Minimum Spanning Trees (MST) y Single-Source Shortest Paths (SSSP)

Camilo Rocha & Miguel Romero

19 de septiembre de 2019

Pontificia Universidad Javeriana de Cali
Análisis y Diseño de Algoritmos

Minimum Spanning Trees (MST)

Problema

Suponga que tiene un conjunto de n ciudades que desea interconectar por medio de carreteras. Para interconectar las n ciudades, se pueden utilizar $n-1$ carreteras, cada una conectando dos ciudades. De todas las formas posibles de interconectar las ciudades, solo nos interesa aquella que utilice las carreteras mas cortas.

Problema

Entrada
Un grafo no dirigido $G = (V, E)$ con pesos, cuya función de pesos $w : E \to \mathbb{R}$ mapea arcos en valores reales.

Salida
Un subconjunto acíclico $T \in E$ que conecta todos los vértices V y cuyo peso total

$$w(T) = \sum_{(u,v) \in T} w(u, v),$$

es mínimo.

Problema

Puesto que T es acíclico y conecta todos los vértices, entonces debe formar un árbol, el cual se denomina *árbol de cubrimiento* puesto que cubre el grafo G.

Problema

Dado un grafo conexo no dirigido $G = (V, E)$ con función de pesos $w : E \to \mathbb{R}$, se quiere encontrar el árbol mínimo de cubrimiento para G.

La estrategia voraz consiste en expandir el árbol mínimo de cubrimiento un arco a la vez. Dado un conjunto de arcos A se mantiene la siguiente invariante: Antes de cada iteración, A es un subconjunto de algún árbol mínimo de cubrimiento.

Problema

En cada paso se selecciona un arco (u, v) que se puede agregar a A sin violar el invariante, de tal forma que $A \cup \{(u, v)\}$ es también un árbol mínimo de cubrimiento. Dicho arco se denomina **arco seguro** para A, puesto que se puede agregar sin riesgo de violar el invariante.

Problema

Teorema 1
Sea $G = (V, E)$ un grafo conexo y no dirigido con función de pesos $w : E \to \mathbb{R}$. Sea A un subconjunto de E que está incluido en algún árbol mínimo de cubrimiento para G, sea $(S, V - S)$ un corte de G que respeta A, y sea (u, v) un arco ligero que cruza $(S, V - S)$. Entonces(u, v) es seguro para A.

Problema

Demostración.
Sea T árbol mínimo de cubrimiento que incluye A, y asuma que T no contiene el arco ligero (u, v), puesto que si lo contiene la demostración esta completa. Debemos construir un árbol mínimo de cubrimiento T' que incluya $A \cup \{(u, v)\}$, y por lo tanto demostrar que (u, v) es seguro para A. $\square$

Problema

Demostración.
El arco (u, v) forma un ciclo con los arcos en el camino simple p de u a v en T. Puesto que u y v están en lados opuestos del corte $(S, V - S)$, por lo menos un arco en T esta en el camino simple p cruza el corte. Sea (x, y) dicho arco.

El arco $(x, y) \notin A$, porque el corte respeta A. Puesto que (x, y) esta en el único camino simple de u a v en T, eliminar (x, y) rompe T en dos componentes. Agregar (u, v) vuelve a conectar estos componentes para formar un nuevo árbol de cubrimiento mínimo $T' = T - \{(x, y)\} \cup \{(u, v)\}$. $\square$

Demostración.
Probamos ahora que T' es un árbol mínimo de cubrimiento.

Puesto que (u, v) es un arco ligero que cruza $(S, V - S)$ y (x, y) también cruza el corte, $w(u, v) \leq w(x, y)$. Por lo tanto,

$$w(T)' = w(T) - w(x, y) + w(u, v)$$

$$w(T)' \leq w(T).$$

□

Demostración.
Pero T es un árbol mínimo de cubrimiento, entonces $w(T)' \leq w(T)$, por lo tanto, T' debe ser un árbol mínimo de cubrimiento también.

Resta probar que (u, v) es realmente un arco seguro para A. Se tiene que $A \subseteq T'$, puesto que $A \subseteq T$ y $(x, y) \notin A$; entonces, $A \cup \{(u, v)\} \subseteq T'$. Por consiguiente, dado que T' es un árbol mínimo de cubrimiento, (u, v) es seguro para A. $\square$

Single-Source Shortest Paths (SSSP)

Problema

Suponga que quiere encontrar la ruta con la menor distancia entre dos puntos de una ciudad y dispone de una mapa de la ciudad en el cual la distancia entre dos intersecciones está dada, ¿Como puede determinar la ruta de menor distancia?

Problema

Una opción es enumerar todos las rutas entre los dos puntos de la ciudad, sumar las distancias de cada ruta y seleccionar la de menor distancia. Es fácil de darse cuenta que se tendrían que examinar muchas posibilidades, muchas de las cuales no vale la pena considerar.

Problema

Entrada
Un grafo dirigido $G = (V, E)$ con pesos, cuya función de pesos $w : E \to \mathbb{R}$ mapea arcos en valores reales, y un vértice de origen s.

Salida
La distancia mínima entre el vértice de origen s y todos los demás vértices $V(G) \backslash \{s\}$.

Problema

El **peso** $w(p)$ de un camino $p = \langle v_0, v_1, \ldots, v_k \rangle$ es la suma de los pesos de los arcos que lo componen:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Problema

El **peso del camino de distancia mínima** $\delta(u, v)$ de u a v se define como:

$$\delta(u,v) = \begin{cases} \text{mín}\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{si existe un camino de } u \text{ a } v, \\ \infty & \text{de lo contrario.} \end{cases}$$

Un **camino de distancia mínima** de un vértice u a un vértice v se define como cualquier camino p con peso $w(p) = \delta(u, v)$.

Variaciones del problema

Note que los arcos de los pesos pueden representar métricas diferentes a la distancia, tales como, costo, penalizaciones, perdidas o cualquier otra cantidad que se acumule linealmente a lo largo del un camino y que se desea minimizar.

Variaciones del problema

- **Single-destination shortest-paths problem:** Encontrar la distancia mínima a un vértice destino t desde todos los vértices v.
- **Single-pair shortest-path problem:** Encontrar la distancia mínima de u a v para dos vértices u y v dados.
- **All-pairs shortest-paths problem:** Encontrar la distancia mínima de u a v para todos los pares de vértices del grafo.

Solución

Grafo sin pesos

El algoritmo de búsqueda en amplitud (BFS) visto en clase es el algoritmo de distancia mínima para grafos sin peso, es decir, grafos en los cuales cada arco tiene peso 1.

Sub-estructura optima del camino de distancia mínima

Los algoritmos de distancia mínima se basan en el hecho de que el camino de distancia mínima entre dos vértices contiene otros caminos de distancia mínima.

Lema 2
Sea $G = (V, E)$ un grafo dirigido con pesos, cuya función de pesos es $w : E \to \mathbb{R}$, $p = \langle v_0, v_1, \ldots, v_k \rangle$ el camino de distancia mínima de un vértice v_0 a un vértice v_k y, para cualquier i y j tal que $0 \leq i \leq j \leq k$, sea $p_{ij} = \langle v_i, v_{u+1}, \ldots, v_j \rangle$ un sub-camino de p del vértice v_i al vértice v_j. Entonces, p_{ij} es el camino de distancia mínima del vértice v_i al vértice v_j.

Arcos de costo negativo

Algunas instancias del problema de distancia mínima de un vértice a todos los demás (SSSP) pueden incluir arcos cuyos pesos son negativos. Si el grafo $G = (V, E)$ contiene ciclos de costo no negativo alcanzables desde el vértice s, entonces $\delta(u, v)$ se mantiene bien definido, incluso si el peso es negativo. Sin embargo, si el grafo contiene un ciclo de costo negativo alcanzable desde el vértice s, $\delta(u, v)$ no esta bien definido.

Relajación

El algoritmo utiliza una técnica de **relajación**. Para cada vértice $v \in V$, se tiene un atributo $v.d$, el cual es una cota superior del peso del camino de distancia mínima de un vértice u a v. Se denota $v.d$ como el **estimado del camino de distancia mínima**.

La inicialización del estimado del camino de distancia mínima se realiza en tiempo $\Theta(V)$.

Relajación

El proceso de **relajar** un arco (u, v) consiste en verificar si es posible mejorar la distancia mínima a un vértice v encontrada hasta el momento, a través de un camino que incluya el vértice u. Si es posible, entonces actualizamos el estimado del camino de distancia mínima $v.d$.

Algoritmo de Dijkstra

El *algoritmo de Dijkstra* resuelve el problema de distancia mínima de un vértice a todos los demás (SSSP) en un grafo dirigido con pesos $G = (V, E)$ para el casa en el cual todos los arcos tienen pesos **no negativos**. Se asume que $w(u, v) \geq 0$ para todos los arcos $(u, v) \in E$.

Algoritmo de Bellman-Ford

El *algoritmo de Bellman-Ford* resuelve el problema de distancia mínima de un vértice a todos los demás (SSSP) en el caso general en el cual los pesos de los arcos pueden ser negativos. Dado un grafo dirigido con pesos $G = (V, E)$, origen s y una función de pesos $w : E \to \mathbb{R}$, el algoritmo de Bellman-Ford indica si existe o no un ciclo de costo negativo alcanzable desde el vértice de origen s. Si no existe dicho ciclo, el algoritmo genera los caminos más cortos con sus respectivos pesos.

Algoritmo de Bellman-Ford

El algoritmo relaja los arcos, disminuyendo progresivamente el estimado de la distancia mínima del origen s a todos los demás vértices $v \in V$ hasta alcanzar el camino de distancia mínima $\delta(u, v)$. El algoritmo retorna **Verdadero** si y solo si el grafo no contiene ciclos de costo negativo alcanzables desde el origen.

Algoritmo de Bellman-Ford

La complejidad temporal del algoritmo de Bellman-Ford es $O(VE)$.

```
def BELLMAN-FORD(V, w, s):
  dist = [INF for _ in range(V)]
  # relax all E edges V-1 times
  for i in range(V - 1):
    # these two loops = O(E), overall O(VE)
    for u in range(V):
      for j in range(len(AdjList[u])):
        v, w = AdjList[u][j]
        # relax
        dist[v] = min(dist[v], dist[u] + w)
```

Algoritmo de Bellman-Ford

```
# after running the O(VE) BF algorithm shown above
hasNegativeCycle = False
# one more pass to check
for u in range(V):
  for j in range(len(AdjList[u])):
    v, w = AdjList[u][j]
    # if this is still possible
    if (dist[v] > dist[u] + w):
      # then negative cycle exists!
      hasNegativeCycle = True
retrun hasNegativeCycle
```

¿Preguntas?

¡Gracias!