Not Clique-Bait: Algorithmic Approaches to the Maximum Clique Problem

Kellen Knop, Brendan Banfield, Aaron Banse, Sophie Quinn

What is the Maximum Clique Problem?

The Maximum Clique Problem asks the question: what is the largest group of fully connected nodes in a graph?

Input: An undirected, weighted graph $G = (V, E)$
Output: A complete graph $G' \subseteq G$, such that
$G' = (V', E')$ and $||V||$ is maximized

This seemingly simple question is **NP-hard**, meaning it becomes computationally expensive as graphs grow in size, so it's important to have efficient algorithms that can solve it in a reasonable amount of time.

What did we do?

We chose to implement 4 different algorithms:

2 exact: Bron-Kerbosch (with pivoting) and Branch and Bound

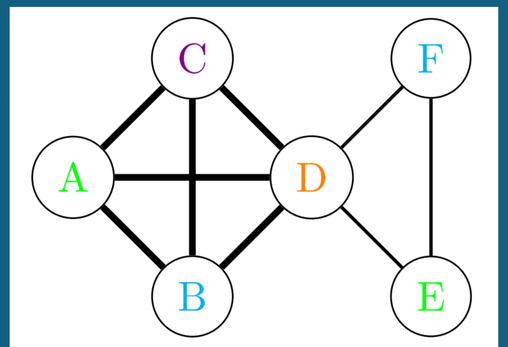
2 approximation: Genetic and Simulated Annealing

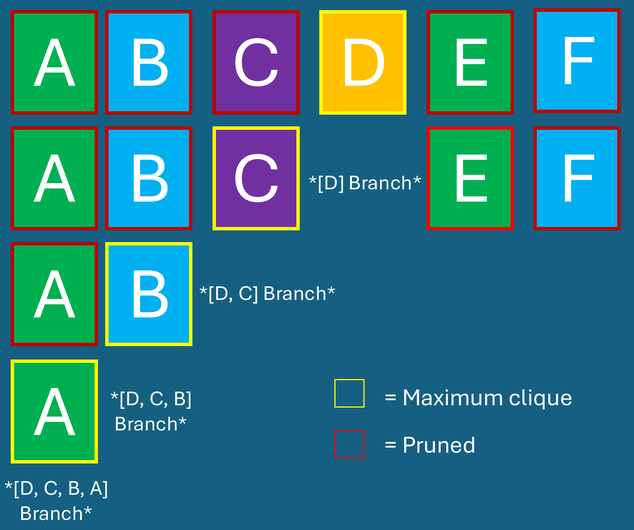
We utilized exact algorithms as a benchmark to assess the time required to obtain an exact solution for the maximum clique problem. Additionally, we evaluated the runtime of approximation algorithms in comparison and measured how closely their outputs aligned with the correct solution. For approximation algorithms, our primary focus was minimizing runtime rather than achieving perfect accuracy, whereas for exact algorithms, we prioritized output precision. Our experiments were conducted on both randomly generated graphs and DIMACS benchmark graphs, the latter providing known maximum cliques for accuracy validation. After receiving correct outputs for DIMACS, we tested our algorithms on 50 randomly generated graphs across five size groups—25, 50, 100, 200, and 400 vertices—while also analyzing each graph's number of edges, edge density, edge variance, and clustering coefficient. In the interest of time, algorithms that did not output within 600 seconds were terminated early.

Branch and Bound Algorithm

This branch-and-bound algorithm operates by greedily coloring the graph and then recursively "branching" to add nodes to the current clique, prioritizing those with the highest assigned color. It utilizes an upper bound based on the largest clique found so far. If a branch is determined to be incapable of producing a larger clique, it is "pruned," terminating the search early to improve efficiency

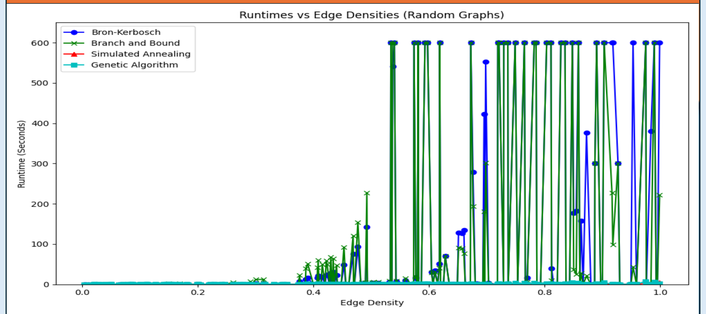
Example:

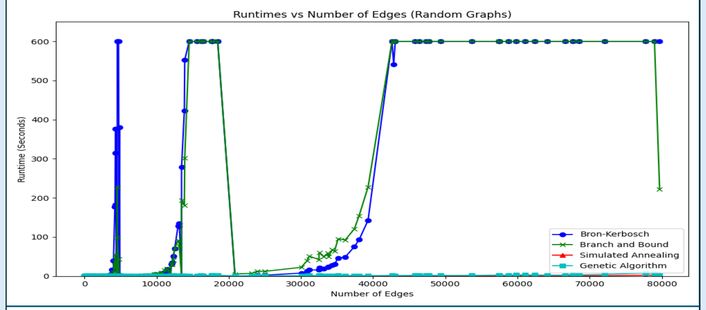


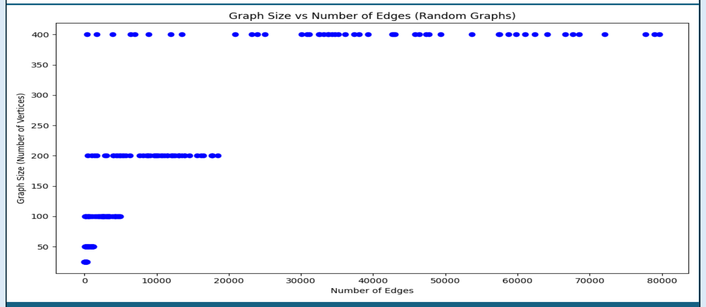


The algorithm begins by adding every possible vertex to the current clique, prioritizing those with the highest assigned color. After a vertex is added, . This process results in the clique [D, C, B, A]. Once no more neighboring vertices can be added, the algorithm compares the current clique's size to the maximum clique found so far. If it is larger, it becomes the new maximum clique. The algorithm then backpropagates the maximum clique size (4) to the last branch containing unadded vertices ([D] branch). Next, it evaluates vertex F, which has a color of 2, indicating that at most two additional vertices could be added to the current clique [D], forming a clique of size 3. Since this is smaller than the previously found maximum clique, the algorithm backpropagates a final maximum clique size of 4.

Random Graphs Results







Edge number and density had the strongest correlations with runtime. The spikes of Bron-Kerbosch and Branch and Bound in the "Number of Edges" graph is probably due to the number of vertices. The approximation algorithms outperformed the exact on most large (|V|=400) graphs.

Reference

Konc and Janežič (2007). *An Improved Branch and Bound Algorithm for the Maximum Clique Problem*

Acknowledgements

I want to thank my family, team, and professors, who supported me during compsing