IA-Practica 1

Búsqueda local

Grupo: Sergi Avila Sanguesa, Francisco Sánchez Costas, Raul Montoya Pérez

ÍNDICE

Parte descriptiva **3**
- El problema 3
  - Por qué busqueda local? 3
- Representación del problema 4
  - Parte estática 4
  - Parte dinámica 5
  - Espacio de búsqueda 6
- Operadores 6
  - Mover 6
  - Intercambiar 7
- Estado inicial 8
  - Vacío 8
  - Lleno sin criterio 8
  - Lleno optimizado 8
- Heurística 9

Parte experimental **10**
- Elección del heurístico de referencia y combinaciones propuestas 10
- Experimento para estado inicial y operadores 10
- Experimento con parámetros del SA 13
- Experimento variando el tamaño del problema en HC y SA 15
- Experimento variando el número de centros 18
- Experimento variando el coste por kilómetro recorrido 20
- Experimento variando horas de trabajo 22

Trabajo en equipo y organización **24**

1. Parte descriptiva

1.1. El problema

El problema de esta práctica consiste en gestionar los camiones de una empresa de distribución de gasolina. El objetivo de esta práctica es crear una ruta para que los camiones de la empresa atiendan las diferentes peticiones de las gasolineras **durante un solo día** para obtener el máximo beneficio.

El problema tiene las siguientes limitaciones y características:

- **Área geográfica:** Para simplificar el problema, se considera que el área es una cuadrícula de 100 x 100 km^2. Cada cuadro del área es de 1 x 1.

- **Movimiento:** el movimiento mínimo que puede realizar un camión es de 1km y únicamente en vertical u horizontal.

- **Elementos en el mapa:** En el mapa Hay un número ***n*** de centros de distribución, estos contienen uno o varios camiones, y un número ***g*** de gasolineras.

- **Peticiones:** Cada una de las gasolineras contiene entre 0 y 3 peticiones que los camiones tienen que atender, y estas llevan 0 o más días pendientes. Todas las peticiones son de la misma cantidad de gasolina.

- **Límite de los camiones:** Cada camión puede hacer un máximo de ***k*** kilómetros, puede realizar un máximo de ***v*** viajes y tienen gasolina para atender 2 peticiones. Estos valores son comunes para todos los camiones de la zona. Un viaje se define como: salir del centro de distribución, atender una o dos peticiones (pueden ser 2 peticiones de la misma gasolinera) y volver al centro.

- **Beneficio por atender una petición:** el valor de un depósito es ***x*** y el dinero que obtiene es el resultado de multiplicar el valor por el siguiente porcentaje:
  - Si los días que la petición lleva pendiente es 0: 102%
  - En caso contrario: $(100 - 2^{\text{dias pendiente}})\%$

- **Coste por km:** Aunque en este problema no hay que tener en cuenta el combustible interno de cada camión (se trata como si fuera infinito), si que se considera un coste fijo de dinero por km recorrido.

1.1.1. Por qué busqueda local?

Aunque no se nos pide elegir el tipo de búsqueda que hay que aplicar en este problema, nadie discute que sea un problema de búsqueda local, pues se ajusta a las características de estos.

En este punto del curso hemos visto 4 familias de algoritmos de búsqueda: exhaustiva o no informada, heurística, local y de satisfacción de restricciones.

Las dos primeras se basan en la búsqueda de una única y óptima solución, y pueden llegar a expandir y visitar todos los nodos del espacio de búsqueda.
En nuestro problema, aun siendo una versión muy simplificada de la realidad, el espacio es demasiado grande como para mantener en memoria todos los nodos. Además, no nos interesa que el ordenador esté mucho rato para buscar un estado ligeramente mejor al que ya ha encontrado.

Utilizar satisfacción de restricciones queda directamente descartado, pues sólo encuentran una solución, no permiten la optimización de ninguna variable.

Los algoritmos de búsqueda local, en cambio, se mueven en el espacio de soluciones, es decir, cada nodo del espacio es una solución posible, y queremos encontrar una solución buena pero no necesariamente óptima.
Además, no guardamos todos los estados visitados en memoria. Una gran ventaja de esto es que se puede parar la ejecución del programa en cualquier momento y que tenga una solución ,aunque no sea demasiado buena.

1.2. Representación del problema

1.2.1. Parte estática

La parte estática representa el problema en general, independientemente del estado en que nos encontremos y las operaciones que hayamos realizado.

En el enunciado ya se dan unas clases para representar esta parte, así que justificamos porque nos parecen adecuadas:

La parte principal de la representación del problema es el área geográfica donde transcurre. Al ser una cuadrícula de 100 x 100, lo mas natural es pensar en una matriz de 100 x 100 pero es altamente ineficiente, pues sólo unas pocas de esas 10.000 casillas están ocupadas. Así pues, lo mejor es que el tablero esté implícito, es decir, trabajar con coordenadas sabiendo que el rango de las ***x*** y las ***y*** será [0,99] pero sobre un tablero inexistente.

Dentro de este tablero hay gasolineras y centros de distribución, que se pueden agrupar en en dos listas. Las gasolineras son representadas por sus coordenadas y una lista de las peticiones que tienen pendientes con los días que hace que se realizó cada petición. Los centros de distribución se representan sencillamente con sus coordenadas. En las clases del enunciado, varios centros pueden estar en las mismas coordenadas para indicar que varios camiones salen del mismo sitio.

1.2.2. Parte dinámica

La parte dinámica de la representación es específica de cada estado, y se modifica mediante los operadores.

Lo que define un estado de nuestro problema son las rutas que van a hacer nuestros camiones durante todo el dia, es decir, cuántos viajes harán y qué peticiones atenderán con cada viaje.

Para encontrar una representación del estado hay que pensar en todo el problema, pues la eficiencia de tanto el heurístico como los operadores depende de ello, y tampoco nos podemos pasar con el uso de memoria, ya que aunque la búsqueda local no es tan exigente en memoria podría darnos problemas.

Para facilitar la escritura y comprensión de código, nuestra representación tiene una forma muy parecida a la de las clases del enunciado para la parte estática. Además separamos la representación del estado en dos partes:

Peticiones: un array de arrays donde **[i] [j]** representan la gasolinera número **i**, y la petición número **j** de esa gasolinera respectivamente. El valor de la posición **[i] [j]** es el número del camión que atiende esa petición, o -1 si la petición no está atendida por ningún camión. Esta simple representación permite buscar rápidamente que peticiones han sido atendidas, así como calcular los beneficios.

Camiones: un array donde cada posición contiene: los km totales que recorre el camión, un array con los km que recorre en cada viaje, y otro array con las peticiones que atiende (representadas por pares **i j** que indican la gasolinera y el número de la petición de esa gasolinera respectivamente, con la posibilidad de que ambos valores sean -1 -1 si en esa posición no atiende ninguna petición).

El camión en la posición **k** es el que sale del centro de distribución **k**, los viajes van implícitos: cada dos posiciones del array de peticiones atendidas es un viaje, si la segunda posición es -1 -1 es que hace un viaje donde atiende a una sola petición. Esta parte está pensada para optimizar el control de km que recorre el camión. Tener un array para los km de cada viaje permite que al aplicar un operador solo haya que hacer cálculos sobre ese viaje en concreto y no sobre todo el recorrido del camión, lo que es muy útil, ya que los operadores van a estar comprobando constantemente que los recorridos no superen los km permitidos. Los viajes implícitos cada 2 peticiones atendidas es una forma de simplificar la representación.

En general, los -1 para representar algo vacío permiten generar los array con tamaño completo desde un comienzo, y poder usar los mismos índices para referirnos a elementos de la representación estática y dinámica.

1.2.3. Espacio de búsqueda

Tenemos un tablero con ***n*** centros de distribución, cada uno con un camión, pudiendo haber varios centros en la misma posición, ***p*** peticiones en total, contando con las de todas las gasolineras, y cada camión puede hacer ***v*** viajes y por lo tanto puede atender un máximo de ***2*v*** peticiones.

Llamemos a los espacios disponibles para colocar la peticiones ***e=n*2*v***. Vamos a hacer las consideraciones siguientes:

- El caso ***e>p*** lo tratamos como si fuera ***e=p***.
- Cada petición se puede colocar en cualquier posición de las ***e*** (o ***p***) peticiones que se pueden atender.
- El orden en que colocamos las peticiones importa **siempre**.

Estas consideraciones, aunque no vayan necesariamente a cumplirse permiten calcular una cota superior, pues estamos considerando muchos estados equivalentes o que van a ser peores que otros sin importar la posición de la gasolinera o el centro.

Hechas todas estas consideraciones, nos encontramos en un caso de variaciones si ***e<p***, en el que tenemos ***p*** elementos, y elegir ***e*** de ellos, o de permutaciones si ***e=n***. Así que el espacio de búsqueda será:

- O(p!) si $e \geq p$
- O($\frac{p!}{(p-e)!}$) si e<p

1.3. Operadores

1.3.1. Mover

Mover, en sus dos versiones, es el operador con el que realizaremos los experimentos. En general, recibe como parámetros un camión ***k***, una de sus peticiones ***l***, y una petición **no atendida** de alguna gasolinera (***i , j)***. Al acabar la operación, queda marcado que el camión ***k*** atiende la petición (***i , j)*** y se han actualizado los km que recorre el camión en total y en el viaje correspondiente a la petición ***l***.

Antes de aplicar la operación se realizan las siguiente comprobaciones:

- El resultado de aplicar la operación no hará que el camión ***k*** recorra más de los km totales permitidos.

- En caso de que la petición sea una petición fantasma (valor (-1,-1)), será la de índice más pequeño del camión ***k.*** Esto reduce el factor de ramificación y sirve como filtro de soluciones peores, ya que la distancia recorrida al atender dos peticiones en un mismo viaje siempre es menor o igual que la recorrida al atenderlas por separado.

La última condición es la que nos marca la separación de mover en dos versiones y surge de la siguiente pregunta: “la petición ***l*** de un camión tiene que ser fantasma necesariamente?”

En caso de que lo sea, el mover actuaría como un operador “añadir” que solamente añade peticiones no atendidas a camiones con huecos libres. En caso contrario mover podrá cambiar una petición ya atendida en un camión por otra.

Pensamos que la segunda versión podría resultar útil cuando hubiera muchas más peticiones que las que se pueden atender en un dia. También se tiene que tener en cuenta que la primera versión que funciona como un añadir, solo serviría para soluciones iniciales vacías o con huecos libres, ya que sinó no se movería nunca una petición a un camión.

Para el cálculo del factor de ramificación consideremos que hay ***n*** peticiones por atender, y ***k*** camiones. Cada uno puede atender un máximo de ***2*v*** peticiones, siendo ***v*** el número de viajes que puede realizar, y que el camión c_i con $0 \leq i < k$ atiende, en un cierto estado del tablero, pa_i peticiones.

En la primera versión de mover, la equivalente a añadir, el factor de ramificación es de $n \times k$

En la segunda versión, que permite añadir aunque el camión ya tenga una petición, es de $n \times \sum_{i=0}^{k} pa_i$.

1.3.2. Intercambiar

Intercambiar es un operador que descartamos rápidamente. No puede ser un operador único como mover, pues no sirve para simular un operador “añadir” y aunque le añadimos restricciones, el tiempo de ejecución no dejaba de ser de minutos. Por este motivo, no llegamos a realizar ningun experimento serio con este operador.

La idea era que dados dos camiones con peticiones, se intercambiara que petición atiende cada camión, es decir, que la petición del primer camión se movía al segundo camión y la del segundo camión al primero.

Para analizar el factor de ramificación volvamos a las consideraciones del apartado anterior: ***n*** peticiones por atender, ***k*** camiones y pa_i peticiones atendidas con $0 \leq i < k$. Para

simplificar la fórmula, supongamos que no probaremos de intercambiar peticiones de un camión entre ellas para intentar optimizar la ruta.

El factor de ramificación es : $\sum_{i=0}^{k} \left(pa_i \times \left(\sum_{j=0, j\neq i}^{k} pa_j \right) \right)$

Como intercambiar no puede ser un operador único, el factor de ramificación de intercambiar se sumará al de mover, como mínimo $n \times k$, en la función generadora de sucesores.

1.4. Estado inicial

Para los estados iniciales, seguimos la opción que nos recomendó nuestra profesora y consideramos 3 configuraciones iniciales: una totalmente vacía, una intentando llegar a una solución buena, y otra intermedia.

1.4.1. Vacío

La solución inicial más simple. Se genera en tiempo constante y no toma ninguna decisión que nos pueda alejar de una buena solución

1.4.2. Lleno sin criterio

Configuración intermedia. Se genera de la siguiente manera: recorremos todas las peticiones y las asignamos al camión de índice menor que la pueda atender sin romper las reglas del problema. Sin preocuparnos demasiado por la eficiencia de la implementación, su coste puede llegar a ser O(n × k × p) siendo ***n*** el número de peticiones, ***k*** el número de camiones, y ***p*** el número de peticiones de un camión. Como **p** es una constante ya que cada camión lleva 10 peticiones, el coste quedaría determinado por O(n × k).

Aunque la implementación sea ineficiente, el mayor problema de esta configuración inicial es que toma decisiones que no tienen porqué ser óptimas y por lo tanto puede provocar que acabemos en un estado peor que con el resto de configuraciones.

1.4.3. Lleno optimizado

Como solución inicial “inteligente” se han asignado las peticiones a los camiones más cercanos que tuvieran espacio para asignarles la petición, es decir, que tenían menos de 10 peticiones asociadas. El coste vendría dado por la expresión O(n × k) siendo ***n*** el número de peticiones y ***k*** el número de camiones, igual que en el método anterior.

El razonamiento en el cual nos hemos basado para generar este estado inicial es que el precio por kilómetro es de 2 euros y por lo tanto, minimizar la distancia recorrida por los camiones nos proporcionará más beneficio. De esta manera, generamos una solución buena, aunque mejorable, como punto de partida y no se expandirán tantos nodos ya que tendremos un buen beneficio desde el inicio.

1.5. Heurística

Respecto a la heurística, además de la que se pide en el enunciado que calcula el beneficio como el dinero ganado con las peticiones menos el precio perdido por recorrer cada kilómetro, hemos generado diferentes heurísticas minimizando o maximizando diferentes elementos del problema.

Primero hemos hecho un heurístico como el que pedía el enunciado pero además se ha tenido en cuenta la pérdida por no atender una petición.

También hemos probado heurísticas dónde multiplicamos el peso de los kilómetros por tres y por cuatro en lugar de los dos que decía el enunciado, que han demostrado funcionar mejor que el primer heurístico creado.

Más tarde, decidimos hacer unos cuantos heurísticos más para probar su eficiencia en el problema, estos heurísticos se encargaban de:

-Minimizar días de las peticiones atendidas -> Si han pasado pocos días en las peticiones que atendemos antes, el beneficio que obtendremos será el mayor posible.

-Maximizar días de las peticiones atendidas -> Si atendemos las que más días llevan esperando primero, las que quedarán serán solo las que tengan menos días por lo que podríamos cobrar más.
-Minimizar días de las peticiones no atendidas -> Si las peticiones que quedan por atender son las que tienen menos días, cuando las atendamos obtendremos más beneficio porque las más viejas ya las habremos atendido.

-Maximizar días de las peticiones no atendidas -> Si las peticiones que quedan por atender son las que tienen más días, eso quiere decir que las que hemos atendido son las que tienen menos por lo que podríamos maximizar el beneficio.

Una vez probados los heurísticos, hemos podido observar que no han dado muy buen resultado, excepto en los que se variaba el coste del kilómetro. De las dos opciones planteadas que dan más peso a los kilómetros recorridos que el heurístico inicial, se ha comprobado que si multiplicábamos por 3 el coste por kilómetro se obtenían mejores resultados que si lo hacíamos por 4 y los resultados obtenidos respecto el heurístico inicial era mejor o pero en función de la seed. Después de hacer varias pruebas nos hemos

decantado por utilizar el heurístico inicial con 2 de coste por kilómetro recorrido ya que parecía funcionar mejor.

2. Parte experimental

2.1. Elección del heurístico de referencia y combinaciones propuestas

Para no desviarnos del objetivo principal de la práctica, los primeros experimentos para determinar el estado inicial y operadores definitivos los hicimos con un heurístico simple que sólo tuviera en cuenta los beneficios obtenidos

Aun teniendo 2 operadores y 3 estados iniciales posibles, no todas las combinaciones entre ellos tienen sentido. Con un estado inicial no vacío, el mover que actúa únicamente como añadir no tiene sentido, pues se aplicará en ninguna o muy pocas ocasiones. Descartando estas combinaciones, realizaremos los experimentos con las siguientes (nos referiremos a mover limitado al que únicamente actúa como añadir, y al que permite mover siempre como mover normal):

1. Estado inicial vacío, mover limitado
2. Estado inicial vacío, mover normal
3. Estado inicial lleno sin criterio, mover normal
4. Estado inicial lleno optimizado, mover normal

2.2. Experimento para estado inicial y operadores

Como nos encontramos en un problema de búsqueda local no sabemos qué método de inicio es mejor o peor ya que no sabemos la forma que tendrá el espacio de búsqueda. Por esta razón, vamos a realizar un experimento probando diferentes métodos de inicialización para ver si hay unos que son mejores que otros.

Consideramos tres métodos de inicialización: estado vacío, estado en el que se llenan los camiones de forma ordenada a partir de los índices (de los camiones, gasolineras, y peticiones), y estado inteligente donde se asignan las peticiones al camión más cercano que tenga un hueco para poder atender la petición.

Como operador tenemos un mover, el cual, mueve una petición no atendida de una gasolinera a un camión, indicando la posición de la petición dentro del camión, ya que un camión puede atender hasta diez peticiones en un dia. Este operador puede estar restringido o no, es decir, no estará permitido mover una petición a un camión que en la posición indicada ya tenga una petición, o de lo contrario siempre se podrá mover aunque ya atienda una petición. Para el primer modo descrito anteriormente, que haría una función de añadir, le llamaremos mover1 mientras que para el segundo, que funciona como un intercambiador, le llamaremos mover2.

Con las diferentes configuraciones que se han descrito, se realizarán diez réplicas del experimento en diez seeds elegidas al azar que también se usarán para el resto de los experimentos. Se medirá tanto el beneficio obtenido como el tiempo que ha tardado el programa en ejecutarse. Omitimos el número de nodos expandidos para llegar a la solución, ya que está altamente correlacionado con el tiempo de ejecución por lo que nos llevaría a las mismas conclusiones.

De esta manera, el experimento se resumiría de la siguiente forma:

Observación: Pueden haber combinaciones de soluciones iniciales y operadores que son mejores que otros.

Planteamiento: Escogemos diferentes combinaciones de soluciones iniciales y operadores y observamos sus soluciones.

Hipótesis: Todas las combinaciones son igual de buenas (H0) o hay combinaciones mejores que otras.

Método: Elegimos diez seeds diferentes al azar, cada una se utilizará para hacer un experimento para cada una de las combinaciones posibles, se experimenta en problemas de 100 gasolineras, 10 centros y un camión por centro usando el algoritmo de Hill Climbing y se mide tanto el tiempo de ejecución como el beneficio obtenido.

Resultados: A continuación se muestran los resultados obtenidos al realizar los experimentos para las diferentes combinaciones:

	Beneficio obtenido				Tiempo de ejecución (ms)			
Seed	Vacío mover1	Vacío mover2	Lleno Índices	Lleno inteligente	Vacío mover1	Vacío mover2	Lleno Índices	Lleno inteligente
464	95780	95828	93216	95356	3162	5767	1909	1155
1923	95428	95688	94176	95472	3084	5969	2434	1991
850	94620	94696	94064	94264	3209	5553	2511	1615
3010	95212	95360	92752	95284	3146	6156	1808	1282
1378	94576	94576	92928	94636	3224	5645	2031	1484
2981	95732	95852	94560	95724	3269	5675	2309	1863
4368	94292	94344	93148	93572	3242	5536	1984	1788
311	94796	95068	94556	94660	3249	5688	2964	2065
1458	95476	95616	94620	94888	3618	7063	3901	2955

4386	90980	90992	91576	92092	3331	5762	2022	1201

Si hacemos la media de los beneficios y del tiempo de ejecución de cada una de las combinaciones en las diez seeds, obtenemos los siguientes valores:

	Vacío mover1	Vacío mover2	Lleno Índices	Lleno inteligente
Beneficio	94689,2	94802	93559,6	94594,8
Tiempo (ms)	3253,4	5881,4	2387,3	1739,9

Primero de todo vamos a comparar las combinaciones llenas asumiendo que son igual de buenas (H0). Podemos considerar que la probabilidad de que un método dé una mejor solución que otro se distribuye como una función binomial y podemos comprobar cuál es la probabilidad de que H0 sea cierta, es decir, que tengan la misma probabilidad de ser mejor solución, que al experimentar con dos combinaciones, que sería de 0,5. Podemos observar que el estado lleno inteligente da siempre mejor beneficio y mejor tiempo que el estado lleno a partir de los índices en todas las seeds. Si miramos a las tablas de la binomial, vemos que la probabilidad de que un estado sea mejor que otro diez veces de diez posibles es de 0,001, es decir, bastante remota, por lo que podemos decir que el estado lleno inteligente da mejores soluciones que el estado lleno a partir de los índices, por lo tanto descartamos este último.

Ahora comparamos los dos estados vacíos con los diferentes modos de operadores mover1 y mover2. Podemos observar que sucede los mismo que en el caso anterior, mover2 da siempre mejores beneficios que mover1 pero en este caso el tiempo de ejecución de mover2 es mucho mayor (casi el doble). Como a parte de ver que método da mejor solución también hay que tener en cuenta el tiempo que tardan en ejecutarse podemos llegar a la conclusión que como la diferencia de beneficio entre los dos métodos es relativamente pequeña, sería mejor escoger mover1 como operador ya que ganamos mucho en tiempo y perdemos muy poco en beneficio y además en escenarios más grandes la diferencia de tiempo aún sería mucho mayor.

Lo mismo pasa si comparamos el estado vació con mover2 y el estado inteligente. El primero es mucho más lento que el segundo (unas tres veces) y además el estado inteligente obtiene más beneficio que el estado vacío con mover2 en dos seeds. Nos quedaríamos con el estado inteligente por la misma razón que antes, ya que la diferencia de tiempo es aún mayor que en el caso anterior.

Finalmente si comparamos el estado vacío con mover1 y el estado inteligente vemos que el primero da mejores beneficios que el segundo en seis de los diez seeds. Si miramos la tabla de la binomial vemos que la probabilidad de que suceda (si las dos soluciones son igual de buenas) es de 0,205, una probabilidad más alta que en los otros casos vistos anteriormente.

Si observamos los beneficios obtenidos por los dos métodos vemos que la diferencia es mínima pero en cambio el tiempo de la solución inteligente es casi la mitad que el de la solución vacía por lo que escojeríamos el estado inteligente por el valor del tiempo de ejecución obtenido.

Como conclusión, después de realizar los experimentos hemos visto que hay métodos que proporcionaban mejores beneficios y otros que proporcionaban mejor tiempo de ejecución. Las soluciones llenas han resultado ser mucho más rápidas que las vacías. Una explicación podría ser que estos estados se encuentran más cerca de la solución final, por lo que no expanden tantos nodos y como consecuencia tardan menos en ejecutarse.

A partir de los razonamientos anteriores, se ha decidido escoger el estado inicial lleno inteligente como estado inicial y operador mover sin restricciones para el desarrollo de la práctica y de los siguientes experimentos.

2.3. Experimento con parámetros del SA

El otro algoritmo con el que hemos trabajado es el Simulated Annealing, que imita el proceso térmico de los metales. A diferencia del Hill Climbing, su función sucesora solo pide un estado generado al azar. Si es mejor lo acepta, si es peor lo aceptará o no según una función de probabilidad.

SA tiene 4 parámetros:

- Máximo de iteraciones: Cuantos sucesores generará el algoritmo.
- K y λ Parámetros de una función exponencial que determina la probabilidad de aceptar un estado peor que el actual.
- Iteraciones por paso: Este algoritmo se divide en fases, con los mismos parámetros en cada una. Por lo tanto, este valor tiene que ser divisor del máximo de iteraciones.

El objetivo de este experimento es determinar qué valores de los parámetros funcionan mejor para nuestra representación del problema. La función de probabilidad es la siguiente:

$F(T) = k \cdot e^{-\lambda \cdot T}$

$P(\text{estado}) = e^{\left(\frac{\Delta E}{F(T)}\right)}$

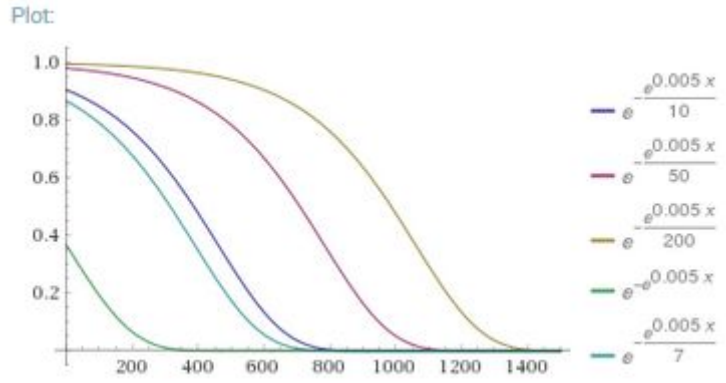
Como podemos ver, es una función exponencial, y por lo tanto la probabilidad de aceptar un estado peor nunca será exactamente 0. Aun así, si podemos determinar los valores de adecuados, podemos hacer que en el tramo final del algoritmo la probabilidad de aceptar un estado peor sea prácticamente 0.

Son 4 parámetros a determinar sin ninguna referencia sobre qué valores dar. Además, por un error de aima, SA no permite saber el camino que el algoritmo ha tomado para llegar al estado final, pues si lo hacemos a través del search agent, este nos muestra una referencia al estado final en vez de una lista con las acciones para llegar a este.

Como no tenemos otra opción, decidiremos los valores de K y λ de la siguiente manera: fijamos primero λ con un valor arbitrario 0.005 y modificamos K para intentar encontrar con qué valor da mejores resultados. Una vez fijada K haremos lo mismo para λ. El número máximo de iteraciones será el número de iteraciones para que la función de probabilidad acepte estados el 0.00001 (en tanto por uno) de las veces, de forma que en la última fase no se acepten estados peores. Finalmente, y para simplificar los experimentos, siempre habrá 10 fases.

A continuación se mostrará el resultado de los experimentos para determinar el valor de K. Todos los tiempos y beneficios son la media de 5 ejecuciones del algoritmo con las mismas condiciones, con una seed al azar, la 464.

Valor de k	Iteraciones necesarias	Beneficio obtenido	Tiempo
10	855	94599,2	0,374
50	1143	94518,4	0,4866
200	1350	93652	0,534
5	**738**	**94484**	**0,3248**
1	441	93924	0,247
7	792	94530,4	0,35

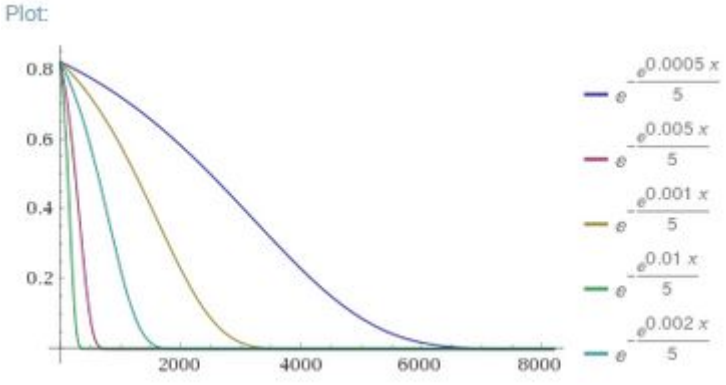


Gráfica hecha con Wolfram Alpha, formula de la función extraída del enunciado de la práctica

Como se puede ver, partimos con un valor de k igual a 10 y fuimos subiendo y bajando su valor hasta encontrar un punto que diera un beneficio alto sin tardar demasiado. Nos quedamos con un valor de k igual a 5, ya que un valor más alto provoca unas mejoras ínfimas pero tarda bastante más.

Resultados para determinar el valor de λ, con K ya fijada en 5:

Valor de λ	Iteraciones necesarias	Beneficio obtenido	Tiempo
0,0005	8100	95297,6	1,71
0,001	4000	95104	1,0006
0,005	**738**	**94484**	**0,3248**
0,01	400	94014,66	0,3346
0,002	**2000**	**95009,6**	**0,5292**



Para estos experimentos, los valores que probamos ya no fueron tan caóticos, hicimos pruebas para 2 valores más bajos, uno más alto y uno intermedio entre los dos mejores resultados. Finalmente, aceptamos el valor λ=0,002, de nuevo buscando un equilibrio entre beneficio y tiempo.

Por lo tanto, los valores elegidos para el simulated annealing son:

- Máximo de iteraciones: 2000
- Iteraciones por fase: 200
- K: 5
- λ: 0,002

2.4. Experimento variando el tamaño del problema en HC y SA

Para estudiar cómo evoluciona el tiempo de ejecución y el beneficio obtenido para valores crecientes de los parámetros del problema se ha realizado un experimento donde se miden estos dos valores en escenarios aumentando el número de camiones y de gasolineras siguiendo la proporción 10:100 para centros y gasolineras hasta que veamos una diferencia significativa.

Utilizaremos la solución inicial y operadores escogidos en el primer experimento, el estado lleno inteligente, y se harán diez réplicas, una en cada una de las seeds escogidas al azar que se han fijado para todos los experimentos.

El experimento se realizará utilizando tanto el algoritmo de Hill Climbing como el de Simulated Annealing y se observará si hay diferencias en el comportamiento de estos dos algoritmos al aumentar el número de camiones y de gasolineras. En el caso de Simulated Annealing se utilizarán los parámetros fijados en el experimento anterior, los cuales son máximo de iteraciones 2000, 200 iteraciones por fase y un valor de k igual a 5 y λ igual a 0.002.

Resultados utilizando el algoritmo de Hill Climbing:

	Beneficio obtenido			Tiempo de ejecución (ms)		
Seed	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam
464	95356	192000	287808	1155	14564	152919
1923	95472	193156	291232	1991	19040	226635
850	94264	192044	291492	1615	19978	212844
3010	95284	192612	292144	1282	22260	197901
1378	94636	192412	290044	1484	18314	165339
2981	95724	194336	290936	1863	19129	215014
4368	93572	192112	289612	1788	16879	281780
311	94660	193916	292976	2065	22069	201117
1458	94888	192928	289180	2955	31728	303695
4386	92092	189844	290264	1201	20915	334239

Si hacemos la media de los beneficios y del tiempo de ejecución de cada una de las combinaciones en las diez seeds, obtenemos los siguientes valores:

	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam
Beneficio	94594,8	192536	290568,8
Tiempo (ms)	1739,9	20487,6	229148,3

Resultados utilizando el algoritmo de Simulated Annealing:

	Beneficio obtenido			Tiempo de ejecución (ms)		
Seed	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam
464	94804	189884	281728	625	1879	3988
1923	94776	190664	287040	690	1808	3832
850	93968	189168	287568	876	1747	3580
3010	94552	189364	288316	784	1782	3521
1378	93832	190192	285664	694	1642	3284
2981	94940	191628	285168	1079	1891	3868
4368	92668	189980	285404	1291	1806	3760
311	93824	191316	288644	884	1912	3499
1458	94216	189888	283700	887	2112	4332
4386	90316	185792	284196	991	1852	3786

Si hacemos la media de los beneficios y del tiempo de ejecución de cada una de las combinaciones en las diez seeds, obtenemos los siguientes valores:

	100 gas 10 cam	200 gas 20 cam	300 gas 30 cam
Beneficio	93789.6	189787.6	285742.8
Tiempo (ms)	880.1	1843.1	3745

Podemos ver que en ambos casos se produce un incremento en el beneficio y en el tiempo debido al aumento de gasolineras y camiones que inducimos variando los parámetros del estado inicial. Sin embargo, podemos observar que la diferencia entre ambos algoritmos es muy significativa en cuanto al tiempo, pero no en cuanto al beneficio. Mientras que Hill Climbing da un beneficio algo mayor, la diferencia de tiempo entre ambos algoritmos es bastante más exagerada, Simulated Annealing el doble de rápido para el caso de los 10 camiones y las 100 gasolineras, 10 veces más rápido para el caso de los 20 camiones y las 200 gasolineras y aproximadamente 61 veces más rápido para el caso de los 30 camiones y las 300 gasolineras, como es de esperar dada la naturaleza de dicho algoritmo, ya que genera muchos menos estados que Hill Climbing. Además podemos observar que el beneficio que nos otorga Simulated Annealing al incrementar el número de camiones y

gasolineras es bastante satisfactorio por lo que podemos reafirmarnos en que los valores correctos para k y λ son respectivamente 5 y 0.002.

A partir de las medias de los resultados de ambos experimentos podemos observar no solo que los parámetros hallados para Simulated Annealing dan buenos resultados al aumentar el tamaño de nuestro problema, sino que además su teórica mejora en eficiencia (vista como beneficio/tiempo) es en efecto cierta, ya que con Simulated Annealing se obtiene un beneficio un poco menor, muy similar a la del algoritmo de Hill Climbing, pero sin embargo la diferencia de tiempo de ejecución es muy grande.

2.5. Experimento variando el número de centros

Hasta ahora, hemos asumido que tener centros de distribución no tiene ningún coste y que tenemos siempre un camión por centro.

En este experimento vamos reducir el número de centros a la mitad, es decir, seguiremos teniendo 100 gasolineras y 10 camiones pero el número de centros de distribución vamos a reducirlo a 5. De esta manera, habrá dos camiones por centro y queremos ver cómo se verán afectados los kilómetros recorridos por los camiones y el beneficio obtenido durante el dia.

Para medir estos dos elementos utilizaremos la solución inicial y operadores escogidos en el primer experimento, el estado lleno inteligente, y se harán diez réplicas, una en cada una de las seeds escogidas al azar que se han fijado para todos los experimentos.

De esta manera, el experimento se resumiría de la siguiente forma:

Observación: Puede que el número de centros de distribución influya en el beneficio obtenido y en el número de kilómetros recorrido por los camiones.

Planteamiento: Reducimos el número de centros a la mitad (5), mantenemos el número de gasolineras (100) y el de camiones (10) y observamos si hay diferencias entre el escenario inicial que tenía 10 centros.

Hipótesis: El número de centros no influye en el beneficio obtenido ni en los kilómetros recorridos (H0) o de lo contrario, sí que afecta el hecho de de tener más centros de distribución.

Método: Elegimos diez seeds diferentes al azar fijadas anteriormente, cada una se utilizará para hacer un experimento para cada una de las posibilidades planteadas (con 5 centros o con 10 centros), se experimenta en problemas de 100 gasolineras y 10 camiones usando el algoritmo de Hill Climbing, con los operadores y solución inicial del primer experimento, y se mide tanto el número de kilómetros recorridos como el beneficio obtenido.

Resultados: A continuación se muestran los resultados obtenidos al realizar los experimentos para las dos configuraciones:

	Configuración con 10 centros		Configuración con 5 centros	
Seed	Beneficio	Kilómetros	Beneficio	Kilómetros
464	95356	2642	92944	3748
1923	95472	2624	93852	3394
850	94264	3198	91412	4584
3010	95284	2768	92752	3844
1378	94636	2842	93236	3592
2981	95724	2598	94228	3316
4368	93572	3304	88756	4872
311	94660	2820	94356	3072
1458	94888	2826	92768	3816
4386	92092	4004	83520	4360

Si hacemos la media de los beneficios y de los kilómetros de cada una de las configuraciones en las diez seeds, obtenemos los siguientes valores:

	Configuración con 10 centros	Configuración con 5 centros
Beneficio	94594,8	91782,4
Kilómetros	2962,6	3859,8

Asumiendo que las dos configuraciones son igual de buenas (H0), podemos considerar que la probabilidad de que una dé mejores beneficios y menos kilómetros recorridos que la otra se distribuye como una función binomial y podemos comprobar cuál es la probabilidad de que H0 sea cierta, es decir, que tengan la misma probabilidad de dar mejores valores, que al experimentar con dos configuraciones, sería de 0,5.

Podemos observar que la configuración con 10 centros da siempre mejor beneficio y un número de kilómetros menor que la que tiene 5 centros. Si miramos a las tablas de la binomial, vemos que la probabilidad de eso suceda es 0,001, es decir, bastante remota, por lo que podemos decir que la configuración con 10 centros es mejor que la de 5. Además, si miramos los valores medios de todas las réplicas, se puede ver que hay una diferencia

considerable entre el beneficio obtenido en ambas configuraciones y también en el número de kilómetros recorridos.

Como conclusión, después de realizar los experimentos hemos visto que la configuración con 10 centros de distribución daba mejores resultados que la 5 centros, lo que significa que el número de centros influye tanto en el beneficio obtenido como en el número de kilómetros recorridos. Esto se debe a que al haber menos centros, la distancia que recorren los camiones para atender las peticiones será más grande o igual a la distancia que se recorrería si hubiesen más centros. Como consecuencia, el aumento de los kilómetros recorridos hará que el beneficio obtenido baje, ya que recorrer un kilómetro tiene un coste que afectará negativamente en el beneficio.

2.6. Experimento variando el coste por kilómetro recorrido

Hasta ahora, habíamos considerado que cada kilómetro recorrido tenía un coste de 2.

Para este experimento vamos a ir multiplicando el precio por kilómetro por 2 para estudiar el comportamiento de las peticiones atendidas según vamos aumentando dicho precio.

Utilizaremos el escenario del primer experimento, con los operadores, estados iniciales y las diez seeds escogidas también en ese experimento y mediremos el número de peticiones que se atienden en un dia en cada rango de aumento del precio.

De esta manera, el experimento se resumiría de la siguiente forma:

Observación: Puede que el número de peticiones atendidas en un dia se vea afectado si aumentamos el precio por kilómetro.

Planteamiento: Utilizamos el escenario del primer experimento que tiene 100 gasolineras, 10 centros y un camión por centro, y observamos si el número de peticiones atendidas en un dia se ve alterado al aumentar el precio por kilómetro.

Hipótesis: El número de peticiones atendidas en un dia no se ve afectado al aumentar el precio por kilómetro (H0) o de lo contrario, sí que se se ve alterado cuando aumentamos el precio por kilómetro.

Método: Elegimos diez seeds diferentes al azar fijadas en el primer experimento, cada una se utilizará para hacer un experimento con diferentes costes por recorrer cada kilómetro, el cual, se irá multiplicando por 2, se experimenta en problemas de 100 gasolineras, 10 centros y un camión por centro, se usa el algoritmo de Hill Climbing, con los operadores y estado inicial del primer experimento y se mide el número de peticiones atendidas en un dia.

Resultados: A continuación se muestran los resultados obtenidos al realizar los experimentos para cada uno de los rangos de precios por kilómetro:

Seed/Precio	x2 (original)	x4	x8	x16	x32	x64	x128	x256
464	100	100	100	98	91	61	40	26
1923	100	100	100	100	90	63	47	43
850	100	100	100	97	88	61	33	24
3010	100	100	100	100	94	70	46	42
1378	100	100	100	100	90	67	40	32
2981	100	100	100	99	96	59	31	30
4368	100	100	100	100	88	59	42	35
311	100	100	100	99	91	59	34	27
1458	100	100	100	99	97	79	57	46
4386	100	99	98	94	86	57	33	30

Si hacemos la media del número de peticiones atendidas de cada uno de los rangos de precios en las diez seeds, obtenemos los siguientes valores:

	x2	x4	x8	x16	x32	x64	x128	x256
peticiones atendidas	100	99,9	99,8	98,6	91,1	63,5	40,3	33,5

Asumiendo que el número de peticiones atendidas es el mismo para todo los rangos de costes del kilómetro (H0), podemos considerar que la probabilidad de que en un rango de precios se atiendan más peticiones que en otro se distribuye como una función binomial y podemos comprobar cuál es la probabilidad de que H0 sea cierta, es decir, que tengan la misma probabilidad de dar más peticiones atendidas, sería de 0,5 para cada par de rangos a comparar.

Podemos observar que en rangos donde el coste por kilómetro es multiplicado por 8 o menos no influye en el número de peticiones atendidas, ya se atienden 100 peticiones en cada una de las seeds menos en una, por lo que la probabilidad de que sean iguales es de casi el 100%. Pero a partir de ese multiplicador de precio, el número de peticiones atendidas disminuye en todas las seeds según vamos aumentando el coste, por lo que si comparamos el coste original con, por ejemplo, el coste multiplicado por 256, vemos que en todas las seeds se atienden menos peticiones debido al aumento del precio. De esta manera, si miramos a las tablas de la binomial, vemos que la probabilidad de eso suceda es 0.001, es decir, bastante remota, por lo que podemos decir el número de peticiones atendidas es mayor con el precio original.

Como conclusión, después de realizar los experimentos hemos visto que a partir del rango donde se multiplica el coste por 8, cada vez se atienden menos peticiones. Esto se debe a que al aumentar el precio por kilómetro no sea factible atender una petición si está lo suficientemente lejos como para que lo que ganemos por atender dicha petición sea menor que el precio a pagar por kilómetro recorrido. Por esta razón, cada vez se atienden menos peticiones según vamos aumentando el coste.

2.7. Experimento variando horas de trabajo

Hasta ahora, habíamos considerado que se trabaja durante un día, es decir, 24 horas.

Para este experimento vamos a considerar que se trabaja durante 23 horas o 25 horas para ver cómo afecta este cambio en el beneficio obtenido.

Utilizaremos el escenario del primer experimento, con los operadores, estados iniciales y las diez seeds escogidas también en ese experimento y mediremos el beneficio que se obtiene si se trabaja durante 23 horas y durante 25 horas.

De esta manera, el experimento se resumiría de la siguiente forma:

Observación: Puede que el beneficio obtenido en un dia se vea afectado si se trabaja una hora más o una hora menos.

Planteamiento: Utilizamos el escenario del primer experimento que tiene 100 gasolineras, 10 centros y un camión por centro, y observamos si el beneficio obtenido en un dia se ve alterado al aumentar o disminuir una hora de trabajo.

Hipótesis: El beneficio obtenido es el mismo que si se trabaja una hora más o una hora menos (H0) o de lo contrario, el beneficio se ve alterado si se trabaja una hora de más o una hora de menos.

Método: Elegimos diez seeds diferentes al azar fijadas en el primer experimento, cada una se utilizará para hacer un experimento cambiando los kilómetros que puede recorrer un camión en un dia, 80 kilómetros en concreto, se experimenta en problemas de 100 gasolineras, 10 centros y un camión por centro, se usa el algoritmo de Hill Climbing, con los operadores y estado inicial del primer experimento y se mide el beneficio obtenido en un dia.

Resultados: A continuación se muestran los resultados obtenidos al realizar los experimentos al aumentar o disminuir una hora de trabajo:

Seed/Precio	23 horas	24 horas	25 horas
464	95488	95356	95404

1923	95472	95472	95472
850	94424	94264	94392
3010	94992	95284	95284
1378	94508	94636	94636
2981	95852	95724	95640
4368	93588	93572	93572
311	94680	94660	94580
1458	95064	94888	94932
4386	92100	92092	92332

Si hacemos la media de los beneficios obtenidos en cada uno de los casos en las diez seeds, obtenemos los siguientes valores:

	23 horas	24 horas	25 horas
Beneficio	94616,8	94594,8	94624,4

Primero vamos a comparar el caso original y el caso en el que se trabaja una hora menos. En los resultados podemos observar que en 7 seeds el beneficio es mayor si se trabaja una hora menos, en una es el mismo y en dos de ellas es menor que en el original aunque las diferencias del beneficio son mínimas, si observamos los valores medios en las diez réplicas vemos que el beneficio es prácticamente el mismo.

Para comprobar si son iguales o no lo son, vamos a realizar un test de t-Student para pares de muestras. Con un intervalo de confianza del 95%, obtenemos un p-valor igual a 0,64167. Como este valor es más grande que 0,05 que es nuestro alpha, no tenemos evidencia suficiente para rechazar la hipótesis nula y por lo tanto, no podemos decir que haya diferencia entre los beneficios obtenidos al trabajar 23 horas o 24 horas.

Seguidamente, vamos a comparar el caso original y el caso en el que se trabaja una hora más. En los resultados se puede observar que en 4 seeds los resultados son los mismos, en 4 los beneficios son mayores si se trabaja una hora más, y en dos de ellas el beneficio es menor si se trabaja una hora más. Si observamos las medias de las 10 seeds, podemos ver que los beneficios son bastante similares pero hay un poco más de diferencia que en el caso anterior.

Para comprobar si son iguales o no lo son, vamos a realizar otro test de t-Student para pares de muestras. Con un intervalo de confianza del 95%, obtenemos un p-valor igual a 0,35495. Como este valor es más grande que 0,05 que es nuestro alpha, tampoco tenemos evidencia suficiente para rechazar la hipótesis nula y por lo tanto, no podemos decir que haya diferencia entre los beneficios obtenidos al trabajar 24 horas o 25 horas.

Como conclusión, después de realizar los experimentos para los diferentes casos, hemos visto que no tenemos evidencia suficiente como para decir que el beneficio obtenido es alterado si se trabaja una hora de más o una hora de menos. Esto podría deberse a que como se minimizan los kilómetros recorridos de los camiones, pocas veces un camión recorra más de 560 kilómetros, que serían los que corresponden a trabajar 23 horas, y por lo tanto, el beneficio obtenido sería el mismo o muy parecido. Como consecuencia, esto también pasaría en el caso de trabajar una hora más de lo habitual, ya que se obtendría el mismo beneficio o similar por el mismo motivo.

3. Trabajo en equipo y organización

Nuestro equipo está formado por dos estudiantes de la especialidad de computación, Francisco y Sergi, y uno de software, Raul. Aunque había más presión en quienes hacen IA como materia obligatoria, la diferencia no ha influido en nuestra forma de trabajar y repartirnos el trabajo.

Debido a la diferencia en horarios, la comunicación se ha realizado principalmente por whatsapp, donde discutíamos la mayoría de aspectos del problema, avisábamos cada vez que se realizaba un cambio en alguna clase del proyecto y criticábamos los códigos del resto de miembros. El código lo hemos compartido a través de github.

Hemos repartido la carga de trabajo de forma dinámica, cada uno iba escribiendo código cuando podía y confiando en que el resto del equipo lo revisaría y programaría otras partes. Este método es un poco arriesgado, ya que se basa en la confianza entre los miembros, pero ha resultado satisfactorio, los tres hemos tenido una carga de trabajo bastante parecida y entendemos todo el código, incluyendo las partes que han programado los otros componentes del equipo.