Graph Coloring Problem with Linear Optimization

Nguyen Bao Dung, 22028017

***Abstract*—Graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices are of the same color. In this paper, a solution purposed to solve the problem is called linear optimization. The solution is ran across multiple graph with different number of weights and edges to compare the algorithm strength.**

I. Introduction

In this introduction section, both vertex coloring problem and linear optimization problem will be discussed. After this section, the approach to the first problem using the second problem solution is proposed. Then propose the setup and experiment results on different graphs with different number of nodes and edges.

A. Vertex coloring problem

The vertex coloring problem remains one of the NP-complete tasks that we're trying to optimize. The problem is stated as follow: Given a graph with N nodes and E edges, state a way to color the vertices such that no 2 nodes connected by an edge is given the same color. And the number of unique colors being used is minimized.

B. Linear optimization

The linear optimization problem is stated as an the optimization of a linear objective function subject to one or many linear constraints. In formula, we can express the problem as follow:

$$\begin{array}{lll} \text{minimize} & f(x) & x \in R^n \\ \text{subject to} & g(x) \leq 0 & \\ & h(x) = 0 & \end{array} \tag{1}$$

with x is an n variables vector, and $f(x), g(x), h(x)$ all being linear functions of x, or $f(x) = c^T x$ with c being the coefficient vector.

II. Objectives

A. Problem statement

Our objective is, if the first problem is an NP-complete task, however, the second problem is solvable in linear complexity time. How can we formulate the concept of the first problem into a solvable problem as stated in the second problem?. First, in linear optimization, we have two main parts, the objective function and the constraints. While in the first problem, we have the objective as the minimized number of unique colors being used, and the constraint being no adjacent vertices could have the same color.

B. Approach

Suppose we have N nodes and E edges in the graph. When formulating the first problem, we have to answer three main questions: what are variables, the constraints on them, the objective function to minimize. identify variables, constraints, objective functions. Conventionally and intuitively, we perform these tasks in sequential order.

1) Variables: Denote X as the N matrix variables and w as a vector of N variables where

$$X_{u,i} = \begin{cases} 1 & \text{if node u has color i} \\ 0 & \text{otherwise} \end{cases}$$

and

$$w_i = \begin{cases} 1 & \text{if color i is used} \\ 0 & \text{otherwise} \end{cases}$$

2) Constraints:

1) One node has only one color: $\sum_N^{i=1} X[u,i] = 1$
2) Two adjacent nodes (u, v) having different color:

$$X[u,i] + X[v,i] \leq w_i$$

3) Objective function: Minimizing the number of unique color being used is synonymous with

$$\text{minimize} \quad \sum_{i=1}^{N} w_i$$

III. Experiments and Results

A. Setup

The tool being used in the experiment is ORtools, with GLOP being the solver. The experiment is run on a selection of 7 graph with different architecture, different nodes and edges being: Running this program on a selection of different graphs provides the following results:

- Peterson Graph
- One Line Path
- Crown Graph
- Simple Graph
- Mycielski Graph

B. Results and Conclusion

- 10 nodes, 15 edges Peterson Graph – Run Time: 0.2568 s
-
- 70 Vertices One Line Graph–Run Time: 0.6353 s
- 140 Vertices One Line Graph–Run Time: 2.2397 s
- 200 Vertices One Line Path Graph–Run Time: 8.9123 s
- 60 Vertices, 3600 edges Crown Graph – Run Time: 19.1123 s

- 5 nodes, 10 edges Simple Graph
- 6 nodes, 15 edges Simple Graph
- Mycielski Graph of Order 6 – Run Time: 338.1341 s

As one can see, within the similar graph architecture, increasing the number of vertices and edges does increase the run time for graph. However, having a lot of vertices and edges does not inherently make the program run longer. The longest running graph has less vertices and edges than the 60 vertices crown graph but due to more complicated structures of the graph itself, it caused the program to take over 5 minutes to find the solution.

REFERENCES

[1] Wikipedia, *Graph Coloring*
[2] Wikipedia, *Linear Optimization*