UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)

9 de enero de 2017

etsinf
Escuela Técnica Superior de Ingeniería Informática

NOMBRE:

1 — 2 puntos

El problema de la búsqueda de un ciclo hamiltoniano se puede resolver mediante la técnica de Vuelta Atrás. Como el ciclo hamiltoniano puede empezar en cualquier vértice, es una práctica habitual empezar la comprobación por el primer vértice del grafo (asumimos que es el 0). Instancia los métodos siguientes para resolver el problema, utilizando una representación de grafo en forma de listas de adyacencia:

```
self.is_feasible(s)
self.branch(s)
self.is_promising(s)
```

Te proporcionamos el código del esquema general de Vuelta Atrás:

```
def backtracking(self, s):
  """Explora el subarbol que tiene por raiz a s y devuelve la primera
  solucion factible que encuentra (si la hay). Si s no contiene al
  menos una solucion factible, devuelve None"""
  if self.is_complete(s):
    if self.is_feasible(s):
      return s
  else:
    for child in self.branch(s):
      if self.is_promising(child):
        found = self.backtracking(child)
        if found != None:
          return found
    return None

def solve(self, s):
  return self.backtracking([0])

def is_complete(self, s):
  return len(s)==len(self.G)+1
```

y un ejemplo de grafo dirigido representado mediante listas de adyacencia:

```
self.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
```

2 **1 punto**

Responde brevemente a las siguientes cuestiones:

1) ¿Cuáles son las **diferencias** fundamentales entre los esquemas de Vuelta Atrás y Ramificación y Poda?
2) ¿Puedes utilizar el esquema de Vuelta Atrás para resolver un **problema de optimización**?
3) ¿Puedes utilizar el esquema de Ramificación y Poda para resolver un **problema de no optimización**?

3 **3 puntos**

En una fábrica, hay N productos que pueden ser fabricados por M máquinas diferentes, si bien tardan un tiempo distinto en fabricarlo, y cada máquina puede fabricar p_i productos como máximo. Una asignación de N productos a las M máquinas se puede representar mediante un vector x de Ncomponentes, $(x_1, x_2, \ldots, x_N)$ donde $x_i = j$ indica que el producto i-ésimo se ha asignado a la máquina j, $1 \leq j \leq M$. Nos interesa minimizar el coste total en tiempo. Por ejemplo, la siguiente asignación de 5 productos a 3 máquinas (2, 2, 1, 3, 1) con la siguiente tabla de tiempos $t_{i,j}$ que indica el tiempo que la máquina i tarda en finalizar el trabajo j, y asumiendo que cada máquina no puede fabricar más de 2 productos:

	T1	T2	T3	T4	T5
M1	4	4	3	5	6
M2	2	3	4	3	4
M3	5	2	4	4	3

genera un coste de $2 + 3 + 3 + 4 + 6 = 18$. Realiza una traza de un algoritmo de Ramificación y Poda sobre la instancia presentada basada en *poda explícita* que calcule la asignación de productos a máquinas que minimice el tiempo total. Explica la cota que utilizas. La traza se debe mostrar como el *conjunto de estados activos* de cada iteración del algoritmo indicando la cota optimista de cada estado (ej: como superíndice) y subrayando el estado que se selecciona para la siguiente iteración. Hay que indicar también si se actualiza la variable mejor solución y la poda explícita. En caso de inicializar la mejor solución inicial a una solución arbitraria, debes indicar el algoritmo que usas y su coste.

4 **4 puntos**

Tenemos un talonario de millas aéreas acumuladas que va a caducar próximamente y queremos planificar un viaje para utilizarlo en su totalidad. El talonario nos permite realizar N vuelos, independientemente de la distancia de este vuelo. Las restricciones que nos impone la compañía son:

- sólo podemos utilizar vuelos directos (nos han proporcionado el grafo de conexiones de la compañía, $conexiones[i, j] = conexiones[j, i] = m$ indica que las ciudades i y j están conectadas por vuelo directo a una distancia de m millas; si $conexiones[i, j] = -1$, indica que las ciudades no están conectadas),
- debemos partir de nuestra ciudad origen, Valencia, y volver de nuevo a Valencia, y
- no podemos volar a una ciudad más de una vez (excepto a Valencia, de la que debemos partir y llegar al final).

Las ciudades a las que vuela la compañía están numeradas de 0 a C (la ciudad 0 es Valencia). Nuestro objetivo es, con las restricciones impuestas por la compañía, volar el máximo de millas. Diseña un algoritmo de Ramificación y Poda para maximizar las millas. Para ello:

a) Expresa formalmente el conjunto de soluciones factibles, la función objetivo a maximizar y la solución óptima buscada.
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.
   2) Condición para que un estado sea solución.
   3) Identifica el estado inicial que representa todo el conjunto de soluciones factibles.
c) Define una función de ramificación. Analiza su coste temporal.
d) Diseña una cota optimista no trivial. Estudia cómo se puede realizar el cálculo de la cota de forma eficiente. ¿Cuál sería su coste temporal? Justifícalo.