Q1 2020-2021

Azamon

Documentación de la práctica de búsqueda local

Trabajo realizado por:
Carles Pamies
Albert Obradors
Álvaro Mañoso

Implementación del estado

Descripción

Los elementos de nuestra clase Estado son los siguientes:

- **listaPaquetes**: Objeto de la clase Paquetes que contiene todos los paquetes (objetos de la clase Paquete) que deben ser asignados a las ofertas de transporte.

- **listaOfertas**: Objeto de la clase Transporte que contiene todas las ofertas de transporte (objetos de la clase Oferta) de cada compañía local a las que se le debe asignar los paquetes.

- **asignacionPaquetes**: Vector de enteros donde cada posición hace referencia al paquete con esa misma posición en listaPaquetes. El contenido en cada posición indica a la oferta a la que está asignado el paquete.

  Por lo tanto, si asignacionPaquetes[i] = k, el Paquete de posición i en listaPaquetes esta asignado a la oferta de posición k en listaOfertas.

- **capacidadRestanteOferta**: Vector de enteros donde cada posición hace referencia a la oferta con esa misma posición en listaOfertas. El contenido de cada posición indica cuantos kilogramos de peso todavía tiene disponibles para poder asignar paquetes esa oferta.

  Por lo tanto si asignacionPaquetes[i] = k, a la Oferta de posición i en listaOfertas todavia se le pueden asignar k kilogramos más.

- **costeAlmacenamieno**: Número real que contiene el valor del coste actual de almacenamiento del estado.
- **costeTransporte**: Número real que contiene el valor del coste actual de transporte del estado.

- **felicidad**: Número entero que contiene el valor de la felicidad actual del estado.

Justificación

Hemos elegido esta implementación del estado ya que para asignar un Paquete a una Oferta tan solo basta con modificar una posición del vector asignacionPaquetes que tiene un coste temporal muy reducido.
Además el cálculo de coste, felicidad y capacidadActual también tiene un coste temporal reducido ya que se va calculando dinámicamente sin necesidad de tenerlos que recalcular.

Como se puede observar no hemos tenido en cuenta las ganancias de cada paquete, por lo que el coste representa solo las pérdidas. Hemos decidido esto porque pensamos que al ser las ganancias un valor constante de cada paquete, no haría variar nuestras soluciones y de esta manera nos ahorramos tiempo en calcularlo.

Implementación de los operadores

Descripción

Los operadores que hemos definido son:

- **moverPaquete**: Mueve un paquete de una oferta a otra haciendo todas las comprobaciones para realizar dicha acción:
  - La oferta destino no es igual a la oferta origen.
  - El paquete se entregaría en un plazo compatible con su prioridad.
  - El paquete cabe en la oferta.

- **intercambiarPaquete**: Intercambia un paquete con otro, es decir si hay dos paquetes, p1 y p2, asigna el paquete p1 a la oferta a la que estaba asignado p2 y viceversa. También hace todas las comprobaciones necesarias:
  - Los dos paquetes no son el mismo paquete.
  - Los dos paquete no están asignados a la misma oferta.
  - Cada paquete se entregaría en un plazo compatible con su prioridad.
  - Los dos paquetes caben en las dos ofertas a las que van dirigidos.

Justificación

Utilizamos estos dos operadores ya que con ellos abarcamos todo el espacio de soluciones del problema.
No utilizamos un operador para quitar paquetes

Implementación de las soluciones iniciales

Descripción

Hemos definido dos métodos que generan soluciones iniciales, son los siguientes:

- **generarEstadoInicial1**: Este método funciona de tal manera que ordena la lista de ofertas de más baratas a más caras (respecto a €/Kg) y los paquetes de más prioritario a menos.
  Tras el proceso anterior asigna todos los paquetes de manera ordenada en la lista de ofertas.
  De esta manera conseguimos un estado inicial muy bueno y próximo al máximo local.

- **generarEstadoInicial2**: Este método funciona de tal manera que ordena las ofertas de más caras a más baratas (respecto €/Kg) y los paquetes de más prioritario a menos.
  Tras el proceso anterior asigna todos lo paquetes de manera ordenada en la lista de ofertas.
  De esta manera conseguimos un estado inicial muy malo por lo que los algoritmos de búsqueda local pueden llegar a un resultado final mejor al estar lejos de los máximos locales

Justificación

Utilizamos estos os funciones generadoras ya que de esta manera podemos observar los resultados de dos casuísticas completamente diferentes.

Implementación de las funciones heurísticas

Descripción

Hemos definido las dos funciones heurísticas siguientes:

- **FuncionHeuristica**: La definición de este heurístico se basa en tener en cuenta solo el coste del estado devolviendo el valor del mismo.

- **FuncionHeuristicaDoble**: La definición de este heurístico se basa en tener en cuenta el coste y felicidad del estado de la siguiente manera: coste-felicidad*K.

  Hemos decidido multiplicar el valor de la felicidad por K para regular la importancia y observar en los experimentos sobre que valor de K deberíamos decidirnos.

Justificación

Hemos decidido definir así el segundo heurístico ya que de esta manera conseguimos darle la misma importancia a cada uno de los dos valores.

Experimentos

1 | Determinar qué conjunto de operadores da mejores resultados para una función heurística que optimice el primer criterio con un escenario en el que el número de paquetes a enviar es 100 y la proporción del peso transportable por las ofertas es 1, 2. Deberéis usar el algoritmo de Hill Climbing. Escoged una de las estrategias de inicialización de entre las que proponéis. A partir de estos resultados deberéis fijar los operadores para el resto de experimentos.

Observaciones

Podemos obtener mejores resultados dependiendo de los operadores que utilicemos.

Planteamiento

Escogemos diferentes operadores, los combinamos de diferentes maneras y obtenemos los resultados.

Hipótesis

Todos las combinaciones de operadores dan los mismos resultados (H0) o los hay que devuelven mejores.

Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y combinación de los operadores.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta de 1,2.
- Usaremos el algoritmo Hill Climbing para hallar la solución.
- Recogeremos diferentes valores para comparar las soluciones.

Resultados

Mover		Intercambiar		Mover + Intercambiar	
Coste	Tiempo	Coste	Tiempo	Coste	Tiempo
796	116	1.000	78	795	345
984	67	1.194	42	983	173
1.008	38	1.296	39	1.008	114
816	32	996	20	816	123
925	63	1.139	51	925	123
1.018	26	1.154	33	1.016	98
968	13	1.120	5	968	68
911	21	1.112	21	908	81
800	19	929	18	800	74
964	16	1.149	23	963	85
919	41,1	1.109	33	918	128,4

Tiempo calculado en milisegundos

Conclusiones

Como podemos observar, la mejor opción respecto al resultado podría ser usar el operador de mover tanto como el de mover + intercambiar ya que devuelven soluciones casi idénticas, pero observando el tiempo de ejecución usar solo el operador de mover es mucho más eficiente.

2 | Determinar qué estrategia de generación de la solución inicial da mejores resultados para la función heurística usada en el apartado anterior, con el escenario del apartado anterior y usando el algoritmo de Hill Climbing. A partir de estos resultados deberéis fijar también la estrategia de generación de la solución inicial para el resto de experimentos.

Observaciones

Podemos obtener mejores resultados dependiendo del estado inicial que generemos.

Planteamiento

Escogemos dos maneras diferentes de generar el estado inicial y analizaremos los resultados.

Hipótesis

Los dos métodos de generar el estado inicial dan el mismo resultado (H0) o uno consigue dar mejor resultado que el otro.

Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y método de generación.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta de 1,2.
- Usaremos el algoritmo Hill Climbing para hallar la solución.
- Recogeremos diferentes valores para comparar las soluciones.

Resultados

Funcion Generadora 1		Funcion Generadora 2	
Coste	Tiempo	Coste	Tiempo
795	84	795	367
946	69	946	161
1.012	17	1.011	131
898.76	33	899	143
888	35	888	121
875	4	875	93
786	14	785	87
1.008	25	1.008	109
960	17	960	166
957	16	957	89
914	31,4	912	146,7

Tiempo calculado en milisegundos

Conclusiones

Como podemos observar, las dos funciones generadoras obtienen prácticamente el mismo resultado, pero la primera consigue obtenerlo de una manera mucho más eficiente que la segunda.

Aún así para realizar los futuros experimentos utilizaremos la 2a opción debido a que nos interesa saber el coste mínimo por poca diferencia que tenga para acotar bien los problemas.

Además de que nuestra Función Generadora 1 siempre nos sitúa muy cerca del máximo local, cosa que no nos interesa si queremos analizar el comportamiento de los algoritmos de Aima.

3 | Determinar los parámetros que dan mejor resultado para el Simulated Annealing con el mismo escenario, usando la misma función heurística y los operadores y la estrategia de generación de la solución inicial escogidos en los experimentos anteriores.

Observaciones

Podemos obtener mejores resultados para dependiendo de los parámetros escogidos para el Simulated Annealing.

Planteamiento

Escogemos varios valores diferentes para los parámetros de: número total de iteraciones, iteraciones por cada cambio de temperatura, y los parámetros k y λ de la función de aceptación de estados. Posteriormente, analizaremos los resultados.

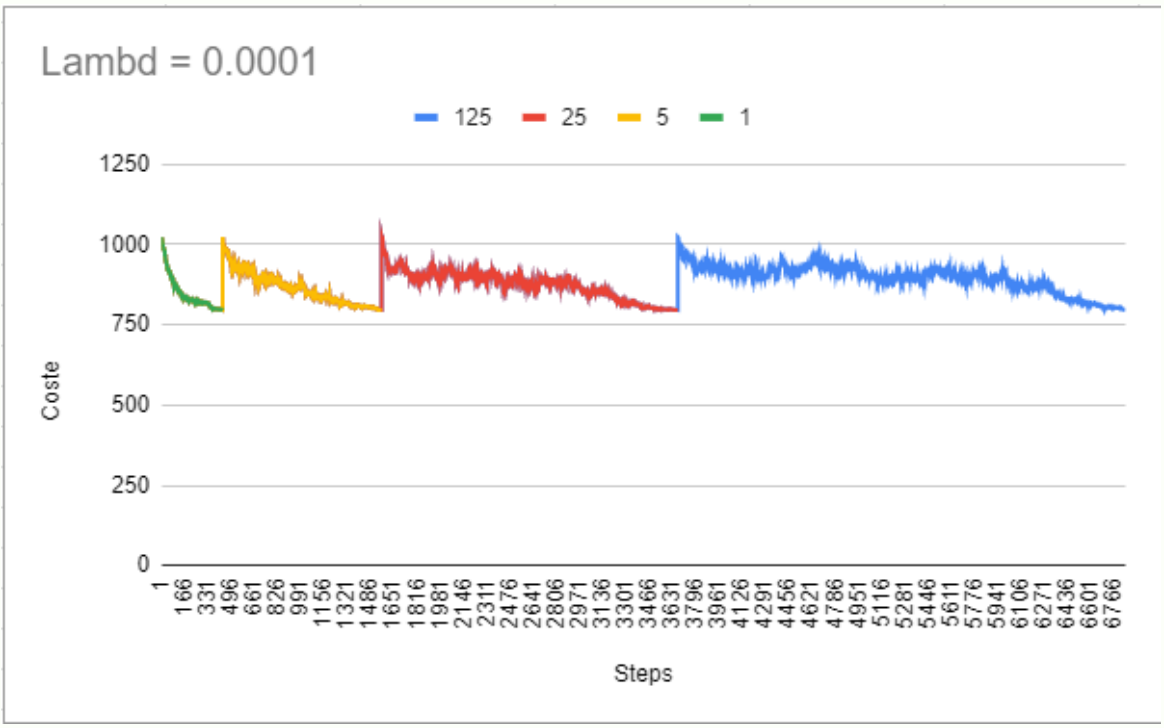
Hipótesis

Cambiar los parámetros del Simulated Annealing no hace que el resultado varíe (H0) o unos parámetros en concreto consiguen dar mejor resultado que el resto.

Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y el primer método de generación.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta de 1,2.
- Usaremos el algoritmo Simulated Annealing para hallar la solución.
- Usaremos un número de iteraciones grandes para k = {1, 5, 25, 125} y para λ = {1, 0.1, 0.01, 0.0001}.
- Recogeremos diferentes valores para comparar las soluciones.

Resultados



Tiempo de ejecución en milisegundos en función de la proporción de las ofertas

796,025	795,415	795,685	796,405

Valores finales para los diferentes valores de K

Conclusiones

Como podemos ver, se puede apreciar muy bien como funciona el parámetro K en Simulated Annealing.

Tras probar diferentes semillas (la anterior gráfica representa la semilla 1234) hemos podido observar que los resultados son prácticamente iguales para los diferentes valores de K, por lo tanto, en nuestro experimento es mejor usar K = 1 para ser más eficientes.

4 | Dado el escenario de los apartados anteriores, estudiad como evoluciona el tiempo de ejecución para hallar la solución en función del número paquetes y la proporción de peso transportable.

Observaciones

El tiempo empleado en hallar la solución va a variar en función del número de paquetes y la proporción del peso transportable, en este experimento queremos cuantificar ésta diferencia.

Planteamiento

- Fijamos el número de paquetes en 100 y vamos aumentando la proporción del peso transportable desde 1, 2 de 0, 2 en 0, 2 hasta ver la tendencia.
- Fijamos la proporción de peso en 1, 2 y vamos aumentando el número de paquetes desde 100 de 50 en 50 hasta ver la tendencia.

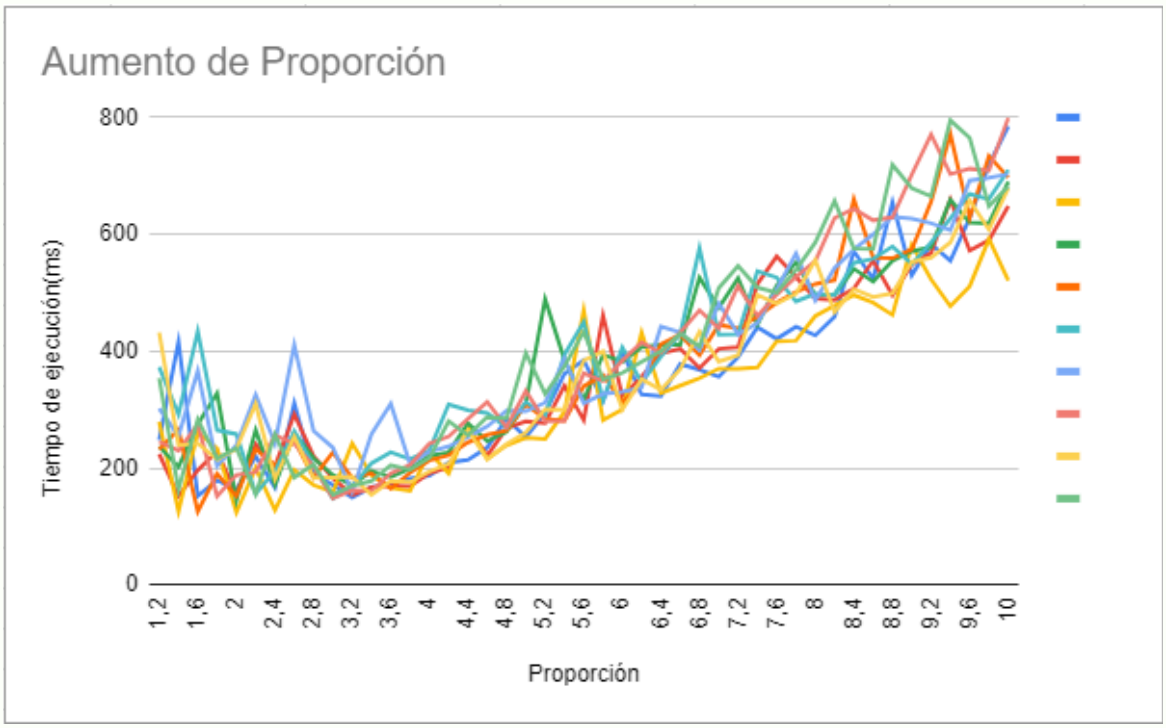
Hipótesis

Aumentar la cantidad de paquetes y la proporción hace que el tiempo en hallar una solución incremente proporcionalmente(H0), o hay un valor o valores a partir de los cuales el tiempo aumenta exponencialmente.

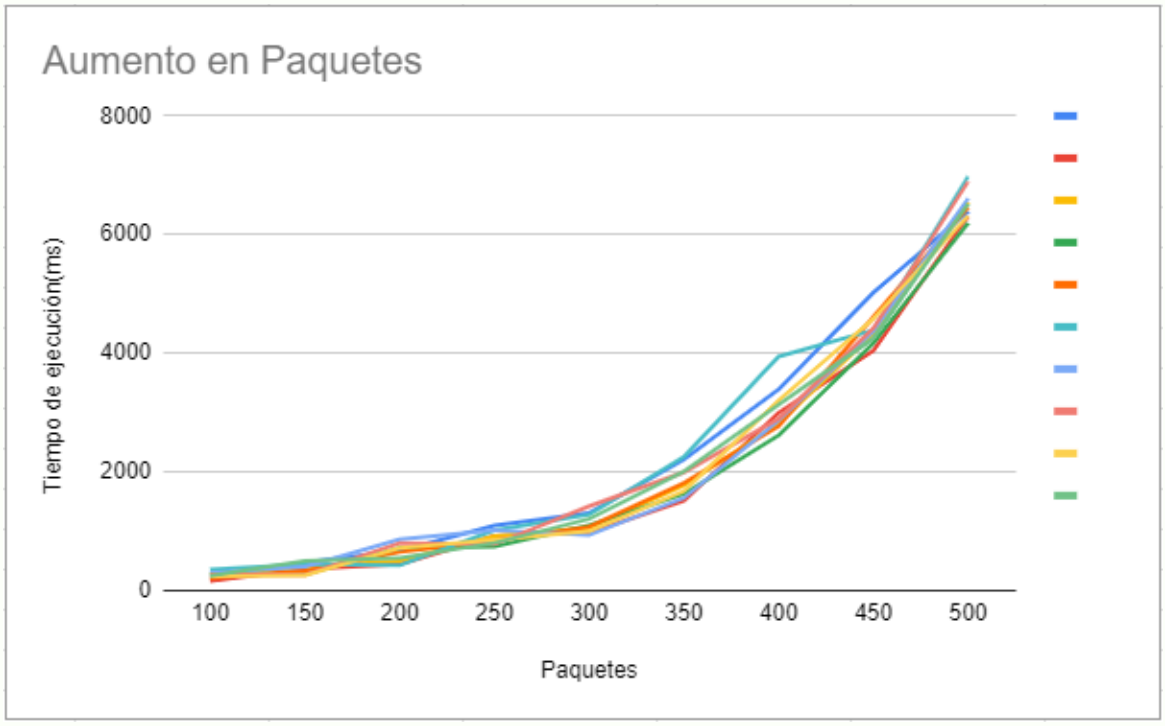
Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y el primer método de generación, con la misma heurística.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta inicial de 1,2 con un aumento progresivo de 0,2 hasta 2.
- Experimentos de 100 paquetes iniciales con un aumento de 50 paquetes por experimento y proporción del peso paquetes/oferta inicial de 1,2.
- Usaremos el algoritmo Hill Climbing para hallar la solución.

Resultados



Tiempo de ejecución en milisegundos en función de la proporción de las ofertas



Tiempo de ejecución en milisegundos en función de la cantidad de paquetes

Conclusiones

Podemos sacar las conclusiones siguientes:

- Respecto a aumentar la proporción de las ofertas
  Vemos que el crecimiento del coste temporal es de forma lineal ya que la proporción solo afecta al número de las ofertas y este solo implica aumentar de la misma manera el uso de los operadores.
  Por lo tanto, dada una función sucesora de coste O(n*m) siendo n el número de paquetes y m el número de ofertas, una oferta más es usar un operador una vez más por paquete, es decir, aumentar m en 1.

- Respecto a aumentar la cantidad de paquetes
  Vemos que el crecimiento del coste temporal es de forma cuadrática ya que cuantos más paquetes tiene el problema, más ofertas tiene también.
  Por lo tanto, dada una función sucesora de coste O(n*m) siendo n el número de paquetes y m el número de ofertas, un paquete más puede implicar una oferta más, así que, n*m tendería a n^2.

5 | Analizando los resultados del experimento en el que se aumenta la proporción de las ofertas ¿cual es el comportamiento del coste de transporte y almacenamiento? ¿merece la pena ir aumentando el número de ofertas?

Observaciones

Nos preguntamos como variará el coste de transporte y almacenamiento al ir aumentando la proporción de las ofertas.

Planteamiento

Fijamos el número de paquetes en 100 y vamos aumentando la proporción del peso transportable desde 1, 2 de 0, 2 en 0, 2 hasta ver la tendencia.

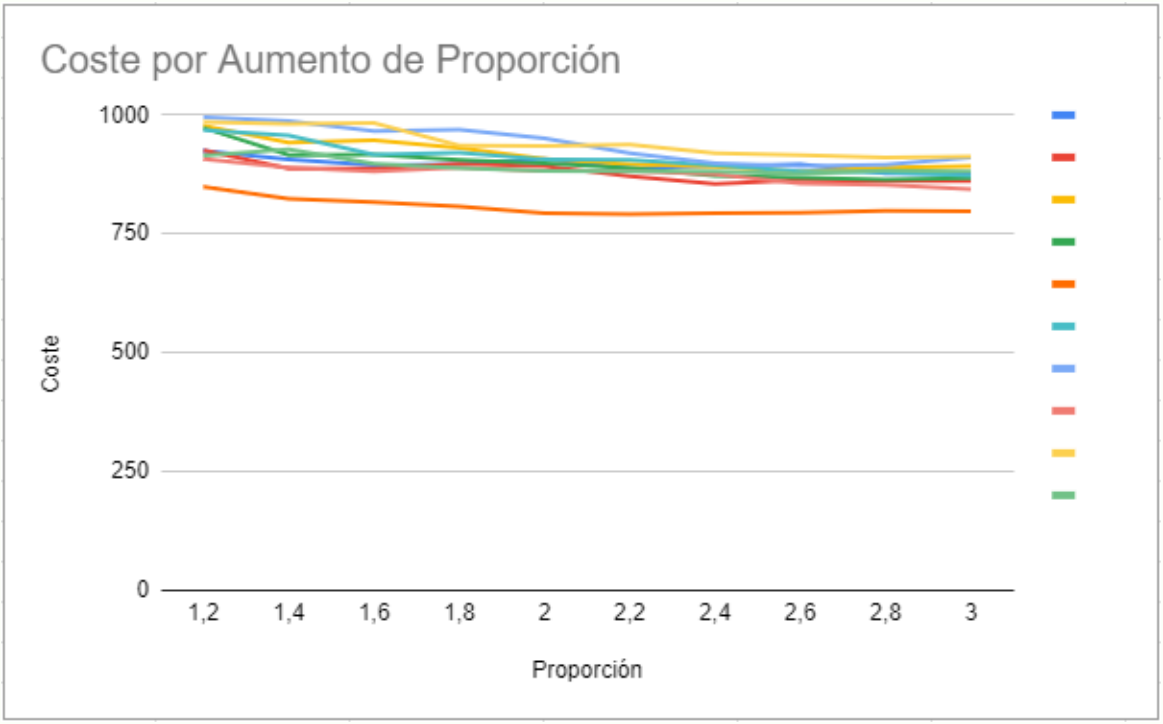
Hipótesis

Aumentar la proporción hace que el coste no cambie (H0), o el coste cambia según la proporción de las ofertas.

Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y el primer método de generación, con la misma heurística.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta inicial de 1,2 con un aumento progresivo de 0,2.
- Usaremos el algoritmo Hill Climbing para hallar la solución.

Resultados



Coste del estado en función de la proporción de las ofertas

Conclusiones

Como podemos observar, la variación es muy poca y solo sucede al principio, llegado un punto el coste se mantendría constante, por lo que no merece la pena ir aumentando el número de ofertas.

Esto pasa porque con una proporción acotada, los paquetes deben distribuirse en un número límite de ofertas, pero si aumentamos la proporción llegará el punto que las pocas ofertas más baratas, quedando ofertas vacías, podrán contener todo el rango de paquetes del problema.

6 | Ahora queremos estudiar como afecta la felicidad de los usuarios al coste de transporte y almacenamiento.
Dado el escenario del primer apartado, estimad como varían los costes de transporte y almacenamiento y el tiempo de ejecución para hallar la solución con el Hill Climbing cambiando la ponderación que se da a este concepto en la función heurística. Deberéis pensar y justificar como introducís este elemento
en la función heurística y como hacéis la exploración de su ponderación en la función. No hace falta que la exploración sea exhaustiva, solo hace falta que veáis tendencias.

Observaciones

Nos preguntamos como variará el coste de transporte y almacenamiento al ir cambiando la ponderación que le damos a la felicidad en la segunda heurística.

Planteamiento

Para ver si el coste cambia, iremos aumentando progresivamente la ponderación de la felicidad.

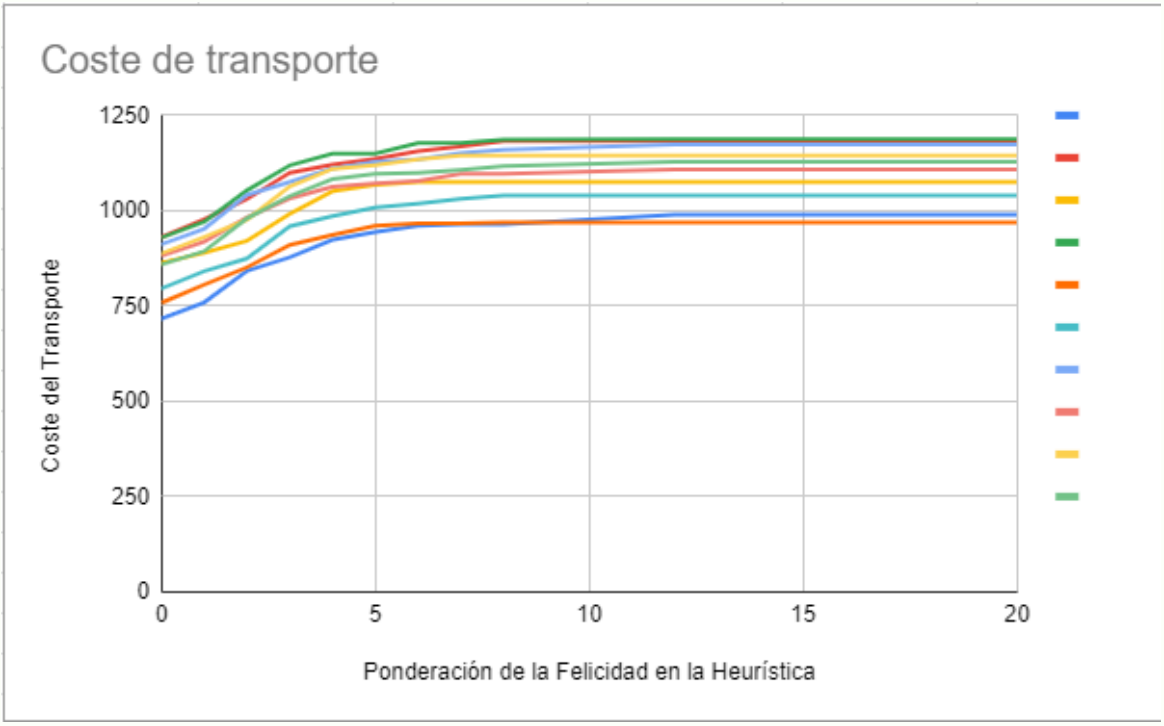
Hipótesis

Aumentar la ponderación hace que el coste incremente(H0), o el coste no cambia al aumentar la ponderación de la felicidad.

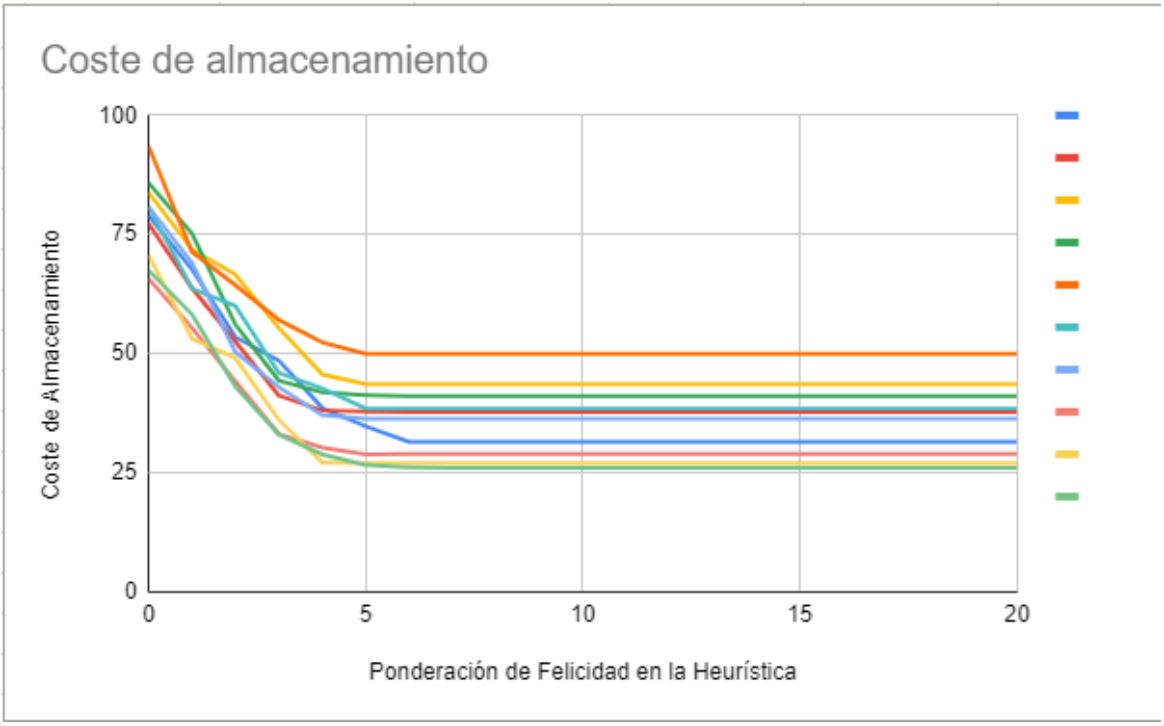
Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y el primer método de generación, con la segunda heurística.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta inicial de 1,2.
- Empezando con una ponderación de 0 para la felicidad, vamos a ir aumentando la misma progresivamente hasta 20.
- Usaremos el algoritmo Hill Climbing para hallar la solución.

Resultados



Coste del transporte en función de la felicidad en la heurística



Coste del almacenamiento en función de la felicidad en la heurística

Conclusiones

Como podemos observar, el coste de transporte tiende a disminuir i el de almacenamiento a bajar. Esto es debido a que si nos basamos más en la felicidad, entregamos los paquetes antes (disminuye el coste de guardarlos más días) y las ofertas más rápidas son más caras (aumenta el precio de transporte).

Al final, si hacemos la suma de los dos veremos que el coste final cada vez es más alto, pues el coste de transporte es generalmente más pronunciado que el de almacenamiento y este se antepone.

7 | Repetid los experimentos con Simulated Annealing y comparad los resultados. ¿Hay diferencias en las tendencias que observáis entre los dos algoritmos?

Observaciones

Nos preguntamos como variará el coste de transporte y almacenamiento al ir cambiando la ponderación que le damos a la felicidad en la segunda heurística, esta vez con el algoritmo de Simulated Annealing.

Planteamiento

Para ver si el coste cambia, iremos aumentando progresivamente la ponderación de la felicidad.

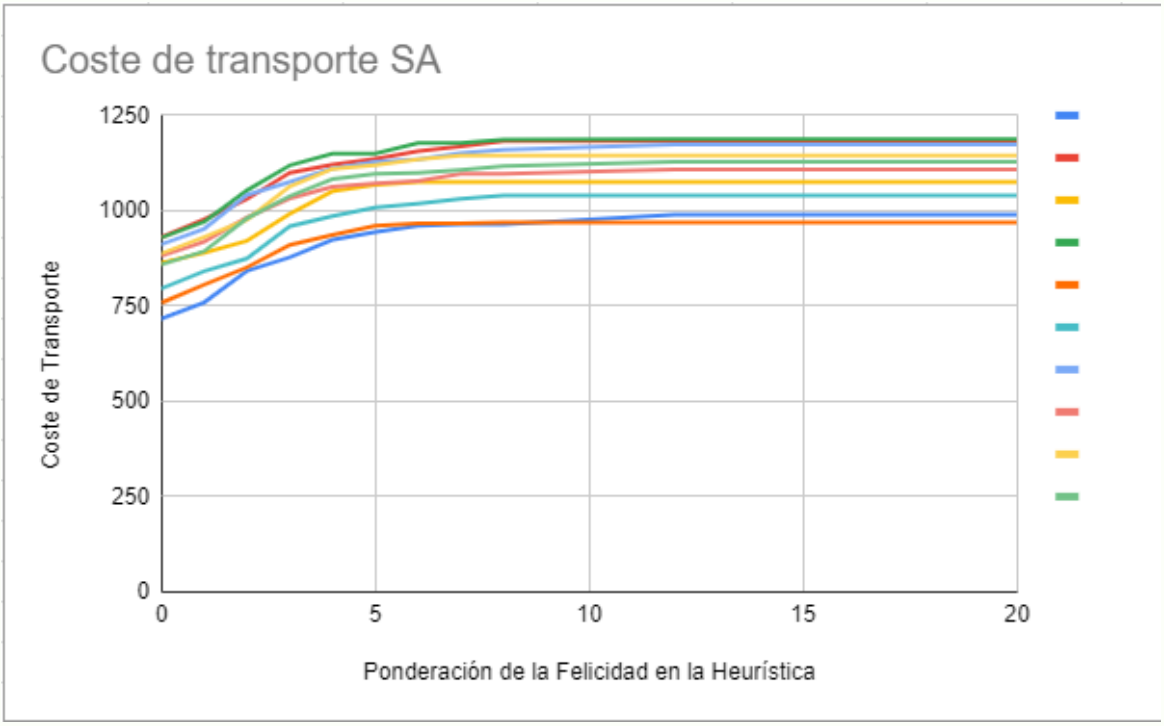
Hipótesis

Aumentar la ponderación con el algoritmo de Simulated Annealing hace que el coste incremente igual que en el Hill Climbing(H0), o el coste cambia de manera diferente al aumentar la ponderación de la felicidad.

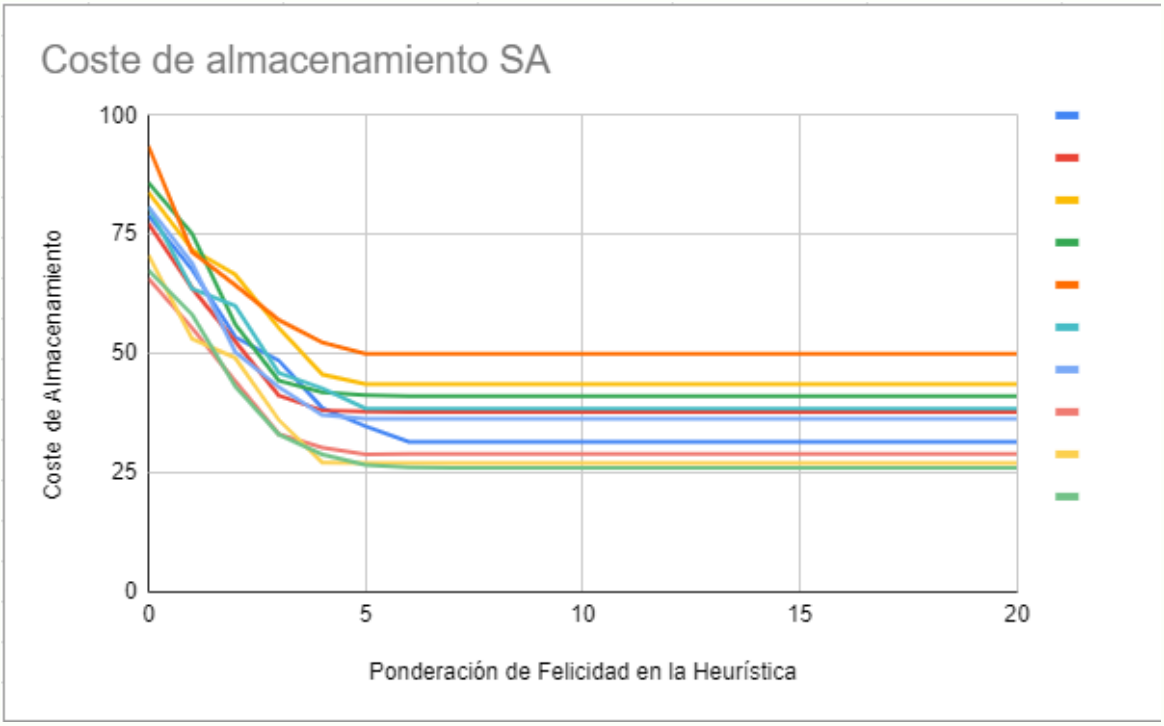
Método

- Elegiremos 10 semillas diferentes.
- Ejecutaremos un experimento para cada semilla y el primer método de generación, con la segunda heurística.
- Experimentos de 100 paquetes y proporción del peso paquetes/oferta inicial de 1,2.
- Empezando con una ponderación de 0 para la felicidad, vamos a ir aumentando la misma progresivamente hasta 20.
- Usaremos el algoritmo Simulated Annealing para hallar la solución.

Resultados



Coste del transporte en función de la felicidad en la heurística



Coste del almacenamiento en función de la felicidad en la heurística

Conclusiones

Al igual que en el experimento anterior con el algoritmo de Hill Climbing, el coste de transporte tiende a disminuir i el de almacenamiento a bajar. Esto es debido a que si nos basamos más en la felicidad, entregamos los paquetes antes (disminuye el coste de guardarlos más días) y las ofertas más rápidas son más caras (aumenta el precio de transporte).

Al final, si hacemos la suma de los dos veremos que el coste final cada vez es más alto, pues el coste de transporte es generalmente más pronunciado que el de almacenamiento y este se antepone.

Concluimos que el Algoritmo Simulated Annealing realiza un trabajo casi igual al Hill Climbing.

8 | Hemos asumido un coste de almacenamiento fijo diario de 0,25 euros por kilo. Sin hacer ningún experimento ¿Cómo cambiarían las soluciones si variamos este precio al alza o a la baja?

Si variamos el precio de almacenamiento a la baja, creemos que el coste total disminuirá pero la felicidad será inferior puesto que se penalizará menos que un paquete permanezca más tiempo en el almacén y por consecuencia el paquete no necesariamente llegaría antes al cliente.

Si variamos el precio al alza, creemos que el coste total aumentará y puede que también aumente la felicidad en según que casos. El sistema intentará tener los paquetes en el almacén el mínimo tiempo posible y eso hará que envíe algunos de ellos antes de tiempo a los clientes.