MAXIMAL CLIQUE ENUMERATION ALGORITHMS AND ANALYSIS

CS F364 : Design and Analysis of Algorithms

Prof. Apurba Das

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCES, PILANI, HYDERABAD CAMPUS

Maximal Clique Enumeration

By
Group - 9

Name (ID)	Contribution
Jeeru Harshith Reddy (2022A7PS0233H)	Listing All Maximal Cliques in Sparse Graphs in Near Optimal Time
Vishal Varma Bhupathiraju(2022A7PS0174H)	The worst-case time complexity for generating all Maximal Cliques
Sri Jaitra Saketh Goparaju(2022A7PS0183H)	Arboricity and subgraph listing algorithms
Vineeth Ulavala(2022A7PS0071H)	Arboricity and subgraph listing algorithms
Abhinav Sai Yekkali(2022A7PS0012H)	Listing All Maximal Cliques in Sparse Graphs in Near Optimal Time

1. Introduction

In this project, we implement and compare three different maximal clique enumeration algorithms to evaluate their performance on real-world datasets. These algorithms were selected based on their distinct theoretical foundations, each offering unique advantages in different types of graph structures.

The first algorithm is based on worst-case complexity analysis, ensuring a structured approach to handling arbitrary graphs with a focus on theoretical guarantees. The second employs degeneracy ordering, which is particularly effective for sparse graphs by reducing redundant computations and improving recursive efficiency. The third method leverages arboricity-based techniques, optimizing clique enumeration by considering the minimal number of edge-disjoint spanning forests required to cover the graph.

By evaluating these approaches across diverse datasets, we aim to assess their computational efficiency, scalability, and suitability for different graph topologies. The results provide insights into the practical trade-offs between worst-case guarantees, sparsity-driven optimizations, and structural decomposition techniques, ultimately guiding the choice of algorithm based on the characteristics of the input graph.

2. Algorithm Descriptions

2.1 Procedure CLIQUE

(Paper 1:The worst-case time complexity for generating all maximal cliques and computational experiments : Tomita, Etsuji, Akira Tanaka, and Takahashi)

Overview

This algorithm is designed to generate all maximal cliques in an undirected graph. The algorithm builds on the depth-first search (DFS) approach used in the Bron–Kerbosch algorithm but introduces pruning techniques to improve efficiency. A key contribution of the paper is proving that the worst-case time complexity of the algorithm is **$O(3^{n/3})$**, which is optimal for an n-vertex graph. Additionally, the algorithm outputs maximal cliques in a tree-like format, reducing memory consumption compared to traditional methods.

Algorithm Summary

1. **Recursive Expansion:** The algorithm recursively grows candidate sets of vertices to generate maximal cliques while pruning unnecessary branches.
2. **Depth-First Search (DFS):** Uses a recursive **EXPAND** function to traverse the search space efficiently.
3. **Pruning Strategies:**
    a. Maintains a **CAND** set (candidates for expansion) and a **FINI** set (processed vertices).
    b. Selects pivot vertices to minimize recursive calls and reduce redundant searches.
4. **Ordering Strategy:** Prioritizes vertices that maximize clique expansion, ensuring efficient search.
5. **Output Optimization:** Uses a tree-structured format to save space without explicitly storing all cliques.

AnalysisTime Complexity: The algorithm runs in **$O(3^{n/3})$** time, which is optimal because it matches the worst-case number of maximal cliques in an n-vertex graph, as established by Moon and Moser.

- **Space Complexity:** The space usage is **$O(n^2)$ for adjacency matrices** and **$O(n + m)$ for adjacency lists**, making it efficient since it avoids storing all maximal cliques explicitly.

2.2 Bron-Kerbosch with Degeneracy Ordering

(Paper 2: Listing all maximal cliques in sparse graphs in near-optimal time Eppstein, D., Löffler, M., & Strash, D.)

Overview

The paper introduces a novel approach that leverages graph degeneracy to improve performance in real-world networks, many of which are naturally sparse. By optimizing the recursive search strategy and reducing unnecessary computations, the algorithm achieves a worst-case time complexity of O(dn3d/3), which is nearly optimal for sparse graphs.Uses degeneracy ordering, which allows the algorithm to focus on vertices with lower degrees first, making it faster for sparse graphs.For sparse graphs with low degeneracy, the algorithm is exponentially faster than Tomita's method.

Algorithm Overview

1. **Degeneracy Ordering:** Processes low-degree vertices first, ensuring a bounded recursion depth and reducing redundant computations.
2. **Pivot Selection:** Chooses optimal pivot vertices to minimize the number of recursive calls.
3. **Modified Bron-Kerbosch Algorithm:**
    - Iterates over vertices in degeneracy order.
    - Applies the **pivoting strategy** to minimize recursive calls.
    - Efficiently generates and reports maximal cliques.

Analysis

- **Time Complexity**: The algorithm runs in O(d n 3^d/3) time, making it highly efficient for sparse graphs since degeneracy (d) is often much smaller than n in real-world datasets.
- **Space Complexity**: With a depth-first search approach, it requires O(n + m) space, benefiting from degeneracy-based ordering to reduce redundant computations.

2.3 Arboricity-Based Clique Listing

(Paper 3: Arboricity and subgraph listing algorithms:Chiba, Norishige, and Takao Nishizeki)

Overview

Arboricity (a(G)) is the minimum number of edge-disjoint spanning forests needed to cover the graph.The algorithm follows a degree-based vertex ordering strategy to efficiently list cliques.

Algorithm Overview

- **Arboricity-Based Approach:** Uses **arboricity (a(G))**, the minimum number of edge-disjoint spanning forests needed to cover the graph, to optimize clique listing.
- **Degree-Based Vertex Ordering:** Sorts vertices in **non-decreasing degree order**, ensuring efficient traversal and clique expansion.
- **Efficient Clique Listing:**
    - **Recursive Processing:** Expands cliques efficiently while ensuring no duplication.
    - **Pivoting Strategy:** Reduces redundant computations by carefully selecting pivot vertices.
    - **Lexicographic Filtering:** Avoids redundant clique generation, ensuring minimal unnecessary computations.

Analysis

Time Complexity: The algorithm runs in **O(a(G) m) per clique**, making it highly efficient for sparse graphs (e.g., planar graphs run in **O(m)** time) but can degrade to **$O(m^{3/2})$** in dense graphs where **a(G) is large**.

Space Complexity: It requires **O(n + m) space** as it relies on edge-searching techniques rather than deep recursion, keeping memory usage minimal.

3. Experimental Setup

Dataset Descriptions

1.Dataset 1: AS-Skitter (Autonomous Systems Internet Topology)

- **Type:** Undirected graph
- **Source:** Traceroutes run daily in 2005 by CAIDA Skitter
- **Nodes: 1,696,415** (including **22,622 isolated nodes**)
- **Edges: 11,095,298**

Represents the internet topology at the **Autonomous System (AS)** level, where nodes correspond to ASes, and edges indicate direct network connections observed through traceroutes. This dataset is useful for studying internet routing, resilience, and large-scale network structures.

2.Dataset 2: Enron Email Network

- **Type:** Undirected graph
- **Source:** Enron Corporation email exchanges
- **Nodes: 36,692**
- **Edges: 367,662**

Represents email exchanges between Enron employees, where an edge between two nodes signifies that at least one email was exchanged. This dataset is widely used for social network analysis, fraud detection, and communication pattern studies.

3.Dataset 3: Wikipedia Vote Network

- **Type:** Directed graph
- **Source:** Wikipedia voting data (till January 2008)

- **Nodes: 7,115**
- **Edges: 103,689**

Captures voting behavior in Wikipedia's administrator elections, where a directed edge **A** → **B** means user **A** voted for user **B** to become an administrator. This dataset is used for studying influence, trust, and decision-making in online communities.

Metrics Evaluated:

a. Execution time on each dataset.
b. Total number of maximal cliques found.
c. Largest clique size per dataset.
d. Histogram-based analysis of execution time and clique size distribution.

4. Results and Analysis

4.1 Largest Clique Size in Each Dataset

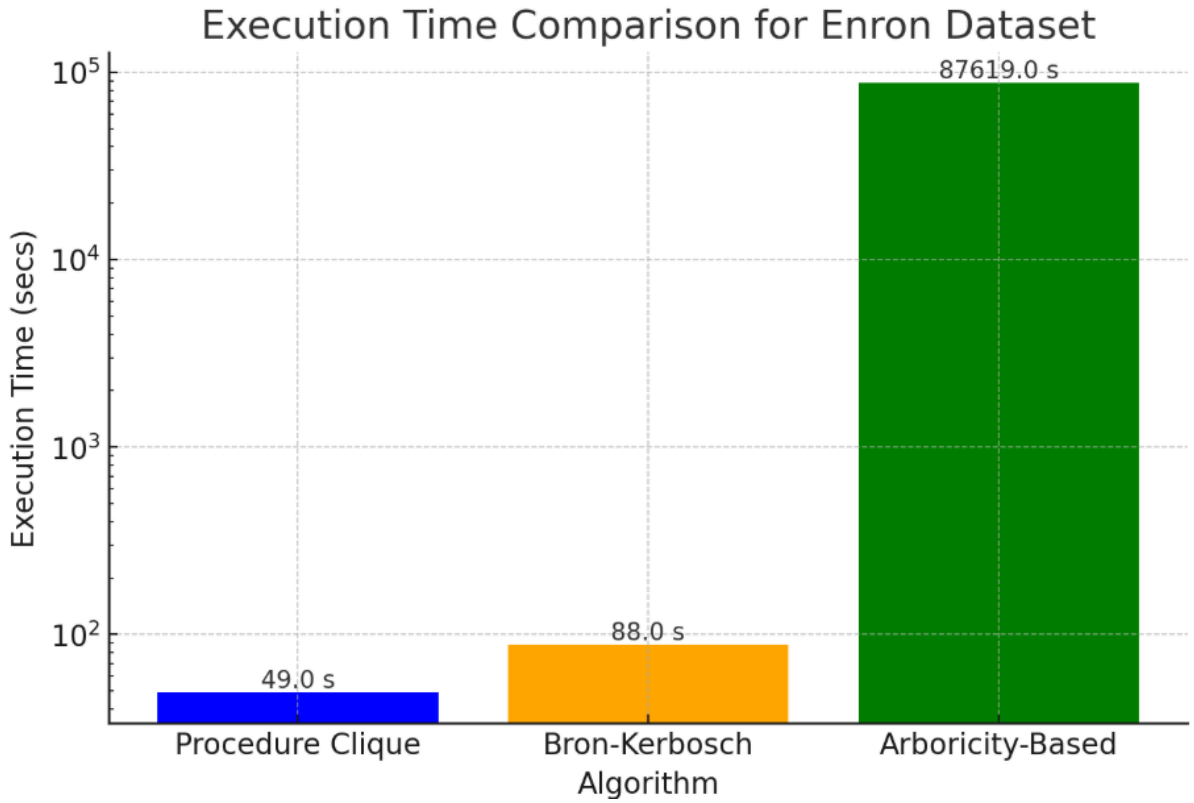
- Enron: 20
- AS-Skitter: 67
- Wiki-Vote: 17

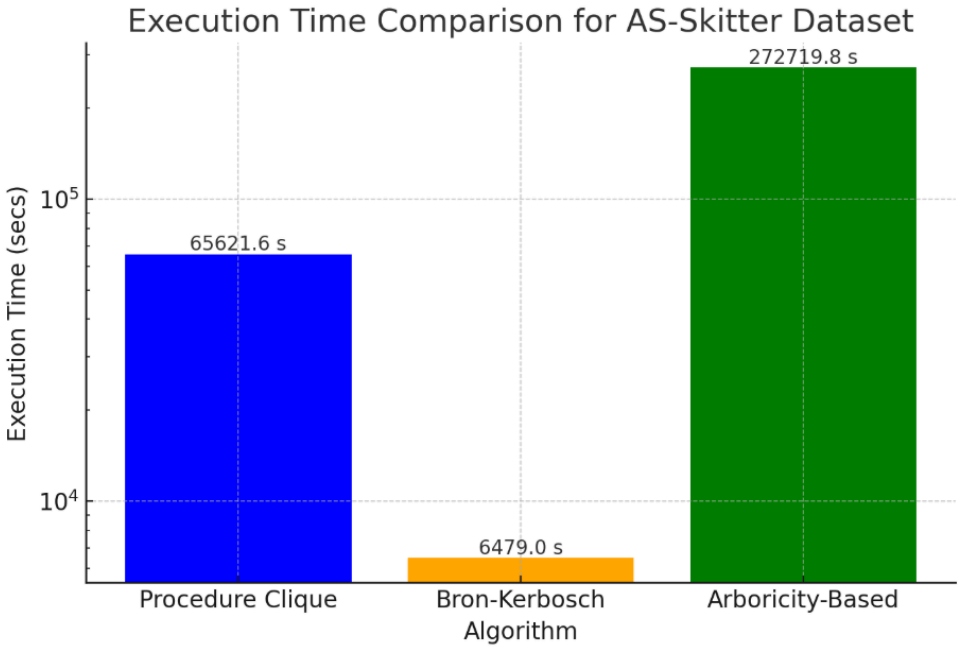
4.2 Total Number of Maximal Cliques

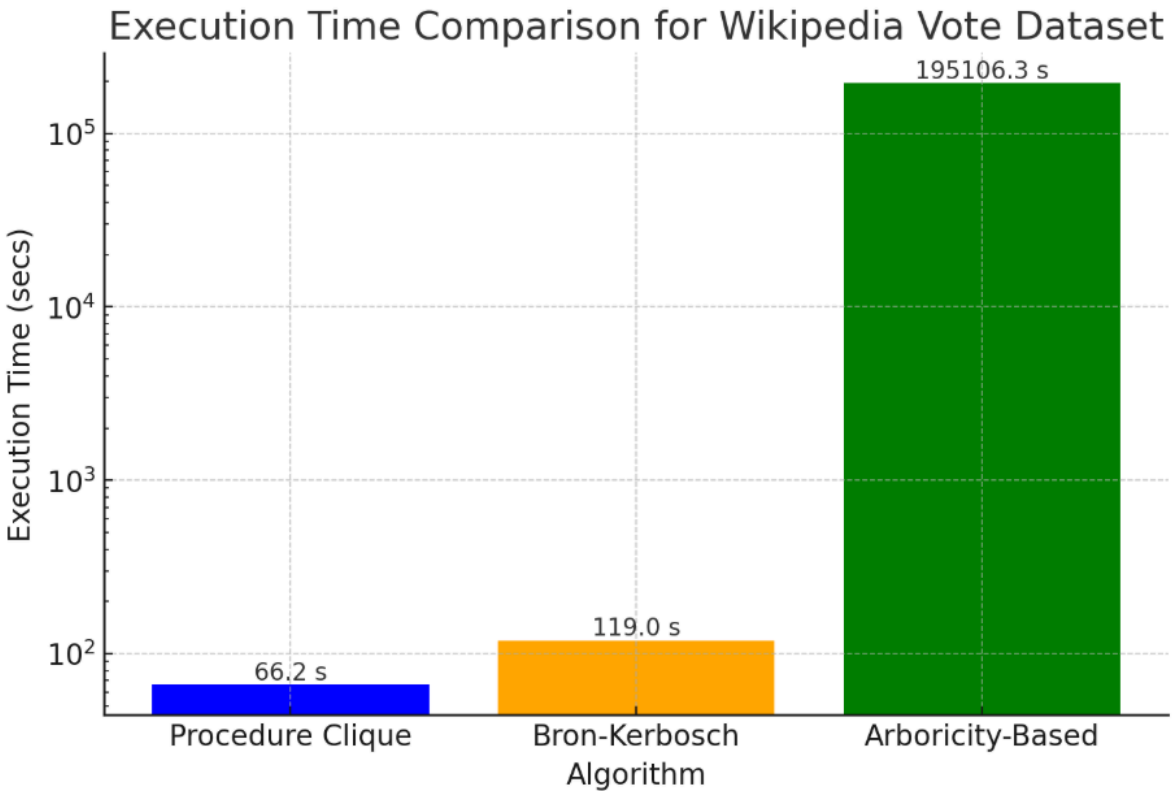
- Enron: 226859
- AS-Skitter: 37322355
- Wiki-Vote: 459002

4.3 Execution Time Comparison (Histograms)

Dataset	Procedure Clique	Bron-Kerbosch	Arboricity-Based Algorithm
Enron	49 secs	88 secs	87619 secs
AS-Skitter	65621.6 secs	6479 secs	272719.8 secs
Wikipedia Vote	66.2 secs	119 secs	195106.3 secs


Execution Time Comparison for Enron Dataset
Execution Time (secs)
10^5
10^4
10^3
10^2
87619.0 s
88.0 s
49.0 s
Procedure Clique
Bron-Kerbosch Algorithm
Arboricity-Based

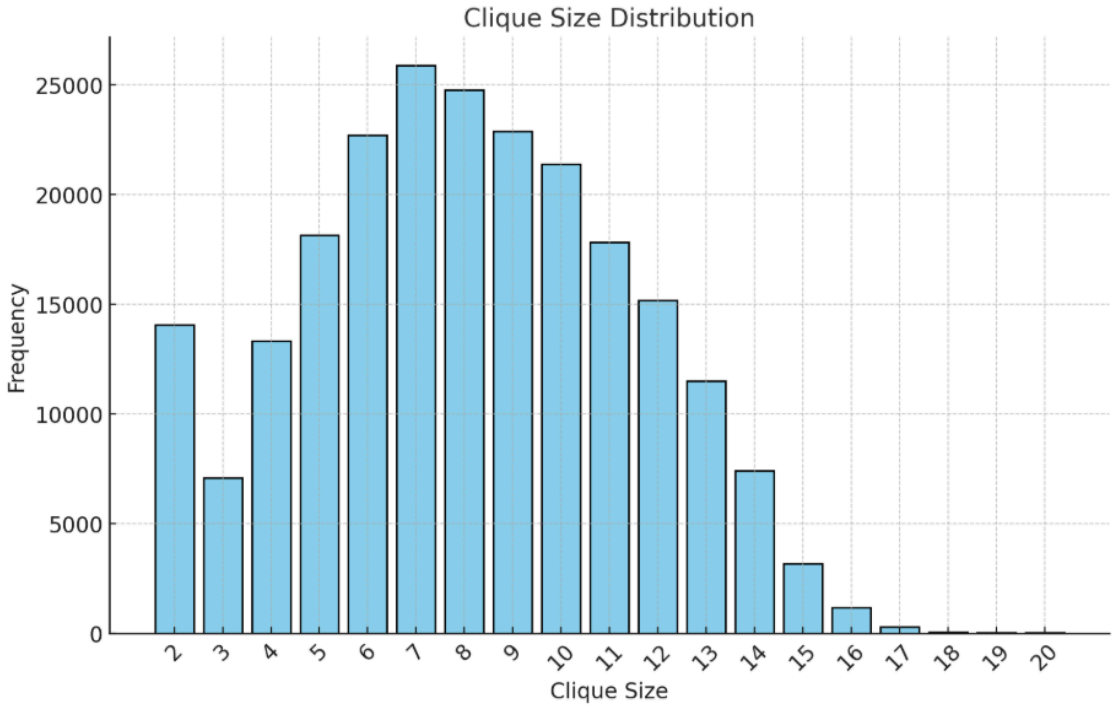

Execution Time Comparison for AS-Skitter Dataset
Execution Time (secs)
10^5
10^4
272719.8 s
65621.6 s
6479.0 s
Procedure Clique
Bron-Kerbosch Algorithm
Arboricity-Based



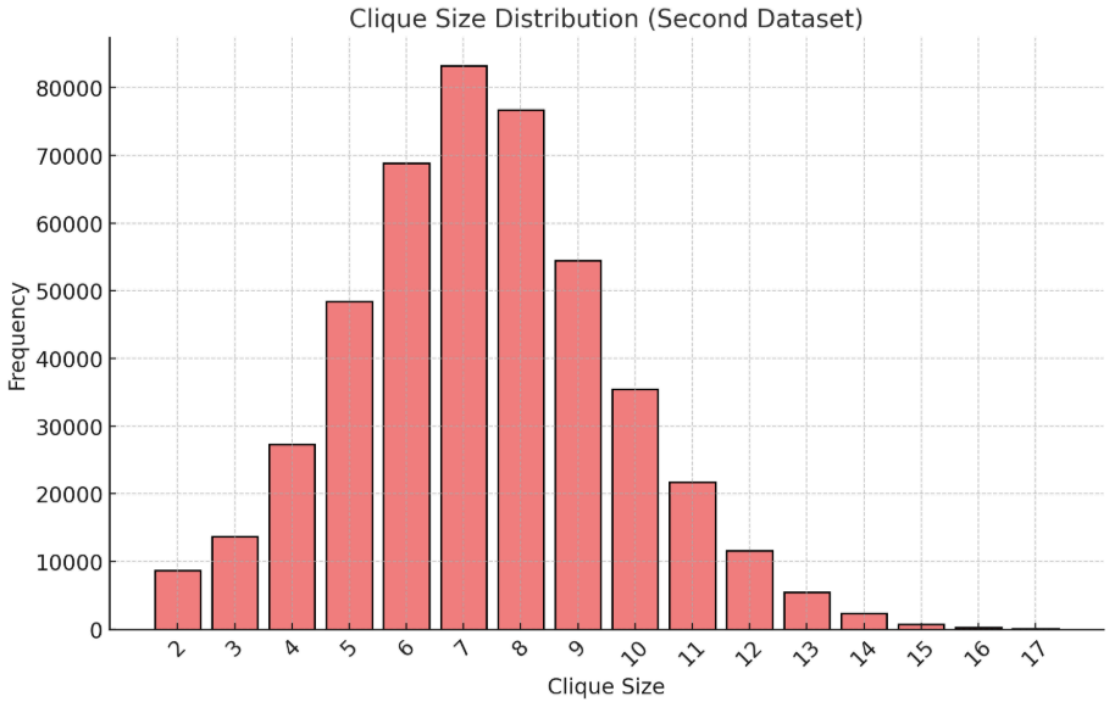
4.4 Clique Size Distribution (Histograms)

Histograms showing the distribution of different clique sizes for each dataset.

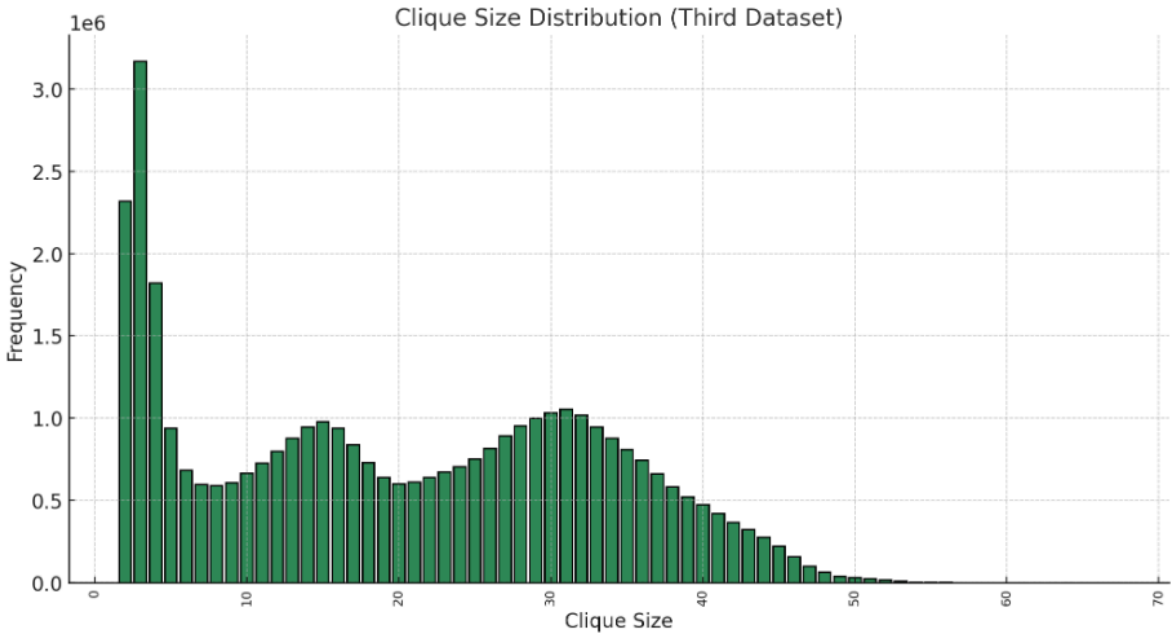
1) **Email-Enron Dataset Clique Size Distribution**



2) WikiVote Dataset Clique Size Distribution



3) Skitter Dataset Clique Size Distribution



5. Conclusion

The results highlight significant differences in the efficiency of the algorithms across datasets of varying sizes and densities. The Bron-Kerbosch algorithm consistently performs well, demonstrating its suitability for real-world networks with moderate clique density. The arboricity-based method, while theoretically promising, struggles with execution time, particularly on dense graphs, suggesting that its practical utility is limited to graphs with low arboricity. The number of maximal cliques varies significantly between datasets, indicating that the structural properties of graphs play a crucial role in algorithm performance. Overall, the findings emphasize the importance of choosing clique enumeration techniques based on the specific characteristics of the input graph, as no single approach is universally optimal.

7. References

1. Chiba, Norishige, and Takao Nishizeki. "Arboricity and subgraph listing algorithms." *SIAM Journal on Computing*, vol. 14, no. 1, 1985, pp. 210-223.

2. Tomita, Etsuji, Akira Tanaka, and Haruhisa Takahashi. "The worst-case time complexity for generating all maximal cliques and computational experiments." *Theoretical Computer Science*, vol. 363, no. 1, 2006, pp. 28-42. Elsevier.

3. Eppstein, David, Maarten Löffler, and Darren Strash. "Listing all maximal cliques in sparse graphs in near-optimal time." *arXiv preprint arXiv:1006.5440*, 2010.