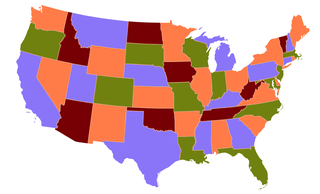
University of Minnesota
MORRIS

Vertex Coloring and Applications

Shawn Seymour
Advisor: Dr. Peh Ng



Abstract

Consider the map of the 48 contiguous states in the USA, and suppose we want to color each state so that no two states that share a boundary have the same color. In general, we could represent every state with a vertex and draw an edge between two states that share a border. This problem can be modeled by a mathematical structure called a graph. A graph, denoted G = (V, E), is a set of vertices V and a set of edges E. The Vertex Coloring problem on G aims to find the minimum number of colors (the chromatic number) needed to color the vertices such that no two adjacent vertices have the same color. Vertex coloring can solve real-world problems such as finding the minimum number of time slots to schedule a final exam period so that no two courses (taken by the same student) are scheduled at the same final exam time slot. In general, there is no known efficient time algorithm to find the chromatic number of a graph and there will likely not be one. This class of problem is known in computer science as NP-hard. Hence, there is interest in finding heuristics or approximation algorithms to find the chromatic number. In this research, we present three heuristics used to find good approximate vertex colorings even though they may not give us the optimal minimum coloring of a graph. This is important as it allows us to approximately solve complex problems in minutes rather than hours. We present computational results comparing the efficiency (time and quality) of these heuristics.

Introduction & Background

A simple graph $G = (V, E)$ is an undirected graph containing no loops or multiple edges. For our purpose, we assume all graphs are simple. An edge, denoted $(v, u) \in E$, connects vertex v to vertex u where $v, u \in V$. Two vertices are said to be *adjacent* if they are connected by an edge. The *degree* of a vertex $v \in V$, denoted $d_G(v)$, is the number of edges incident to v. For example, $d_{G_1}(2) = 3$. The *saturation degree* of a vertex $v \in V$ is the number of different colors used for vertices adjacent to it.

Figure 1: Example proper vertex colorings of a graph G_1 where $\chi(G_1) = 2$

A *proper vertex coloring* assigns colors to each vertex of G such that no two adjacent vertices share the same color. The *chromatic number* of G, denoted $\chi(G)$, is the minimum number of colors needed to properly color G. The *vertex coloring problem* (VCP) on G is to find $\chi(G)$.

The VCP has been shown by [3] to be NP-hard based on a reduction from 3-SAT, a well-known NP-hard problem. This means the VCP cannot be optimally solved in polynomial-time, so we must use heuristics (approximation algorithms) to efficiently find "good", but non-optimal, solutions.

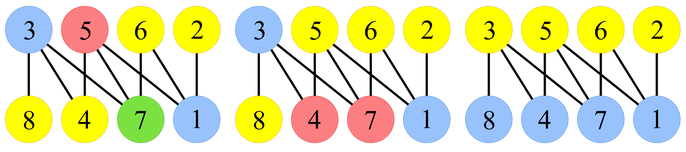


Figure 2: Greedy, Welsh-Powell, and DSATUR vertex colorings of a graph G_2, respectively

Acknowledgements & Information

Special thanks to Peh Ng for her support, feedback, and insightful ideas throughout this project. Thanks to Engin Sungur and Humza Haider for helping with the statistical analysis and data plotting. Source code: `https://devshawn.com/coloring`.

Heuristics

We focused on comparing three heuristics for approximating the vertex coloring problem:

- **Greedy**: Label each vertex in V (i.e. $v_1, v_2, \ldots, v_n$). Iterate through the vertices and assign the smallest available color.
- **Welsh-Powell** [4]: Label each vertex in V (i.e. $v_1, v_2, \ldots, v_n$) such that the vertices are sorted in decreasing order of their degree (i.e. $d_G(v_1) \geq d_G(v_2) \geq \ldots \geq d_G(v_n)$). Iterate through the vertices and assign the smallest available color.
- **DSATUR** [2]: Choose an uncolored vertex $v \in V$ with maximum saturation degree. If there is a tie, choose the vertex with maximum degree. Color v with the smallest available color. Repeat until there are no uncolored vertices.

Methods

The three heuristics were implemented in Java and we ran simulations according to the following:

- Graphs were generated based on the *Erdős–Rényi* model. This model generates a random graph with approximately p percent of edges.
- We focused on graphs with $90\%, 91\%, \ldots, 99\%$ edges. These were the most interesting.
- Simulations were run on 4 sets of graphs: 50 vertices, 100 vertices, 250 vertices, and 500 vertices.
- Each heuristic can produce a different amount of colors used, as shown in Figure 2. Thus, we focused on comparing the quality (number of colors used) and the efficiency (running time) of the three heuristics.

Results

- The heuristics were compared based on the percent difference of four properties: average colors used, minimum colors used, maximum colors used, and running time. Two example comparisons are shown in Figure 3 and Figure 4.

Figure 3: Percent difference of average colors used for 250 vertices

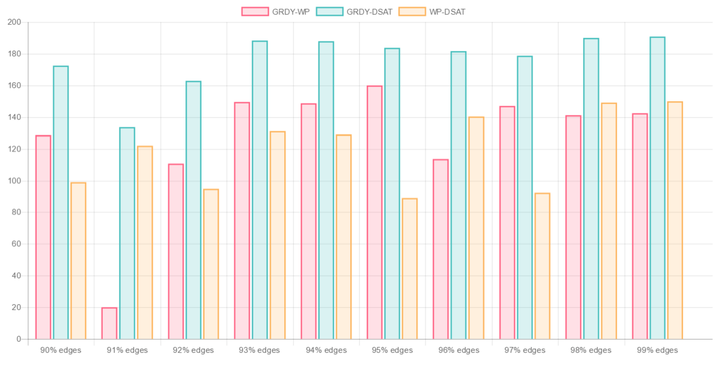


Figure 4: Percent difference of running time for 250 vertices

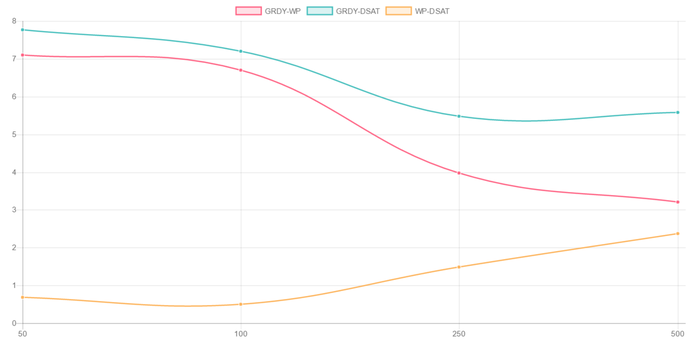


Figure 5: Percent difference of avg. colors used as vertices increase (93% edges)

- As the number of vertices increased, the percent difference between Greedy and Welsh-Powell as well as Greedy and DSATUR decreased. We also noticed that the percent difference between Welsh-Powell and DSATUR increased. This is shown in Figure 5.
- By looking at our comparisons, we saw that there is a *partial order*.
  - Welsh-Powell on average used $4 - 5\%$ less colors than Greedy.
  - DSATUR on average used $6 - 7\%$ less colors than Greedy.
  - DSATUR on average used $1 - 2\%$ less colors than Welsh-Powell.
- This comparison stayed consistent when we looked at both minimum and maximum number of colors used.
- This led us to the following partial order based on colors used:

$$\textbf{Greedy} \leq \textbf{Welsh-Powell} \leq \textbf{DSATUR}$$

- By looking at our running time comparisons, we saw:
  - Welsh-Powell on average was $90 - 110\%$ slower than Greedy.
  - DSATUR on average was $160 - 180\%$ slower than Greedy.
  - DSATUR on average was $80 - 100\%$ slower than Welsh-Powell.

Conclusion

- Although we were not able to compare the heuristics colorings to the optimal coloring, we were able to determine what heuristics gave the closest to optimal colors.
- Based on our results, we would suggest using the Welsh-Powell heuristic on most graphs.
  - We decided that it would be worth 90-100% more time to use 4-5% less colors over Greedy. As most of these simulations ran in seconds, it is worth it.
  - we decided it is *not* worth using another 80-100% more time to only use 1-2% less colors if we use DSATUR over Welsh-Powell.

References

[1] Bondy, J.A. and U.S.R. Murty [1976], *Graph theory with applications*, American Elsevier Publishing, New York, NY.

[2] Brélaz, D. [1979]. *New methods to color the vertices of a graph*. Communications of the ACM, 22(4), pp.251-256.

[3] Sharma, P. C. and Chaudhari, N.S. [2012], *A new reduction from 3-SAT to graph k-colorability for frequency assignment problem*, Int. J. Comp. Applic, 23-27.

[4] Welsh, D. J. and Powell, M. B. [1967], *An upper bound for the chromatic number of a graph and its application to timetabling problems*, The Computer Journal, 10(1), 85-86.