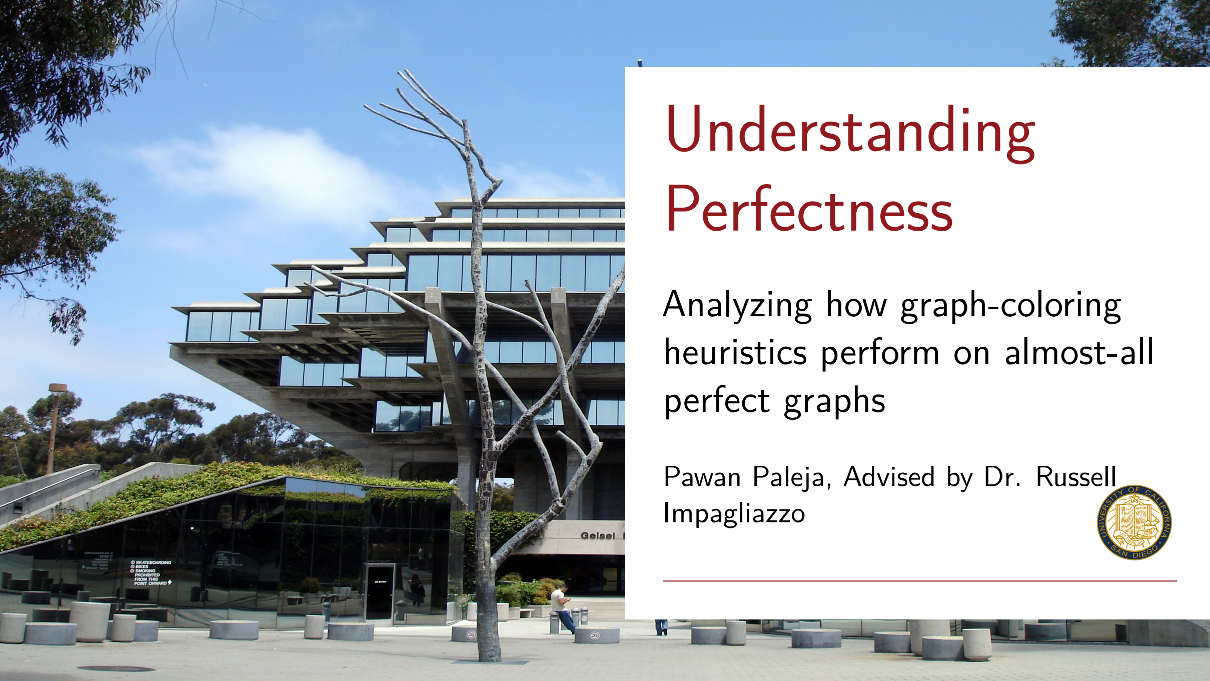


Understanding Perfectness

Analyzing how graph-coloring heuristics perform on almost-all perfect graphs

Pawan Paleja, Advised by Dr. Russell Impagliazzo

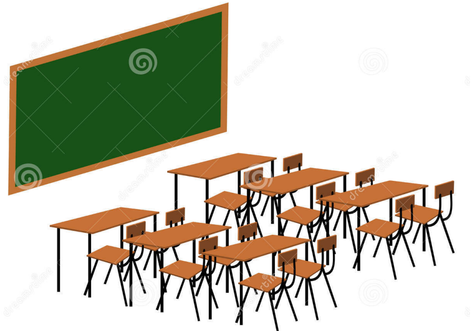
Introduction: About me

Interests!

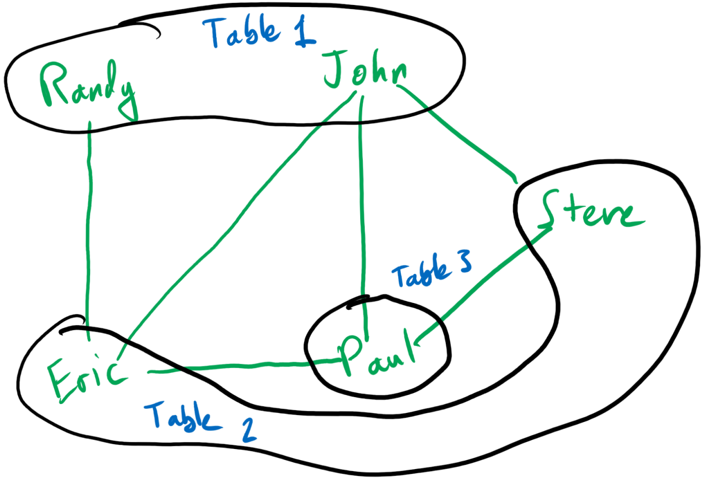
- Surfing!
- Roller Skating!
- Cooking!
- Exclamation points!

Introduction: The Graph Coloring Problem

The Graph Coloring Problem
Example: Let's say I'm a teacher, and I noticed that my students aren't getting their work done because they're too busy chatting with their friends. My plan then, is to place them at tables *away* from their friends. How can I arrange their seats so that no student has any friends at a given table?



Introduction: The Graph Coloring Problem



Introduction: The Graph Coloring Problem

Answer: Graph Coloring!
Generally, the graph coloring problem asks given a set of nodes connected by edges (a graph), how can I color (think: a labelling that induces a partitioning) the nodes such that no node is colored the same as one of it's neighbors.
In our example:

- Nodes: Students.
- Edges: Student a is connected to Student b if they are friends.
- Colors: Tables that we assign them to.
- **Remark!** Notice that if we have a group of students G that are all friends with each other (they form a social *clique*), then we need to use at least that many tables in our assignment (because they all need their own table). This will be important later!

In most cases, we want to get an *optimal* coloring, that is one that uses the least amount of colors possible.

Introduction: Heuristics

This problem is actually very difficult to solve for arbitrary graphs! In fact, if we want a guarantee of success on any graph, our best algorithms don't fare much better than just checking every possible coloring until the optimal one is found (which, for even moderately large graphs, can be more particles than there are in the observable universe!).
So, what do we do?
Answer: Heuristics!

- **The Good:** They tend to do well on many graphs we test them on (i.e. they find an acceptable solution).
- **The Bad:** They have much looser and less rigorous guarantees and can do abysmally poor on graphs that others do well.

Moreover, it can be hard to compare heuristic A to heuristic B in any meaningful way besides through empirical results. How do you know if A is better suited to your dataset's needs or B is?

Introduction: Perfect Graphs

For my project, we looked particularly on graphs sampled uniformly at random from the space of all *perfect graphs* on n nodes (denoted as $G \sim P_n$ for arbitrary n) and asked for our chosen heuristic **what guarantees can we give on performance or solution quality?** But, first, what are Perfect Graphs?
First, it must be noted that for any graph, we have a lower bound on the number of colors it must take to color it: The size of the maximum clique in the graph (recall our example!)

$$\text{Clique } \# \leq \text{Chromatic } \#$$

In perfect graphs, this lower bound is actually an equality, and we also have:

$$\text{Clique } \# \text{ of } H = \text{ Chromatic } \# \text{ of } H$$

for any subgraph $H \subset G$ of our graph. This leads to some very nice properties that make the class of graphs quite 'friendly'.

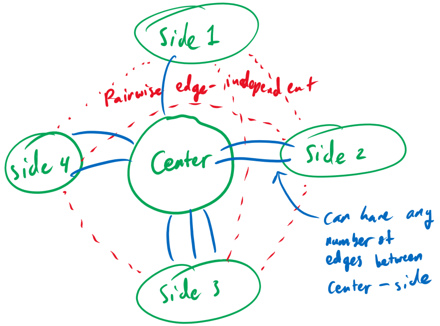
Introduction: Random Perfect Graphs

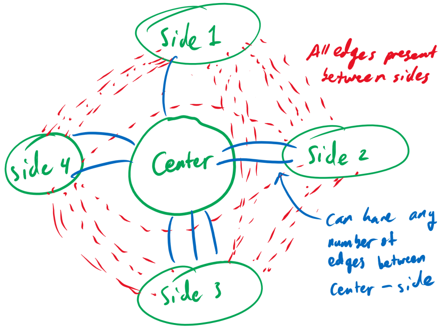
- In [7], Prömel and Steger were able to show that almost all perfect graphs actually fall into the class of *Generalized Split Graphs*.
- McDiarmid and Yolov showed in [6] how to sample uniformly from this family of graphs (GSG_n).
- In the same paper, they showed how this allows one to sample almost uniformly from P_n by sampling from GSG_n.

Thus, we move to the study of Generalized Split Graphs.

Generalized Split Graphs

GSGs are a family of graphs comprised of the union of two subclasses of perfect graphs, *unipolar* (GSG_n^+) and *co-unipolar* (GSG_n^-) graphs.


Side 1
Side 2
Side 3
side 4
Center
Pairwise edge-independent
Can have any number of edges between center - side


Side 1
Side 2
Side 3
side 4
Center
All edges present between sides
Can have any number of edges between center - side

Heuristics Analyzed

Random MaxIS [6]

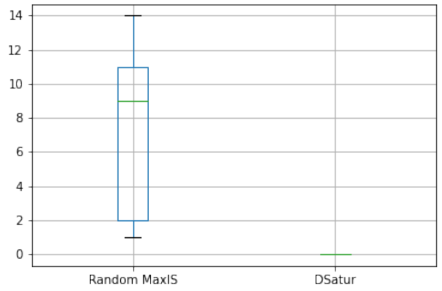
- High-level: Greedily construct maximal independent sets and color in a sequential order using the independent sets you construct.
- Able to get roughly twice the number of colors in a random graph w.h.p. [5]
- Can be shown to require $O(n)$ colors to color a 3-colorable graph [3].

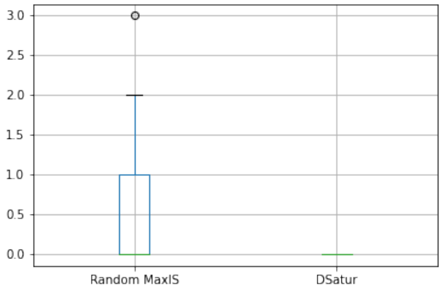
DSatur [1]

- High-level: Color vertices in 'saturation' order (has the most distinct neighboring colors, i.e. the most constrained vertex), breaking ties using degree.
- Known to be optimal for many classes of perfect graphs including bipartite [1] and chordal graphs (not necessarily perfect) [4].
- Can be shown to require $O(n)$ colors on some 3-colorable graphs (not necessarily perfect) [2]

Empirical Results

Figure: Found Chromatic Number - True Chromatic Number; Left: Unipolar Case, Right: Co-Unipolar Case





Theoretical Results

Theorem

With high probability, the Random MaxIS heuristic finds a coloring of $G \sim P_n^+$ *using* $\chi(G) + O(\ln(n))$ *colors.*

Theorem

With constant probability, the Random MaxIS heuristic find a coloring of $G \sim P_n^-$ *with at most* $\chi(G) + 1$ *colors.*

Theorem

With high probability, DSatur finds an optimal coloring on $G \sim P_n^+$*.*

Theorem

DSatur finds the optimal coloring for arbitrary $G \in GSG_n^-$*.*

Future Work

Conjecture

With at least constant probability, the Random MaxIS heuristic finds a coloring of $G \sim P_n$ *using at most* $\chi(G) + O(\ln(\ln(n)))$ *colors.*

Conjecture

There exists $G \in GSG_n$ *such that with high probability, Random MaxIS fails to find an optimal solution.*

Conjecture

DSatur finds the optimal coloring for arbitrary $G \in GSG_n$*.*

Implications

- Perfect graphs are highly studied and are still of interest to be solved as they come up quite often in scheduling and time tabling.
- Giving more analysis for commonly used graph coloring heuristics.
- Insight into analyzing more complex heuristics that often use the above techniques as a subroutine
- Insight into a general combinatorical algorithm for coloring perfect graphs.

References I

- Daniel Brélaz. “New Methods to Color the Vertices of a Graph”. In: *Commun. ACM* 22.4 (Apr. 1979), pp. 251–256. ISSN: 0001-0782. DOI: 10.1145/359094.359101. URL: https://doi.org/10.1145/359094.359101.
- Robert Janczewski et al. “The smallest hard-to-color graph for algorithm DSATUR”. In: *Discret. Math.* 236 (2001), pp. 151–165.
- D. W. JOHNSON. “Worst case behavior of graph coloring algorithms”. In: *Proceedings of the 5th Southeast Conference on Combinatorics, Graph Theory, and Computing, 1974* (1974), pp. 513–527. URL: https://ci.nii.ac.jp/naid/10025066790/en/.
- Benjamin Lévêque and Frédéric Maffray. “Coloring Meyniel graphs in linear time”. In: *Electronic Notes in Discrete Mathematics* 22 (Oct. 2005), pp. 25–28. DOI: 10.1016/j.endm.2005.06.005.
- Colin McDiarmid. “Colouring random graphs.”. In: *Annals of Operations Research* 1 (1984).

References II

Colin McDiarmid and Nikola Yolov. *Random Perfect Graphs*. 2017. arXiv: 1604.00890 [math.CO].

Hans Jürgen Prömel and Angelika Steger. "Almost all Berge Graphs are Perfect". In: *Combinatorics, Probability and Computing* 1.1 (1992), pp. 53–79. DOI: 10.1017/S0963548300000079.

Thank you to my advisor, Dr. Impagliazzo, the UCSD Physical Sciences Department, and the generous donors who made this possible!