Report on Maximal Clique Enumeration Algorithms

Date: March 23, 2025

Design and Analysis of Algorithms

CS F364

Submitted By:

Name	**ID**
Sthitaprajna	2021B3A71082H
Kushagra Mishra	2021B5A72970H
Riya Agrawal	2021B3A70996H
Dhruv Choudhary	2021B3A73142H
Waleed Iqbal Shaikh	2021B3A70559H

Table of Contents

1. Introduction

Graph algorithms are crucial in many application domains, including social network analysis, bioinformatics, and communication networks. In this report, we focus on the problem of maximal clique enumeration—finding all complete subgraphs (cliques) that cannot be further extended.

One key concept in graph analysis is the clique—a subset of vertices in which every pair of vertices is connected by an edge. In many applications, it is essential not only to find cliques, but to identify maximal cliques, which are cliques that cannot be extended by including any adjacent vertex without losing their complete connectivity.

An important measure that influences clique-finding algorithms is arboricity. Arboricity is defined as the minimum number of edge-disjoint spanning forests needed to cover all edges in a graph. This property captures the inherent sparsity or density of a graph and can be used to design algorithms with improved performance. In particular, when the arboricity is low, efficient techniques can be applied to enumerate cliques faster, making arboricity a valuable parameter in the analysis of large-scale networks.

Our work reviews and implements three distinct algorithms based on pioneering research:

- A DFS-based algorithm with effective pruning (Tomita et al.),
- A variant of the Bron–Kerbosch algorithm that exploits degeneracy ordering,
- An approach leveraging the concept of graph arboricity.

This report outlines the theoretical background, implementation challenges, and empirical results obtained on benchmark datasets.

---

2. Overview of Algorithms

2.1. Tomita's DFS-Based Approach

Tomita and colleagues improved on the traditional Bron–Kerbosch method by incorporating an efficient pivot selection strategy. The algorithm uses a depth-first search (DFS) to systematically explore potential cliques. A key aspect is the careful maintenance of three vertex sets:

- **R:** the current clique,
- **P:** candidate vertices for expansion,
- **X:** vertices already considered.
  Pruning is applied by selecting a pivot that minimizes the number of recursive calls,

leading to an optimal worst-case performance relative to the number of vertices.

By cleverly selecting the pivot, the algorithm avoids redundant exploration of similar subgraphs. The worst-case time complexity is expressed in terms of the number of maximal cliques, and the pivoting strategy ensures that the method performs near the theoretical lower bound. The DFS structure allows the algorithm to delve deep into the graph, reporting a maximal clique only when both P and X are empty, ensuring that each clique is maximal.

Tomita's algorithm is particularly effective for medium-sized graphs where the number of cliques is manageable. It has been widely adopted due to its balance between theoretical optimality and practical efficiency.

Time Complexity: O(3^(n/3)), where n is the number of vertices

2.2. Bron–Kerbosch with Degeneracy Ordering

This variant of the Bron–Kerbosch algorithm begins by ordering the graph's vertices based on degeneracy—a measure of sparsity indicating that every subgraph has a vertex with a limited number of neighbors. By processing vertices in this order, the algorithm restricts the candidate set at each recursive call, reducing redundant computations. The pivoting strategy further narrows the search space, making the approach particularly effective for large, sparse graphs.

The concept of degeneracy refers to the property that every subgraph of the graph has at least one vertex with a limited number of neighbors. By ordering the vertices in non-decreasing order of their degree, the algorithm restricts the number of candidate vertices to be considered during the recursive expansion. This ordering drastically reduces the size of the candidate set in each recursive call.

Similar to Tomita's method, the algorithm employs a recursive process using three sets (R, P, and X). However, by processing vertices in degeneracy order, the algorithm ensures that many vertices in the candidate set have already been excluded or processed, leading to a more efficient search. A pivot is selected at each step to further prune the search space by eliminating those vertices that do not contribute to forming a larger clique.

Time Complexity: O(d·n·3^(d/3)), where n is the number of vertices and d is the degeneracy

2.3. Arboricity-Based Clique Listing

The third algorithm exploits the notion of graph arboricity, defined as the minimum number of edge-disjoint spanning forests that cover all edges in the graph. This method organizes vertices by increasing degree and employs a recursive procedure to expand candidate cliques. Its time complexity is closely tied to the arboricity, making it ideal for moderately dense graphs. Although theoretically appealing, its performance can deteriorate on very large datasets due to increased computational overhead.

Initially, vertices are ordered by increasing degree. The algorithm then uses a recursive expansion process (similar in spirit to the EXPAND procedure used in other methods) to generate candidate cliques. For each vertex, only the neighbors that come later in the order are considered, thereby avoiding duplicate clique formations. The process involves rigorous maximality tests and lexicographical ordering checks to ensure that only unique maximal cliques are reported.

The algorithm achieves a time complexity of $O(a(G) \cdot m)$ per clique, where $a(G)$ is the arboricity and m is the number of edges. This makes it particularly efficient for graphs with low arboricity (such as planar graphs). However, in practice, for very large or highly dense graphs, the overhead of managing extensive candidate sets may lead to increased runtimes, as observed in some experimental evaluations.

Time Complexity: $O(a(G) \cdot m)$, where m is the number of edges and a(G) is the arboricity

---

3. Implementation Summary

The algorithms have been implemented in C++ with emphasis on efficiency and clarity. Common implementation strategies include:

- **Graph Input and Preprocessing:**
  Graphs are read from text files and represented using adjacency lists. Vertices are re-indexed as necessary (typically using 0-indexing) to streamline processing.

- **Core Recursive Procedures:**
  Each algorithm employs a recursive procedure (either EXPAND or a modified Bron–Kerbosch) that builds cliques vertex by vertex. Key operations include:

    - **Intersection of Candidate Sets:**
      Rapidly identifying common neighbors to ensure that the clique property is maintained.

    - **Pivot Selection and Pruning:**
      Strategies to reduce unnecessary recursion by eliminating vertices that cannot contribute to a maximal clique.

- **Optimization Techniques:**
  Use of STL containers and in-place operations to minimize memory overhead and improve runtime efficiency.

Run instructions were provided with each implementation, detailing the compilation commands (using optimization flags such as `-O3`) and execution guidelines for different operating systems.

---

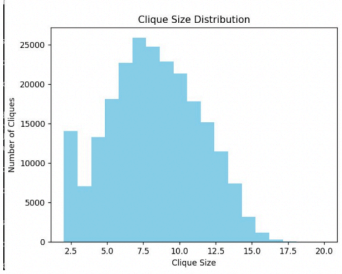
4. Experimental Observations and Runtime Comparison

The algorithms were tested on three real-world datasets:

- **Email-Enron Dataset:**
    - Nodes: 36,692
    - Edges: 183831
    - Observations: Moderate clique sizes were observed with the largest clique having 20 vertices.
    - Runtime: The DFS-based and degeneracy-ordered approaches completed in minutes, while the arboricity-based algorithm required several hours.
- **Wikipedia Vote Dataset:**
    - Nodes: 7,115
    - Edges: 103689
    - Observations: The largest clique comprised 17 vertices, and all methods produced consistent results.
    - Runtime: The DFS-based and degeneracy-ordered approaches completed in minutes, while the arboricity-based algorithm required several hours.
- **AS-Skitter Dataset:**
    - Nodes: 16,96,415
    - Edges: ~11 million
    - Observations: The dataset posed the greatest challenge with the largest clique reaching 67 vertices.
    - Runtime: The degeneracy-based method demonstrated the best scalability, whereas the DFS-based approach took significantly longer in hours. The arboricity-based method struggled with resource constraints on this dataset.

Clique Distribution

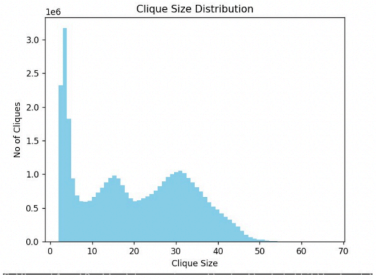
Wiki-Vote Network



Largest Clique Size: **17**

Total Maximal Cliques: **459002**

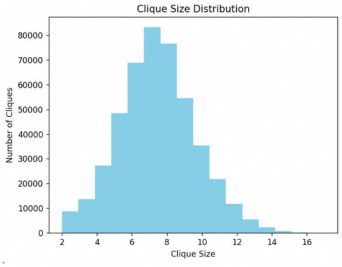
Skitter Network



Largest Clique Size: **67**

Total Maximal Cliques: **37322355**

Enron Network

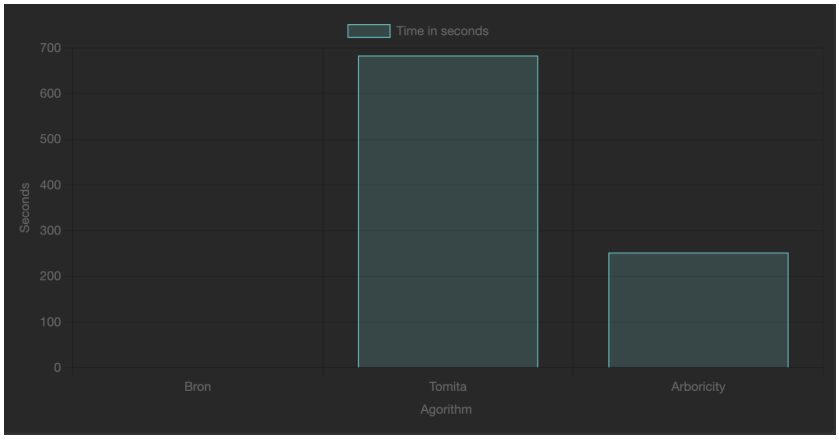


Largest Clique Size: **20**

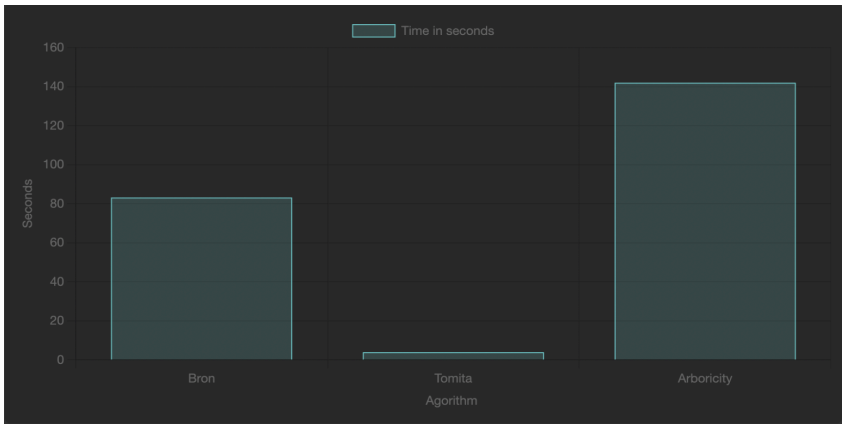
Total Maximal Cliques: **226859**

Execution time of implementation on each dataset (histogram).

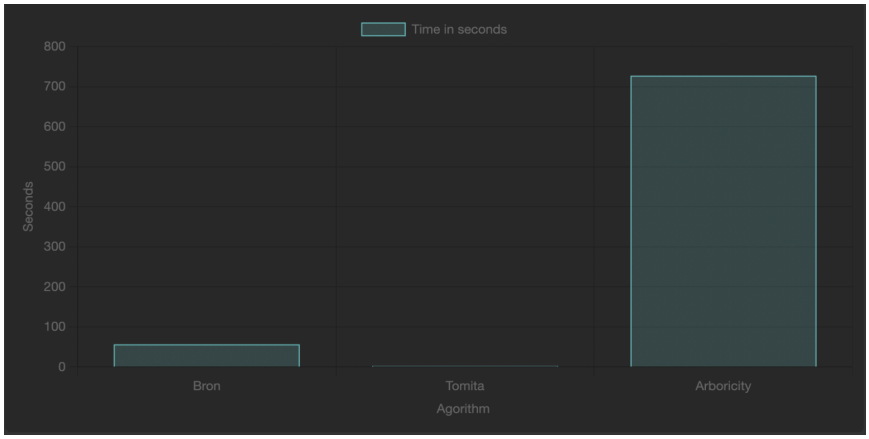
Skitter Dataset:



Wiki-Vote Dataset:

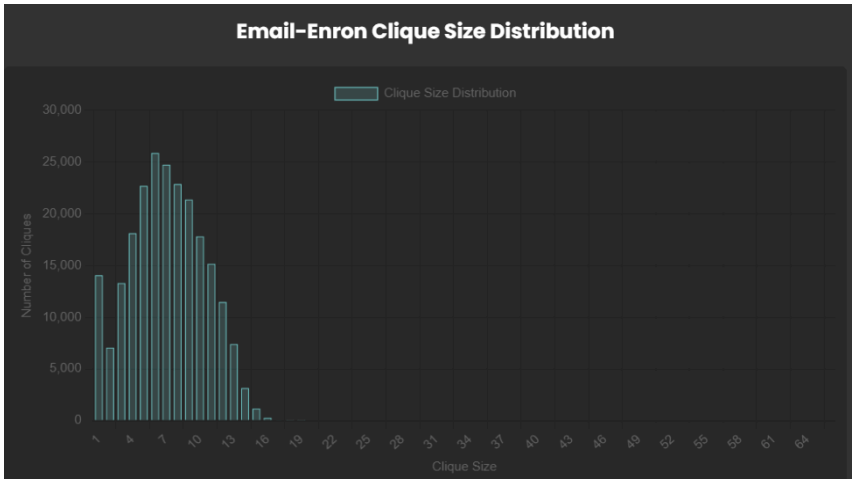


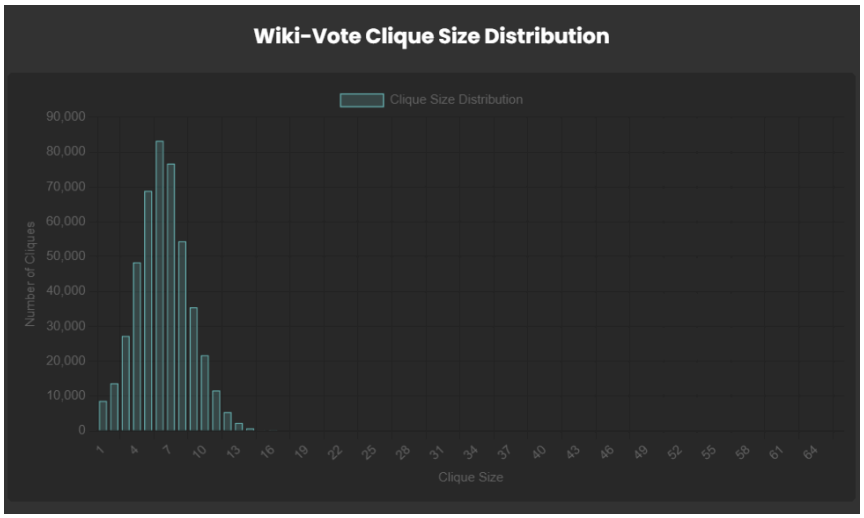
Email-Enron Dataset:

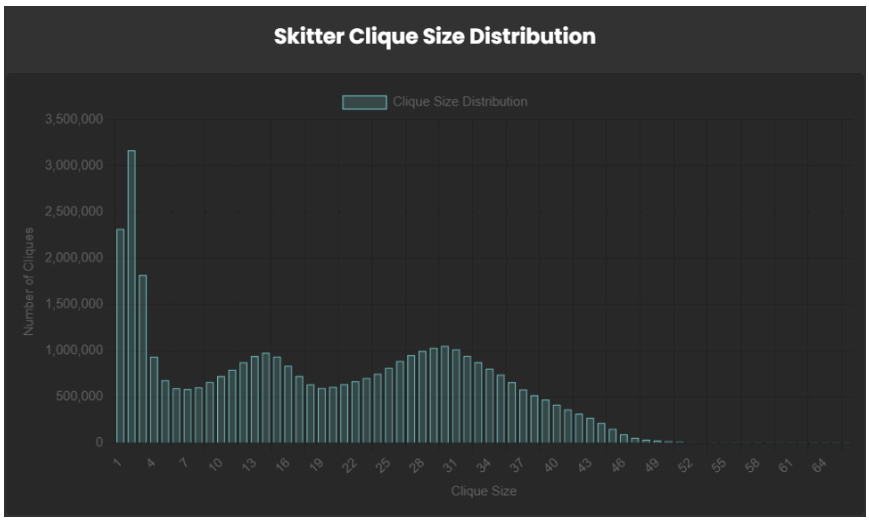


Distribution of different size cliques (histogram).

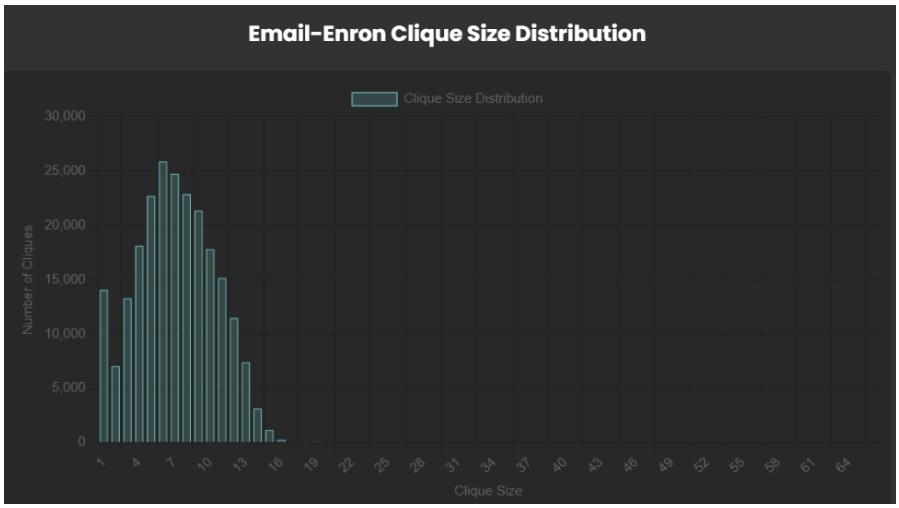
Tomita:

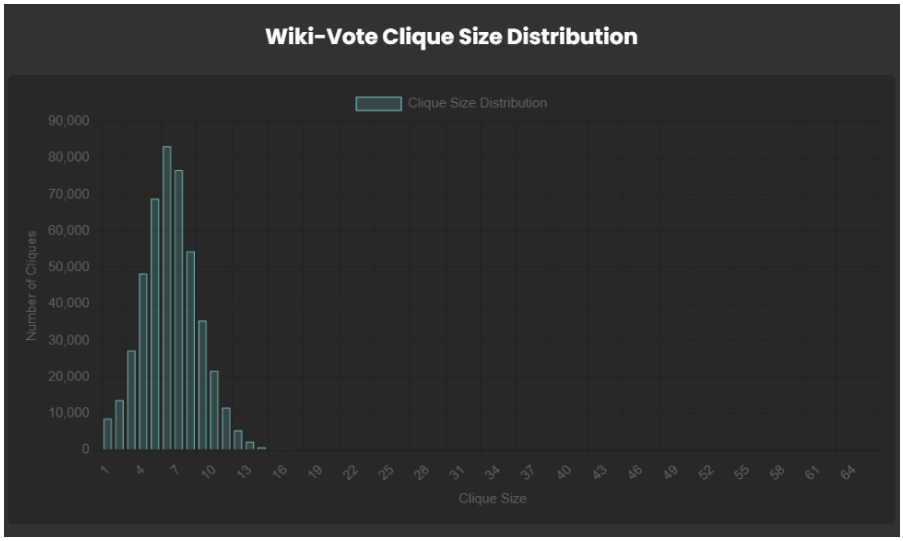


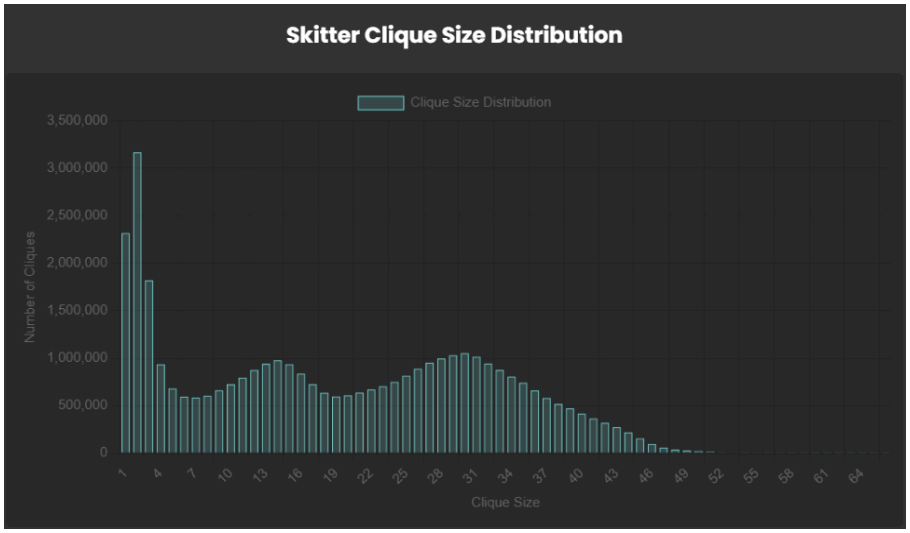




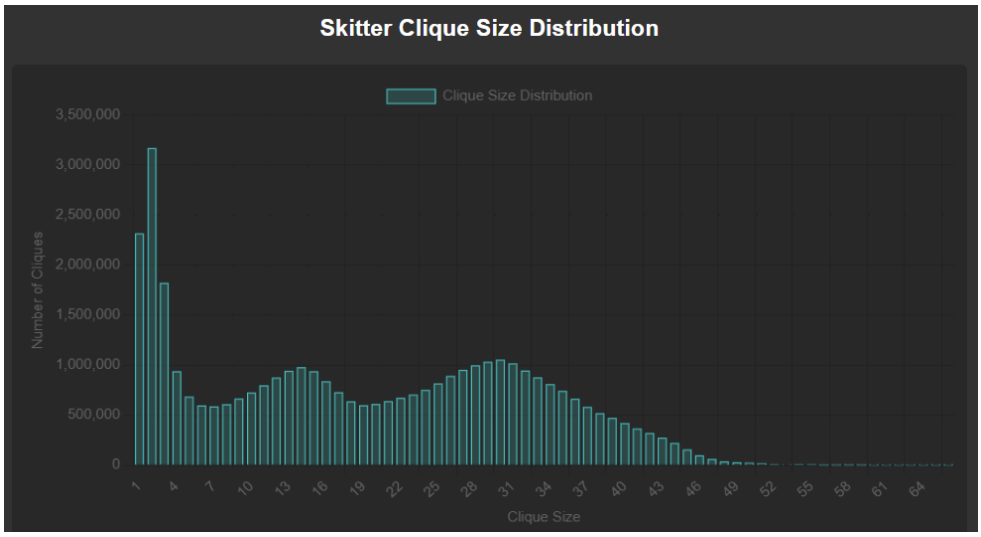
Bron-Kerbosch:

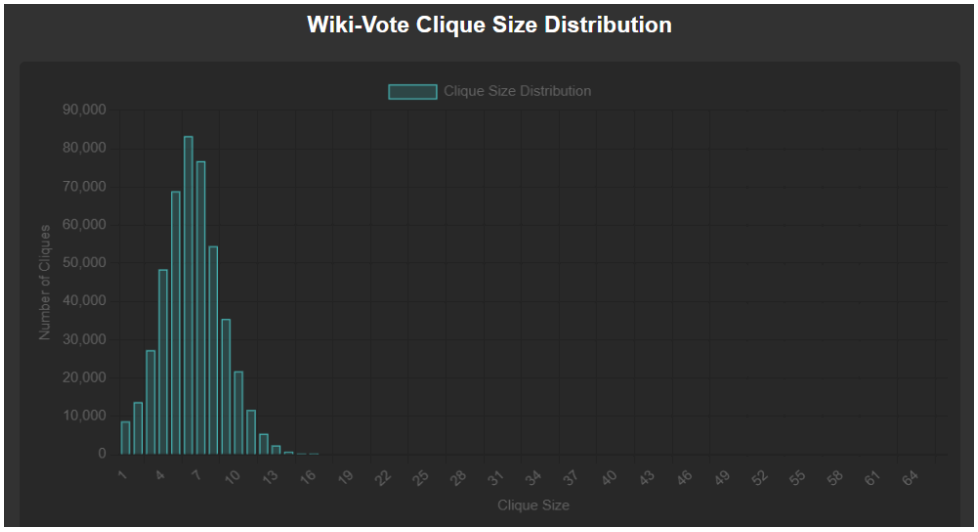


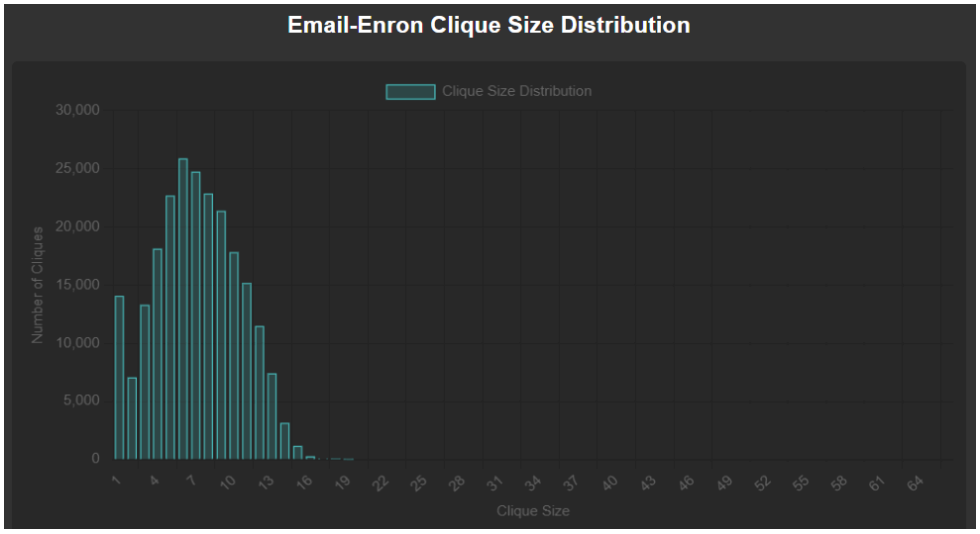




Arboricity


Skitter Clique Size Distribution
Clique Size Distribution
Number of Cliques
3,500,000
3,000,000
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64
Clique Size


Wiki-Vote Clique Size Distribution
Clique Size Distribution
Number of Cliques
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64
Clique Size


Email-Enron Clique Size Distribution
Clique Size Distribution
Number of Cliques
30,000
25,000
20,000
15,000
10,000
5,000
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64
Clique Size

---

5. Discussion and Findings

Our experimental analysis reveals several key insights:

- **Pruning Strategies:**
  Effective pivot selection and candidate set reduction are vital for reducing recursive overhead.

- **Graph Properties:**
  The choice of algorithm should consider the specific properties of the dataset. Sparse graphs benefit more from degeneracy ordering, while denser graphs may require different heuristics.

- **Scalability:**
  Modern implementations (Tomita and Bron–Kerbosch with degeneracy) scale significantly better on large datasets compared to older approaches.

These findings underline the importance of algorithmic refinements that are sensitive to the graph's structural properties.

---

6. Conclusion

In conclusion, our comparative study of maximal clique enumeration algorithms demonstrates that:

- Modern DFS-based and degeneracy-ordered methods offer substantial performance improvements over traditional approaches.

- The arboricity-based algorithm, while theoretically sound, may face practical limitations on very large datasets.

- Choosing the right algorithm depends on both the size and structure of the graph, as well as available computational resources.

Future work may involve further optimizations, parallel implementations, and exploration of hybrid methods to handle extremely large-scale graphs more efficiently.

Link to Project Website: