Resolución de problemas
con metaheurísticos

The Graph Partitioning Problem

Laura Rodríguez Navas

March 2019

Abstract

In mathematics, a graph partition is the reduction of a graph to a smaller graph by partitioning its nodes into mutually exclusive groups. It is commonly used in scientific computing, VLSI circuit design, and task scheduling in multiprocessor computers, among others. Actually, finding partitions that simplifies graph analysis is a hard problem and the graph partitioning problem has gained importance due to its application for clustering and detection of cliques in social, pathological and biological networks.

Since graph partitioning is a hard problem, the practical solutions are based on heuristics, because typically graph partition problem fall under the category of NP-hard problems. For tackling this problem, I propose three algorithms based on heuristics, including local and global strategies. The two first algorithms presented are Kernighan–Lin and Fiduccia-Mattheyses, effective 2-way cuts by local search strategies. The third is the Spectral Bisection algorithm. Finally, I demonstrate with some examples my approach in an evaluation of these algorithms using a dataset.

Table of Contents

1. The problem

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 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 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 interconexión era muy elevado por el que el número de interconexiones debía ser minimizado.

La aplicación descrita anteriormente fue presentada en [1], debido a Kernighan y Lin (1970), en la cuan 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.

Este problema ha sido demostrado como un problema NP-completo (`https://es.wikipedia.org/wiki/NP-hard`), lo que implica que las soluciones para él no pueden ser encontradas en tiempos razonables.

En el presente trabajo se pretende estudiar el algoritmo descrito en [1], [2] i

1.3 Partición de grafos

2. Algorithms

2.1 Kernighan-Lin

The Kernigan-Lin algorithm[1], often abbreviated as K/L, is one of the earliest graph partitioning algorithm and was originally developed to optimize the placement of electronic circuits onto printed circuit cards so as to minimize the number of connections between cards.

The K/L algorithm does not create partitions but rather improves them iteratively (there is no need for the partitions to be of equal size). The original idea was to take a random partition and apply Kernigan-Lin to it. This would be repeated several times and the best result chosen. While small graphs this delivers reasonable results, it is quite inefficient for problem sizes.

```
from graph import Graph

def switch(graph, A, B):
        D = computeD(graph, A, B)
        costs = []
        X = []
        Y = []

        for i in range(int(graph.getSize() / 2)):
                x, y, cost = maxSwitchCostNodes(graph, A, B, D)
                A.remove(x)
                B.remove(y)

                costs.append(cost)
                X.append(x)
                Y.append(y)

                D = updateD(graph, A, B, D, x, y)

        maxCost, k = getMaxCostAndIndex(costs)

        if maxCost > 0:
                A = Y[:k + 1] + X[k + 1:]
                B = X[:k + 1] + Y[k + 1:]
                return A, B, False
        else:
                A = [i for i in X]
                B = [i for i in Y]
                return A, B, True

def k_lin():
        graph = createGraph()
        A = [i for i in range(int(graph.getSize() / 2))]
        B = [i for i in range(int(graph.getSize() / 2), graph.getSize())]
        done = False

        while not done:
                A, B, done = switch(graph, A, B)

        print("Partition A: ", end=' ')
                for i in A:
                print(graph.getNodeLabel(i), end=' ')
```

```
print("\nPartition B: ", end=' ')
        for i in B:
        print(graph.getNodeLabel(i), end=' ')
```

2.1.1 Example

2.2 Fiduccia-Mattheyses

2.3 Spectral Bisection

3. Experimental set-up

Text

4. Results and discussion

Text

5. Conclusion

Text

Bibliography

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

[2] 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. 1

A. Appendix

In this file (Appendices/Appendix_A.tex) you can add appendix chapters, just as you did in the Document.tex file for the 'normal' chapters. You can also choose to include everything in this single file, whatever you prefer.