UIMP
Universidad Internacional
Menéndez Pelayo

Formalización del problema de la partición del grafo

Laura Rodríguez Navas
rodrigueznavas@posgrado.uimp.es

Junio 2020
Resolución de problemas con metaheurísticos

Resumen

Dentro del amplio campo de la Teoría de Grafos, en este trabajo se considera el problema de la partición del grafo: dado un grafo no dirigido cuyo número de vértices es par y cuyas aristas tienen asociado un peso, se trata de encontrar la partición del conjunto de vértices en dos subconjuntos de tamaño similar de manera que se minimice la suma de los pesos de aquellas aristas que unen vértices que están en diferentes conjuntos

En el capítulo inicial se presenta una breve introducción al problema. En el segundo capítulo, se proponen tres codificaciones para este problema, señalando sus aspectos más relevantes (positivos y negativos), además de ejemplos. Concretamente, las codificaciones son implementaciones en Python 3 de los algoritmos basados en técnicas metaheurísticas: Kernighan-Lin, Spectral Bisection y Multilevel Spectral Bisection. Y en el tercer capítulo, se concluye con una comparativa entre las diferentes codificaciones.

Las codificaciones se encuentran en el repositorio:

`https://github.com/lrodrin/masterAI/tree/master/A2`

Tabla de Contenidos

1. Introducción a la Partición de Grafos

1.1 Introducción

En los últimos años el problema de la partición de grafos ha sido ampliamente estudiado. El problema se deriva de situaciones del mundo real por lo que tiene aplicación en varias ramas de la ingeniería y de la investigación. La partición de grafos es una disciplina ubicada entre las ciencias computacionales y la matemática aplicada. El problema de la partición de grafos es un problema importante que tiene aplicaciones extensas en muchas áreas ya que se busca minimizar los costes o maximizar los beneficios a través de la correcta distribución de los recursos.

La primera aplicación del problema fue en la partición de componentes de un circuito en un conjunto de tarjetas que juntas realizaban tareas en un dispositivo electrónico. Las tarjetas tenían un tamaño limitado, de tal manera que el dispositivo no llegara a ser muy grande, y el número de elementos de cada tarjeta estaba restringido. Si el circuito era demasiado grande podía ser dividido en varias tarjetas las cuales estaban interconectadas, sin embargo, el coste de la interconexión era muy elevado por lo que el número de interconexiones debía ser minimizado.

La aplicación descrita fue presentada en [1], en la cual se define un algoritmo eficiente para encontrar buenas soluciones. En la aplicación, el circuito es asociado a un grafo y las tarjetas como subconjuntos de una partición. Los nodos del grafo son representados por los componentes electrónicos y las aristas forman las interconexiones entre los componentes y las tarjetas.

1.2 Descripción el problema

El problema de partición de grafos puede formularse como un problema de programación entera.

Los problemas de programación entera generalmente están relacionados directamente a problemas combinatorios, esto genera que al momento de resolver los problemas de programación entera se encuentren restricciones dado el coste computacional de resolverlos; por ese motivo se han desarrollado algoritmos que buscan soluciones de una forma más directa, la restricción de los mismos es que no se puede garantizar que la solución encontrada sea la óptima.

El problema de partición de grafos ha sido denominado como un problema NP-Hard[2], lo que implica que las soluciones para él no pueden ser encontradas en tiempos razonables.

2. Algoritmos basados en metaheurísticas

Debido a la *"intratabilidad"* del problema de la partición de un grafo, no existe un algoritmo concreto que permita obtener en tiempo polinómico una solución óptima a la partición de cualquier grafo. Es por ello que, los algoritmos codificados para este trabajo se basan en la metaheurística, con el objetivo de obtener soluciones de buena calidad en tiempos computacionales aceptables, a pesar de que los algoritmos metaheurísticos no garantizan que se vaya a obtener una solución óptima al problema, incluso podría suceder que no se obtenga ninguna solución.

En el siguiente apartado se analizan los algoritmos Kernighan-Lin[1] (ver apartado 2.1), Specrtal Bisection (ver apartado 2.2) y Multilevel Spectral Bisection (ver apartado 2.3), tres algoritmos diseñados específicamente para la resolución del problema de la partición de grafos. Cualquiera de estos algoritmos proporciona una solución factible al problema, pudiendo ser esta óptima o no. Además de describir cada uno de los algoritmos detalladamente, las principales propiedades de cada uno de ellos, también se describirán algunos ejemplos.

2.1 Kernighan-Lin

El algoritmo Kernighan-Lin[1], a menudo abreviado como K-L, es uno de los primeros algoritmos de partición de grafos y fue desarrollado originalmente para optimizar la colocación de circuitos electrónicos en tarjetas de circuito impreso para minimizar el número de conexiones entre las tarjetas.

El algoritmo K-L no crea particiones, sino que las mejora iterativamente. La idea original era tomar una partición aleatoria y aplicarle Kernighan-Lin[1]. Esto se repetiría varias veces y se elegiría el mejor resultado. Mientras que para grafos pequeños esto ofrece resultados razonables, es bastante ineficiente para tamaños de problemas más grandes.

Decimos que es un algoritmo:

- **Iterativo**. Esto significa que el grafo inicialmente ya está particionado, pero la aplicación del algoritmo intentará mejorar u optimizar la partición inicial.
- **Voraz**. Esto significa que el algoritmo hará cambios si hay un beneficio inmediato sin considerar otras formas posibles de obtener una solución óptima.
- **Determinista** porque se obtendrá el mismo resultado cada vez que se aplique el algoritmo.

Hoy en día, el algoritmo se usa para mejorar las particiones encontradas por otros algoritmos. Como veremos, K-L tiene una visión algo *"local"* del problema, tratando de mejorar las particiones mediante el intercambio de nodos vecinos. Por lo tanto, complementa muy bien los algoritmos que tienen una visión más *"global"* del problema, pero tienden a ignorar las características locales. Un ejemplo de este tipo de algoritmos seria la partición espectral (ver sección 2.2).

C. Fiduccia y R. Mattheyses realizaron importantes avances prácticos en [3] quienes mejoraron el algoritmo de tal manera que una iteración se ejecuta en $O(n^2)$ en lugar de $O(n^2 \log n)$.

2.1.1 Descripción

Ventajas:

- El algoritmo es robusto.

Desventajas:

- Los resultados son aleatorios porque el algoritmo comienza con una partición aleatoria.

- Computacionalmente es un algoritmo lento.
- Solo se crean dos particiones del mismo tamaño.
- Las particiones tienen que tener el mismo tamaño para que el algoritmo no intente encontrar tamaños de partición óptimos que ya existan.
- No resuelve muy bien los problemas con aristas ponderadas.
- La solución dependerá en gran medida del primer intercambio.

2.1.2 Ejemplo

2.2 Spectral Bisection

2.2.1 Descripción

2.2.2 Ejemplo

2.3 Multilevel Spectral Bisection

El particionamiento de grafos por multilevel es un método moderno que reduce el tamaño de las particiones del grafo con la combinación de los vértices y las aristas sobre varios niveles, creando particiones del grafo cada vez más pequeñas y extensas con muchas variaciones y combinaciones de diferentes métodos.

METIS es un algoritmo de partición que se enfoca en minimizar el número de bordes de partición cruzados y distribuir la carga de trabajo de manera uniforme entre las particiones. Con METIS, los gráficos se dividen en tres fases. La primera fase es la fase de engrosamiento, la segunda la fase de partición y la tercera y última fase la fase de no engrosamiento (Figura X). La fase de particionamiento contiene particiones bi-particionadas y K-way. A diferencia de la partición K-way, la bi-partición se realiza de forma recursiva. Las subseccion a continuación contienen una explicación más extensa de estas diversas fases.

2.3.1 Descripción

2.3.2 Ejemplo

3. Comparativa entre los diferentes algoritmos

Para finalizar, veremos una comparativa entre los algoritmos que hemos analizado en las secciones previas, tras ser aplicados sobre grafos aleatorios. Pero, en primer lugar definimos el concepto de grafo aleatorio.

En el tabla siguiente, se presenta una comparación entre los algoritmos codificados de este trabajo en términos de tiempo computacional. Se muestra el tiempo (en segundos) en completar ejecuciones sobre un conjunto de grafos aleatorios para distintos números de vértices, n. Para ello se ha utilizando un MacBook Pro, con un procesador de cuatro núcleos de 2.8 GHz y con 16 GB de memoria RAM. El conjunto de datos que se ha utilizado durante la evaluación de los algoritmos se puede encontrar en el repositorio de GitHub, dentro de la carpeta *dataset*.

n	307	552	861
Kernighan-Lin	0.0117	0.0214	0.0451
Spectral Bisection	0.0021	0.0031	0.0060
Multilevel Spectral Bisection	0.0025	0.0031	0.0037

Podemos ver como los algoritmos Kernighan-Lin, Spectral Bisection y Multilevel Spectral Bisection tienen unos tiempos de ejecución similares. Aunque se puede observar que el algoritmo Multilevel Spectral Bisection es el más eficiente que se ha codificado.

Es de esperar que a mayor número de vértices los algoritmos tengan mayor tiempo de ejecución.

4. Conjunto de datos

Para la realización de este trabajo se ha utilizado tres conjuntos de datos. Cada conjunto de datos se ha creado aleatoriamente utilizando la herramienta generatedata.com y se ha guardado en un fichero *CSV*. Los ficheros *CSV* contienen información que representan parejas de nodos (vértices), con pesos en las aristas que forman un grafo no dirigido.

El primer conjunto de datos lo forma el grafo de la figura 4.1. Se ha utilizado para describir sobre los ejemplos de los diferentes algoritmos.

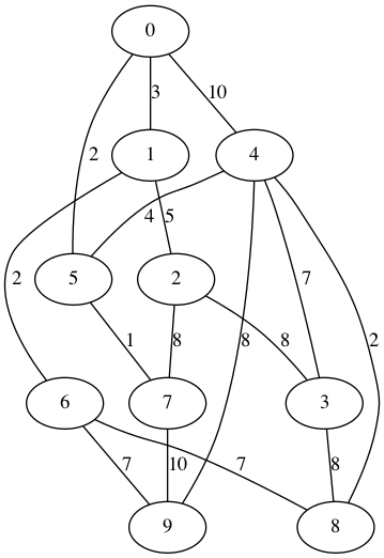

Figura 4.1: Grafo no dirigido de 10 nodos

El segundo conjunto de datos lo forma el grafo de la figura 4.2

El tercer conjunto de datos lo forma el grafo de la figura 4.3.

El segundo y tercer conjunto de datos se han utilizado para la comparativa entre los diferentes algoritmos.

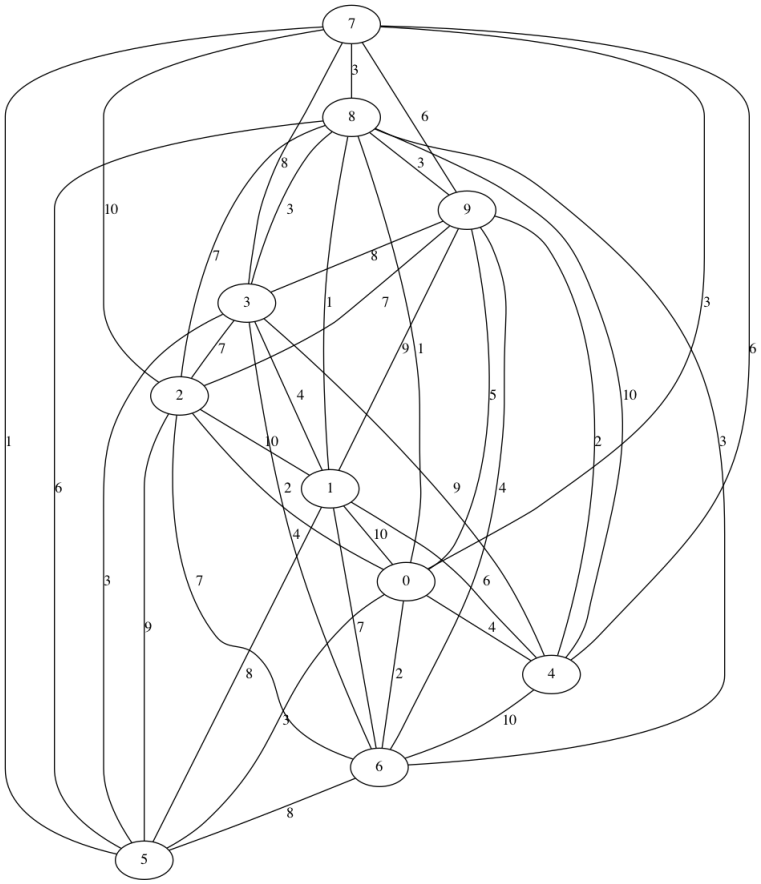

Figura 4.2: Grafo no dirigido de 10 nodos

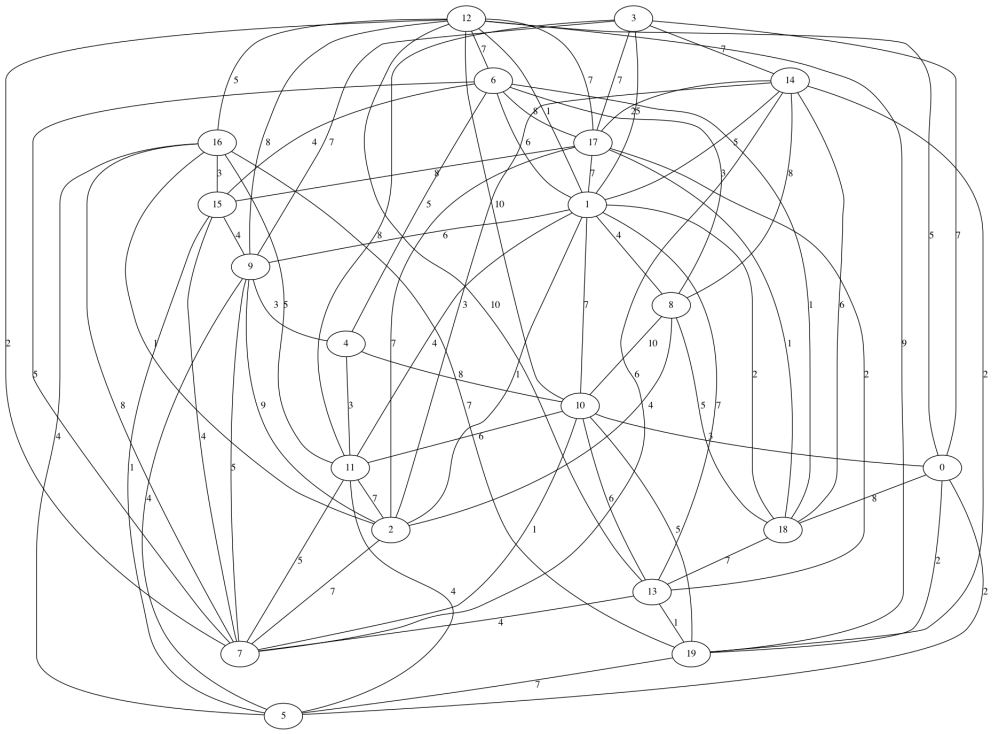

Figura 4.3: Grafo no dirigido de 20 nodos

Bibliografia

[1] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970. 3, 4

[2] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998. 3

[3] C.M. Fiduccia and R.M. Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference, Las Vegas, NV, USA, 14-16 June, 1982*. IEEE, 1982. 4