Practica de Métodos Algorítmicos de Resolución de Problemas

Pablo Vazquez Gomis
Universidad Complutense de Madrid

January 4, 2020

Abstract

Practica opcional del primer cuatrimestre para la asignatura de Métodos Algorítmicos de Resolución de Problemas.

1 Introducción

1.1 Definicion

Implementar o Java o en C++ el algoritmo de Kruskal junto con una estructura de partición con compresión de caminos.

1.2 Descripción del Algorítmo de Kruskal

En 1956, Joseph B. Kruskal[1] propuso varios metodos para encontrar el arbol de recubrimiento minimo (ARM).

A Mientras la solucion S (Sets de las aristas) no genere un AR con respecto al grafo $G = (V, A)$, añadir a S la arista de menor valor (longitud) que no forme ciclos con la solución. Eventualmente S formará un ARM.

B Sea V' un subset de los vertices del grafo y mientras S no genere un AR. Añadir a la solución el la arista de menor valor que conecte V' o los vertices ya en S

Notese que A es una generalización de B cuando $V' = V$. El objetivo de dicho articulo era presentar un algorítmo mas simple a los ya propuestos en la época, por ello, Kruskal describe el algoritmo con poca precisión y a muy alto nivel sin preocuparse por el coste de dicho metodo. Durante estos años, se han propuesto varias opciones para cubrir los detalles tecnicos que Kruskal deja a la imaginación, como puede ser mantener la lista ordenada de aristas o comprobar si añadir una arista a la solución forma algún ciclo y por tanto ha de ser descartada.

1.3 Implementación

Grafo Con intención de simplificar la implementacion del grafo y optimizar la cota amortizada con respecto a las operaciones *(*insertar) y *(*kruskal), en vez de utilizar una estructura de datos basada en listas de adjacencia, el grafo esta definido mediante una lista ordenada de aristas y otra de vertices (esta última duplica información pero ayuda a simplificar la implementación de otras funciones).

Demostración Utilizando el metodo de agregación podemos calcular el coste de implementar la operación insertar k veces y la operacion kruskal j veces. Los costes reales son los siguientes para cada implementación.

- **Listas de adjacencia**:
  - **Insertar**: 1
  - **Kruskal**: $m \log m + m \log n$ (La primera parte, ordenar la lista, la segunda el algoritmo en si)
- **Set ordenador de aristas**:
  - **Insertar**: $logm$
  - **Kruskal**: $m \log n$ (No es necesario ordenar)

Utilizando estos valores podemos calcular el coste amortizado y comparlo para encontrar la implementacion mas eficiente. El lado derecho de la ecuacion correspondera con el set de aristas ordenadas y el izquierdo con la lista de adjacencia.

$$\sum_{i=0}^{k} \log i + jk \log(n) \leq k + jk \log k + jk \log n \tag{1}$$

$$k \log k + jk \log(n) \leq k + jk \log k + jk \log n \tag{2}$$

$$k \log(k) \leq jk \log(k) \tag{3}$$

$$1 \leq j \tag{4}$$

Es evidente que unicamente en el caso $j = 1$ ambas implementaciones son igual de optimas. En cualquier otra situacion, utilizar un set ordenada de aristas mejora el coste amortizado.

2 El algoritmo

El algoritmo implementa la propuesta A de Kruskal explicada en el punto 1.2. Debido a que el grafo se implementa como una lista de ordenada de aristas encontrar la arista con menor valor (longitud) es trivial.

Por otra parte, comprobar si $S \cup \{e'\}$ forma un AR, siendo S la solución parcial y e' la arista mas prometedora del set restante, no es tan trivial. Usando una estructura de partición P se pueden ir uniendo sets, originalmente definidos como conjuntos unitarios de cada vertice, a medida que se introducen aristas en la solución. Esto es: $(a, b) \in S \Leftrightarrow \{a\} \cup \{b\} \in P$.

El coste de la busqueda de ciclos usando la estructura de partición con compresión de caminos esta en orden $O(\log n)$, en el peor de los casos esto se ha de repetir para todas las aristas por lo que el coste total del algoritmo esta en orden $O(m \log n)$ siendo m el numero de aristas y n el numero de vertices de G.

3 Conclusión

Es evidente la progresión exponencial del algoritmo, aun asi la aplicacion de cotas aun más costosas por nodo limita el numero de nodos y permite solucionar grafos más grandes y densos.

El problema del cliqué es un problema estudiado en profundidaz como cualquier otro problema NP-Completo. Aun habiendo echo grandes avances en el campo que permiten en un tiempo mas o menos razonable gestionar grafos con muchas aristas, aun queda espacio por recorrer dado que la curva exponencial de su complejidad sigue limitando la resolución del problema incluso para ordenadores modernos.

References

[1] Kruskal, J. B. (1956). *On the shortest spanning subtree of a graph and the traveling salesman problem.* Proceedings of the American Mathematical Society. 7(1), 48–50.

[2] Carraghan, R. & Pardalos, P.M. (1990). *An exact algorithm for the maximum clique problem.* Operations Research Letters, 9(6), 375–382.

[3] E. Tomita, T. Seki, *An efficient branch-and-bound algorithm for finding a maximum clique*, Lecture Notes in Computer Science 2631 (2003) 278- 289.

[4] J. Konc & D. Janežič, D. (2007). *An improved branch and bound algorithm for the maximum clique problem.* MATCH - Communications in Mathematicaland in Computer Chemistry, 58, 569-590.