# Algorítmica (11593)

16 de enero de 2014

Escuela Técnica Superior de Ingeniería Informática

**NOMBRE:** _____ **PROFESOR:** _____

**1** **2.5 puntos**

En el problema de la existencia de un ciclo Hamiltoniano, dado un grafo $G = (V, E)$, deseamos encontrar un camino que parta de un vértice, termine en el mismo vértice y visite todos los vértices del grafo una sola vez (excepto en el caso del vértice de partida, que deberá coincidir con el de llegada).

Se pide que implementes un algoritmo que, utilizando la técnica de Búsqueda con Retroceso, devuelva la solución encontrada, si es que existe. Utiliza una representación de grafo en forma de listas de adyacencia. Recuerda que, como el ciclo hamiltoniano puede empezar en cualquier vértice, es una práctica habitual empezar la comprobación por el primer vértice del grafo, $G.V[0]$. Para ello:

1. Explica brevemente la estrategia a seguir.
2. Escribe un algoritmo (en seudo-código o python) que implemente la estrategia anterior.

**Solución:**

```
# un grafo dirigido representado mediante listas de adyacencia seria
# simplemente una lista de listas del tipo:

G = [[1,2,3],       # del vertice 0 vamos a los vertices 1,2,3
     [0,3,4],       # del vertice 1
     [0,3,5],       # del vertice 2
     [0,1,2,4,5,6], # del vertice 3
     [1,3,7],       # del vertice 4
     [2,3,6,8],     # del vertice 5
     [3,5,7,8,9],   # del vertice 6
     [4,6,9],       # del vertice 7
     [5,6,9],       # del vertice 8
     [6,7,8]]       # del vertice 9

# para este tipo de grafo se puede definir el siguiente codigo de backtracking:

def hamiltonian_cycle(G):
  def backtracking(path):
    if len(path)==len(G):
      if path[0] in G[path[-1]]: return path+[0]
    else:
      for v in [x for x in G[path[-1]] if x not in path]:
        found = backtracking(path+[v])
        if found!=None: return found
    return None
  return backtracking([0])

# ejemplo de uso:
print hamiltonian_cycle(G)
```

**2** **4 puntos**

Una sociedad desea invertir un cierto capital $C$. Ha hecho una prospección del beneficio que puede alcanzar invirtiendo en varios países $P$ y varios sectores $S$ (energías renovables, turismo, eléctricas, telefonía, ...). Esa estimación la tienen tabulada y se puede consultar en $b(c, i, j)$, que devuelve el beneficio que se puede alcanzar invirtiendo una cierta cantidad $c$, $1 \leq c \leq C$, en el país $i$, para $1 \leq i \leq P$, en el sector $j$, para $1 \leq j \leq S$. Hay dos restricciones:

1. Sólo se puede invertir, como máximo, en un sector en cada país.
2. Cuando se invierte en un sector, hay que invertir una cantidad fija que depende del sector $j$ y del país $i$, $s[i, j]$.

Una instancia sería: un capital de $C = 400$ millones de euros, un conjunto de 3 países y 5 posibles sectores donde invertir. Las cantidades prefijadas a invertir en cada país y sector $s[i, j]$ son:

| | Eléctricas | Telefonía | Turismo | Energía undimotriz | Energía eólica |
|---|---|---|---|---|---|
| País 1 | 10 | 200 | 125 | 40 | 400 |
| País 2 | 20 | 100 | 125 | 20 | 100 |
| País 3 | 30 | 200 | 100 | 20 | 220 |

Diseña un algoritmo de Ramificación y Poda que proporcione la mejor forma de realizar la inversión. Para ello:

a) Formaliza el problema en términos de optimización:

   1) Identifica y expresa el conjunto de soluciones factibles $X$.
   2) Identifica y expresa la función objetivo a optimizar $f$.
   3) Identifica y expresa la solución óptima buscada $\hat{x}$.
   4) ¿Cuál debe ser la condición para que, dada una instancia del problema, exista al menos una solución factible? Pon un ejemplo de solución factible para la instancia presentada.

b) Describe los siguientes conceptos sobre los estados que serán necesarios para el algoritmo.

   1) Representación de un estado (no terminal) y su coste espacial. Pon un ejemplo para la instancia presentada.
   2) Condición para que un estado sea solución. Pon un ejemplo para la instancia presentada.
   3) Identifica el estado inicial que representa todo el conjunto de soluciones factibles.

c) Define una función de ramificación. Analiza su coste temporal. Pon un ejemplo para la instancia presentada.

d) Diseña una cota superior no trivial. Estudia cómo se puede realizar el cálculo de la cota superior de forma eficiente. ¿Cuál sería su coste temporal? Analízalo.

**Solución:** Una solución factible puede expresarse con una $P$-tupla en el que la componente $i$-ésima indica el sector en el que se invierte en el ese país $i$. Por ejemplo, para la instancia presentada, una posible solución es: invertir en 200 millones en Telefonía en el País 1, 125 en Turismo en el País 2 y 30 en Eléctricas en el País 3. Si numeramos de 1 a $S$ los sectores, la solución se puede representar con la tupla $(2, 3, 1)$. El espacio de soluciones factibles es

$$X = \{(x_1, x_2, \ldots, x_P) \in [0..S]^P \mid \sum_{1 \leq i \leq P} s[i, x_i] \leq C\},$$

donde $x_i = j$ indica que en el país $i$ se invierte en el sector $j$ una cantidad $s[i,j]$, para $1 \leq j \leq S$; si $x_i = 0$ no se invierte en el país $i$, y se define $s[i,0] = 0$. Deseamos maximizar el beneficio total de la inversión, por lo que nuestra función objetivo es

$$f((x_1, x_2, \ldots, x_P)) = \sum_{1 \leq i \leq P} b(s[i, x_i], i, x_i).$$

y buscamos:

$$(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_P) = \arg \max_{(x_1, x_2, \ldots, x_P) \in X} \sum_{1 \leq i \leq P} b(s[i, x_i], i, x_i).$$

Las soluciones son secuencias formadas por $P$ elementos. Los estados no unitarios serán, pues, secuencias con menos de $P$ elementos, es decir, tuplas de la forma $(x_1, x_2, \ldots, x_k, ?)$ con $k < P$. La función de ramificación generará, a partir de un estado $(x_1, x_2, \ldots, x_k, ?)$, el conjunto de estados de la forma $(x_1, x_2, \ldots, x_k, x_{k+1}, ?)$ para $x_{k+1}$ en $[0..S]$, siempre que la suma de la cantidad invertida hasta el momento no supere la cantidad total $C$:

$$branch((x_1, x_2, \ldots, x_k, ?)) = \{(x_1, x_2, \ldots, x_k, x_{k+1}, ?) \mid 0 \leq x_{k+1} \leq S \mid \sum_{1 \leq i \leq k+1} s[i, x_i] \leq C\}$$

Si mantenemos en cada estado la cantidad total invertida hasta el momento (o la cantidad que aún resta por invertir), la ramificación es un proceso realizable en tiempo $O(S)$, es decir, $O(1)$ por estado.

Se puede diseñar una cota optimista (superior) suponiendo que no hay límite en la cantidad a invertir. En tal caso, el máximo beneficio que se puede obtener vendría dado por el máximo beneficio de la inversión que se puede obtener en cada país. El beneficio de la inversión así obtenido es mayor o igual que el que obtendríamos si nos limitásemos a una cantidad finita de capital:

$$F((x_1, x_2, \ldots, x_k, ?)) = \sum_{1 \leq i \leq k} b(s[i, x_i], i, x_i) + \sum_{k+1 \leq i \leq P} maxbeneficio_i,$$

donde $maxbeneficio_i = \max_{0 \leq j \leq S} b(s[i,j], i, j)$. Estos valores se pueden precalcular con un coste $O(PS)$. La cota puede calcularse en tiempo constante por estado si se realiza incrementalmente:

$$F((x_1, x_2, \ldots, x_k, ?)) = F((x_1, x_2, \ldots, x_{k-1}, ?)) + b(s[k, x_k], k, x_k) - maxbeneficio_k.$$

## 3 — 3.5 puntos

Dado un tablero de ajedrez de tamaño $N \times N$ donde cada casilla tiene asignado un valor, haz una traza de un algoritmo Ramificación y Poda que sitúe $N$ torres en dicho tablero de modo que las torres no se amenacen entre sí y que la suma de las casillas ocupadas por las torres sea máxima. (Nota: las torres se mueven libremente en horizontal y vertical.) Haz la traza para este ejemplo con $N = 4$.

| | | | | |
|---|---|---|---|---|
| *4* | 5 | 2 | 1 | 2 |
| *3* | 4 | 6 | 3 | 2 |
| *2* | 7 | 3 | 1 | 7 |
| *1* | 3 | 2 | 4 | 6 |
| | *1* | *2* | *3* | *4* |

Ten en cuenta:

a) Sigue una estrategia por primero el mejor (en caso de empate, el estado más cercano a una solución) y con poda implícita.

b) Describe la cota optimista que vas a utilizar y analiza su coste temporal.

c) Lista 3 ejemplos de soluciones factibles (si es que existen), y su valor de función objetivo.

d) Si para la traza utilizas el esquema que inicializa la variable mejor solución $\hat{x}$ a una solución factible o a una cota pesimista, describe qué algoritmo o método utilizas para calcularla y cuál es su coste temporal.

e) En cada iteración de la traza, deberás indicar:

- El conjunto de estados activos (aunque en realidad esté representado con un max-heap, represéntalo como un conjunto desordenado, teniendo en cuenta que siempre estará accesible el estado con mayor puntuación).
- Marca el estado seleccionado para explorar.

**Solución:** La solución vendrá dada por una configuración válida de las torres en el tablero, que se puede representar como una tupla de 4 elementos $(x_1, x_2, x_3, x_4)$ que indica la posición de la torre de la columna $i$ en la fila $x_i$. La restricción es que no puede haber dos torres en la misma fila (en la misma columna está implícito por la representación elegida), por lo que, en realidad, las soluciones factibles serán el conjunto de permutaciones de 1 a $N$. Podemos hacer una inicialización temprana de la variable mejor solución con una solución cualquiera, por ejemplo, la permutación $(1, 2, 3, 4)$, que tiene asociado un valor de $3 + 3 + 3 + 2 = 11$. El coste del cálculo de esta solución y su valor de función objetivo es $O(N)$. Otros ejemplos de soluciones factibles para esta configuración serían: $(1, 3, 2, 4)$ con un valor $3 + 6 + 1 + 2 = 12$ y $(4, 3, 2, 1)$ con un valor $5 + 6 + 1 + 6 = 18$.

Una posible cota superior para este problema vendría dada por el máximo valor de las filas en cada columna y asumir que las torres restantes se colocan ahí, sin tener en cuenta si la fila que tiene esa puntuación ya está ocupada. Estos valores se pueden preprocesar con un coste $O(N^2)$ y tienen un valor para la instancia: $maxv = (7, 6, 4, 7)$. Si el cálculo de la cota se realiza de forma incremental, el coste será constante:

$$F((x_1, x_2, \ldots, x_k, ?)) = F((x_1, x_2, \ldots, x_{k-1}, ?)) + v(k, x_k) - maxv_k.$$

Inicializamos el conjunto de estados activos con el estado que representa todo el espacio de búsqueda (ninguna torre posicionada), con un valor de cota igual a $7 + 6 + 4 + 7 = 24$. En cada iteración seleccionaremos como siguiente a ramificar el más prometedor de acuerdo a su valor de cota. La variable mejor solución es $(1, 2, 3, 4)$ con un valor de 11. Aunque en el algoritmo de ramificación y poda el conjunto de estados activos se representaría con un maxheap, en la traza lo representaremos un conjunto sin ordenar.

$$A^0 = \{\overbrace{(?)}^{24}\}.$$

Al seleccionar el estado (?) se obtienen 4 nuevos estados:

$$A^1 = \{\overbrace{(1,?)}^{20}, \overbrace{(2,?)}^{24}, \overbrace{(3,?)}^{21}, \overbrace{(4,?)}^{22}\}.$$

Seleccionamos para ramificar el estado $(2, ?)$:

$$A^2 = \{\overbrace{(2,1,?)}^{20}, \overbrace{(2,3,?)}^{24}, \overbrace{(2,4,?)}^{20}, \overbrace{(1,?)}^{20}, \overbrace{(3,?)}^{21}, \overbrace{(4,?)}^{22}\}.$$

Seleccionamos para ramificar el estado $(2, 3, ?)$:

$$A^3 = \{\overbrace{(2,3,1,?)}^{24}, \overbrace{(2,3,4,?)}^{21}, \overbrace{(2,1,?)}^{20}, \overbrace{(2,4,?)}^{20}, \overbrace{(1,?)}^{20}, \overbrace{(3,?)}^{21}, \overbrace{(4,?)}^{22}\}.$$

Al ramificar el estado $(2, 3, 1, ?)$ se obtiene la solución $(2, 3, 1, 4)$ que mejora $\hat{x}$ con un valor 19. Seguimos iterando y seleccionamos $(4, ?)$ para ramificar:

$$A^4 = \{\overbrace{(4, 3, ?)}^{22}, \overbrace{(2, 3, 4, ?)}^{21}, \overbrace{(2, 1, ?)}^{20}, \overbrace{(2, 4, ?)}^{20}, \overbrace{(1, ?)}^{20}, \overbrace{(3, ?)}^{21}\}.$$

Los estados $(4, 1, ?)$[18] y $(4, 2, ?)$[19] no se guardan. En esta iteración, el estado ramificado es el $(4, 3, ?)$:

$$A^5 = \{\overbrace{(4, 3, 1, ?)}^{22}, \overbrace{(4, 3, 2, ?)}^{20}, \overbrace{(2, 3, 4, ?)}^{21}, \overbrace{(2, 1, ?)}^{21}, \overbrace{(2, 4, ?)}^{20}, \overbrace{(1, ?)}^{20}, \overbrace{(3, ?)}^{21}\}.$$

Al ramificar $(4, 3, 1, ?)$ se obtiene una nueva y mejor solución de coste 22: $\hat{x} = (4, 3, 1, 2)$. En la siguiente iteración, el conjunto de estados activos es:

$$A^6 = \{\overbrace{(4, 3, 2, ?)}^{20}, \overbrace{(2, 3, 4, ?)}^{21}, \overbrace{(2, 1, ?)}^{20}, \overbrace{(2, 4, ?)}^{20}, \overbrace{(1, ?)}^{20}, \overbrace{(3, ?)}^{21}\}.$$

El mejor estado es $(2, 3, 4, ?)$ con una cota de 21, peor que la mejor solución solución obtenida hasta el momento. Esto significa que ninguno de los estados activos puede mejorar $\hat{x}$, con lo que se vacía $A$ y el algoritmo termina devolviendo la solución $\hat{x} = (4, 3, 1, 2)$.