Práctica 5 de Algorítmica
(Ramificación y poda)

Ordenamiento de los vértices de un grafo

Curso 2017-2018

Ordenamiento de los vértices de un grafo

En esta práctica de **tres sesiones** vamos a resolver, de manera exacta el mismo problema que vimos en la práctica anterior.

El objetivo es, dado un grafo dirigido $G = (V, E)$ con $N = |V|$ vértices, ponderado y posiblemente con ciclos, encontrar un ordenamiento de los vértices del grafo que maximice la suma de las aristas que van hacia la derecha menos la suma de las aristas que van hacia la izquierda.

Para ello vamos a utilizar la estrategia *branch and bound* o *ramificación y poda*.

Ejercicio 1 Formaliza este problema de optimización. Para ello se pide:

- El espacio de búsqueda X.
- La función objetivo $f(x)$.
- El criterio de optimización.

Ordenamiento de los vértices de un grafo

Ejercicio 2 Implementa un algoritmo de *ramificación y poda* que resuelva este problema. Para ello te proporcionamos algunos detalles así como un código inicial para completar.

Concretamente, una vez formalizado el problema, necesitamos poder determinar:

- Cómo se representará una solución parcial.
- Función de ramificación.
- Cómo determinar si una solución parcial es terminal o completa.
- Calcular una **cota optimista**. A ser posible, que coincida con la función objetivo cuando la solución sea terminal o completa.
- El esquema básico de ramificación y poda a utilizar.
- Una primera solución inicial.

Veamos cada uno de estos puntos:

- En general, la solución parcial se representará mediante una lista python cuyo contenido e interpretación dependerá de tu formalización, aunque normalmente será una lista de los k primeros vértices de la ordenación.
- La solución será terminal cuando contenga todos los vértices.

Ordenamiento de los vértices de un grafo

- La cota optimista es, en general, la suma de la parte conocida más una estimación optimista de la parte que queda por completar:
  La parte conocida contiene la contribución de 2 cosas:
  - las aristas entre los vértices ya fijados en la solución parcial
  - las aristas entre vértices de la solución parcial y los vértices que todavía no hemos situado.

  La parte desconocida, de la que necesitamos una estimación optimista, corresponde a:
  - las aristas entre los vértices que todavía no hemos situado.

  Para esta última parte te proponemos simplemente sumar todas las aristas y no restar ninguna. En este punto es importante haber aplicado previamente el ejercicio 2 de la práctica anterior.
- El esquema básico de ramificación y poda será el de *poda implícita* utilizando una lista de estados activos ordenada por la propia cota optimista. Para ello se te proporciona un esquema a completar.
- Para la inicialización de la primera solución se propone utilizar un algoritmo voraz de la práctica anterior.

Ordenamiento de los vértices de un grafo

- El conjunto de estados activos se representará mediante una cola de prioridad basada en un *heap*. Los elementos a guardar serán tuplas donde el primer elemento será el *score* obtenido mediante la cota optimista y el segundo la lista python que representa una solución parcial.

Para representar la lista de estados activos podemos utilizar un *heap* o montículo. Hay uno disponible en la biblioteca estándar de python 3 que puedes consultar en el siguiente enlace. Cuando usamos esta biblioteca los heaps se representan simplemente mediante una lista python como en este ejemplo:

```
import heapq # ¡¡ojo!! es un minheap
A = [] # empty priority queue
heapq.heappush(A,5)  # insertar elemento
heapq.heappush(A,10) # insertar elemento
print(A[0]) # consultar el menor sin extraer
menor = heapq.heappop(A) # consulta extrayendo
print(len(A)) # tamaño de la cola de prioridad
```

El problema de esta estructura es que representa un *minheap* adecuado para extraer el **mínimo** aunque realmente a nosotros nos interesa extraer el **máximo**. Lo que haremos es guardar los scores **negados**.

Ordenamiento de los vértices de un grafo

Un posible esquema del algoritmo de ramificación y poda es como sigue:

```
x,fx = inicializar_solucion_y_su_score() # alg. voraz
A = [] # empty priority queue
s = [] # estado inicial:
opt = optimistic(s)
heapq.heappush(A,(-opt,s)) # lo añadimos, observa -opt
while len(A)>0 and -A[0][0] > fx: # PODA IMPLÍCITA
  score_s,s = heapq.heappop(A)
  score_s = -score_s # ahora ya no está negado
  for child in branch(s):
    if is_complete(child): # si es terminal
      cota_child = optimistic(child) # calcula evaluate
      if cota_child > fx: # si mejoramos lo que hay
        x, fx = child, cota_child
    else: # no es terminal
      opt_child = optimistic(child)
      if opt_child > fx: # poda por cota optimista
        heapq.heappush(A,(-opt_child,child)) # observa -opt_child
return x,fx
```

Ordenamiento de los vértices de un grafo

Ejercicio 3 Se pide calcular la cota optimista de manera **incremental**. Es decir, como una diferencia respecto a la cota optimista del estado padre. Para ello observa que cuando pasamos un vértice desde la parte desconocida hasta el final de la parte conocida hay que tener en cuenta que las aristas desde la parte desconocida hasta él antes sumaban y ahora restan, por lo que hay que restarlos 2 veces para actualizar la cota.