Problemas de eliminación de aristas

Existen, en teoría de grafos, muchos problemas de eliminación de aristas, los cuales pueden ser difíciles de resolver de manera eficiente, o incluso, imposibles de hacerlo de manera de correcta para un número considerable de nodos y aristas.

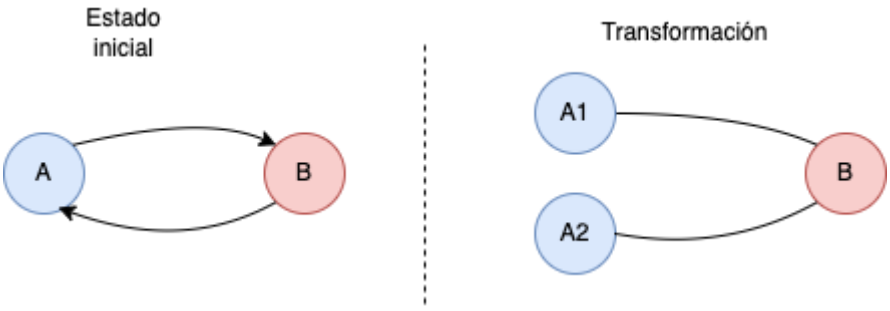
El problema que se presenta es uno de estos:

Problema

Luego de leer el problema en Readme podemos reducirlo a:

> Decir si es posible obtener un grafo bipartito a partir de la eliminación de hasta k aristas de un grafo G

Debemos tener en cuenta que si un planeta apunta a otro entonces existe una arista entre ellos. En caso de apuntarse mutuamente, entonces tenemos que modificar el grafo inicial tomando uno de esos nodos y duplicándolo, de tal manera que uno de ellos tenga la arista que representa el ataque de salida y el otro nodo tenga la arista del ataque de entrada. En la figura siguiente se muestra la transformación:



Si nos damos cuenta, el resultado será el mismo, ya que el objetivo es demostrar que existe o no un grafo bipartito. Como un planeta no se ataca a sí mismo, al duplicar un nodo, estos estarán en la misma parte de la bipartición y la eliminación de aristas sigue siendo inalterable.

Por lo tanto, tenemos, en general, el siguiente problema:

> Entrada:
> Un grafo $G(N, E)$ y un entero k que será el número máximo de aristas a eliminar.
>
> Salida:
>
> - $True$ si $\exists\, p \leq k \mid G(N, E - p)$ es bipartito.

- $False$ en caso contrario

Demostrando NP-completitud

Este problema (llamémosle A) es *NP-completo*, es decir, es un problema de decisión sobre el que podemos probar que existe otro problema *NP-completo* (llamémosle B) tal que: B $\leq p$ A y A $\in$ NP.

Primero demostremos que el problema se encuentra en la clase de problemas NP:

1. Tomemos una instancia de la solución, es decir, una lista de hasta k aristas que se eliminan del grafo G para obtener un grafo bipartito.
2. Comprobemos que la instancia es verdadera. Se debe eliminar las aristas de la lista del grafo G y comprobar si el grafo resultante es bipartito. Esto se puede hacer en tiempo polinómico, ya que la verificación de si un grafo es bipartito se puede hacer en tiempo O(V+E) utilizando el algoritmo BFS.
3. Como se puede verificar en tiempo polinómico si esta instancia es verdadera, el problema se encuentra en la clase de problemas NP.

Ahora, demostremos que es NP- completo aplicando la reducción de uno de los 21 problemas de Karp a este problema.

Problema de corte máximo

Dado un grafo *G* y un entero k *,* determinar si hay un corte de tamaño al menos k en *G.*

Se sabe que este problema es NP-completo . Es fácil ver que el problema está en NP: una respuesta *afirmativa* es fácil de probar presentando un corte lo suficientemente grande. La NP-completitud del problema se puede mostrar, por ejemplo, mediante una reducción de la máxima 2-satisfacibilidad (una restricción del problema de máxima satisfacibilidad. La versión ponderada del problema de decisión fue uno de los 21 problemas NP-completos de Karp Él mostró que era NP-completo por una reducción del problema de partición.

Reducción

(=>) Sea $G = (N, E)$ un grafo de tal manera que al eliminar al menos k aristas este es bipartito.

- Creamos un nuevo grafo G´ que es copia de G y obtengamos el entero p para usarlo en el problema de corte máximo.
- Al eliminar k aristas de G y obtener un grafo bipartito nos quedamos con aristas que van de una bipartición a otra, por lo tanto, si nos quedamos con esas aristas podemos separar con un corte ambas biparticiones. Así que $p = |E| - k$.

- Esto nos permite decir que si existe el grafo bipartito al eliminar al menos k aristas, entonces para G´ y un entero p existe un corte de tamaño al menos p. Ahora, ¿cómo sabemos que el tamaño del corte es al menos p?
- Como la cantidad de aristas que se eliminan de G, es menor o igual que k entonces el número de aristas que quedan cruzando las biparticiones es mayor o igual que p, por lo tanto, el tamaño del corte es mayor o igual que p.

(<=) Sea $G´ = (N´, E´)$ un grafo que presenta un corte de tamaño al menos p.

Seguimos de la misma manera que la demostración anterior:

- Tomamos un grafo G =G´
- Como p es el tamaño del corte que divide en dos particiones al grafo, tomamos el número de todas las aristas que no cruzan ese corte, es decir, $k = |E´| - p$. Estas aristas son las que se encuentran dentro de las particiones (que necesitamos eliminar para obtener el grafo bipartito).
- Así obtenemos que dado G y k determinamos que eliminando a lo sumo k aristas, G es bipartito.

> **Conclusión**
>
> Como puede reducirse el problema de corte máximo al problema presentado y teniendo en cuenta que es NP-completo, podemos concluir que nuestro problema es NP-completo.

Archivos

Solución con backtrack

Se presenta una solución ineficiente pero exacta de nuestro problema, eliminando aristas y comprobando si el grafo resultante es bipartito.

K-aproximación

Un algoritmo de aproximación común para los problemas de eliminación de aristas es el algoritmo de eliminación de aristas aleatorias. Este algoritmo funciona de la siguiente manera:

1. Seleccionamos aleatoriamente k aristas del grafo original y las eliminamos.
2. Verificamos si el grafo resultante es bipartito. Si es así se devuelva el subgrafo bipartito resultante. Si no, volvemos a elegir aleatoriamente k aristas

Este algoritmo es $\frac{k}{2}$ - aproximación del problema, lo que significa que la solución que devuelve siempre tendrá al menos la mitad del tamaño del subgrafo bipartito óptimo. Además, el algoritmo se ejecuta en tiempo polinómico en el tamaño del grafo.

Vamos a demostrar este algoritmo aleatorio de $\frac{k}{2}$ - aproximación.

Podemos garantizar esta aproximación ya que, en promedio, la mitad de las aristas eliminadas serán necesarias para obtener un subgrafo bipartito. Veamos cómo probamos esto:

Sabemos que la probabilidad de que una arista sea necesaria para obtener un subgrafo bipartito es de 1/2 (es o no necesaria). Por lo que la esperanza del conjunto K de aristas aleatorias tomadas es:

$E[K] = \frac{1}{2} + \frac{1}{2} * (E[K-1] + 1)$

Donde $\frac{1}{2}$ representa la probabilidad de que la primera arista eliminada sea necesaria para obtener un subgrafo bipartito y $\frac{1}{2} * (E[K-1] + 1)$ representa la probabilidad de que la primera arista eliminada no sea necesaria y se tenga que eliminar una arista adicional.

La ecuación anterior es una ecuación de recurrente fácil de resolver:
$E[K] = \frac{1}{2} + \frac{1}{2} * E[K-1] + \frac{1}{2}$
$E[K] = 1 + \frac{1}{2} * E[K-1]$
Vemos que $E[K-1] = 1 + \frac{1}{2} * E[K-2]$
Al sustituir en la ecuación original:
$E[K] = 1 + \frac{1}{2} * (1 + \frac{1}{2} * E[X-2]$
$E[K] = 1 + \frac{1}{2} + \frac{1}{4} * E[K-2]$
$E[K] = \frac{3}{2} + \frac{1}{4} * E[K-2]$
Esto se realiza k veces hasta llegar a $E[0] = 0$ y como se puede notar obtenemos $E[K] = \frac{k}{2}$. Esto significa que, en promedio, el algortimo se aproxima en $\frac{k}{2}$ aristas a la solución correcta.

La complejidad temporal en el peor de los casos es $O(E)$ ya que se puede probar eliminar todas las aristas del grafo en el conjunto K.

Solución con metaheurísitca

Además de la solución aleatoria demostrada anteriormente desarrollamos una metaheurística basada en colonia de hormigas.:

1. Se inicializa una colonia de hormigas. Cada hormiga se mueve por el grafo y construye una solución parcial.

2. Se asigna una feromona inicial a cada arista del grafo.
3. La solución parcial está dada por:

- La hormiga comienza en un vértice aleatorio del grafo.

- La hormiga se mueve a un vecino aleatorio del vértice actual, eligiendo una arista con una probabilidad proporcional a la cantidad de feromona en la arista.

- Si la arista elegida conecta dos vértices del mismo color, la hormiga la elimina y continúa construyendo la solución.

- Si la arista elegida conecta dos vértices de diferente color, la hormiga la mantiene y continúa construyendo la solución.

- Si la hormiga ha eliminado k aristas, termina la construcción de la solución.

4. Se Actualiza la cantidad de feromona en cada arista de acuerdo a la calidad de las soluciones construidas por las hormigas.
5. Se Repiten los pasos 3 y 4 hasta que se alcance un número de iteraciones
6. Se devuelve la mejor solución encontrada.

Bibliografía

1. M. YANNAKAKIS, Node and edge-deletion NP-complete problems, Proc. of Tenth Annual ACM Symposium on Theory of Computing, 1978, pp. 253-264.
2. M. YANNAKAKIS, Edge-deletion problems, 1981
3. R. Karlin, Anna. An Improved Approximation Algorithm for the Minimum k-Edge Connected Multi-subgraph Problem
4. G. I. ORLOVA, AND Y. G. DORFMAN, Finding the maximum cut in a graph, Eng. Cybernetics, 10 (1972), pp. 502-506.