



Formalización del problema de la partición del grafo

Laura Rodríguez Navas
rodrigueznava@posgrado.uimp.es

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

Resumen	1
1 Introducción a la Partición de Grafos	3
1.1 Introducción	3
1.2 Descripción el problema	3
2 Algoritmos basados en metaheurísticas	4
2.1 Kernighan-Lin	4
2.1.1 Descripción	4
2.1.2 Ejemplo	5
2.2 Spectral Bisection	5
2.2.1 Descripción	5
2.2.2 Ejemplo	5
2.3 Multilevel Spectral Bisection	6
2.3.1 Descripción	6
2.3.2 Ejemplo	6
3 Comparativa entre los diferentes algoritmos	8
4 Conclusiones	10
Bibliografía	11

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), Spectral 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 sería 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

Para la codificación del algoritmo se ha utilizado la librería de Python: `NetworkX`. El algoritmo divide un grafo en dos bloques usando el algoritmo Kernighan-Lin[1]. Es decir, divide un grafo en dos conjuntos intercambiando iterativamente pares de nodos para reducir el peso de las aristas entre los dos conjuntos.

El grafo aleatorio de ejemplo es:

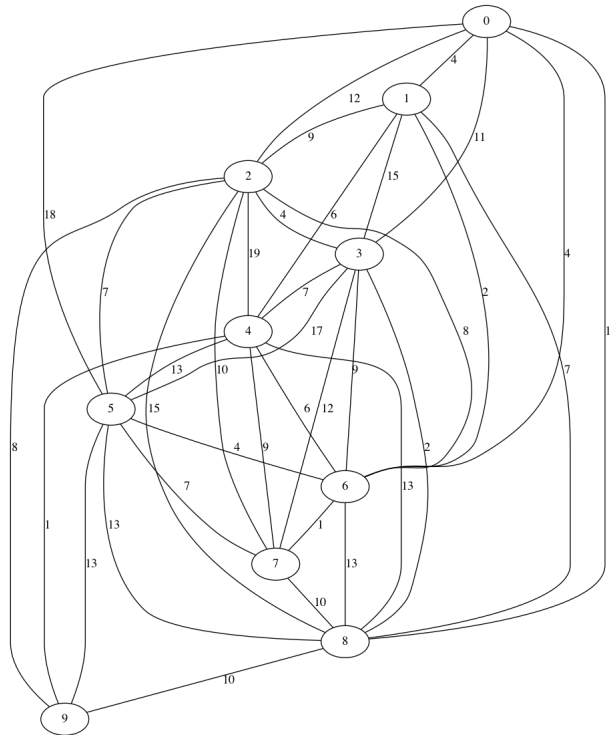


Figura 2.1: Grafo no dirigido de 10 nodos.

Después de la partición inicial obtenemos los conjuntos: $A = \{3, 4, 6, 8, 9\}$ y $B = \{0, 1, 2, 5, 7\}$. Los conjuntos tienen el mismo tamaño. Están ordenados.

2.2 Spectral Bisection

2.2.1 Descripción

2.2.2 Ejemplo

El grafo de ejemplo inicialmente,

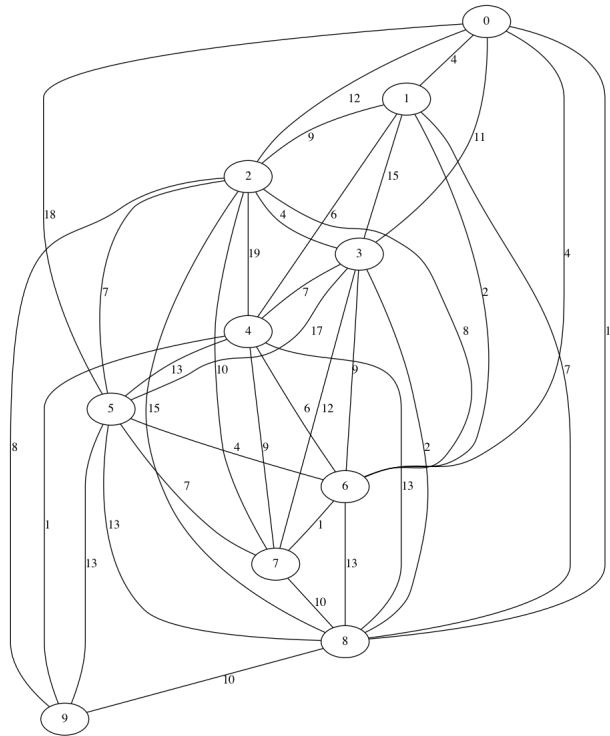


Figura 2.2: Grafo no dirigido de 10 nodos.

Después de la partición obtenemos los conjuntos: $A = \{1, 2, 3, 4, 6, 9\}$ y $B = \{0, 8, 5, 7\}$.

Los conjuntos tienen no son del mismo tamaño. No están ordenados.

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 subsección a continuación contienen una explicación más extensa de estas diversas fases.

2.3.1 Descripción

2.3.2 Ejemplo

El grafo de ejemplo inicialmente, vemos que sigue siendo el mismo que en el primer ejemplo (ver Figura 2.2).

Después de la partición obtenemos los conjuntos: $A = \{0, 1, 3, 7, 8\}$ y $B = \{2, 4, 5, 6, 9\}$.

Los conjuntos tienen el mismo tamaño. Están ordenados.

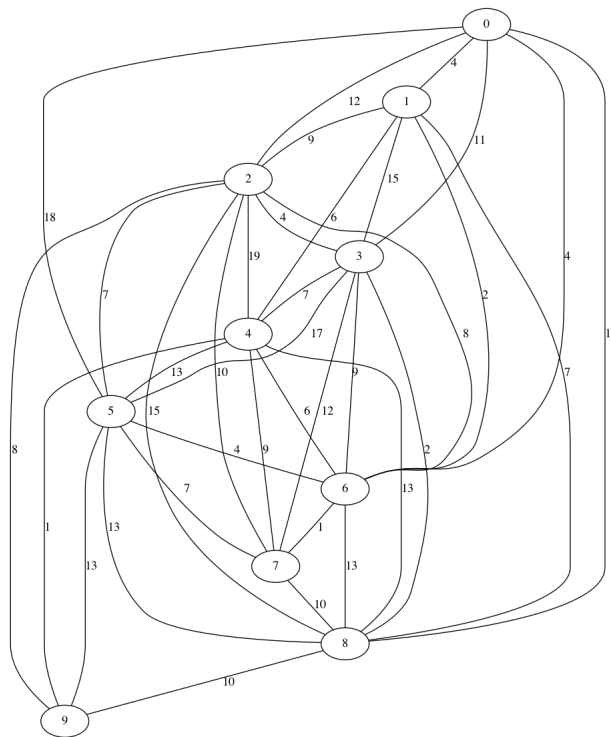


Figura 2.3: Grafo no dirigido de 10 nodos.

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, también conocidos como grafos de Erdős y Renyi o grafos binomiales. Pero, en primer lugar, definimos el concepto de grafo aleatorio.

Se dice que un grafo es aleatorio si la presencia y colocación de sus aristas siguen una distribución aleatoria. Por tanto, un grafo aleatorio no puede diseñarse con ningún criterio concreto. La noción del modelo de un grafo aleatorio incluye el límite entre cada par de nodos con igual probabilidad, independientemente de los extremos. Concretamente del modelo se denomina: modelo de Erdős y Renyi. El nombre del modelo proviene de los matemáticos Paul Erdős y Alfréd Rényi[4], quienes lo presentaron por primera en 1959.

El modelo Erdős y Renyi (a veces nombrado en la literatura abreviado como modelo ER), es el método que se ha empleado en la generación de los grafos aleatorios. Para ello, un grafo se construye mediante la conexión de los nodos al azar. Cada arista se incluye en el grafo con una probabilidad de p independiente de las otras aristas del grafo. De manera equivalente, todos los grafos con n nodos y m aristas tienen la misma probabilidad de:

$$p^m(1-p)^{\binom{n}{2}-m}$$

Para la implementación de los grafos aleatorios se ha utilizado la librería de Python: NetworkX. Concretamente el método que se ha utilizado nos devuelve un grafo no dirigido, donde se elige cada una de las aristas posibles con probabilidad $p = 0.7$, según el número de nodos de entrada n . En particular, el caso $p = 0.7$ se corresponde con el caso en el que todos los grafos en n vértices se eligen con mayor probabilidad. Los pesos de las aristas también se han generado aleatoriamente dentro de un intervalo de pesos de 1 a 20.

Podemos observar la codificación utilizada a continuación, que se ejecuta en tiempo $O(n^2)$.

```
G = nx.erdos_renyi_graph(n, 0.7)

for u, v in G.edges():
    if u != v:
        G[u][v]['label'] = random.randrange(1, 20)
```

Finalmente, en la tabla siguiente, se presenta una comparación entre los algoritmos codificados sobre los grafos aleatorios de este trabajo en términos de tiempo computacional. Se muestra el tiempo (en segundos) en completar ejecuciones sobre conjuntos de grafos aleatorios para distintos números de vértices, n . Para ello se ha utilizado un MacBook Pro, con un procesador de cuatro núcleos de 2.8 GHz y con 16 GB de memoria RAM. Es de esperar que, a mayor número de vértices, los algoritmos tengan mayor tiempo de ejecución.

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 cuando el número de vértices n es más pequeño. Fenómeno que cambia cuando casi se duplican el número de vértices.

Si nos fijamos con el algoritmo Kernighan-Lin, cuando el número de vértices inicialmente casi se dobla, el tiempo de ejecución también lo hace. Parece que el algoritmo va aumentando el tiempo de ejecución a medida que aumenta el número de vértices de manera proporcionada.

En cambio, el algoritmo Spectral Bisection obtiene unos resultados muy parecidos con Multilevel Spectral Bisection, pero cuando el número de vértices empieza a ser muy elevado, la eficiencia baja drásticamente. Que no es el caso del algoritmo Multilevel Spectral Bisection.

Como conclusión global a la comparación entre los algoritmos podemos decir que el algoritmo Multilevel Spectral Bisection es el más eficiente que se ha codificado, y el algoritmo Kernighan-Lin es el menos eficiente.

4. Conclusiones

Hemos utilizado el mismo grafo aleatorio para los tres ejemplos descritos de las codificaciones de los algoritmos.

El algoritmo Multilevel Spectral Bisection es el más eficiente que se ha codificado, y el algoritmo Kernighan-Lin es el menos eficiente.

Para la comparativa solo se ha tenido en cuenta el tiempo de ejecución de los algoritmos. No se ha añadido el tiempo de ejecución empleado en la creación de los grafos, antes y después de la partición.

Cada vez que ejecutamos un algoritmo las soluciones cambian. Así que por eso, los hemos comparado por tiempo de ejecución. Ya que al ser algoritmos metaheurísticos, todas las soluciones pueden ser óptimas y válidas. Eso quiere decir, que aunque un algoritmo sea más eficiente, no obtendrá la solución más óptima.

Bibliografía

- [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, 5
- [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
- [4] P. Erdős and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959. 8