Comer hongos solo daa problemas

Sheyla Leyva Sánchez - Darío Fragas Gonzalez

Curso 2023

Índice

1. Problema

La extraña aeventura de Sheyla

Sheyla veía a Darío dormido en el sofá mientras pensaba: "Qué flojera, le ganó el cansancio luego de unos pocos champiñones. Yo comí muchos más que él y aquí estoy, enterita". Fue entonces cuando sintió sonidos de motores provenientes del patio. Cuando salió, se encontró con una inmnensa nave espacial de la que salía una criatura himanoide color azul marino con grandes ojos y cuello largo. La criatura gritaba tratando de hacerse oír sobre los sonidos del motor de la nave: Çomandante, monte, la galaxia la necesita". Sheyla montó en la nave sin pensarlo mucho pues, aunque no conocía a la criatura, alguien color azul marino no podía ser malvado.

Una vez en la nave, la criatura le explicó la situación. Sucede que los n planetas de la galaxia se estaban apuntando entre sí con m láseres de largo alcance. El mapa de ataques entre planetas conformaba un grafo. Sheyla podía intervenir para cancelar algunos de estos ataques, pero había un número límite $k < m$ de forma que, como mínimo, $m - k$ ataques debían realizarse. Para reducir al mínimo el daño de la galaxia, los altos mandos decidieron que lo mejor sería cancelar ataques de forma que se pudiera separar a los planetas en dos facciones. Estas facciones formarían un grafo bipartito en el que miembros de una facción no se atacarían mutuamente y así al menos la guerra se pudiera controlar. Ayude a Sheyla a encontrar si existe una forma de cancelar hasta k ataques de forma que se obtenga un grafo bipartito.

2. Resumen del problema

Determinar, dado un grafo no dirigido $G = (V, E)$ y un número entero k, si es posible convertir G en un grafo bipartito eliminando a lo sumo k aristas de E.

3. Demostración de la NP-completitud

El problema es un caso particular de un problema de modificación de aristas. Los problemas de modificación de aristas involucran realizar pequeños cambios en el conjunto de aristas de un grafo de entrada para obtener un grafo con una propiedad deseada. Incluyen problemas de complementación, eliminación y edición. La mayoría de estos problemas están demostrados como problemas NP-completos, en la siguiente tabla de [1] se muestra un buen resumen:

Como se puede observar, el problema de bipartición, eliminando aristas es NPC.

En el artículo [2], que es el mismo artículo que se cita en la tabla para el caso de bipartición con la operación de eliminación, se hace habla de una equivalencia entre el problema de bipartición de aristas con eliminación y el popular problema

Summary of complexity results for some edge modification problems

Graph Classes	Completion	Deletion	Editing
Perfect	NPC [26,6]	NPC [26,6]	NPC [26,6]
Chordal	NPC [35]	NPC [30]	NPC [30]
Interval	NPC [14,35]	NPC [16]	**NPC**
Unit Interval	NPC [35]	NPC [16]	**NPC**
Circular-Arc	NPC [30]	NPC [30]	**NPC**
Unit Circular-Arc	NPC [30]	NPC [30]	**NPC**
Proper Circular-Arc	NPC [30]	NPC [30]	**NPC**
Chain	NPC [35]	NPC [30]	?
Comparability	NPC [20]	NPC [36]	NPC [26]
Cograph	NPC [12]	NPC [12]	?
AT-Free	?	NPC [30]	?
Threshold	NPC [23]	NPC [23]	?
Bipartite	irrelevant	NPC [15]	NPC [15]
Split	NPC [26]	NPC [26]	P [21]
Cluster	P [30]	NPC [12]	NPC [30]
Trivially Perfect	NPC [35]	NPC [30]	?
Permutation	**NPC**	**NPC**	**NPC**
Circle	**NPC**	**NPC**	**NPC**
Weakly Chordal	**NPC**	**NPC**	?
Bridged	?	**NPC**	?
Clique-Helly Circular-Arc	**NPC**	**NPC**	**NPC**
Clique-Helly Chordal	**NPC**	**NPC**	**NPC**
Clique-Helly Perfect	**NPC**	**NPC**	**NPC**
Clique-Helly Comparability	**NPC**	**NPC**	**NPC**
Clique-Helly Permutation	**NPC**	**NPC**	**NPC**

Boldfaced results are obtained in this work, "NPC" indicates an NP-complete problem, "P" a polynomial problem, and "?" an open problem.

Figura 1: Problemas de modificación de aristas

de corte máximo simple [1].

El problema de corte máximo simple se define de la siguiente manera:
Encontrar una partición de los vértices de un grafo en dos conjuntos de manera que el número de aristas entre esos conjuntos sea lo más grande posible.

El problema de decisión correspondiente sería:
Dado un grafo G y un entero k, determinar si existe un corte de tamaño al menos k en G.

Karp probó la NP-completitud del problema de corte máximo reduciéndolo al problema de partición.

Aunque ya conocemos que nuestro problema, al ser un problema de bipartición con eliminación es NPC, realicemos la reducción al problema de decisión correspondiente al corte máximo.

Demostración.
Problema 1 (Bipartición con k-eliminación): Dado un grafo no dirigido $G = (V, E)$ y un entero k, determinar si existe un subconjunto E' de E con $|E'| <= k$ tal que $G' = (V, E - E')$ es bipartito.

Problema 2 (Corte con tamaño k): Dado un grafo no dirigido $G = (V, E)$ y un entero k, determinar si existe una partición de V en $V1$ y $V2$ tal que $|E(V1, V2)| >= k$.

[1] Se considera simple pues es la versión del problema con aristas no ponderadas

a) Demostremos que el problema 1 es NP:

Se puede comprobar en tiempo polinomial si el grafo resultante de quitar k aristas $G' = (V, E - F)$ (grafo resultante) es bipartito, esto toma $O(|V| + |E|)$ tiempo usando un algoritmo de verificación bipartita, por lo tanto, el algoritmo es NP.

b) Reduzcamos desde el problema 2 para mostrar la NP-completitud:

Dada la entrada (G, k) del problema 2:

1. Se crea un grafo G' idéntico a G.
2. Se establece $k' = |E| - k$
3. Se devuelve (G', k')

Supongamos que G tiene una solución para el Problema 1 eliminando E' aristas: $\Rightarrow G' = (V, E - E')$ es bipartito y $|E'| <= k$ por lo que $|E - E'| >= |E| - k$

Supongamos que G tiene una solución para el Problema 2 con una partición $V1, V2$: $\Rightarrow |E(V1, V2)| >= k$, Se necesita eliminar como máximo $|E| - k$ aristas para hacer bipartito a G'.

Por lo tanto, existen soluciones equivalentes entre los problemas.

Si G tiene un corte de tamaño al menos k, entonces G' se puede hacer bipartito eliminando $k' = |E| - k$ aristas. A la inversa, si G' se puede hacer bipartito eliminando k' aristas, entonces G tiene un corte de tamaño al menos $|E| - k'$.

La entrada original (G, k) se usa para construir (G', k') en tiempo polinomial. Como Problema 2 es NP-Completo, esto demuestra que el Problema 1 también lo es.

□

4. Solución exponencial

Ahora que conocemos que el problema es NP-completo, sabemos que no habrá una solución polinomial exacta [2] para este, sin embargo, el problema se puede resolver sin problemas con algoritmo de complejidad temporal exponencial.

Un primer algoritmo para dar solución al problema podría ser:

1. Enumerar todas las biparticiones posibles de V. Son $2^{|V|}$
2. Luego para cada bipartición calcula cuántas aristas se deben eliminar para que G coincida con la bipartición

[2] se podría tratar de aproximar la solución con un algoritmo polinomial pero sería solo una aproximación

3. Si la cantidad de aristas a eliminar es $<= k$, agrega la bipartición a una lista de candidatos (o devolver True directamente).

4. Finalmente si la lista está vacía (o si nunca se devolvió True), devolver False

Este algoritmo fue implementado en Python en el script *exponential.py* y tiene una complejidad temporal de $O(2^V)$

5. Primera solución

Un algoritmo FPT (Fixed-Parameter Tractable) es un algoritmo para resolver problemas NP-completos que es exponencial solamente en algunos parámetros de la entrada, no en el tamaño total de la entrada. La compresión iterativa es una técnica comúnmente usada para diseñar algoritmos FPT.

Bipartición de aristas fue uno de los primeros problemas que demostraron admitir un algoritmo FPT usando la técnica de compresión iterativa. [3]

Una primera idea de algoritmo FPT podría ser la implementada en *ftp1.py*

Este algoritmo funciona de la siguiente manera:

Si el grafo es bipartito, devolvemos True. Luego generamos todos los posibles subconjuntos de aristas de tamaño como máximo k usando *find-cuts*. Esto usa el parámetro k para limitar el espacio de búsqueda.

Para cada corte, eliminamos las aristas del grafo y verificamos si el grafo restante es bipartito.

Las funciones *find-cuts* y *bipartite* tienen un tiempo de ejecución de $O(2^k)$ y $O(n + m)$ respectivamente, donde n es el número de nodos y m es el número de aristas.

En total, el tiempo de ejecución total es $O(2^k * (n + m))$ que es FPT en k.

La idea es limitar el espacio de búsqueda a recortes de tamaño como máximo k, en lugar de probar todos los posibles subconjuntos de aristas o biparticiones (como se hizo en el primer algoritmo); esto hace que el algoritmo sea FPT en k.

En la actualidad la mejor complejidad temporal que se ha podido lograr para este problema es $O(1{,}977^k * nm)$ [3]

6. Solución aproximada

Un enfoque común para aproximar problemas de eliminación de aristas es el método de eliminación de aristas aleatorias. Este método funciona de la siguiente manera:

- Se seleccionan al azar k aristas del grafo original y se eliminan.
- Se comprueba si el grafo restante es bipartito. Si lo es, se devuelve el subgrafo bipartito resultante. De lo contrario, se vuelven a seleccionar al azar k aristas.

Esto se repite según el número de iteraciones por tanto tendrá una complejidad temporal de $O(iterations * (m + n))$

Este algoritmo es $\frac{k}{2}$ - aproximación del problema. Podemos garantizar esta aproximación mediante un análisis probabilístico que muestra que, en promedio, la mitad de las aristas eliminadas serán necesarias para un subgrafo bipartito.

Demostración. Cada arista tiene probabilidad $\frac{1}{2}$ de ser necesaria para un subgrafo bipartito, es decir, probabilidad $\frac{1}{2}$ de que sea eliminada. Si eliminamos k aristas:

La esperanza (promedio) de la primera arista eliminada es $1/2$, ya que tiene probabilidad $1/2$ de ser necesaria.

La esperanza de la segunda arista eliminada será $\frac{1}{2} * (E[K-1]+1)$, donde E[K-1] es la esperanza de las k-1 aristas restantes y el 1 representa la primera arista.

Esta es una ecuación recursiva que se resuelve como: $E[K] = K/2$.

Lo que significa que, en promedio, nuestro algoritmo se aproxima en $\frac{k}{2}$ aristas a la solución óptima. □

7. Solución metaheurística

Por último decidimos que sería buena idea intentar una solución con metaheurísticas. Cualquier meteheurística pensamos que podía ser adecuada para una aproximación; sin embargo según [**?**] GRASP y búsqueda local pueden ser buenas opciones.

Referencias

[1] Pablo Burzyna, Flavia Bonomoa, Guillermo Duránb. "NP-completeness results for edge modification problems"

[2] M. Garey, D. Johnson, L. Stockmeyer. "Some simplified NP-complete graph problems"

[3] Marcin Pilipczuk, Michał Pilipczuk, Marcin Wrochna. ."Edge Bipartization Faster than 2^k"

[4] ilberto F. de Sousa Filho, Lucidio dos Anjos F. Cabral2, Luiz Satoru Ochi, Fabio Protti. "Metaheuristic GRASP for the Bicluster Editing Problem"