.UBAfiuba
FACULTAD DE INGENIERÍA

Branch and Bound: Problema del viajante de comercio

Teoría de Algoritmos I (75.29 / 95.06)

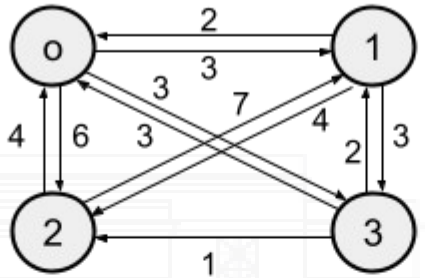
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

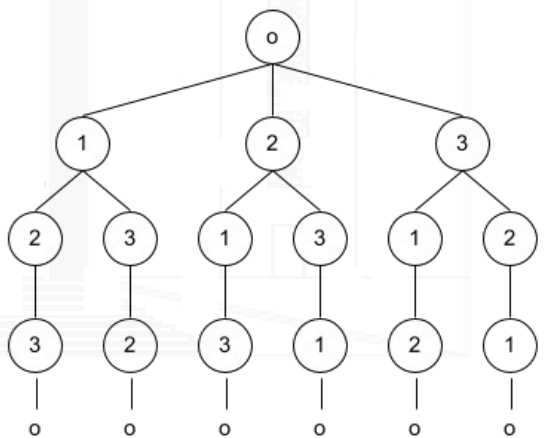
Problema del viajante de comercio

- **Contamos con un conjunto de “n” ciudades a visitar.**
  - Existen caminos que unen pares de ciudades.
- **Cada camino**
  - inicia en una ciudad “x” y finaliza en la ciudad “y”
  - tiene asociado un costo de tránsito de wx,y.
- **Partiendo desde una ciudad inicial y finalizando en la misma se quiere**
  - construir un circuito que visite cada ciudad una y solo una vez minimizando el costo total.

Árbol de estados del problema

- **La raíz representa el comienzo del viaje partiendo de la ciudad inicial**
- **Cada nodo representa la inclusión de una ciudad en el recorrido que no fue previamente visitada**
  - Tiene como posibles estados descendientes las posibles elecciones de próximas ciudades (restringidas por las previamente visitadas).
  - Al elegir una opción se debe sumar el costo del traslado
- **El camino desde la raíz hasta el nodo representa el recorrido realizado desde la ciudad inicial hasta el momento**
  - El costo del recorrido es la suma de los costos de los trayectos entre las ciudades realizadas
- **El árbol para n ciudades tiene un total de** $1+\sum_{i=1}^{n}\prod_{j=1}^{i}(n+1)-i$ **estados del problema**





Costo del Ciclo

- Dado una posible solución correspondiente a un ciclo C=[o,x_1,x_2,x_{n-1},o], podemos calcular su costo como:

$$Costo(C) = w_{o,x_1} + \left(\sum_{j=1}^{n-1} w_{x_j,x_{j+1}}\right) + w_{x_{n-1},o}$$

- Podemos separar el circuito en dos partes C_1= [o,x_1,…,x_i] y C_2 =[x_{i+1},…,x_{n-1},o],

$$Costo(C) = Costo(C_1) + Costo(C_2)$$

$$Costo(C) = \left[w_{o,x_1} + \left(\sum_{j=1}^{i-1} w_{x_j,x_{j+1}}\right)\right] + \left[\left(\sum_{j=i}^{n-1} w_{x_j,x_{j+1}}\right) + w_{x_{n-1},o}\right]$$

Existen varios Ciclos que comparte el orden de visitas de las primeras i ciudades

Para todas ellas el costo C_1 es el mismo

Función costo “l”

Dado un nodo a profundidad “i” del árbol de estados

Sabemos las ciudades visitadas y podemos calcular el costo acumulado C_1

Llamamos X al conjunto de todas las ciudades a recorrer

podemos calcular el subconjunto Y ⊆ X como aquellas ciudades aún no visitadas al llegar a ese nodo.

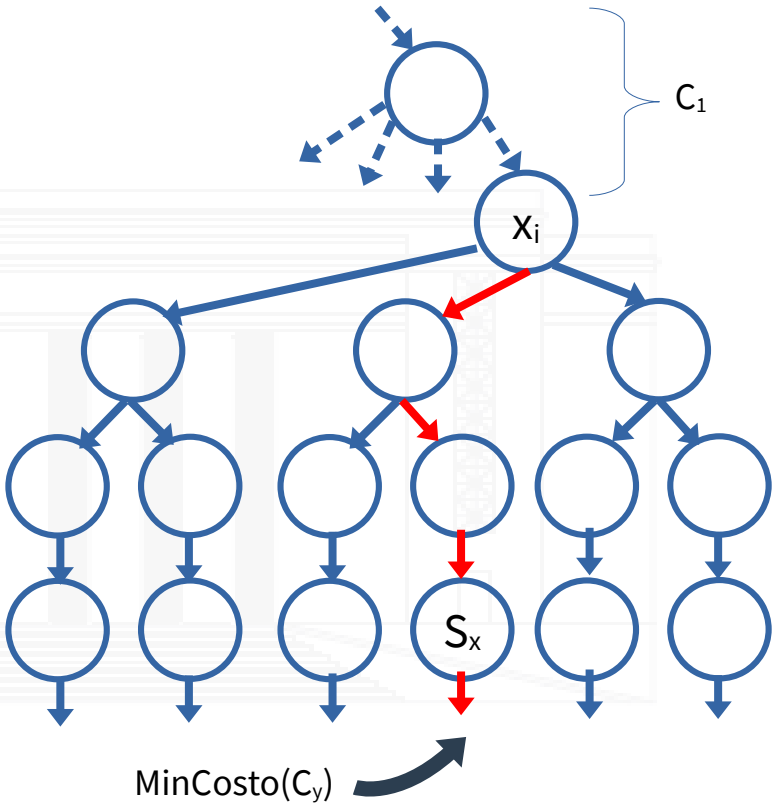
Existirán posiblemente varias alternativas para generar el camino C_2.

En el caso extremo todas las permutaciones posibles de las ciudades en Y, cada una de estas con un costo diferente.

Nos interesa poder medir el menor de los costos posibles al que llamaremos MinCosto(Y).

Si MejorActual ≤ l = Costo(C_1) + MinCosto(Y), sabemos que no hay un circuito mejor al existente que comience con el camino C_1.

Buscaremos estimar MinCosto(Y)



Estimación de MinCosto(Y)

Supongamos que todas las ciudades en Y están comunicadas entre sí y además comunicadas con la última ciudad x_i de C_1

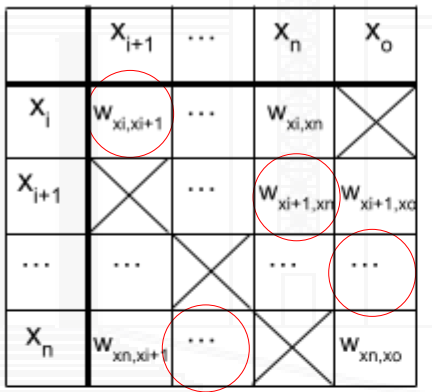
Podemos expresar los valores en una grilla donde la columna representa la ciudad de origen y la fila la de destino

Un camino C_2 representado en la grilla corresponde a

una celda por fila y columna

Las permutaciones posibles son los posibles caminos

La suma de los valores de las celdas seleccionadas corresponde al costo del camino.



No llegar hasta el ultimo paso

No se puede ir de una ciudad a si misma

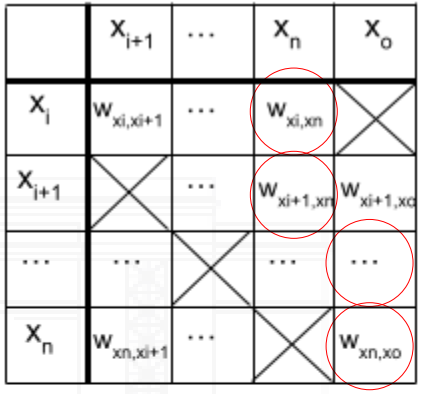
Estimación de MinCosto(Y) - Continuación

Llamaremos MinCosto(C_y) a la estimacion de MinCosto(Y)

Se propone tomar por cada fila de la matriz el menor peso disponible

Como precaución no seleccionar para xi la ciudad de origen
(a menos que sea la única posibilidad).

$$min\left\{w_{x_i,z} / z \in Y\right\} + \sum_{y \in Y} min\left\{w_{y,z} / z \in Y \cup \{o\}\right\}$$



Las celdas seleccionadas podrían no corresponder a un camino válido

(Puede incluir 2 o más celdas de la misma columna).

Pero me aseguro que la sumatoria de los pesos van a ser igual o menor al costo del menor camino mínimo válido posible en C_y : MinCosto(C_y) ≤MinCosto(Y)

Función costo “l” estimada

Con la estimación de MinCosto(Y)

La podemos reemplazar en la función costo “l”

Nos servirá para determinar qué nodo podar.

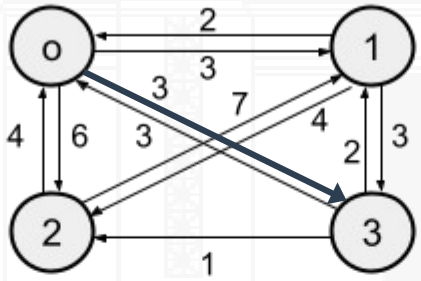
Si MejorActual ≤ Costo(C1) + MinCosto(Y), entonces MejorActual ≤ Costo(C1) + MinCosto(C_y) y por lo tanto no hay forma de – explorando esa rama del árbol – encontrar una mejor solución a la actual.

Nos servirá para determinar qué nodo explorar primero

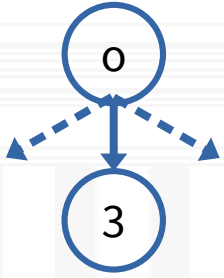
Si comparamos entre dos nodos, la que tiene mejor “l” posiblemente tenga un ciclo mejor entre sus descendientes.

Ejemplo

Supongamos la siguiente instancia del problema:



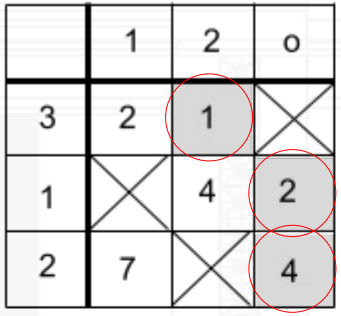
Debemos calcular la funcion costo “l” del nodo [o,3]:



C_1=[o,3]

Y={1,2}

Costo(C_1)=3



Costo(Y)=7

l=3+7 = 10

Recorrido del árbol de estados

Para recorrer el árbol utilizaremos Best-First.

Se incluye en la cola de prioridad la raíz del árbol con su valor de función de costo “l”

Se toma el camino parcial desde la ciudad “o” en la cola de mayor valor de costo que supere a la mejor solución encontrada

Determinamos si corresponde a un camino de longitud n en ese caso un podemos cerrar el ciclo

Sino, obtenemos las ciudades que aun no se incluyen y expandimos cada posible camino parcial.

Por cada uno de los caminos parciales verificamos si supera la función limite y se agregan en la cola de prioridad

Branch & Bound – Pseudocódigo

```
Sea C[x,y] el costo de ir de la ciudad x a la y
Sea A[y] la lista de ciudades adyacentes de la ciudad y.
Sea camino el recorrido realizada hasta el momento
Sea minimoCosto el costo del camino minimo encontrado
Sea minimoCamino el circuito de menor costo encontrado

camino={o}.
minimoCosto=infinito
minimoCamino={}

Definir fc=0 para costo de camino
Agregar camino con fc a caminosDisponibles
```

Branch & Bound – Pseudocódigo

```
Mientras queden caminos estos en caminosDisponibles
     Sea caminoActual el camino de mayor fc en caminosDisponibles
     Sea x la ultima ciudad visitada en caminoActual.
     Si longitud del camino es n
          Si la ciudad o se encuentra en A[x]
               Agregar al final de caminoActual la ciudad o
               Sea costoCamino la suma de los costos de caminoActual
               Si costoCamino < minimoCosto
                    minimoCosto = costoCamino
                    minimoCamino=camino
     Sino
          Por cada ciudad y en A[x] no visitada previamente excepto “o”
               Sea caminoAmpliado = caminoActual+{y}
               Determinar Y ciudades aun no visitadas en caminoAmpliado
               Sea costoCamino la suma de los costos de caminoAmpliado
               Calcular fc=costoCamino +MinCosto(x,Y)
               Si fc < minimoCosto
                    Agregar caminoAmpliado  con fc a caminosDisponibles
```

Complejidad Temporal

Debemos explorar

en el peor de los casos explorar todos los estados del árbol de estados en O(n!).

Calcular para cada nodo

su MinCosto(Y) con complejidad $O(n^2)$

Calcular en O(n) todos sus descendientes posibles

Realizar operaciones O(1)

Agregar y eliminar un nodo a la cola de prioridad

log(A) conde A es la cantidad de nodos agregados. A=n!

Unificando y simplificando tendremos O(n! log n!))

Complejidad Espacial

Cada nodo

requiere O(n) para generar el camino parcial

O(1) por otras operaciones

En de la cola de prioridad

Se almacenan los nodos con el parcial que requiere O(n) y su valor de prioridad “l”

En el peor de los casos

tendremos O(n!) nodos en la cola de prioridad

Tendremos un consumo elevado de memoria de

O(n*n!)