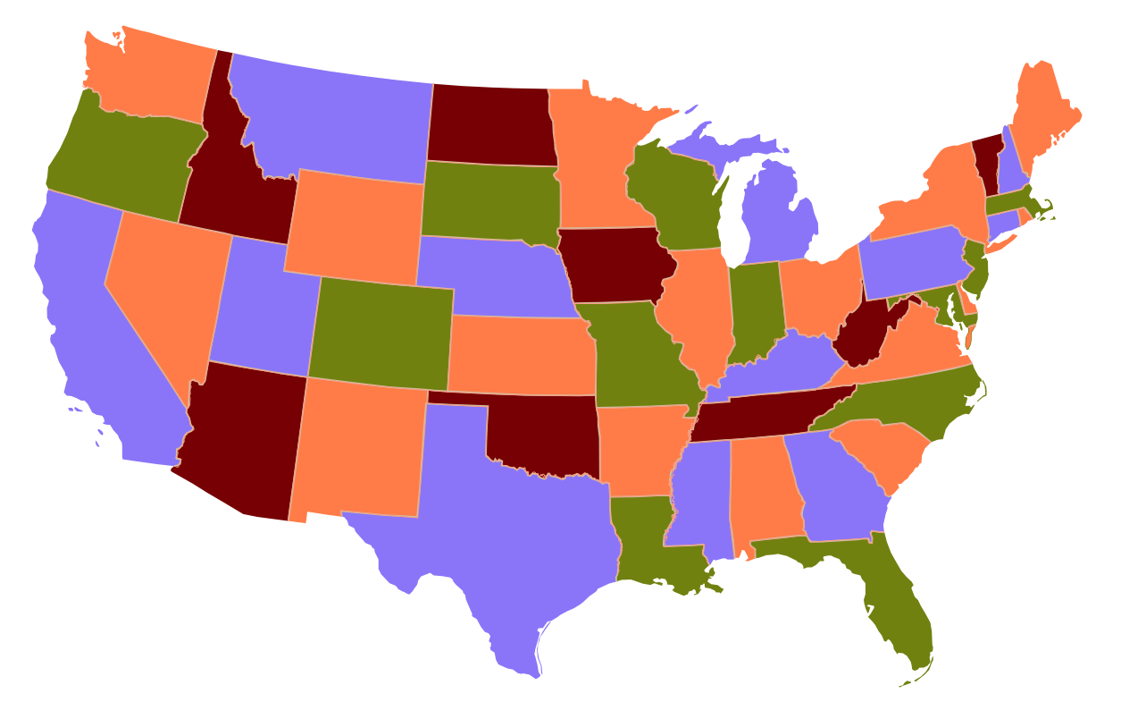
UNIVERSITY OF MINNESOTA
MORRIS

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.

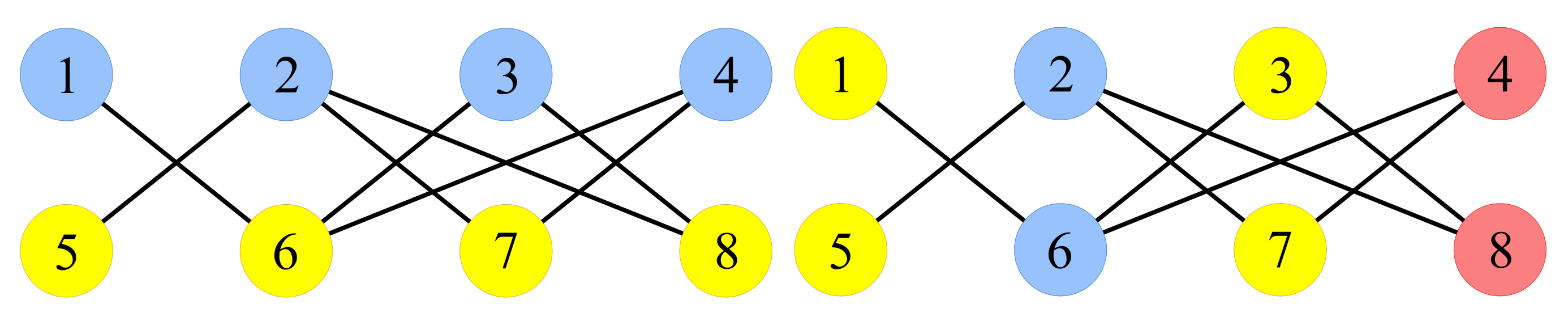


Figure 1: Example proper vertex colorings of a graph G_1 where $\chi(G_1) = 2$

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.

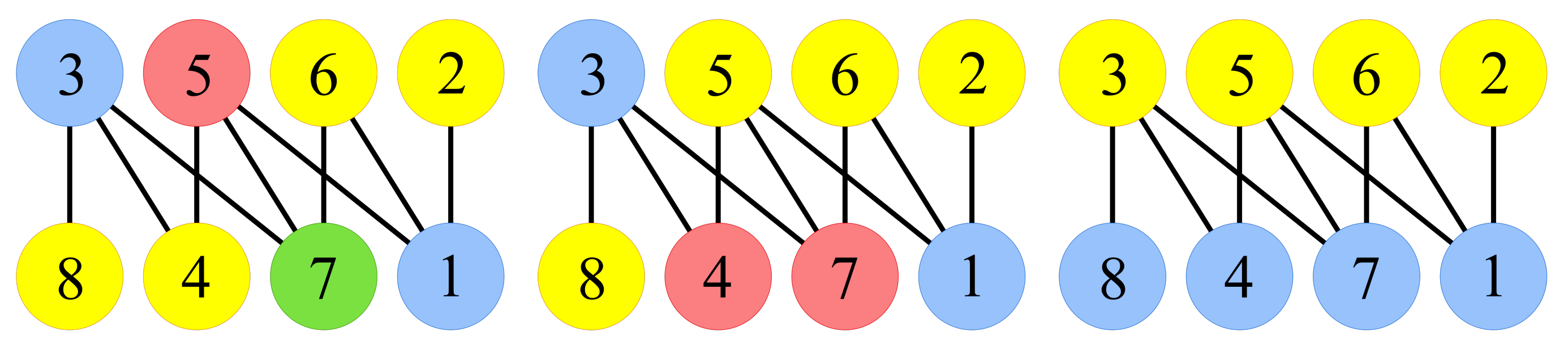


Figure 2: Greedy, Welsh-Powell, and DSATUR vertex colorings of a graph G_2, respectively

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 analyzed graphs with $10\%, 20\%, \ldots, 90\%$ edges (type). This gave us both sparse and dense graphs.
- Simulations were run on graphs with 5 different amount of vertices: 50, 100, 250, 500, 1000.
- 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

We used statistical analysis to compare the number of colors used and the running time of each heuristic based on edge percentage (type) and then also looked at how this changed as the number of vertices changed.

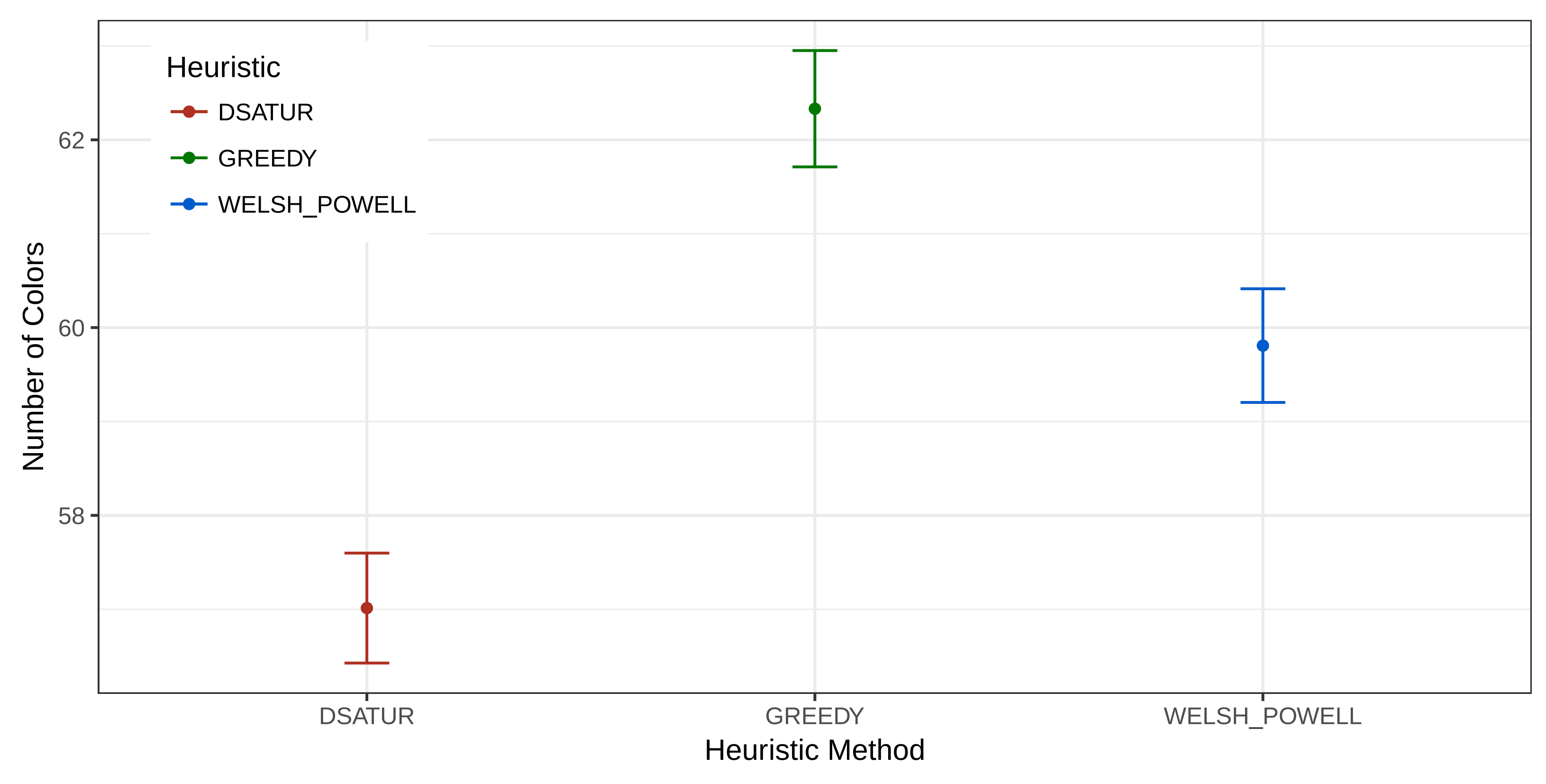


Figure 3: Comparing number of colors used for the three heuristics

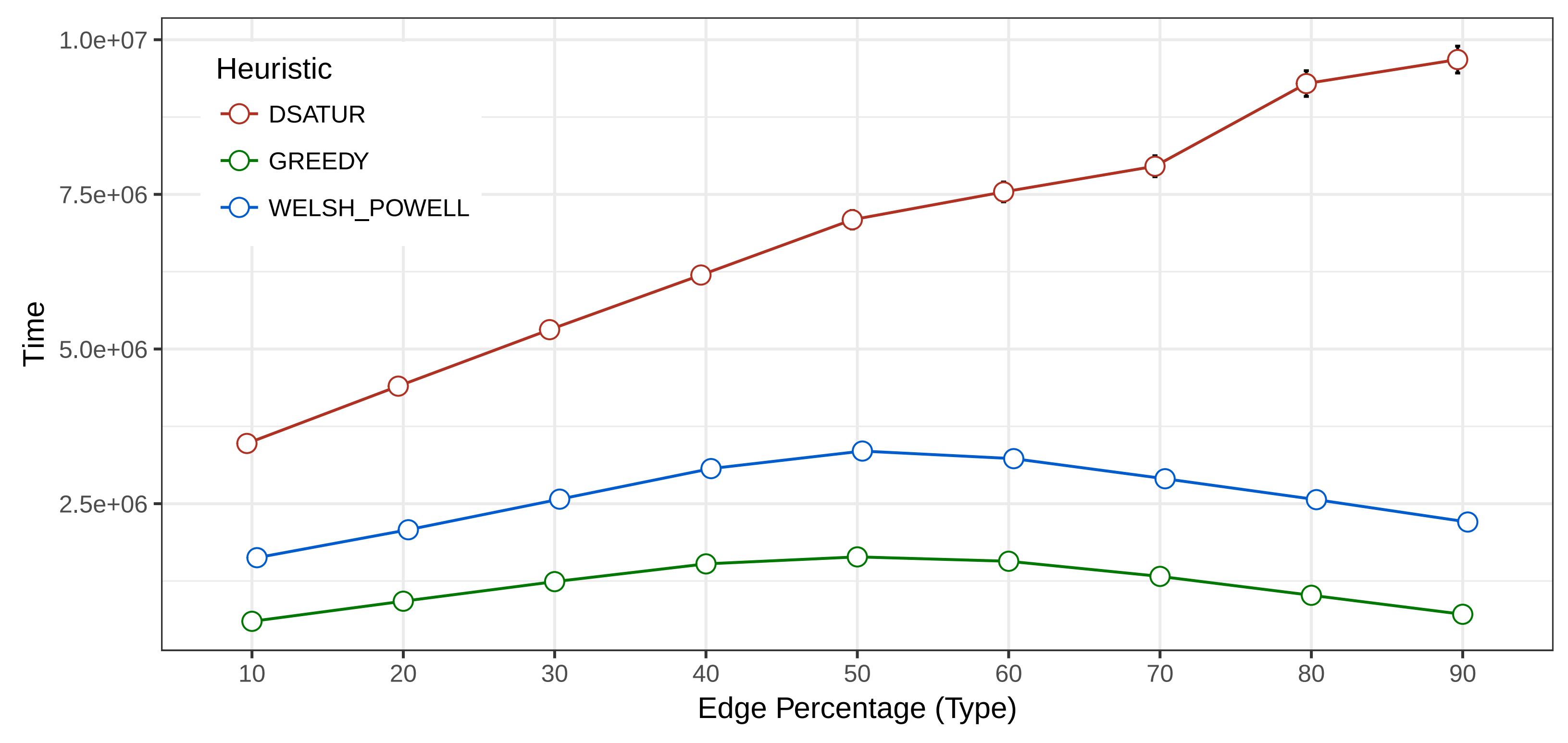


Figure 4: Comparing running time of each heuristic for each edge percentage

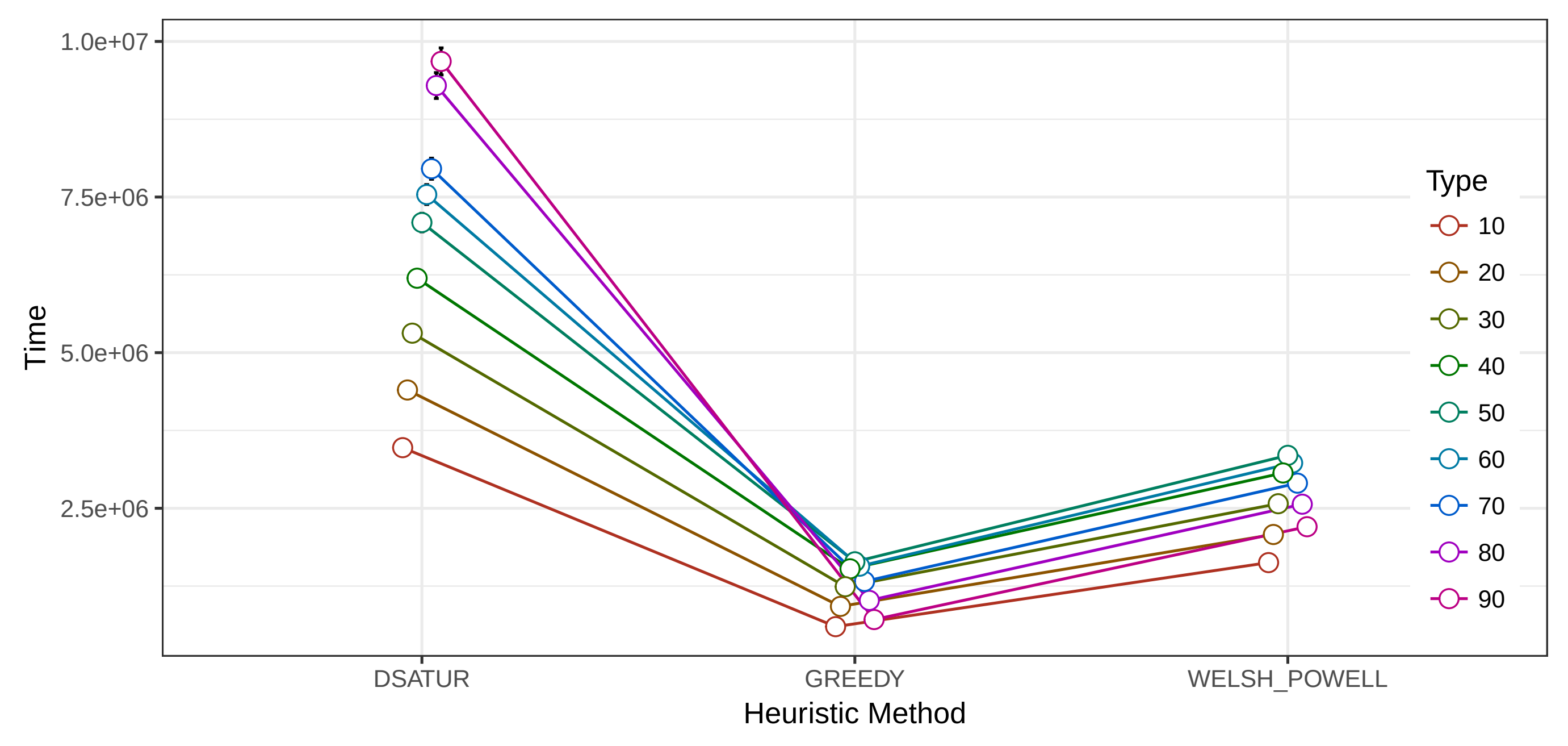


Figure 5: Variance of running time of each heuristic based on edge percentage

Color usage results:
Based on computational data of the amount of colors used, we found that there is a *partial order*. This is shown in Figure 3.

- Greedy uses on average 2.52 colors more than Welsh-Powell.
- Welsh-Powell uses on average 2.79 colors more than DSATUR.

This led us to the following partial order based on colors used:

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

As vertices increased, the partial order defined above stays consistent. DSATUR always used the least colors while Greedy always used the most.

Running time results:
Based on computational data of the running time, we also found a *partial order*, as shown in Figure 4.

- Welsh-Powell is significantly slower than Greedy.
- DSATUR is significantly slower than Welsh-Powell.

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

As the edge percentage increases:

- DSATUR running time increases.
- Greedy and Welsh-Powell running time increases until 50% edges then decreases until 90% edges.
- The difference in running time between DSATUR and Greedy as well as DSATUR and Welsh-Powell grows larger and larger.

Conclusion

Although we were not able to compare the heuristics colorings to the optimal coloring, we were able to determine which heuristics gave the closest to optimal colors. Based on our results, we suggest using **Welsh-Powell** for most graphs. We believe that it is worth the extra time Welsh-Powell takes compared to Greedy but not worth the extra time DSATUR takes compared to 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.