UNIVERSIDAD DE GRANADA
E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Teoría de Algoritmos

Guión de Prácticas

Tercera Práctica: Exploración en grafos

Curso 2003-04

Ingeniería Técnica en Informática de Gestión
Ingeniería Técnica en Informática de Sistemas

2.1. Objetivo

El objetivo de esta práctica es estudiar el comportamiento de las técnicas exactas de diseño de algoritmos *backtracking* y *branch and bound*, basadas en la exploración de grafos, y de las técnicas aproximativas basadas en heurísticas voraces (*greedy*). Para ello se requerirá que el alumno implemente algoritmos basados en estas técnicas para resolver el problema del *viajante de comercio* (PVC) y compare los resultados obtenidos por los diversos algoritmos.

Por otro lado, se persigue fomentar el trabajo en equipo buscando el desarrollo de actitudes y habilidades que permitan la realización exitosa de tareas a través de la colaboración y complementación. Por ello, esta práctica habrá de realizarse en parejas. Para facilitar la organización. Las parejas deberán estar formadas entre alumnos del mismo grupo de prácticas. Una vez organizados, los alumnos deberán comunicar a su profesor de prácticas los integrantes de cada pareja.

2.2. Tareas

Para lograr el objetivo de resolver el problema del viajante de comercio, habrá que diseñar varios algoritmos basados en distintos enfoques y estudiar las propiedades de cada uno de los algoritmos creados (eficacia y eficiencia).

Más concretamente, se deberán realizar las siguientes tareas:

1. Diseñar e implementar tres heurísticas voraces que resuelvan el PVC. Realizar un estudio de la eficiencia de estas heurísticas desde los enfoques teórico, empírico e híbrido.

2. Diseñar e implementar un algoritmo basado en la técnica *backtracking* para resolver el PVC. En este caso se podrá utilizar el esquema que se proporciona en la página `http://decsai.ugr.es/~jmbs/TA` y particularizarlo para este problema. También habrá que calcular una aproximación a la eficiencia de este algoritmo.

3. Diseñar e implementar un algoritmo basado en la técnica *branch and bound* para resolver el PVC. También habrá que calcular una aproximación a la eficiencia de este algoritmo.

4. Realizar un análisis de los resultados obtenidos, comparando el comportamiento de los algoritmos implementados desde el punto de vista

de su eficacia y eficiencia. Para abordar el estudio de la eficacia, los algoritmos se aplicarán a un conjunto de casos del PVC que se facilitarán al alumno.

2.3. Instancias del problema

En `http://decsai.ugr.es/~jmbs/TA` están disponibles nueve casos del PVC, con tamaños comprendidos entre 5 y 25 ciudades, que han sido generados aleatoriamente. En estos casos, cada vértice se describe según sus coordenadas sobre el plano real ($\mathbb{R}^2$). El peso asociado a cada arista es la distancia euclídea entre los dos vértices que une. El nombre de los ficheros es `pvc-X.txt`, donde `X` representa el tamaño del caso.

El formato de dichos ficheros es el siguiente:

- La primera línea presenta un único valor, el tamaño del caso.
- Las siguientes contienen dos valores, los cuales representan las coordenadas de la ciudad correspondiente en el plano. El primer valor se refiere a la coordenada x y el segundo a la y.

Todos los algoritmos implementados deben ser ejecutados sobre los nueve casos, anotándose el tiempo empleado para resolver cada una de ellas, así como el coste de la mejor solución obtenida y **el vector solución que representa ésta**.

2.4. Modo de realizar cada uno de los apartados

2.4.1. Heurísticas Voraces

El alumno deberá implementar tres algoritmos aproximativos para resolver el PVC y poder comparar los resultados obtenidos con las técnicas exactas *backtracking* y *branch and bound*. Estos algoritmos estarán basados en la filosofía voraz.

Existen numerosas heurísticas voraces para resolver el PVC. A continuación, describimos dos heurísticas posibles:

Heurística del vecino más cercano

En este caso, el algoritmo voraz parte de una ciudad concreta, eliminándola del conjunto de nodos y escogiendo como su seguidora en el circuito aquélla

que se encuentre situada a menor distancia. En cada paso se elige de entre las ciudades restantes la que esté a menor distancia de la última que se visitó. El algoritmo finaliza cuando sólo queda una ciudad por visitar.

De este modo, el algoritmo presenta la siguiente forma:

1. *Inicializar el conjunto de nodos con todas las ciudades:* $C \leftarrow V$.
2. *Sol[i]* $\leftarrow$ *j (ciudad de comienzo)*
3. *Eliminar j del conjunto de nodos:* $C \leftarrow C - \{j\}$
4. *Repetir para i=2 hasta n*
   - *Sol[i]* $\leftarrow$ *VecinoMásCercano (Sol[i-1])*
   - *Eliminar Sol[i] del conjunto de nodos:* $A \leftarrow A - \{Sol[i]\}$

Una mejora de esta heurística consiste en aplicarla comenzando cada vez en una ciudad distinta.

Heurística de múltiples fragmentos

Otra posibilidad para el diseño del algoritmo voraz consiste en ir escogiendo en cada paso la arista de menor peso siempre que dicha arista verifique las dos condiciones siguientes:

1. No forme un ciclo con alguna de las aristas ya escogidas (obviamente, excepto en el caso de la última arista escogida, la cual completará el camino volviendo del último nodo al primero).
2. No sea la tercera arista que incida (tenga como destino) en un nodo.

El algoritmo finaliza una vez escogidas n aristas, momento en el cual queda construido un circuito que visita todas las ciudades.

El alumno puede utilizar alguna de estas dos heurísticas o cualquier otra que desee. Una vez implementados los algoritmos, deberá ejecutarlos para resolver los casos proporcionados, construyendo una tabla en la que consten el resultado obtenido y el tiempo necesario para cada instancia.

Finalmente, se habrán de realizar un estudio de la eficacia de las heurísticas desde las perspectivas teórica, empírica e híbrida.

2.4.2. Backtracking

Se ha de implementar un algoritmo exacto basado en la técnica *backtracking* para resolver el PVC. En `http://decsai.ugr.es/~jmbs/TA` se encuentra una implementación en C++ del algoritmo iterativo genérico de esta técnica, que puede ser empleado para implementar el algoritmo solicitado.

De este modo, bastará con adaptar la implementación comentada al PVC, añadiendo las estructuras de datos necesarias para representar el problema e implementando la función de poda concreta para el mismo. Para este problema, dicha función de poda está basada en dos restricciones:

1. El valor escogido para la componente actual no debe de haber sido asignado a ninguna de las componentes anteriores (en ese caso, formaríamos un ciclo en el circuito).

2. El coste acumulado en la solución que está siendo construida no debe superar el coste de la mejor solución obtenida hasta el momento al añadir la nueva ciudad al circuito.

Una vez implementado el algoritmo, deberá ser ejecutado sobre los distintos casos del problema proporcionados, construyendo una tabla que recoja el valor de la solución obtenida y el tiempo empleado para resolver cada uno de ellos.

Se debe obtener también una aproximación del tiempo de ejecución del algoritmo. No es posible obtener una ecuación exacta de este tiempo en algoritmos basados en las técnicas *backtracking* y *branch and bound*, ya que la expresión del mismo depende del número de nodos generados, el cual varía en función de la instancia concreta del problema.

La ecuación de tiempos exacta respondería a la siguiente fórmula:

$$t(n) = N \cdot t_{nodo}(n)$$

donde:

- N es el número de nodos generados en la ejecución del algoritmo sobre la instancia actual del problema.

- $t_{nodo}(n)$ es el tiempo de ejecución de las operaciones efectuadas en cada nodo.

De este modo, para obtener una aproximación general de esta fórmula para nuestro algoritmo deberemos proceder del siguiente modo:

1. Obtener la expresión teórica del tiempo de ejecución de las operaciones realizadas en cada nodo.

2. Calcular empíricamente una tabla de tiempos de esta parte del código cuando resuelve instancia del PVC de distinto tamaño.

3. Haciendo uso de los datos anteriores, obtener la expresión del tiempo de ejecución requerido en cada nodo según el enfoque híbrido (es decir, calcular las constantes ocultas tal y como se hacía en la Práctica 1).

4. Finalmente, ejecutar el algoritmo para los casos proporcionados, contando en cada caso el número de nodos visitado para cada tamaño del problema. Obtener una expresión polinómica aproximada en función de n que aproxime el conjunto de datos obtenido lo mejor posible. Para ello, efectuar cinco regresiones polinómicas: de orden uno ($f(n) = a \cdot n + b$), de orden dos ($f(n) = a \cdot n^2 + b \cdot n + c$), ..., hasta de orden cinco. Escoger aquélla que devuelva menor error con respecto al conjunto de datos.

Una vez efectuados los pasos anteriores, la ecuación de tiempo sería:

$$t(n) = f(n) \cdot t_{nodo}(n)$$

2.4.3. Branch and Bound

Se debe implementar un algoritmo exacto basado en la técnica *branch and bound* para resolver el PVC. Para ello, basta con modificar la implementación del algoritmo *backtracking* realizada para efectuar los dos cambios siguientes:

1. Modificar la técnica de exploración del grafo de estados para que pase a ser una búsqueda con prioridad en lugar de en profundidad.

2. Implementar una función de poda compuesta por dos cotas, una inferior y otra superior. Esta última coincide con la función de poda considerada en la sección anterior para el algoritmo *backtracking*. En cambio, la cota inferior debe realizar una estimación de la mejor solución alcanzable desde el nodo actual. Se evitará la exploración del subárbol cuya raíz es el nodo actual en el caso de que el coste de dicha aproximación sea peor que el de la mejor solución obtenida hasta el momento.

   A continuación se describe una sugerencia para el cálculo del estimador inferior. Dicho estimador está basado en calcular el coste de un circuito ideal considerando:

- El coste real del camino acumulado hasta el momento (es decir, del que parte de la ciudad inicial y recorre las ciudades del camino que llega al nodo actual).
- El mínimo coste necesario para salir del nodo actual hacia otro no visitado aún.
- El mínimo coste necesario para recorrer el resto del camino hasta el último nodo sin pasar por ninguno de los ya visitados.
- El mínimo coste necesario para entrar de nuevo en el nodo inicial, cerrando el circuito.

Se requiere también la obtención de una estimación de la expresión del tiempo de ejecución, que se realizará siguiendo el procedimiento explicado en el apartado anterior.

2.4.4. Estudio comparativo y análisis de los resultados obtenidos

Se deberán construir tablas que recojan los datos referentes a la eficacia (grado de bondad de la solución obtenida para cada instancia del problema) y la eficiencia (tiempo necesario para obtenerla) de los algoritmos implementados. El alumno diseñará a su vez dos gráficas comparativas, una referente a la eficacia de los mismos en cada instancia del problema, y otra representando la eficiencia.

Finalmente, deberá realizar un análisis de los resultados obtenidos, comparando el comportamiento de los algoritmos implementados, desde el punto de vista de los dos parámetros comentados. **Este análisis influirá decisivamente en la calificación de la práctica**.

2.5. Evaluación de la práctica

Para su evaluación, los alumnos deberán entregar una memoria y un disquete, que se ajustarán a las siguientes indicaciones:

2.5.1. Estructura de la memoria

- **¡¡Muy importante!!** La memoria tendrá una portada en la que aparezcan claramente: el título de la práctica, la titulación, los nombres y direcciones de correo electrónico de los alumnos y el nombre de su profesor de prácticas.

Los alumnos se asegurarán de que en la portada aparecen claramente todos los datos indicados.

- Definición del problema del PVC.
- Una sección por cada heurística voraz, que incluya:
  - Diseño
  - Estudio de eficiencia: teórica, empírica e híbrida.
  - Resultados de aplicar los algoritmos a los casos facilitados.
- Algoritmo *backtracking*
  - Diseño del algoritmo especificando sus componentes.
  - Aproximación a la eficiencia del algoritmo.
  - Resultados de aplicar los algoritmos a los casos facilitados.
- Algoritmo *branch and bound*
  - Diseño del algoritmo especificando sus componentes.
  - Aproximación a la eficiencia del algoritmo.
  - Resultados de aplicar los algoritmos a los casos facilitados.

- Análisis comparativo y justificado de las eficiencias y eficacias.

2.5.2. Disquete

Junto con la memoria se entregará un disquete con las implementaciones de los algoritmos indicados. Además, de todos los ficheros fuentes correspondientes, se incluirá un fichero `Makefile` que automatice la compilación.

El disquete irá etiquetado con el título de la práctica, los nombres de los alumnos y su grupo de prácticas.

Finalmente, el disquete se adjuntará dentro de un sobre pequeño pegado a una de las páginas de la memoria.

2.5.3. Otras observaciones

En la página `http://decsai.ugr.es/~jmbs/TA` puede encontrarse este guión en distintos formatos, además de información adicional sobre la práctica.

Fecha límite de entrega: **16 de enero de 2004** a las **14:00**h.