An Integer Program Approach to Generating Proper Colorings of Non-Planar Graphs

Peter Hoffman, Amy (Jia Yuan) Hu, Shivani Konduru, Allison Li

April 28, 2022

Abstract

We present a method to generate proper colorings of non-planar graphs using an integer program driven approach in which we place restrictions on the maximum number of vertices each color can be assigned to. We first provide a mathematical overview of non-planar graphs and their colorings. We then design an integer program to generate a proper coloring of a graph and apply our method to a collection of randomly generated non-planar graphs. We subsequently design an algorithm to efficiently find a lower bound on the chromatic number of a non-planar graph. We then use our method to generate optimal bus stop groupings in which we seek to minimize the number of buses needed to serve a predefined collections of bus stops. We conclude by observing the properties of this resulting coloring and its real world implications.

Keywords: Integer Programming, Non-Planar Graph, Proper Coloring of Non-Planar Graph, Vertex Coloring Problem, Random Graph

Contents

1 Introduction

We begin by providing relevant background on the topics we subsequently study. Towards this goal, we first discuss non-planar graphs, highlighting their many applications. We then introduce the concept of a proper coloring and its applications. Finally, we explain the important role that integer programming plays in graph theory overall.

A graph is said to be non-planar if it cannot be drawn in a plane so that no edges cross. Non-planar graphs, like many objects in the field of graph theory, have many applications in network theory and combinatorial optimization [1]. In addition, non-planar graphs are important for the study of physical systems such as road networks, telephone lines, and biological pathways [2] [3]. Thus, by understanding the properties of non-planar graphs, we better understand the many theoretical and physical phenomena that they model.

One interesting question concerns proper colorings of a graph's vertices. A proper coloring of a graph is an assignment of vertices to colors such that no two adjacent vertices are assigned the same color. The smallest number of colors needed to form a proper coloring for a given graph is known as the chromatic number of that graph, typically denoted as χ. In this paper, we focus on the task of generating proper colorings of non-planar graphs.

Proper colorings have a surprising number of physical applications. For example, consider the task of assigning radio frequencies to radio stations [4] [5]. To avoid interference between the stations, no two stations which broadcast to the same area can be assigned the same frequency. We can formulate this problem as a graph in which each station is a vertex and two vertices share an edge if and only if the stations broadcast to the same area. We color the vertices with frequencies so that a proper coloring of this graph would ensure no two stations interfere with each other. Additionally, the chromatic number would be the smallest number of frequencies needed, which might be of additional logistical or monetary interest. For the curious reader, see [7] for a more thorough survey of this area.

However, so far only relatively small vertex coloring instances can be solved to provable optimality by exact algorithms [4]. This is in contrast to other classical combinatorial optimization problems such as the Travelling Salesman Problem, where large instances can be solved to optimality.

Lastly, we would like to highlight the multiple contributions that integer programming has made to classical combinatorial optimization problems in recent decades. Due to these questions typically being NP-hard, integer programming tends to be more efficient than classical algorithms [8]. More specifically, integer programming often provides a tractable way to prove existence of certain properties of graphs, such as with bipartite matching, shortest paths, and colorings [4] [6]. Furthermore, integer programs have also been used to find, not just prove, these properties. While we discuss some of the specific accomplishments of integer programming with respect to graph theory in greater detail in Section 2.2, we hope we have highlighted the existing precedent of using integer programming in the study of graph theory.

2 Graphs and their properties

In this section we define basic mathematical notation that will facilitate a more technical discussion to follow. The purpose of this is twofold: first to make our work more accessible to those unfamiliar with graph theory, and secondly to eliminate any confusion with conflicting notation encountered elsewhere for readers familiar with graph theory.

2.1 Graph notation

We begin by formally defining what it means for a graph to be simple, bipartite, and planar.

Definition 1. *(Simple graph) A simple graph is an unweighted, undirected graph containing no graph loops or multiple edges. In other words, there exists at most one edge between any two unique vertices, and zero edges between a vertex and itself.*

We next introduce the notion of a bipartite graph. As will be described afterwards, a certain class of bipartite graphs is known as a non-planar graph.

Definition 2. *(Bipartite graph) A graph $G = (V, E)$ is said to be bipartite if there exists two disjoint sets A, B such that $V = A \dot{\cup} B$ and each edge of G is adjacent to exactly one vertex in A and exactly one vertex in B.*

We now turn our attention to the idea of a planar graph.

Definition 3. *(Planar graph) A graph is said to be planar if it can be can be drawn in the plane so that no edges cross each other.*

Planar graphs are well studied objects in discrete mathematics, and the curious reader may find interest adjacent topics such as Euler's formula. Furthermore, we will now see in Section 2.2 that planar graphs have a special property with regards to graph coloring.

2.2 Graphs colorings

An important area of study within graph theory, and the focus of this paper, surrounds the problem of proper colorings of graphs. We first define a proper coloring and then describe a special property of planar graph colorings.

Definition 4. *(Graph coloring) Let $G = (V, E)$ be a graph and $C = \{c_1, c_2, \ldots\}$ be a set of colors. Then a coloring G is a function $f : V \to C$, in other words a function that assigns every vertex $v \in V$ a color $c \in C$. A proper coloring is a coloring f such that no two vertices that share an edge are mapped to the same color.*

An tangential concept is the notion of a chromatic number of a graph.

Definition 5. *(Chromatic number) The chromatic number of a graph G, denoted by $\chi(G)$, is the smallest integer k for which a proper coloring of G exists on C such that $|C| = k$.*

We will now state a famous result on the chromatic number of every planar graph.

Theorem 1. *For every simple planar graph G, its chromatic number $\chi(G) \leq 4$.*

As this is a paper on integer programming and not on graph theory, we omit a proof of this result. However, a curious reader may find additional interest in the "proof by computer" technique that led to this result.

2.3 Properties of non-planar graphs

As previously alluded to, many graphs that arise in physical applications are non-planar by virtue of their large size, connectedness, and randomness. Therefore, while the four color theorem is an important result in graph theory, it brings less value to the physical modeling that we are interested in.

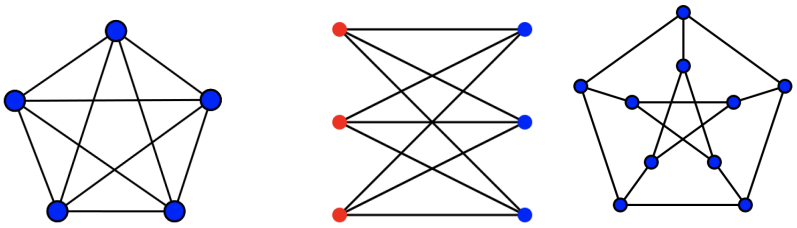
Before describing our integer program to generate proper colorings of non-planar graphs in Section 3, we describe two non-planar graphs that will be important in the formulation of our integer program.

Theorem 2. *(Kuratowski's Theorem) A graph is non-planar if and only if it contains a subgraph that is edge-equivalent to K_5 or $K_{3,3}$.*

In other words, given a graph G, if one is able to "find" either the complete graph on five vertices K_5 or the complete bipartite graph on six vertices $K_{3,3}$, then G is non-planar. See Figure 1 below for depictions of K_5, $K_{3,3}$, and the Petersen graph, which is non-planar by Kuratowski's Theorem.

3 IP-generated colorings of non-planar graphs

We now introduce our IP-driven approach to finding colorings of non-planar graphs in which we place a restriction on the maximum number of vertices each color can be assigned to. The motivation for this additional constraint will become clear in the subsequent discussion of the real-world application.



(a) K_5 (b) $K_{3,3}$ (c) Petersen graph

Figure 1: Non-planar graphs

In the sections below, we lay out the formulation of the problem, decision variables, and data generation.

3.1 Formulation of IP

This subsection contains a mathematical discussion of our integer program model. Let us begin by defining the following:

- Let $V = \{1, \ldots, n\}$ be a set of vertices, with $v, w \in V$
- Let $E = \{(v, w)|$ if v is adjacent to w $\}$.
- Let C be a set of colors, $C = \{c_1, \ldots, c_n\}$.
- Let $y \in \mathbb{N}$ be the maximum number of vertices that can be assigned to any single color.
- Define binary variables $x_{v,c_i} = 1$ if vertex v is colored c_i, and 0 otherwise.
- Define the binary variables $k_{c_i} = 1$ if color c_i is used at least once, and 0 otherwise.

We formulate our model as follows:

$$\text{(1) minimize: } k_{c_1} + \ldots + k_{c_n}$$

such that:

$$\text{(2) for all } (v, w) \in E \text{ and } c_i \in C, \quad x_{v,c_i} + x_{w,c_i} \leq 1$$

$$\text{(3) for all } v \in V, \quad \sum_{c_i \in C} x_{v,c_i} = 1$$

$$\text{(4) for every } k_{c_i}, \quad x_{v,c_i} \leq k_{c_i}$$

$$\text{(5) for every } c_i \in C, \quad \sum_{v \in V} x_{v,c_i} \leq y$$

$$\text{(6) } x_{v,c_i} \in \{0, 1\} \text{ for all } v \in V \text{and } c_i \in C$$

$$\text{(7) } k_i \in \{0, 1\} \text{ for all } i \in [|C|]$$

Explanation of model:

1. Minimize the number of colors used
2. No two adjacent vertices are given the same color
3. Each vertex is assigned exactly one color
4. Logical statement that forces $k_{c_i} = 1$ if color c_i is used
5. Each color can be assigned to at most y vertices
6. x_{v,c_i} is binary
7. k_{c_i} is binary

Note that for the sake of efficiency, we let $|C| = |V|$ since the worst case scenario involves every vertex getting a different color, such as in the case of K_5.

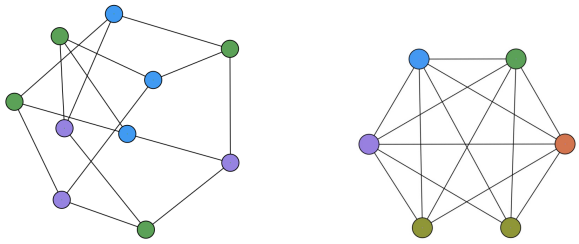
3.2 Data generation

We now create random, non-planar graphs on n vertices to test our above integer program. To guarantee a non-planar graph, we start with a Peterson subgraph or $K_{3,3}$ or the complete bipartite graph $K_{3,3}$.

We then proceed to expand our vertex set V by adding a user-determined number of vertices. Next, we add edges by randomly picking two vertices, checking to see if an edge already exists between them, and adding an edge if not. We then repeat this edge-generating process for a user-determined number of iterations. Please see our corresponding .ipynb file for more detail on this random graph generation.

3.3 Discussion of results

In addition to our IP model and the associated data, we also developed code that depicts graphs and their proper colorings as generated using our integer program. This subsection uses these diagrams to discuss the results of the model in which no restriction was placed on the number of vertices per color (see item 5 in our model above), thus making the problem equivalent to the classical graph coloring problem discussed in Definition 2.2. Figure 2 below gives two such examples.



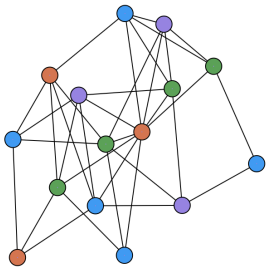
(a) Graph with $\chi = 3$ (b) Graph with $\chi = 5$

Figure 2: Example of coloring solutions

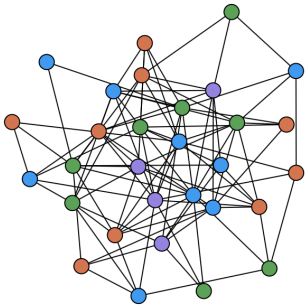
Several surprising results arose out of repeatedly running our IP on randomly generated non-planar graphs. Most notable was that sparse graphs – generally described as graphs with $|V| >> |E|$ – tended to often be 4-colorable, such as the graphs in Figure 3 below.

This trend, albeit on a small sample size, might allow us to draw preliminary conclusions that a sparse graph is likely going to be 4-colorable. More generally, if one ran our model on a large enough sample of data it might be possible to deduce the approximate probability of a graph on v vertices and e edges being k-colorable.

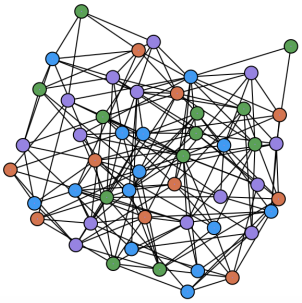
One other surprising result dealt with the run-time of larger, less dense graphs. For graphs in which $|V|$ was large, say ≥ 100, and $|E| \geq 200$, the run-time slowed considerably, often to the point where



(a) $|V| = 15$ and $|E| = 35$



(b) $|V| = 35$ and $|E| = 95$



(c) $|V| = 55$ and $|E| = 175$

Figure 3: Sparse graphs where $\chi = 4$

an optimal solution was not found. However, even for these large graphs the chromatic number was often surprisingly low, such as in the case of the graph in Figure 4 with $|V| = 100$ and $|E| = 200$ where $\chi = 5$. As we present in Section 3.4, we instead develop a classical algorithm to find the chromatic number of a given graph.

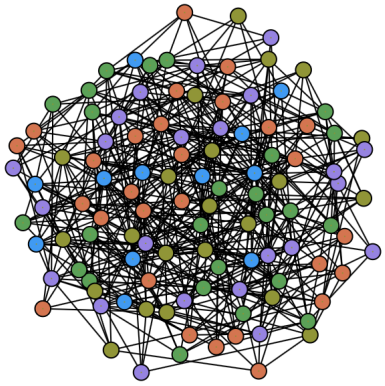


Figure 4: Graph with $|V| = 100$ and $|E| = 200$ where $\chi = 5$

3.4 Chromatic Algorithm

We now develop a classical algorithm to produce a lower bound on the chromatic number χ of a graph. While this algorithm will not produce a solution (i.e. an injective function from vertices to colors) as the integer program does, it will be more efficient at finding a fairly tight lower bound on χ for large, dense graphs.

With this in mind, notice that the existence of a complete subgraph on k vertices on any given graph $G = (V, E)$ implies that the chromatic number $\chi \geq k$. We thus proceed by finding the largest complete

subgraph on k vertices in order to produce a lower bound on χ.

Towards this goal, we iterate through all values of $\chi = k$ where $1 \leq k \leq |V|$, as we clearly need at least 1 color and no more colors than the number of vertices. For each k, any vertex with a degree of at least $k-1$ could potentially be a vertex in a complete subgraph on k vertices. If there exists at least k vertices all with degree of at least $k-1$, we then iterate through all k possible subsets of these vertices to check if a complete subgraph indeed exists. For a given such subset, we check that all $\binom{k}{2}$ edges exist in the edge set E. If this holds, then a complete subgraph on k vertices exists as a subgraph of $G = (V, E)$. If so, we are guaranteed to need at least k colors. We proceed for $1 \leq k \leq |V|$ and return the largest value of k such that the above holds. Using the fact that any graph that is at least k-colorable must also be at least $k-1$-colorable, we perform a binary search on k in order to speed up the algorithm by a factor of $\frac{n}{log_2 n}$, where $n = |V|$ is an upper bound for k since we will never need more colors than vertices. After running our Chromatic Algorithm on the graphs generated in Section 3.2, we notice that the lower bound on χ produced using this method frequently equals the chromatic number produced using our integer program.

4 Real world application

After testing our model on the randomized graphs generated in Section 3.2, we sought to apply it to real world data. This section describes our chosen application and the subsequent results.

4.1 Application and context

One real-world application of non-planar graph coloring concerns a city transportation bureau and their task of assigning buses to bus stops. More formally, consider the task of deciding which bus stops should be visited by which buses where (a) we are seeking to minimize the number of buses needed while (b) ensuring that no two stops that are too far apart are visited by the same bus and (c) that no single bus visits "too" many stops. In other words, we are setting implicit restrictions on the length of each bus route.

We can formulate this problem as follows. Each bus stop is a vertex and two vertices share an edge if the two corresponding bus stops are more than x meters apart. The objective is to minimize the number of colors needed to cover all vertices.

Determining the optimal bus assignment has several tangible benefits. Most notably, an optimal solution would minimize the cost associated with purchasing and maintaining buses, paying for fuel, and hiring drivers. In other words, optimal coloring of the graph helps maximize efficiency of resource use.

4.2 Description of Data

We apply our model to Boston bus stops using Massachusetts Bay Transportation Authority (MBTA) data listing all bus stops and their corresponding latitude and longitude [9].

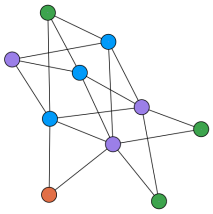
We first filtered the data to extract all bus stops located strictly in Boston. We then sorted the data by latitude and longitude so that it would be easier to work with smaller subsets of the data later on. Our final data set consisted of 1721 Boston bus stops, along with their respective latitude and longitude.

4.3 Methods

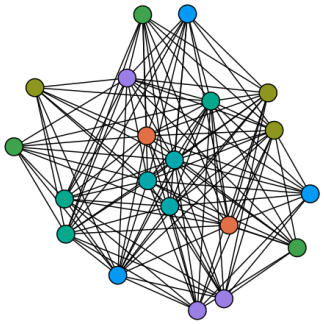
First, we constructed the graph to use in the IP. There is an edge between vertices u and v if the distance between the two bus stops is greater than 400m (chosen because this is the average distance between bus stops in the United States). We then used a brute-force $O(n^2)$ approach to find all such edges, limiting the number of vertices (bus stops) for the sake of the run time of our IP.

4.4 Discussion of results and implications

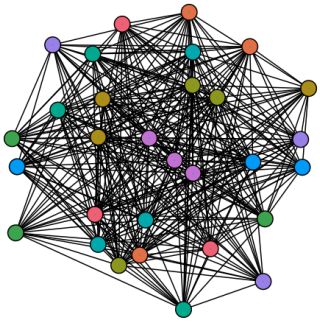
We then used the IP model to solve the graph coloring problem for different-sized subsets of the data up to size 100. Figure 5 shows the colorings produced for graphs of up to 30 vertices. The maximum number of vertices per color was set to 3 in order to demonstrate functionality.



(a) $|V| = 10$, $|E| = 18$, $\chi = 3$



(b) $|V| = 20$, $|E| = 127$, $\chi = 5$



(c) $|V| = 30$, $|E| = 320$, $\chi = 7$

Figure 5: Colorings produced by the IP model where edges are determined based on distance between vertices. The limit for maximum number of vertices assigned to a single color was set to 3.

As we increased the number of bus stops to include, the number of edges in the graph increased as well which in turn increased the chromatic number of each graph. In our graphs, the chromatic number represents how many buses would be needed to cover a certain subset of bus stops in Boston.

Since our IP model limits the max number of vertices to assign to any single color, this directly translates to limiting the number of bus stops assigned to any single bus. This helps us to get a more accurate estimate of the number of buses needed to cover the set of bus stops. This constraint ensures that, in the case where the bus stops happens to be all within close proximity of each other, no single bus is assigned an absurdly large number of stops to cover in a single route. When solving the graph coloring problem for a data set for an entire city, a reasonable number for this limit is 20 vertices. This is a manually computed estimate for the average number of stops on existing bus routes.

There are some limitations to our solutions. Firstly, we were unable to solve the graph coloring problem on the entire set of 1721 bus stops in Boston due to computational limitations. Additionally, it is worth noting that our model does not care about "start" or "end" points of bus routes. Even though all vertices of a certain color can be interpreted as a valid route for a single bus to follow, these routes are unlikely to be optimal because buses all have set starting points and destinations in real life. For example, public buses end up at designated bus garages at the end of the day. Thus, our model is better suited for estimating the number of buses needed for a certain subset of bus stops. For the actual optimal mapping of bus stops to buses, consider utilizing our model's coloring along with additional results from other approaches to get better routing results.

4.5 Sensitivity Analysis

Sensitivity analysis allows us to see how the optimal solution changes if we make slight changes to the problem. More specifically, sensitivity analysis reveals how much the model's cost coefficients or the right-hand side values can be changed without changing the optimal solution. In the graph coloring IP that we formulated above, sensitivity analysis is not very appropriate due to the nature of the model. We cannot alter any of the right-hand side values of the constraints without changing the model into something other than the graph coloring problem that we originally laid out. Additionally, the variable y for the max number of vertices per color cannot be changed as this number is set based on the number of stops that a single bus can feasibly cover in a single day. Decreasing y would only serve to worsen the objective value, while increasing y is not possible in the context of our problem because buses can only visit a certain number of stops within the operating hours in one day.

5 Conclusion

In conclusion, we presented two ways to approach the graph coloring problem on non-planar graphs. The first method involved using a chromatic algorithm to find a lower bound on the chromatic number χ of a graph. In the second approach, we designed an integer program to generate a proper coloring of a non-planar graph.

The IP was applied to the public transit problem where each bus stop is a vertex and two vertices share an edge if the corresponding bus stops are more than x meters (a reasonable value for x would be 400) apart. The "colors" in this context are the specific buses visiting the stops. The goal of our program is to minimize the number of buses needed to cover a given set of bus stops while ensuring no two bus stops that are too far apart from each other are visited by the same bus. In addition, one bus can not visit more than y number of stops (a reasonable value for y would be 20).

After running the MBTA bus stop location data through the integer program, we found that increasing the number of bus stops lead to an increase in the number of edges, causing the number of buses to grow nonlinearly. The chromatic number found by the IP is a good estimate for the number of buses needed to cover a set of bus stops.

While our project focused on the bus stops in Boston, our integer program could be utilized by bus companies beyond Boston to find the optimal number of buses to operate. This would help them minimize the costs associated with maintaining the buses and optimize the efficiency of resource usage.

However, it is worth noting that there are still some limitations to our solution - namely the program would take too long to solve the graph coloring problem on the entire set of 1721 bus stops in Boston. Further study could be conducted on how to make the program faster and more efficient.

References

[1] Annegret Wagler. Combinatorial Optimization: The Interplay of Graph Theory, Integer Programming Illustrated on Network Flow. Large Scale Networks in Engineering and Life Sciences, P. Benner et al. (eds.), Birkhauser, pp.225-262, 2014. hal-02045730

[2] Cardillo, A., Scellato, S., Latora, V., amp; Porta, S. (2006). Structural properties of planar graphs of urban street patterns. Physical Review E, 73(6). https://doi.org/10.1103/physreve.73.066107

[3] Masucci, A., Smith, D., Crooks, A. et al. Random planar graphs and the London street network. Eur. Phys. J. B 71, 259–271 (2009). https://doi.org/10.1140/epjb/e2009-00290-4

[4] Duff, M., amp; Robinson, R. (n.d.). An integer linear programming approach to graph coloring. An Integer Linear Programming Approach to Graph Coloring - CU Denver Optimization Student Wiki. Retrieved March 13, 2022, from http://math.ucdenver.edu/~sborgwardt/wiki/index.php/An_Integer_Linear_Programming_Approach_to_Graph_Coloring

[5] Bóna, M., amp; Stanley, R. P. (2017). A walk through combinatorics: An introduction to enumeration and graph theory. World Scientific.

[6] Jabrayilov, A., amp; Mutzel, P. (n.d.). New integer linear programming models for the vertex coloring problem. New Integer Linear Programming Models for the Vertex Coloring Problem –nbsp;arXiv Vanity. Retrieved March 13, 2022, from https://www.arxiv-vanity.com/papers/1706.10191/

[7] E. Malaguti and P. Toth. A survey on vertex coloring problems. International Transactions in Operational Research, 17:1–34, 2010. doi:10.1111/j.1475-3995.2009.00696.x.

[8] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA, USA, 1979.

[9] Massachusetts Bay Transportation Authority. Dataset of PATI (Plan for Accessible Transit Infrastructure) Bus Stops. Data retrieved from MBTA Open Data Portal, https://mbta-massdot.opendata.arcgis.com/maps/MassDOT::pati-bus-stops/about