Práctica de búsqueda local

Laboratorio de Inteligencia Artificial

Marion Not - Michael Boris Mandirola

Primavera 2010-2011

Índice

Introducción

El objetivo de esta práctica es analizar y resolver un problema de optimización logística mediante algoritmos de búsqueda local. Definiremos la representación del problema como estado y estudiaremos la influencia de los elementos que intervienen en esta búsqueda y que hemos visto a clase (estado inicial, función heurística y operadores) bajo diferentes condiciones sobre los parámetros del problema.

Usaremos dos tipos de algoritmos de búsqueda local : el Hill Climbing y el Simulated Annealing. Como el objetivo de la práctica no es la implementación de estos algoritmos, usaremos las herramientas proporcionadas por el package AIMA. En el caso del Simulated Annealing, también estudiaremos la influencia de los parámetros del algoritmo.

1. Representación del problema

El contexto del problema es el siguiente : Una empresa de transporte está contratada por una compañía para gestionar el encaminamiento de productos desde un almacén central hasta unos centros de producción.

Cada día, los centros realizan un conjunto de peticiones de diferentes tipos y cantidades de productos que se tienen que entregar a una cierta hora. El pago se realiza en función de la cantidad de productos entregados y la puntualidad con la cual han llegado al centro.

La empresa de transporte quiere optimizar estas entregas, sabiendo que manda un transporte a cada centro en cada hora en punto del día.

No tenemos que gestionar la recogida de productos en el almacén, ni el recorrido de los transportes, ni la cantidad de camiones físicos que se necesitan : Suponemos que la empresa dispone de una flota suficiente para hacer todas las entregas en tiempo.

Solucionaremos este problema para un solo día, pues las peticiones no entregadas no estarán desplazadas al día siguiente.

1.1. Identificación de los elementos relevantes

Para cada problema, tendremos unos elementos constantes, unos elementos específicos del problema y unos elementos variables que se tendrán que determinar para llegar a la solución.

1.1.1. Constantes

Centros de producción : Como lo hemos dicho, el número de centros esta fijado a 6.

Horas de entrega : Las entregas se harán a cada hora en punto del día. La primera se hará a las 8 y la última a las 17, para un total de 10 horas de entrega.

Transportes : Tendremos un transporte para cada hora del día y cada centro, es decir 60 transportes en total. La capacidad de los camiones usados sera de 500, 1000 o 2000kgs, en proporción variable.

Peticiones : Las peticiones se harán para una de las horas de entregas y para una cantidad de productos de 100, 200, 300, 400 o 500kgs.

1.1.2. Elementos específicos del problema

Peticiones : Todas las peticiones estarán generadas al principio de la resolución mediante unos parámetros especificados por el usuario. Por lo tanto, serán diferentes para cada problema. Se tendrán que especificar :

- El número de peticiones
- La distribución de probabilidad de cantidad de productos de las entregas
- La distribución de probabilidad de horas de entrega

Las peticiones estarán repartidas de forma equiprobable entre los centros.

Transportes : Se tendrá que especificar la distribución de probabilidad de capacidades de los transportes, es decir cuantos de los 60 transportes tendrán una capacidad de 500kgs, cuantos de 1000 y cuantos de 2000.

Parámetros de la búsqueda : Se precisará que tipo de estado inicial, algoritmo de búsqueda y función heurística se tienen que usar.

1.1.3. Elementos de la solución

Para llegar a la solución, se tendrán que determinar :

- La repartición de los 60 transportes, es decir que capacidad de transporte se asignará à cada hora de cada centro.
- La repartición de las peticiones en los transportes.

1.2. Definición del Estado

Hemos escogido una representación en acuerdo con la decripción del problema de la parte anterior.

1.2.1. Implementación

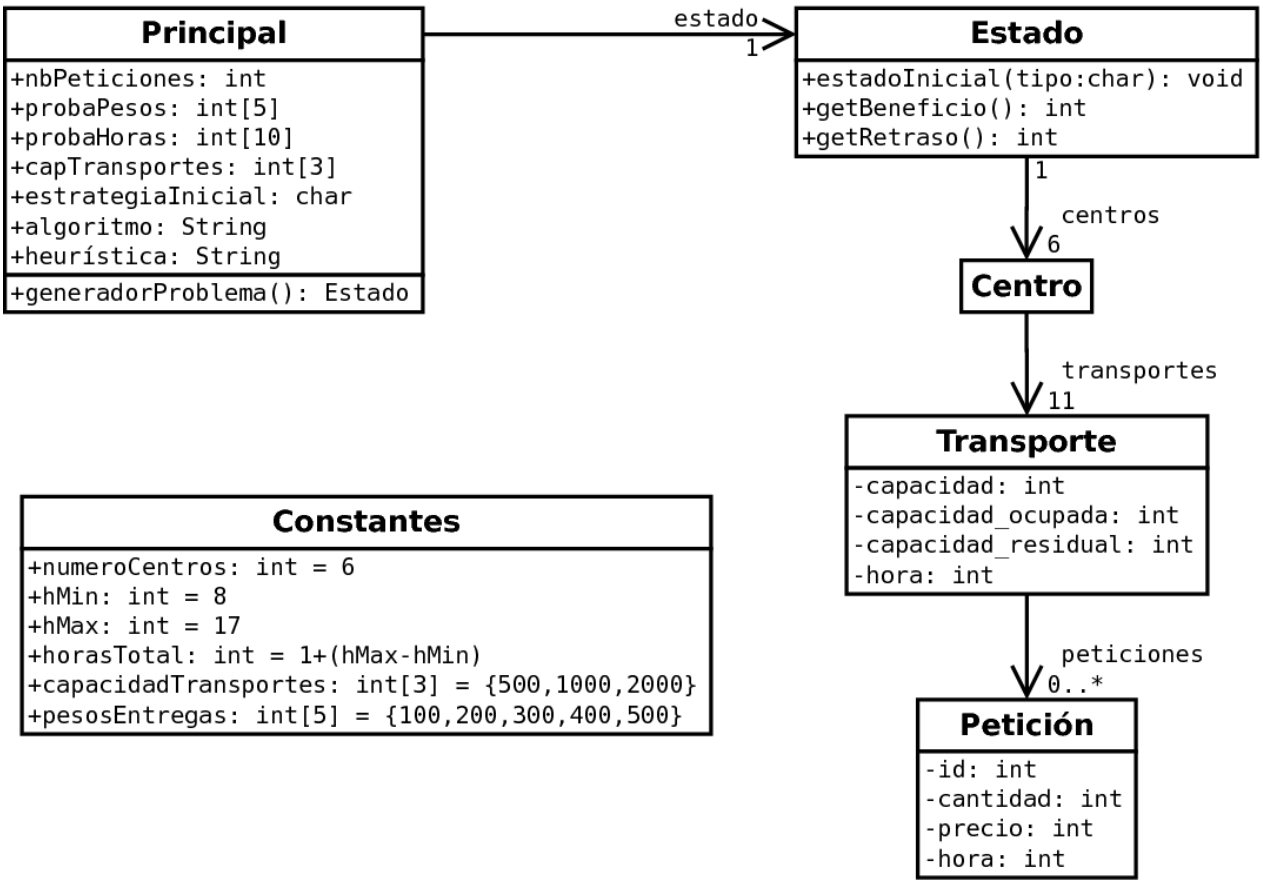

Diagrama de clases de representación del problema

- Una clase singleton Constantes contiene los elementos comunes a todos los problemas.
- Los elementos especificos serán leidos desde un fichero .txt y usados por el Principal para generar un problema dado, es decir un conjunto de instancias de la clase Petición y un Estado “vacio”, donde las peticiones estarán repartidas entre las 6 instancias de clase Centro pero no entregadas. Se crearán 11 instancias de clase Transporte : una para cada hora del día y una para guardar las peticiones no entregadas.
- A continuación, se generará el estado inicial de la búsqueda, es decir que se atribuirá una capacidad a cada unos de los 60 transportes que corresponden a una hora (los Transporte de peticiones no entregadas tendrán capacidad máxima) y se repartirán las peticiones según la estrategia especificada.
- Durante la resolución del problema, las peticiones se desplazarán de un Transporte a otro, dentro del mismo Centro. Tambien se podrá modificar la repartición de las capacidades entre todos los 60 transportes. Ver la sección siguiente para detalles sobre los operadores.

1.2.2. Tamaño del espacio de búsqueda

Si tenemos N peticiones, repartidas igualmente entre 6 centros ($\frac{N}{6}$ peticiones/centro), que se pueden afectar a cualquier de los $10+1$ transportes, contamos con $O(11^N)$ configuraciones posibles para la repartición de las peticiones.
Si añadimos que cada uno de los 60 transportes puede tener una de 3 capacidades, es decir $O(3^{60})$ configuraciones posibles, tenemos un total de $O(11^N x 3^{60})$ *estados posibles.*
Se tiene que tener en cuenta que en verdad la suma de pesos de las peticiones afectadas a un transporte no puede exceder su capacidad, y que tenemos un número limitado de transportes de cada capacidad : Eso reduce el tamaño del espacio de búsqueda.
Este tamaño es bastante importante, pero considerando que hacemos una búsqueda local y que recorriremos solo una fracción del espacio, no es excesivo.

1.3. Operadores

1.3.1. Desplazamiento de una petición dentro de un centro

1.3.2. Intercambio de capacidades de camiones

1.3.3. Operadores no implementados

1.4. Funciones Heurísticas

1.4.1. Maximización de la ganancia

El precio pagado para cada petición entregada sera el indicado en la tabla 1 menos unos 20 % del mismo para cada hora de retraso, es decir que para mas de 5h de retraso le tocara pagar a la empresa de transporte. Si una petición no esta entregada en el día, la empresa de transporte tendrá que pagar el precio de la petición mas unos 20 % para cada hora hasta las 17.

Peso	Precio
100 y 200 kg	*peso* euros
300 y 400 kg	1,5x*peso* euros
500 kg	2x*peso* euros

Tabla 1 : Precio de las entregas.

1.4.2. Minimización de la diferencia entre hora de entrega deseada y efectiva

El retraso en la entrega de una petición es simplemente la diferencia entre la hora de entrega deseada y la efectiva. Si la petición no esta entregada en el día, el retraso es la diferencia entre la hora de entrega deseada y las 8 del día siguiente. No obstante, como solucionemos el problema solo para una jornada, no se tendra que gestionar la entrega de estas peticiones.

1.5. Estados Iniciales

En un primer momento hemos intentado generar estados diferentes con lógicas muy diferentes con la idea de buscar nuestros estados iniciales entre un grupo más amplio y en particular elegir dos estados que tengan características muy diferentes. Es decir tiempo de generación de el estado inicial, complejidad de el algoritmo, posibilidades de evolución, calidad de la solución buscada.

1.5.1. a

A) Este estado divide los camiones de manera ecua entre los centres asignando los camiones con mas capacidad a los transportes mas tempranos. Las entregas están entregadas lo mas pronto posible: se hace una lista de peticiones pertinentes a un centro, se ordenan crecientemente por tiempo de entrega y disminuyendo por precio y se pasa toda la lista poniendo las peticiones en el transporte más pronto que tiene más tiempo libre.

1.5.2. b

B) Este estado ordena las peticiones por precio y hora. Por cada petición, intenta entregarla en la hora pertinente con este algoritmo: Si no hay camión, se pone el camión disponible más pequeño y la petición, si hay camión y bastante capacidad libre también se pone la petición. Si hay camión de capacidad inferior a la capacidad máxima y hay disponibilidad de camiones más grandes, se asigna un camión mas grande en lugar de lo más pequeño y se pone la petición. Además, si no hay camiones libres más grandes intenta hacer lo mismo en las horas más tempranas hasta las 8. Si no tiene éxito, intenta hacerlo en las horas después hasta las 17. Después si hay oras sin camión, se asignan los camiones que se quedan libres.

1.5.3. f

F) Los camiones se asignan como en el estado A. Sin ordinación se iteran todas las peticiones intentando ponerlas en su hora de pertinencia. Después, iterativamente por cada petición se intenta poner, las que se quedan no entregadas, en las horas más tempranas de la suya hasta las 8 y si no se tiene éxito, se intenta hacerlo en las horas después hasta las 17.

1.5.4. v

V) Los camiones se asignan como en el estado A. Las peticiones se quedan todas no entregadas.

2. Implementación

2.1. Generación de problemas aleatorios

2.2. Uso del AIMA

2.3. Ejecucción de tests

2.4. Recuperación de resultados

3. Resultados

3.1. Influencia de la solución inicial

3.2. Influencia de los operadores

3.3. Influencia de la heurística