Sexta práctica voluntaria Métodos Matemáticos y Técn. Comp.
Francisco R. Villatoro y Carmen M. García 9 de Abril de 2003
PUNTOS : 0.25 puntos FECHA ENTREGA: Lunes, 28 de Abril de 2003

Programación entera mediante el método Branch & Bound (B&B).

El problema del viajante (Traveling Salesman Problem, o TSP) consiste en un viajante que debe visitar n ciudades (todas) minimizando el costo total del viaje, gasolina, distancia, o cualquier otra medida definida entre cada par de ciudades.

Considere el siguiente TSP: Un ingeniero de ventas debe recorrer todas las ciudades importantes de Andalucía. Suponiendo que el costo es proporcional a la distancia, ¿cuál es el camino óptimo? Según el "Anuario Estadístico de Andalucía de 1997,"la distancia media entre las principales ciudades andaluzas (o de más de 100000 habitantes) es

(Km.)	AL	AG	JF	CA	CO	GR	HU	JA	MA	SE
AL	–	336	557	615	364	166	608	283	210	503
AG	336	–	123	155	289	312	336	447	144	232
JF	557	123	–	36	231	291	194	303	219	91
CA	615	155	36	–	268	347	248	360	268	145
CO	364	289	213	268	–	200	236	105	172	133
GR	166	312	291	347	200	–	373	134	164	269
HU	608	336	194	248	236	373	–	335	318	96
JA	283	447	303	360	105	134	335	–	236	234
MA	210	144	219	268	172	164	318	236	–	215
SE	503	232	91	145	133	269	96	234	215	–

donde AL, AG, JF, CA, CO, GR, HU, JA, MA, y SE denotan Almería, Algeciras, Jeréz de la Frontera, Cádiz, Córdoba, Granada, Huelva, Jaén, Málaga y Sevilla. Queremos obtener la solución óptima de este problema.

Cuestión 1. Implemente el algoritmo de Branch & Bound (Ramificación y Acotación) utilizando, para resolver los problemas relajados que necesite el método del SIMPLEX de Matlab

```
>> help linprog
 LINPROG     Linear programming.
    X=LINPROG(c,A,b) solves the linear programming problem:
              min c'*x    subject to:   A*x <= b
               x
```

Explique detalladamente su implementación.

Implemente tres versiones de su algoritmo, una eligiendo la rama de forma aleatoria, la otra con la regla de primero en profundidad y el último con la de primero en amplitud.

Cuestión 2. Como modelo de nuestro problema tomaremos un problema de asignación (subtipo de los de transporte). Sea $G = (X, A)$ un grafo completo dirigido, con costes en los arcos c_{ij}. Definiremos las variables de decisión $x_{ij} = 1$, si el arco (i, j) está en el camino óptimo, y $x_{ij} = 0$, en otro caso. Note que no están incluidas las variables x_{ii}.

El camino óptimo será solución del problema de programación entera

$$\begin{aligned} \min \quad & \sum_{i,j} c_{ij}\, x_{ij}, \\ \text{S.A.} \quad & \sum_{i} x_{ij} = 1, \quad \forall i, \\ & \sum_{j} x_{ij} = 1, \quad \forall j, \\ & 0 \le x_{ij} \le 1, \quad x_{ij} \in \mathbb{Z}, \quad i \neq j. \end{aligned}$$

Se puede demostrar que siempre existe un solución entera óptima de este problema.

Cuestión 2.1. Escriba este problema en Matlab para la red de ciudades importantes de Andalucía y aplique el método B&B para obtener su solución, tanto con elección aleatoria, como con primero en profundidad y primero en amplitud. ¿Son iguales las tres soluciones que ha obtenido? ¿Cuántas problemas de programación lineal ha tenido que resolver para cada uno de éstos métodos? ¿Cuánto tiempo ha tardado en obtener la solución para cada uno de estos métodos? ¿Cuál es el más rápido de los tres? ¿Cuál es la mejor solución de las tres?

NOTA: si su ordenador es muy lento, simplifique el problema considerando menos ciudades. Si la práctica se realiza entre varios alumnos, les recomiendo que cada uno implemente una de las 3 variantes (elección aleatoria, primero en profundidad y primero en amplitud).

Cuestión 2.2. El problema fundamental del problema como lo hemos planteado es que pueden darse ciclos. Suponga 5 ciudades, y que hemos obtenido como solución del problema de asignación $x_{12} = x_{24} = x_{45} = x_{53} = x_{31} = 1$. Se puede describir este itinerario como $1 - 2 - 4 - 5 - 3 - 1$. Un itinerario como éste se denomina circuito. Obviamente, si la solución del

problema de asignación produce un circuito, se trata de la solución óptima del TSP (¿por qué?). Sin embargo, el problema de asignación nos puede dar una solución como $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$, que se puede interpretar como dos subcircuitos $1-5-2-1$ y $3-4-3$. Un subcircuito es un itinerario que no pasa por todas las ciudades.

¿Alguna de las soluciones que ha obtenido es correcta? ¿Cuántos subcircuitos ha obtenido en su caso, y de qué longitud, para cada una de las soluciones obtenidas?

Vamos a ver cómo evitar la aparición de subcircuitos.

Cuestión 3. La primera posibilidad nos la da el libro de Winston, sección 9-6, pág. 520 en la edición en castellano. Detectar los subcircuitos y eliminarlos utilizando el algoritmo de B&B. Por ejemplo, si obtenemos como solución $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$, ramificamos a dos problemas, uno con $x_{34} = 0$ y otro con $x_{43} = 0$, evitamos la aparición de dicho ciclo (aunque podrían aparecer otros). Si en la primera rama obtenemos otro circuito ($x_{25} = 1 = x_{52}$), volvemos a ramificar con $x_{25} = 0$ en una rama y $x_{52} = 0$ en la otra. De esta forma acabaremos obteniendo una serie de circuitos "válidosτ entre ellos elegiremos el circuito óptimo.

Cuestión 3.1. Escriba un algoritmo que determine cuántos subcircuitos (si es que hay alguno) tiene una solución del problema TSP. Suponga que recibe un vector solución de la forma $[x_{12}, x_{13}, \ldots, x_{1n}, x_{21}, x_{23}, \ldots, x_{2n}, \ldots, x_{n1}, x_{n2}, \ldots, x_{nn-1}]$.

Cuestión 3.2. Utilizando el algoritmo anterior implemente la solución de evitación de circuitos que propone Winston en su libro utilizando los tres algoritmos B&B desarrollados previamente. ¿Son iguales las tres soluciones que ha obtenido? ¿Cuántas problemas de programación lineal ha tenido que resolver para cada uno de éstos métodos? ¿Cuánto tiempo ha tardado en obtener la solución para cada uno de estos métodos? ¿Cuál es el más rápido de los tres? ¿Cuál es la mejor solución de las tres?

Cuestión 4. Segunda posibilidad (Miller-Tucker-Zemlin, MTZ). Para excluir los subcircuitos introducimos las variables extra u_i, $i = 1, 2, \ldots, n$, y las restricciones

$$\begin{aligned} & u_1 = 1, \\ & 2 \leq u_i \leq n, \quad \forall i \neq 1, \\ & u_i - u_j + 1 \leq (n-1)\,(1 - x_{ij}), \quad \forall i \neq 1, \forall j \neq 1. \end{aligned}$$

La última de estas desigualdades se denomina restricción de arcos. ¿Cómo

excluye los subcircuitos? La restricción de arcos para el arco (i, j) fuerza a que $u_j \geq u_i + 1$, cuando $x_{ij} = 1$, y además, si la solución factible contiene más de un subcircuito, al menos uno no debe contener el nodo 1, con lo que a lo largo del mismo los valores de u_i tienen que crecer hasta infinito. Por ello, el único valor factible para los u_i es la posición del nodo i en el circuito. Una ventaja de esta formulación es que introduce sólo n variables extra y $n^2/2$ restricciones. Otra ventaja es que si es necesario o preferible visitar "pronto"la ciudad i en el circuito, podemos añadir en la función objetivo un término $-\alpha\, u_i$, para algún $\alpha > 0$ con objeto de modelar este problema.

Cuestión 4.1. Implemente el algoritmo MTZ y aplíquelo a nuestro problema, utilizando los tres algoritmos B&B desarrollados previamente. ¿Son iguales las tres soluciones que ha obtenido? ¿Cuántas problemas de programación lineal ha tenido que resolver para cada uno de éstos métodos? ¿Cuánto tiempo ha tardado en obtener la solución para cada uno de estos métodos? ¿Cuál es el más rápido de los tres? ¿Cuál es la mejor solución de las tres?

Cuestión 5. Tercera posibilidad. Para todos los posibles subcircuitos de longitud hasta $n/2$, imponemos la restricción

$$\sum_{i \in S, j \in S} x_{ij} \leq \sharp(S) - 1, \qquad \sharp(S) > 1,$$

donde S es el conjunto de todos los posibles subcircuitos. Los subcircuitos de longitud mayor que $n/2$ no hay que considerarlos ya que vienen acompañados de otros de longitud menor. Esta posibilidad no tiene las ventajas de la formulación MTZ, e introduce un número exponencial de restricciones, ya que hay un número exponencial de subcircuitos. Sin embargo, normalmente se implementa bajo demanda, es decir, obtenemos una solución en la que hay varios subcircuitos, entonces añadimos la restricción de más arriba para cada uno de ellos, solamente, y volvemos a resolver el problema. Si no hay más subcircuitos tenemos el circuito óptimo, si los hay, habrá que añadir más restricciones y continuar.

Cuestión 5.1. Implemente el algoritmo basado en subcircuitos que hemos presentado y aplíquelo a nuestro problema, utilizando los tres algoritmos B&B desarrollados previamente. ¿Son iguales las tres soluciones que ha obtenido? ¿Cuántas problemas de programación lineal ha tenido que resolver para cada uno de éstos métodos? ¿Cuánto tiempo ha tardado en obtener la solución para cada uno de estos métodos? ¿Cuál es el más rápido de los tres? ¿Cuál es la mejor solución de las tres?