k-remove-bipartite

Hansel Blanco Martí
Elena Rodríguez Horta

1. Problema: Subgrafo bipartito eliminando k aristas

Dado un grafo $G =< V, E >$ con n vértices y m aristas y un entero k, $0 \leq k \leq m$ se quiere determinar si es posible obtener un subgrafo bipartito de G, eliminando a lo sumo k aristas de G.

1.1. NP completitud

El problema en cuestión se trata de un problema NP completo.

Demostración

Primeramente se demostrará que el problema es NP.

Para demostrar que el problema es NP, para un grafo $G =< V, E >$ se utilizará como certificado de verificación el conjunto de aristas $E' \subseteq E$ que al eliminarlas queda un subgrafo de G bipartito. El algoritmo de verificación comprobará que E' tenga cardinalidad a lo sumo k para luego verificar que es posible obtener un subgrafo bipartito eliminando las aristas de E'. Un posible algoritmo para esto último es eliminar estas aristas de G, cuya complejidad en general puede asegurarse que es polinomial y luego verificar si el subgrafo G' resultante es bipartito, lo cual también puede hacerse en tiempo polinomial con un recorrido BFS sobre el grafo, como se demostrará más adelante.

A continuación se demostrará que:

DECISION CORTE MAXIMO $\leq_p$ SUBGRAFO BIPARTITO ELIMINANDO K-ARISTAS.

En el problema de CORTE MAXIMO se quiere un subconjunto S del conjunto de vértices tal que el número de aristas entre S y el subconjunto complementario sea el mayor posible. El problema de decisión relacionado, al que se denotará por DECISION CORTE MAXIMO, consiste en dado un grafo G y un entero k, determinar si existe en G un corte cruzado por al menos k aristas. Se sabe que se trata de un problema NP-completo. [1]

Sea una instancia cualquiera $< G, k >$ del problema de decisión de corte máximo, donde G es un grafo de n vértices y m aristas. A partir de esta, se puede construir en tiempo polinomial la instancia $< G, m - k >$ del problema de determinar si es posible obtener un subgrafo bipartito de G eliminando a lo sumo $m - k$ aristas. A continuación se demostrará que la respuesta para el

problema de decisión de corte máximo es afirmativa si y solo si la respuesta para la instancia del subgrafo bipartito es también afirmativa.

Sea $< G, k >$ una instancia del problema de decisión de corte máximo para la cual se sabe que es posible encontrar en G un corte cruzado por al menos k aristas. Sean las aristas que no cruzan dicho corte (son a lo sumo m-k), si se eliminan esas aristas de G, se obtiene un subgrafo bipartito que tiene los mismos vértices que G y sus aristas son un subconjunto de tamaño al menos k de las aristas de G. Luego, si es posible encontrar un corte en G atravesado por al menos k aristas, es posible obtener un subgrafo bipartito de G eliminando m-k aristas a lo sumo.

Sea $< G, m-k >$ una instancia del problema de obtener un subgrafo bipartiro eliminando a lo sumo m-k aristas para la cual se sabe que es posible obtener un subgrafo bipartito de G eliminando a lo sumo m-k aristas. Sea G' el subgrafo bipartito obtenido y sean A y B los conjuntos de vértices independientes de una bipartición del subgrafo, se sabe que $A \cup B = V$ donde V es el conjunto de vértices del grafo G pues la única transformación que se hizo para obtener el subgrafo bipartito fue eliminar aristas. Además se sabe que en G' hay al menos k aristas que cruzan la bipartición. Sea el corte A, V-A en G, este corte tiene al menos a todas las aristas que cruzan la bipartición de G', luego tiene al menos k aristas. Luego, si es posible obtener un subgrafo bipartito de G eliminando m-k aristas a lo sumo, es posible encontrar un corte en G atravesado por al menos k aristas.

Se ha demostrado que es posible reducir en tiempo polinomial una instancia del problema DECISION CORTE MAXIMO a una instancia del problema SUBGRAFO BIPARTITO ELIMINANDO K-ARISTAS, luego si fuera posible resolver este último en tiempo polinomial, entonces por composición de polinomios, también sería posible resolver DECISION CORTE MAXIMO en tiempo polinomial.

En este punto ha quedado demostrado que el problema es NP y NP duro, por tanto es NP completo.

2. Solución por fuerza bruta

Dado un grafo $G =< V, E >$ con n vértices y m aristas y un entero k, $0 \leq k \leq m$ se quiere determinar si es posible obtener un subgrafo bipartito de G, eliminando a lo sumo k aristas de G.

Dividiremos el problema en dos casos: si el grafo tiene a lo sumo k aristas y si el grafo tiene más de k aristas.

Si el grafo tiene cantidad de aristas menor o igual que k, entonces al eliminar todas las aristas de grafo se están eliminando a lo sumo k aristas y el grafo resultante es bipartito.

Si el grafo tiene más de k aristas podemos garantizar que es posible convertirlo en bipartito eliminando a lo sumo k aristas si y solo si es posible convertirlo en bipartito eliminando k aristas.

Demostración

Se sabe que un grafo es bipartito si y solo si no tiene ciclos de longitud impar y además se sabe que eliminar aristas de un grafo no crea ciclos que no estuvieran ya en el grafo, por tanto se puede garantizar que si no es posible obtener un subgrafo bipartito eliminando k aristas, entonces tampoco se podrá obtener elimimando menos de k aristas y por tanto, tampoco a lo sumo k (si elimimando k no se eliminan todos los ciclos de longitud impar, no se eliminarán quitando menos que k aristas). Luego, si es posible hacerlo eliminando a lo sumo k aristas, es posible hacerlo eliminando k aristas. A la misma vez, si es posible obtener un subgrafo bipartito elimimando k aristas entonces es posible hacerlo eliminando a lo sumo k aristas.

Luego puede enforcarse el problema en este caso como: Dado un grafo $G =< V, E >$ con n vértices y m aristas y un entero k, $0 \leq k \leq m$ se quiere determinar si es posible obtener un subgrafo bipartito de G, eliminando k aristas de G.

La solución que se propone verifica si el grafo tiene k aristas o menos y en ese caso retorna que es posible hacerlo bipartito eliminando a lo sumo k ya que un grafo sin aristas es bipartito, y si tiene más de k aristas lo que hace es probar con todas las combinaciones de aristas de tamaño k, si el subgrafo resultante de eliminar esas k aristas es bipartito. La demostración de la correctitud se orientará al segundo de los casos.

2.1. Correctitud

Si existe un conjunto de aristas de tamaño k, tal que al eliminar dichas aristas el grafo resultante es bipartito, entonces el algoritmo encuentra dicho conjunto pues revisa todos los posibles conjuntos de aristas de tamaño k. A la misma vez, cualquier conjunto de tamaño k de aristas que encuentre el algoritmo es subconjunto de E pues estos conjuntos se forman como combinaciones de E de tamaño k.

Quedaría por demostrar la correctitud del algoritmo que determina si un grafo es bipartito. Si este algoritmo fuera correcto, entonces la solución propuesta es correcta pues si existe un conjunto de aristas que al eliminarlas queda un subgrafo bipartito sabemos que el algoritmo las ecuentra y devuelve que en efecto el grafo es bipartito y además si el algoritmo encuentra que para un conjunto de k aristas el grafo que resulta de eliminarlas es bipartito, entonces existe un conjnto de k aristas que al eliminarlas de G el subgrafo resultante es bipartito.

2.1.1. Demostración de la corrección del método `is_bipartite_bfs`

Se sabe que un grafo es bipartito si y solo si es bicoloreable, por lo que el algoritmo determinará si es posible o no bicolorear los vértices del grafo. Además se sabe que un grafo es bipartito si y solo si no tiene ciclos de longitud impar.

El método `is_bipartite_bfs` utiliza una búsqueda a lo ancho (BFS) para recorrer el grafo y colorear los vértices. Comienza por inicializar todos los vértices como no coloreados comienza un recorrido BFS sobre el grafo. Cuando encuentra un vértice que no está coloreado lo colorea con el color opuesto al de

su vértice padre y si encuentra un vértice que ya está coloreado, verifica que el color que tiene sea el mismo que tocaría asignarle en ese momento, si esto no se cumple entonces devuelve que el grafo no es bicoloreable.

Un grafo es bicoloreable si y solo si el algoritmo devuelve que es bicoloreable.

Sea G un grafo bicoloreable. Supóngase que en este caso el algoritmo devuelve que el grafo no es bicoloreable ($falso$). Se sabe que el algoritmo solo devuelve $falso$ si al encontrar una arista de retroceso durante el recorrido BFS, los vértices extremos de dicha arista tienen asignado el mismo color. Como el algoritmo asigna dos colores de forma alternada se sabe que dos vértices tienen el mismo color si y solo si están a distancia par en el árbol de recorrido BFS. Luego la arista de retroceso encontrada forma parte de un ciclo de longitud impar en G, pero G es bicoloreable y por tanto bipartito por lo que no tiene ciclos de longitud impar. Luego no es posible que el algoritmo devuelva $falso$ en este caso, y por tanto devuelve que es bicoloreable($true$)

Sea G un grafo para el que el algoritmo devuelve que sí es bicoloreable ($true$). Supóngase que el grafo no es bicoloreable. Si el algoritmo devuelve $true$ es porque encontró una bicoloración del grafo, pero como el grafo no es bicoloreable entonces hay al menos dos vértices del grafo adyacentes que tienen el mismo color en dicha bicoloración. Llamemos a estos vértices u y v y sea $< u, v >$ la arista que hay entre ellos. Todas las aristas del grafo G en el recorrido BFS son descubiertas una vez. En el momento en que se encuentra la arista $< u, v >$ en el recorrido si alguno de los vértices se está encontrando por primera vez, entonces al vértice que se está descubriendo, sea v sin perder generalidad, se le asigna el color contrario al que tiene el vértice u. Como además una vez coloreado un vértice este nunca cambia de color, en la coloración final u y v tienen colores distintos. Si por el contrario cuando la arista $< u, v >$ aparece ambos vértices ya se habían descubierto antes y por tanto tenían asignado un color, si el algoritmo no devuelve $falso$ en ese momento es porque u y v tienen colores distintos y como una vez coloreado un vértice este nunca cambia de color, en la coloración final u y v no pueden tener el mismo color. Luego no es posible que dos vértices adyacentes tengan el mismo color.

2.2. Complejidad temporal

Se analizan $\binom{m}{k}$ conjuntos de aristas y por cada uno se realiza un recorrido BFS sobre G, por lo que la complejidad temporal es $O(\frac{m!}{(m-k)!k!}\dot{(}n+m))$

3. Solución aproximada

A continuación se propone una metaheurística para encontrar la menor cantidad de aristas que hay que eliminar en un grafo G para obtener un subgrafo bipartito.

El algoritmo que se propone para obtener la menor cantidad de aristas de $G =< V, E >$ tal que al eliminarlas el subgrafo que queda es bipartito consta de los siguientes pasos:

1. Particionar los vértices del grafo en dos conjuntos S y V-S.

2. Mientras haya alguna menera de aumentar la cantidad de aristas que van de un conjunto a otro cambiando un vértice de conjunto, hacerlo.

Se puede garantizar que el algoritmo termina porque cada vez que se hace un movimiento se aumenta la cantidad de aristas que cruzan el corte establecido, y esta cantidad está acotada superiorment epor la cantidad de aristas del grafo.

Cuando el algoritmo termine, aquellas aristas que queden dentro de los conjuntos del corte serán las que hay que eliminar para obtener un subgrafo bipartito. Este algoritmo se trata de una aproximación, no se puede garantizar que converge a un óptimo global. Existen casos con óptimos locales, con un valor subóptimo de k pero donde no hay ningún intercambio de vértices que mejore el valor. Una forma de tratar de escapar de los óptimos locales pudiera ser probar con varias distribuciones iniciales de los vértices y retornar la menor cantidad de aristas a eliminar que se obtenga, pero esta técnica tampoco es infalible.

En el óptimo del algoritmo puede garantizarse que todos los los vértices tienen al menos la mitad de sus aristas de un lado a otro de la bipartición, y por tanto la cantidad de aristas que hay en el interior de los conjuntos, que son las que hay que eliminar, es siempre menor que $\frac{m}{2}$.

3.1. Complejidad temporal

El algoritmo ejecuta un ciclo mientras sea posible aumentar el número de aristas que cruzan el corte definido. En cada iteración aumenta al menos en 1 la cantidad de aristas que cruzan el corte y por tanto este ciclo puede ejecutarse $|E|$ veces a lo sumo. Dentro de este ciclo, se realizan comprobaciones por cada vértice de cada conjunto de la bipartición si se tiene en sus vecinos, potencialmente $|V|$ vecinos, más cantidad de vértices de su mismo conjunto o del contrario, en el caso de que tenga menos aristas de cruce, se cambia de conjunto de la bipartición. Por tanto, por cada iteración del ciclo externo se realizan a lo sumo $|V|^2$ operaciones.

El resto de operaciones está acotado por dicha complejidad.

Luego, la complejidad temporal pertenece a $O(|E| * |V|^2)$.

Referencias

[1] Wikipedia contributors : Maximum cut — Wikipedia, the free encyclopedia, 2023. [Online; accessed 17-June-2023].