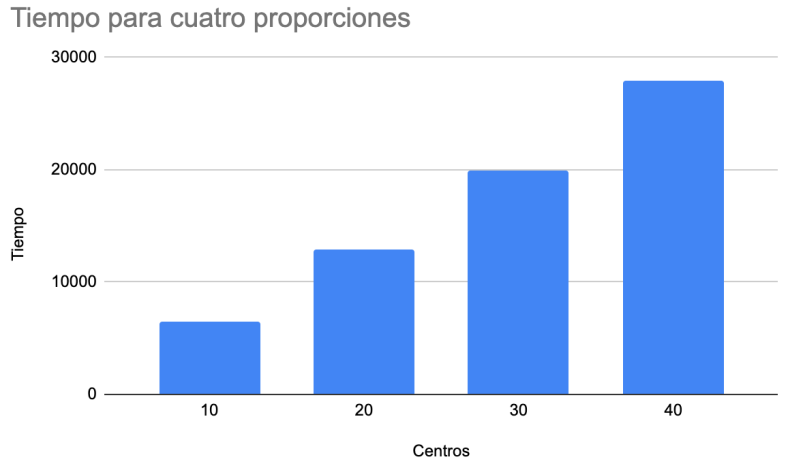


Centros	Gasolineras	Tiempo	Km	Beneficio
10	100	14428	2714,4	93556,8
20	200	256995,3	6206,4	187133,6
30	300	1334035,7	11276	281071,2

Observamos que de media, el tiempo de ejecución aumenta rápidamente al aumentar la densidad del problema. Esto se explica porque recordamos que nuestra función sucesora para el Hill Climbing debe generar todos los estados posibles para poder pasarlos al algoritmo de búsqueda. De modo que el número total de iteraciones que deben hacerse en la sucesora crece exponencialmente y en consecuencia el tiempo que tarda en ejecutarse el programa también lo hace. Pasamos de 14 segundos a unos 4 minutos, y finalmente a más de 22 minutos de media. Podemos predecir que, aunque para Simulated Annealing también aumentará el tiempo, éste no será tan elevado como para el algoritmo de Hill Climbing.

Influencia del tamaño del problema para Simulated Annealing

Con las mismas 10 semillas que hemos usado, resolvemos ahora el problema para las proporciones *nCentros:nGasolineras* 10:100, 20:200, 3:300 y 4:400. Estas cuatro ejecuciones para las 10 semillas nos bastan para observar la tendencia ascendente del tiempo. Después de resolver el problema para cada proporción, recogemos el tiempo que se ha tardado en encontrar dicha solución y volvemos a hacer la media. El gráfico siguiente muestra estos valores de media del tiempo para las tres proporciones observadas. La tabla muestra los valores medios exactos del tiempo, así como de los km y el beneficio.



Centros	Gasolineras	Tiempo	Km	Beneficio
10	100	6526	2780	94043,2
20	200	12958,1	4370,8	188570,8
30	300	19978,5	6043,4	286955
40	400	27872	7885,2	382976,6

Como cabía esperar, el aumento en el tiempo de ejecución no es tan pronunciado como para el Hill Climbing. Esto se debe a que en la sucesora para el Simulated Annealing no exploramos todos los nodos sucesores al estado actual, sino que solamente escogemos uno completamente al azar, para que el algoritmo se encargue de decidir si lo expande o no. Mientras que en el Hill Climbing alcanzamos tiempos superiores a los 22 minutos, con una proporción más de Simulated Annealing no llegamos al minuto.

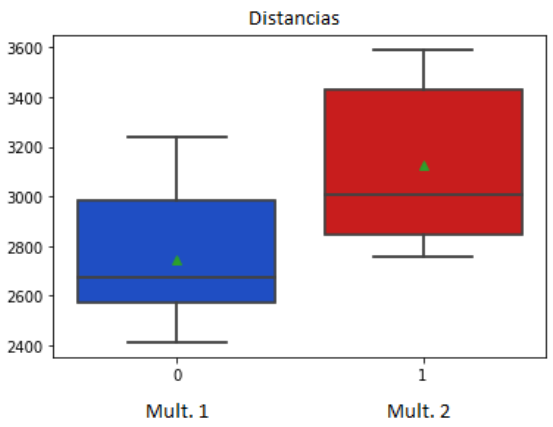
En ambos algoritmos, podemos observar también que los buenos resultados (km bajos y beneficios altos), se mantienen.

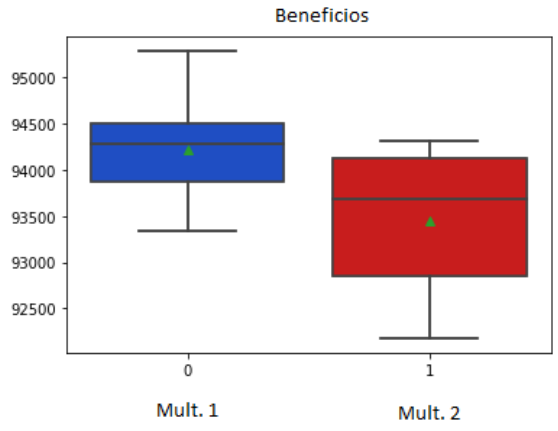
4.5 Influencia de la multiplicidad

Observación	Tener más de un camión por centro puede afectar a los beneficios y los kilómetros recorridos
Planteamiento	Reducir el número de centros a la mitad y observar los cambios en los beneficios y los kilómetros recorridos
Hipótesis	La multiplicidad no afecta a los beneficios y los kilómetros recorridos (H_0) o la multiplicidad puede tener algún efecto en los beneficios y los kilómetros recorridos (H_1)
Método	• Generar 10 semillas • Ejecutar 5 experimentos para cada semilla, ya que se usa un método randomizado para llegar a la solución inicial • Experimentar con problemas de 10 centros de distribución con 10 camiones y 100 gasolineras, y con problemas de 5 centros de distribución con 10 camiones y 100 gasolineras. • Usar el algoritmo de Hill Climbing con la función heurística 1 • Comprobar el beneficio y los kilómetros recorridos resultantes del problema con multiplicidad y compararlos a los del problema sin multiplicidad

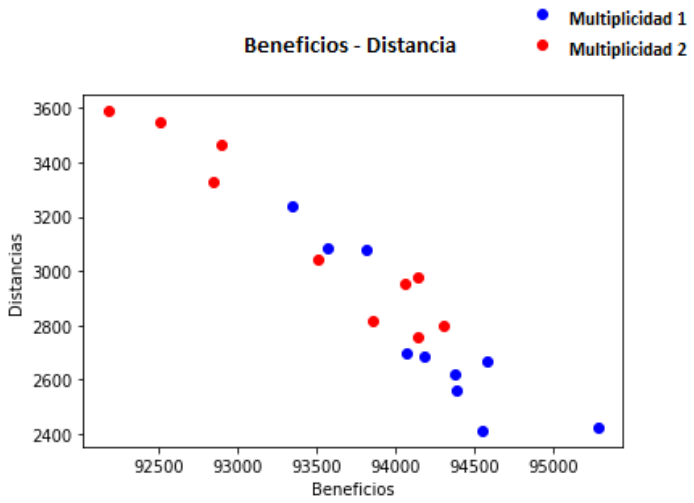
Este experimento consiste en comparar los principales resultados obtenidos en dos problemas cuya única diferencia es el número de centros de distribución. Para ello, se ha ejecutado el problema en el que teníamos una cisterna por centro de distribución y se han comparado las métricas de beneficios y distancias con las obtenidas con la ejecución de un problema en el que teníamos la mitad de centros (o, de forma más precisa, teníamos parejas de centros que compartían las mismas coordenadas, simulando tener más de un camión por centro).

Los siguientes gráficos muestran un resumen visual de las diferencias entre distancias y beneficios obtenidos en los dos problemas. En azul tenemos los resultados del primer problema (10 centros de distribución con 10 camiones y 100 gasolineras). En rojo, tenemos los resultados del problema con 5 centros, 10 camiones y 100 gasolineras, es decir, de multiplicidad 2 (dos camiones por centro).





Se puede observar que al tener 5 centros en vez de diez, manteniendo el número de camiones esto implica un aumento en las distancias, lo que comporta un aumento de los costes y por lo tanto también una disminución de los beneficios.



En el gráfico de dispersión anterior podemos observar como la nube de puntos azules (problema con un camión por centro) se desplaza hacia arriba a la izquierda en reducir el número de centros a la mitad (en rojo), indicando, de la misma forma que los diagramas de cajas anteriores, un aumento en las distancias recorridas y una disminución en los beneficios obtenidos.

Éste es un resultado esperable, dado que al perder la mitad de centros de distribución y habiendo la misma cantidad de gasolineras, tendremos menos margen para optimizar las rutas ya que los centros tendrán que servir a más gasolineras, y aunque ahora tengan dos camiones en vez de uno, perderemos la capacidad de minimizar las distancias entre las gasolineras ya que los camiones deberán salir y volver al mismo centro.

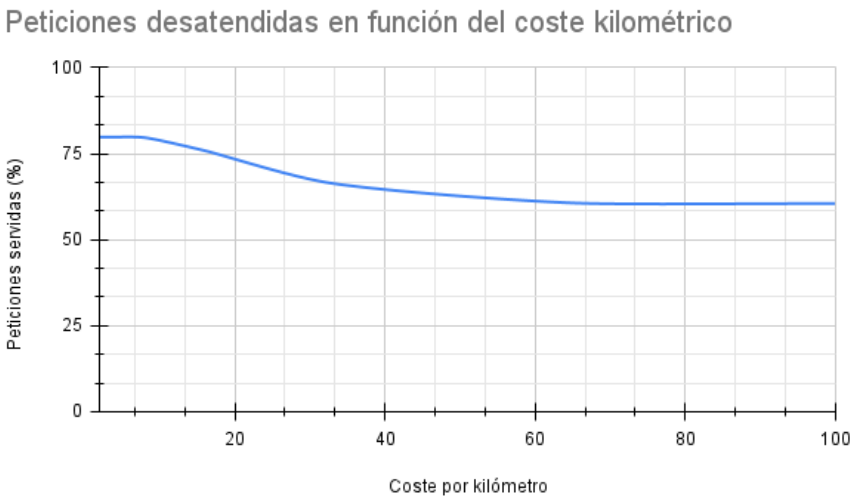
En un escenario real, se podría esperar el resultado contrario, ya que mantener más centros supone también mayores costes en la contratación de personal y en el mantenimiento de las instalaciones. Para tener en cuenta estas posibles circunstancias, se podría añadir un supuesto coste fijo por centro de distribución, que sería restado de los beneficios totales y de esta forma recortar el número de centros de distribución supondría por un lado la ventaja (para la empresa) de una reducción de costes en personal e instalaciones y por el otro lado el inconveniente de un potencial aumento de de la distancia recorrida por las cisternas, y no solamente un inconveniente como nos podrían sugerir los resultados de este experimento.

4.6 Influencia del coste fijo por kilómetro recorrido

Observación	El coste fijo por kilómetro recorrido puede afectar la proporción de peticiones servidas
Planteamiento	Observar la proporción de peticiones servidas para distintos costes por kilómetro que aumentan exponencialmente
Hipótesis	El coste fijo por kilómetro no afecta la proporción de peticiones servidas (H_0) o El coste fijo por kilómetro puede afectar la proporción de peticiones servidas (H_1)
Método	• Generar 10 semillas aleatorias • Ejecutar 5 experimentos para cada semilla, ya que se usa un método randomizado para llegar a la solución inicial • Experimentar con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras • Incrementar exponencialmente el coste por kilómetro hasta que se vea la tendencia • Usar el algoritmo de Hill Climbing con la función heurística 1 • Comprobar la tendencia de la proporción de peticiones servidas a medida que aumenta el coste fijo por kilómetro

Para este experimento se ha ejecutado el problema con diferentes valores para el coste por kilómetro empezando por 2 y doblándose cada vez hasta llegar a 128. A partir de esta cifra, se ha observado que un mayor aumento en el coste ya no generaba grandes cambios en la proporción de peticiones servidas, y por esa razón ya no se han considerado costes mayores a dicha cifra.

Mediante este experimento se ha podido observar que inicialmente con un valor pequeño, de entre 2 y 8 el porcentaje de peticiones servidas se mantiene más o menos constante alrededor del 80%. A medida que se aumenta el coste por kilómetro más allá de 16, el porcentaje de peticiones servidas sigue una tendencia a la baja, hasta un valor aproximado del 60% a partir de costes mayores a 60 por kilómetro. Más allá de esta cifra el porcentaje se mantiene constante. A continuación se muestran gráficamente los resultados del experimento.



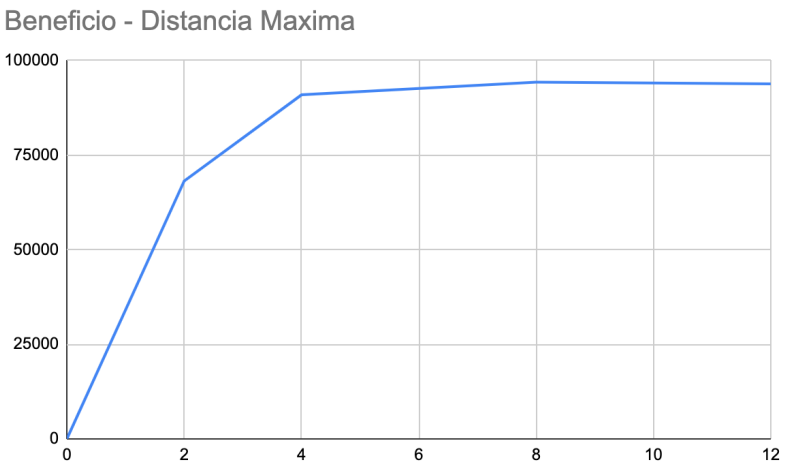
También se puede observar que aún considerando un coste por kilómetro muy alto, se continúan sirviendo una parte importante de las peticiones, lo que en un escenario real no ocurriría ya que causaría pérdidas a la empresa distribuidora. Esta diferencia podría ser reproducida por el algoritmo modificando la función heurística, de forma que devolviese un valor muy negativo para un estado cuya situación generase pérdidas, de forma que, a partir de cierto coste, no se serviría ninguna petición.

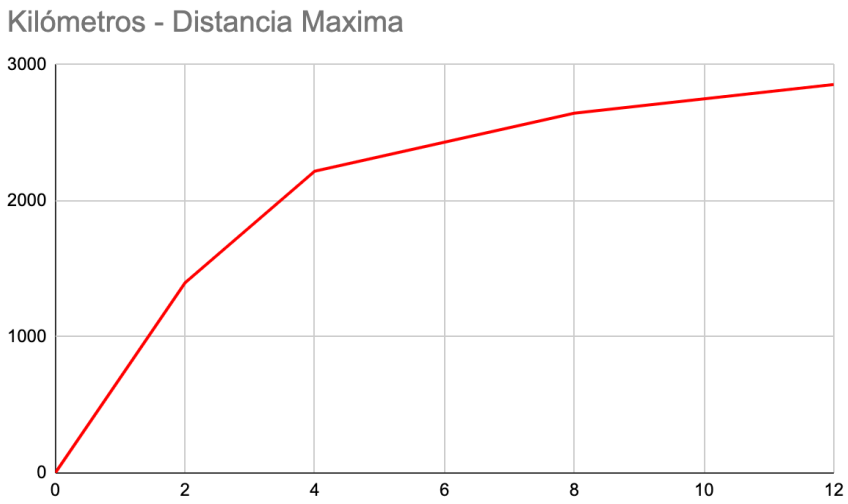
4.7 Influencia de la máxima distancia recorrida por las cisternas

Observación	La distancia màxima que puede recorrer una cisterna puede afectar el beneficio
Planteamiento	Observar el beneficio resultante con diferentes valores para la distancia máxima
Hipótesis	La distancia máxima no afecta el beneficio (H_0) o la distancia máxima puede tener algún efecto sobre el beneficio (H_1)
Método	• Generar 10 semillas aleatorias • Ejecutar un experimento para cada semilla • Experimentar con problemas de 10 centros de distribución sin multiplicidad (un camión por centro) y 100 gasolineras • Usar distintos valores para la distancia máxima, mayores y menores que la distancia máxima dada en el enunciado • Usar el algoritmo de Hill Climbing • Comprobar la relación existente entre la distancia máxima recorrida por una cisterna y el nivel de los beneficios resultantes

En este experimento es importante enfatizar que no cambiaremos el número de viajes, por lo tanto este nos puede crear un límite de kilómetros. Con tal de ver los máximos que este nos podría construir con los diferentes máximos escogimos cuatro valores diferentes. A continuación exponemos los resultados de las ejecuciones.

Horas de Trabajo	Beneficio	Kilometros
0	0	0
2	68178,26 €	1397,4 km
4	90956,9 €	2217,56 km
8	94274,8 €	2644,2 km
12	93855,96 €	2854,0 km





El beneficio tiene una subida empinada de 0 a 4 pero al llegar a 8 horas de trabajo se vuelve más plana la línea. Esto nos da a ver que manteniendo el número de viajes máximo, a partir de las 6 horas trabajadas nos quedamos con el mismo beneficio. En cuanto a kilómetros vemos que aunque de 0 a 4 tenemos una subida más empinada, nunca se nos vuelve vertical esta línea, si no que sigue incrementando.

Por lo tanto vemos, que a partir de las 6 horas no nos es beneficioso seguir subiendo el número de horas trabajadas ya que esto nos acabará decrementando el beneficio total por el aumento en kilómetros que supondrá.

5 CONCLUSIÓN

5.1 Comparación de los Algoritmos

Basándonos en los experimentos de algoritmos y operadores previamente expuestos, vamos a tomar en consideración todo lo que hemos aprendido sobre los algoritmos de Hill Climbing y de Simulated Annealing, para comparar su rendimiento y sus resultados.

Compararemos los dos algoritmos desde los dos aspectos que consideramos durante la fase de experimentación; en función de la calidad de la solución generada, y cómo influye el tamaño del problema al tiempo que tarda en encontrarse.

5.1.1 Optimalidad de la Solución Generada

Una vez encontrados los parámetros adecuados para el Simulated Annealing, podemos observar que los resultados obtenidos con ambos algoritmos son bastante parecidos.

En el segundo experimento (apartado 4.2), obtenemos los siguientes resultados al usar Hill Climbing con solución inicial aleatoria y la función heurística 1: un **beneficio de 94230,02 €** de media, y **2730 Km de media**.

Si recordamos ahora los resultados obtenidos en el experimento 4.4, obtenemos de media y para las mismas semillas, **un beneficio de 94043,2 €, y 2780 Km recorridos.**

Por lo tanto, podemos concluir que para la proporción 10:100 ambos algoritmos rinden de la misma manera. Y como vemos en el mismo experimento 4.4, el rendimiento se mantiene similar a medida que agrandamos el tamaño del problema.

Sin embargo, la diferencia entre ambos algoritmos, se hace mucho más notable cuando comparamos los tiempos de ejecución.

5.1.2 Complejidad Temporal

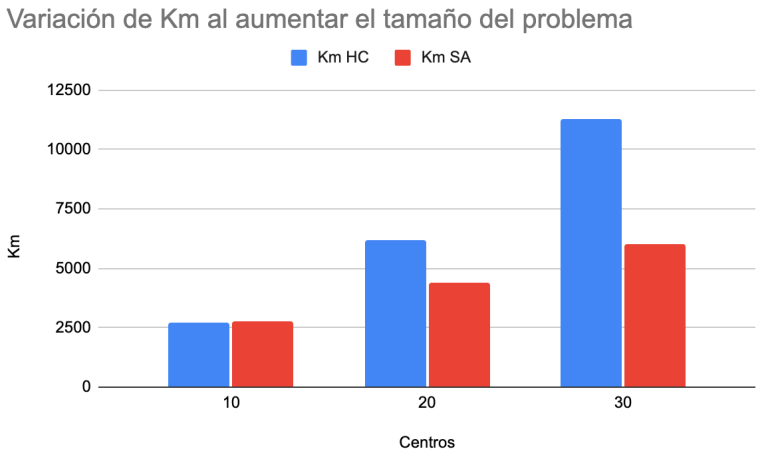
Como ya hemos comentado al realizar el experimento 4.4, esperábamos que el tiempo de ejecución para el Hill Climbing fuera más elevado que para el Simulated Annealing puesto que el comportamiento del primero se ve afectado por el aumento del tamaño del problema mientras que el segundo no.

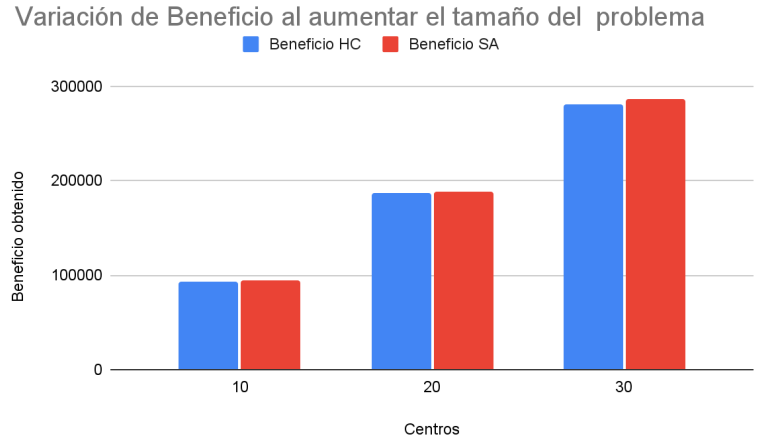
Hill Climbing toma en cuenta todos los nodos sucesores al estado actual (aumenta considerablemente al añadir centros de distribución y gasolineras), mientras que Simulated Annealing, se limita a escoger siempre un solo nodo sucesor al azar, por lo que no se ve directamente influenciado al aumentar el tamaño del problema.

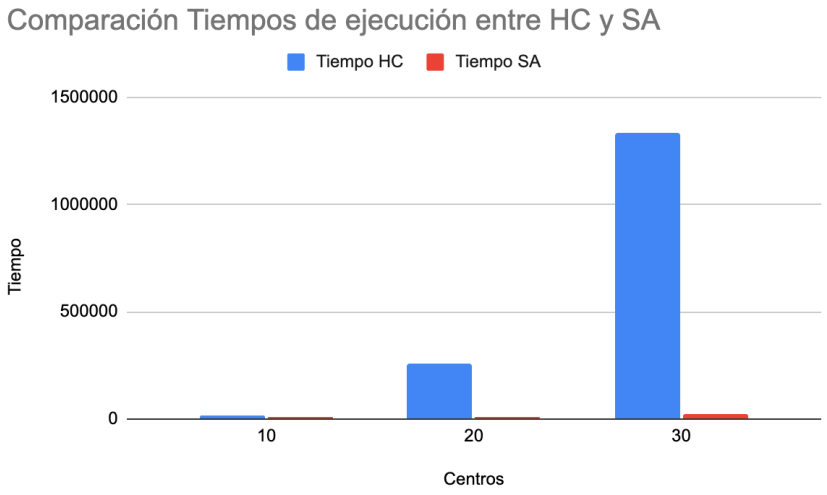
Adjuntamos a continuación tres gráficos, que comparan los kilómetros, el beneficio y nuevamente el tiempo para ambos algoritmos, respecto al aumento de la dificultad del problema.

De ellos podemos sacar como conclusión que para problemas “pequeños”, el rendimiento de ambos es muy parecido y por lo tanto no se observan diferencias significativas al optar por un algoritmo o el otro. No obstante, esto no se cumple a medida que las dimensiones del problema crecen. Hemos visto que como consecuencia, se añaden más nodos sucesores a cada estado dado que aparecen nuevas

combinaciones para aplicar los operadores. Es precisamente por esto que Hill Climbing empieza a rendir más lentamente mientras que Simulated Annealing mantiene un incremento más estable del tiempo de ejecución. Es decir que para problemas grandes, Simulated Annealing sale más a cuenta que Hill Climbing.







5.2 Posibles Extensiones del Experimento

Como posibles extensiones para el experimento realizado, proponemos las siguientes áreas, que en este estudio no han sido consideradas.

5.2.1 Experimentar con diferentes heurísticas

Nuestra propuesta de solución para el problema de abastecimiento cuenta con dos funciones heurísticas, mencionadas en el apartado 3.5. Sin embargo, a la práctica nos percatamos que nuestra primera función heurística nos proporciona resultados más satisfactorios. Es por eso que para llevar a cabo los experimentos, nos servimos únicamente de la primera heurística.

Proponemos como extensión comparar ambas heurísticas y analizar a qué puede deberse la diferencia en la calidad de las soluciones a las que nos llevan.

5.2.2 Experimentar con diferentes números de viajes

Otro factor determinante en el problema de abastecimiento es el máximo número de viajes que cada camión puede realizar. Esto tiene también una gran influencia en el resultado que obtenemos. De ser muy reducido, un camión con máximo número de asignaciones no puede atender otra petición aunque le queden kilómetros suficientes para hacerlo. Y de ser muy elevado, podríamos encontrarnos con una situación opuesta, en la que no podemos aprovechar el máximo número de viajes de un camión cisterna por haber agotado el máximo de kilómetros. Podríamos por lo tanto plantear también la siguiente pregunta: ¿Cuál es la proporción idónea de *max Kilómetros : maxViajes*? Podría ser interesante determinar también si dicha proporción dependerá solo de estos dos factores o si hay algún otro que también influya.

6 TRABAJO DE INNOVACIÓN

6.1 Descripción del tema escogido

Las redes sociales cada vez se están expandiendo más, convirtiéndose en un componente central de nuestro dia a dia. Este trabajo explorará la inteligencia artificial usada en Twitter.

6.2 Reparto del Trabajo

Hemos dividido el trabajo en tres subapartados, cada uno explicando un tema relacionado con la inteligencia artificial aplicada a twitter. Se ha implementado así para ver perspectivas diferentes de los usos de la inteligencia artificial en las redes sociales. Ya que nos hemos centrado en Twitter, mucha de la información que hemos sacado es del blog de esta compañía.

Cada miembro del equipo se ha encargado de buscar un tema que le interese, y buscar publicaciones hechas sobre este tema. Carla ha optado por entender la aplicación de Graph Neural Networks en Twitter. Laia ha optado por investigar sobre la aplicación de métodos de inteligencia artificial en el recorte automático de imágenes que aparecen en la cronología de un usuario. Bea ha optado por estudiar cómo Twitter entrena sus modelos de datos para emplear de manera efectiva ML (aprendizaje automático) en varios aspectos de su plataforma, como los anuncios personalizados, la detección de abusos o la recomendación de contenido.

6.3 Dificultades

En este momento no nos hemos encontrado con baches significativos, esto es dado a que la información que encontramos en el blog de twitter habla ampliamente sobre los algoritmos que utilizan. Esta, como todas, tiene sus limitaciones ya que una compañía como Twitter no divulgará información que les pueda poner en desventaja.

6.4 Bibliografía

Para crear esta bibliografía hemos utilizado el formato MLA.

6.4.1 Neural Networks

[1] Bronstein, Michael, and Fabrizio Frasca. “Simple Scalable Graph Neural Networks.” *Twitter*, Twitter,19 Apr. 2021, https://blog.twitter.com/engineering/en_us/topics/insights/2021/simple-scalable-graph-neural-networks.

[2] Chamberlain, Ben and Bronstein, Michael. “Graph Neural Networks as Neural Diffusion Pdes.” *Twitter*, Twitter, 23 July 2021,https://blog.twitter.com/engineering/en_us/topics/insights/2021/graph-neural-networks-as-neural-diffusion-pdes.

[3] Bronstein, Michael. “Blog Author @Mmbronstein.” *Twitter*, Twitter, https://blog.twitter.com/engineering/en_us/authors.mmbronstein.
Author publishes a lot of information on graphs and networks and their use in twitter.

6.4.2 ML model Training

[4] Bean, Andrew. “Distributed Training of Sparse ML Models - Part 1: Network Bottlenecks.” *Twitter*, Twitter, 23 Sept. 2020, https://blog.twitter.com/engineering/en_us/topics/insights/2020/distributed-training-of-sparse-machine-learning-models-1.

[5] Bean, Andrew. “Distributed Training of Sparse ML Models - Part 2: Optimized Strategies.” *Twitter*, Twitter, 23 Sept. 2020, https://blog.twitter.com/engineering/en_us/topics/insights/2020/distributed-training-of-sparse-machine-learning-models-2.

[6] Bean, Andrew. “Distributed Training of Sparse ML Models - Part 3: Observed Speedups” *Twitter*, Twitter, 23 Sept. 2020, https://blog.twitter.com/engineering/en_us/topics/insights/2020/distributed-training-of-sparse-machine-learning-models-3.

6.4.3 Auto-cropping

[7] Theis, Lucas, and Zehan Wang. “Speedy Neural Networks for Smart Auto-Cropping of Images.” *Twitter*, Twitter, 24 Jan. 2018, https://blog.twitter.com/engineering/en_us/topics/infrastructure/2018/Smart-Auto-Cropping-of-Images.

[8] Chowdhury, Rumman. “Sharing Learnings about Our Image Cropping Algorithm.” *Twitter*, Twitter, 19 May 2021, https://blog.twitter.com/engineering/en_us/topics/insights/2021/sharing-learnings-about-our-image-cropping-algorithm.

[9]Agrawal, Parag and Davis, Dantley. “Transparency around Image Cropping and Changes to Come.” *Twitter*, Twitter, 1 Oct. 200, https://blog.twitter.com/en_us/topics/product/2020/transparency-image-cropping.