Ada Approximation Algorithms

Antonio Toche

14 de diciembre de 2021

1. Problema

Dados n puntos $X_i = (x_i, y_i)$ con una métrica de distancia $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, se nos pide hallar un circuito que pase por cada uno de los n puntos una sola vez y que la longitud del mismo sea la mínima posible.
En términos formales, deseamos una permutación p tal que:

$$\sum_{i=0}^{n-1} d(X_{p_i}, X_{p_{(i+1) \mod n}})$$

Sea mínima.

2. Algoritmo

El algoritmo de aproximación propuesto se basa en hallar un MST (Minimum Spanning Tree) del grafo, luego generar un Euler Tour del árbol y tomar la permutación generada por el mismo como circuito.

Algoritmo 1: TSP - MST + Euler tour

Datos: n puntos $X_i = (x_i, y_i)$
Resultado: Aproximación del circuito de menor longitud que pase por todos los vértices una sola vez
Sea $G = ([1, \ldots, n], \{(i, j) \in [1, n]^2\}), w(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$;
$T \leftarrow MST(G)$;
$p \leftarrow Euler\_Tour(T, 1)$;
Eliminar todas las segundas ocurrencias de cada nodo en p. **Devolver** $costo(p)$;

La complejidad del algoritmo depende principalmente de la complejidad de hallar su MST, pues hallar el Euler Tour de un árbol tiene complejidad lineal (es la ejecución de un DFS), así como el calcular el costo del ciclo.
Para hallar el MST proponemos el algoritmo de Prim, el cual tiene complejidad $O(n^2)$.

Algoritmo 2: Prim

Datos: n puntos $X_i = (x_i, y_i)$
Resultado: MST del grafo generado por los puntos
$vis \leftarrow [0, \ldots, 0]$ de tamaño n ;
$d \leftarrow [\infty, \ldots, \infty]$ de tamaño n ;
$par \leftarrow [-1, \ldots, -1]$ de tamaño n ;
$vis[1] \leftarrow 1$;
$d[1] \leftarrow 0$;
para $step = 1 \rightarrow n - 1$ **hacer**
  $best \leftarrow -1$;
  para $i = 1 \rightarrow n$ **hacer**
    si $vis[i] = 0$ **entonces**
      si $best = -1$ *o* $d[best] > d[i]$ **entonces**
        $best \leftarrow i$;
  $vis[best] \leftarrow 1$;
  para $i = 1 \rightarrow n$ **hacer**
    si $vis[i] = 0$ *y* $\delta(i, best) < d[i]$ **entonces**
      $d[i] \leftarrow \delta(i, best)$;
      $par[i] =\leftarrow best$;
$G \leftarrow (\{1, \ldots, n\}, \varnothing)$;
para $i = 2 \rightarrow n$ **hacer**
  Agregar la arista $(i, par[i])$ al grafo G ;
Devolver G ;

Algoritmo 3: Euler_Tour

Datos: Árbol T, nodo inicial u, nodo padre p, arreglo *orden* (global)
Resultado: Euler tour del árbol T
Agregar u al final de *orden* ;
para $(u, v) \in T$ **hacer**
  si $v \neq p$ **entonces**
    $Euler\_Tour(T, v, u)$;
Agregar u al final de *orden* ;

3. Correctitud

La correctitud del algoritmo se basa en el hecho de que todo ciclo que pase por los n nodos puede ser representado por una permutación de los mismos, de manera que el ciclo representado por la permutación p es:

$$c = \{(p_i, p_{i \mod n+1}) \forall i = 1, \ldots, n\}$$

3.1. Coeficiente de aproximación

Como toda solución aproximada, se debe hallar el coeficiente de aproximación. Probaremos que la solución hallada por el algoritmo propuesto es una 2-aproximación para el problema de TSP; es decir:

$$Sol\_Hallada \leqslant 2 \cdot Sol\_Optima$$

Prueba:

Consideremos que el circuito óptimo es H, entonces podemos quitarle una arista e y se volverá un árbol H^*. Por definición del MST T, se tiene que:

$$costo(T) \leqslant costo(H^*)$$

Además, consideremos el Euler Tour W de T: este visita cada arista 2 veces, por lo cual su costo será:

$$costo(W) = 2 \cdot costo(T) \leqslant 2 \cdot costo(H^*) \leqslant 2 \cdot costo(H)$$

Donde la última relación se obtiene del hecho de que $\delta(i, j) \geqslant 0$ por definición de distancia.
Ahora, notemos que si tomamos tres posiciones consecutivas (cíclicamente) (i, j, k) del Euler Tour, entonces por desigualdad triangular se cumple que:

$$\delta(W_i, W_j) + \delta(W_j, W_k) \geqslant \delta(W_i, W_k)$$

Por lo tanto, al usar la propiedad anterior de manera repetida cada vez que eliminamos una segunda ocurrencia de alguno de los nodos de W, el costo del nuevo camino será menor o igual al de W. Sea W^* la permutación resultante, entonces:

$$costo(W^*) \leqslant costo(W) = 2 \cdot costo(T) \leqslant 2 \cdot costo(H^*) \leqslant 2 \cdot costo(H)$$

De lo anterior, concluimos que:

$$costo(W^*) \leqslant 2 \cdot costo(H)$$

Lo que queríamos demostrar.