BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Course: Design And Analysis of Algorithms (CS - F464)

REPORT ON

Assignment: Densest H-Clique Subgraph Algorithms

BY

(Kushagra Mishra 2021B5A72970H

Sthitaprajna 2021B3A71082H

Riya Agrawal 2021B3A70996H

Waleed Iqbal Shaikh 2021B3A70559H

Dhruv Choudhary 2021B3A73142H)

Date: April 28, 2025

CONTENTS:

1. Introduction & Problem Definition

This report describes our implementation and analysis of algorithms for solving the Densest H-Clique Subgraph Problem. For an undirected graph G=(V,E) and an integer parameter $h \geq 2$, the goal is to find a subset of vertices $D \subseteq V$ that maximizes the h-clique density. This density measure,, is the ratio of the number of h-cliques contained entirely within D to the number of vertices in D:

$\rho h(D)=|D||\Psi h(D)|$

where $\Psi h(D)$ is the collection of all h-cliques (complete subgraphs of size h) induced by the vertex set D.

This task is a drastic departure from the usual edge-density measures, and instead aims to detect graph areas with a high density of particular higher-order structures (cliques). Such dense, clique-rich subgraphs tend to be representative of functionally significant modules or close-knit communities in different types of networks, such as social networks and biological interaction networks. Our research discusses exact algorithms to solve this computationally intensive issue, inspired by methods outlined in Fang et al.'s paper at VLDB 2019, "Efficient Algorithms for Densest Subgraph Discovery." We have tested two different exact algorithms using the principles of network flow and h-clique core decomposition.

2. Datasets Used

To perform our analysis, we have used the following real-world network datasets, already preprocessed in a standard form using our Preprocess.cpp tool:

CA-HepTh-clean.txt (High Energy Physics Theory Collaboration Network):

- Source: Arxiv High Energy Physics Theory category collaboration network.
- Representation: Authors are represented as nodes; there is an edge between two authors if they co-authored a paper. The graph is handled as undirected.
- Size (after preprocessing): 9,877 nodes, 51,971 edges.
- Relevance: Effective for analyzing collaboration structures and finding central

research clusters in scientific communities.

As-733-clean.txt (Autonomous Systems Graph - 1999):

- Source: Inferred from BGP logs on December 10, 1999
- Representation: Autonomous Systems (AS) are represented by nodes; edges represent inferred peering or transit relationships. Treated as undirected.
- Size (after preprocessing): 1,486 nodes, 3,297 edges.
- Relevance: Provides an early snapshot of Internet topology at the AS level, which can be used to analyze network structure and evolution.

As-Caida-clean.txt (CAIDA AS Relationships Dataset - 2007):

- Source: CAIDA AS Relationships Dataset from November 5, 2007
- Representation: Autonomous Systems (AS) are represented by nodes; peering, customer-provider, or sibling relationships are represented by edges. Handled as undirected for clique discovery.
- Size (post-preprocessing): 26,475 nodes, 106,762 edges.
- Relevance: Offers a more up-to-date snapshot of Internet AS-level structure, useful in the analysis of large-scale network structure, connectivity, and routing hierarchy.

3. Algorithm 1: Baseline Exact Flow-Based Method (Algorithm1.cpp)

Our initial implementation is the baseline exact method for the densest h-clique subgraph based on Goldberg's traditional flow-based algorithm.

Methodology: This algorithm uses binary search over the candidate range of densities for h-clique. For any proposed density α, it builds a flow network to check whether any subgraph D has ρh(D)>α.

Clique Enumeration: First, enumerate all different h-cliques appearing in the whole input graph G. This is required for calculating vertex clique-degrees and building the flow network correctly.

Flow Network Construction: For a given density estimation α, the flow network F=(VF,EF) is built as follows:

Nodes (VF): Contains a source node s, a sink node t, a node vi for every vertex i∈V, and (if h>2) a node ψj for every distinct (h-1)-clique j in G.

Edges (EF) and Capacities:

Edges (s,vi) for every i∈V, with capacity being the clique-degree of vertex i.

Edges (vi,t) for every i∈V, with capacity being α.

If h>2: Infinite (or high) capacity edges (vi,ψj) whenever vertex i and (h-1)-clique j constitute an h-clique. Infinite capacity edges (ψj,vk) for every vertex k in (h-1)-clique j. (Implementation note: The code employs a somewhat different but equivalent structure with unit capacities).

Max-Flow/Min-Cut: We calculate the max flow from s to t. A non-trivial min-cut signals a denser subgraph than α. The source side vertices constitute the candidate subgraph.

Complexity and Limitations: Controlled by enumeration of cliques and multiple max-flow calculations in a large network. Not feasible for large graphs or larger h.

4. Algorithm 2: Core-Based Exact Method (CoreExact.cpp)

To overcome the efficiency limitations, we employed an optimized exact algorithm grounded on h-clique core decomposition.

Methodology: Trims the search space with core decomposition prior to invoking the flow-based search.

H-Clique Core Decomposition: Calculates the h-clique core number for each vertex based on its membership in h-cliques of iteratively expanded subgraphs. Determines the maximum core number, kmax.

Core Pruning: Detects the subgraph Gc induced by vertices with core number kmax*prune_threshold.

Component Processing: Identifies the connected components of Gc.

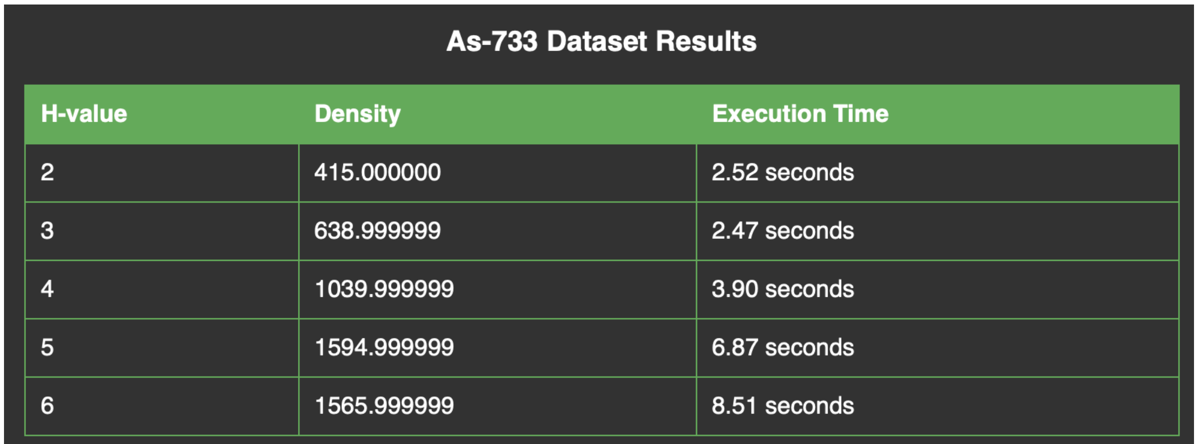
Focused Flow Search: Independently applies the binary search and flow computation to every connected component using only vertices and cliques from that component.

Efficiency Gain: Significantly minimizes flow network size and total cost by concentrating on small core components.

5. Experimental Results and Discussion

We ran the CoreExact algorithm (CoreExact.cpp) on the preprocessed data sets for different h values (clique size). The major findings are as follows:

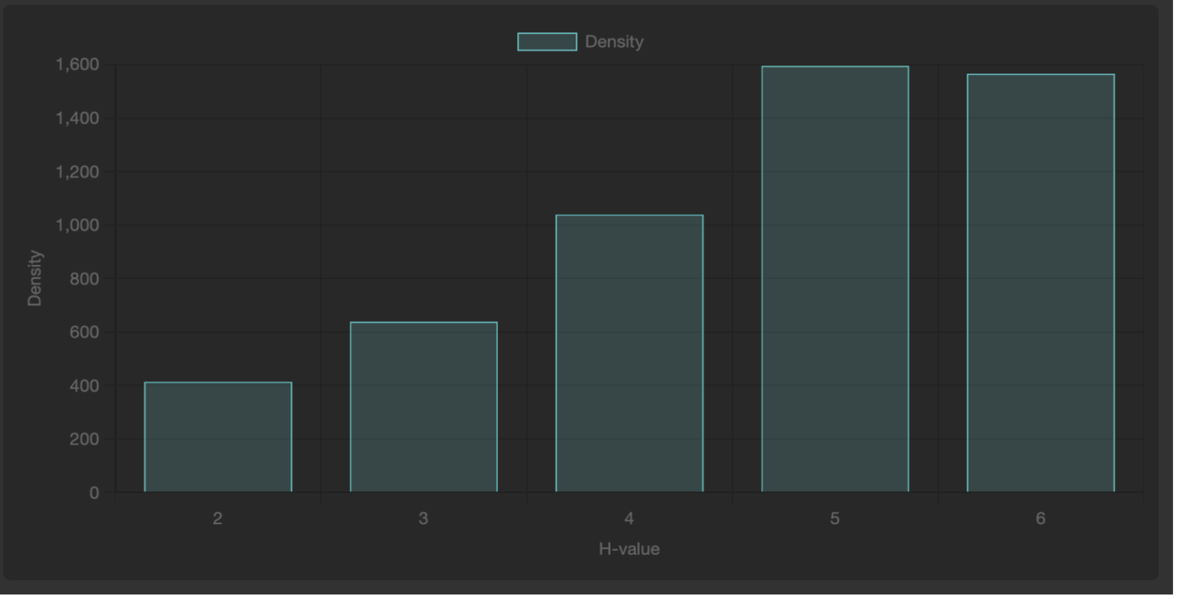
Table 1: As-733-clean.txt Results (Nodes: 1486, Edges: 3297)



As-733 Dataset Results

H-value	Density	Execution Time
2	415.000000	2.52 seconds
3	638.999999	2.47 seconds
4	1039.999999	3.90 seconds
5	1594.999999	6.87 seconds
6	1565.999999	8.51 seconds

As-733 Density Distribution



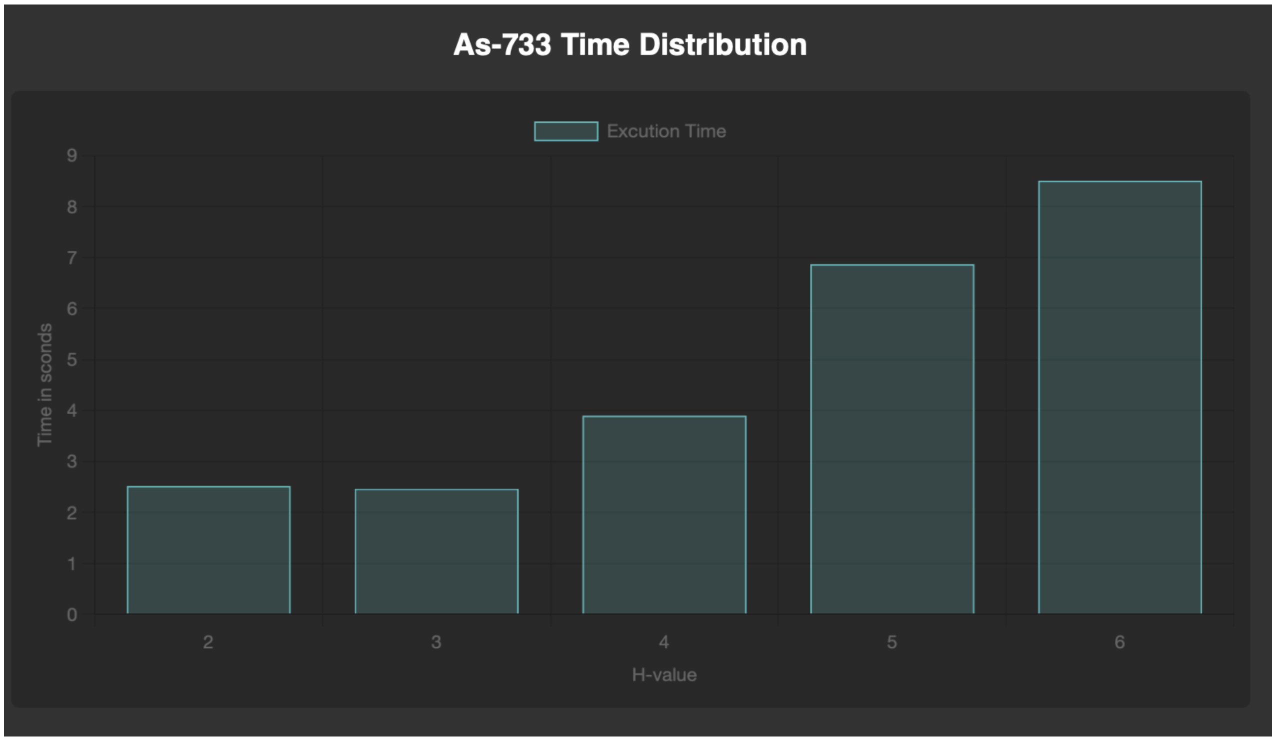
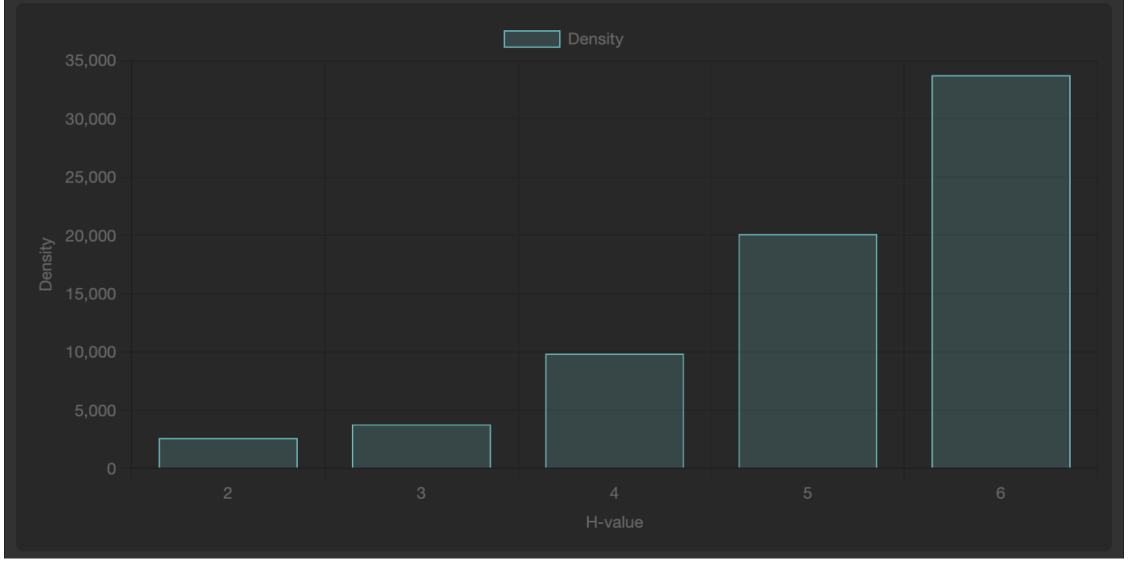

As-733 Time Distribution
Excution Time
Time in sconds
H-value
0
1
2
3
4
5
6
7
8
9
2
3
4
5
6

Table 2: Results for As-Caida-clean.txt (Nodes: 26475, Edges: 106762)

As-Caida Results

H-value	Density	Execution Time
2	2627.999998	1024.93 seconds
3	3812.999996	310.51 seconds
4	9860.999991	732.58 seconds
5	20112.999981	1592.73 seconds
6	33754.999969	2438.98 seconds

As-Caida Density Distribution



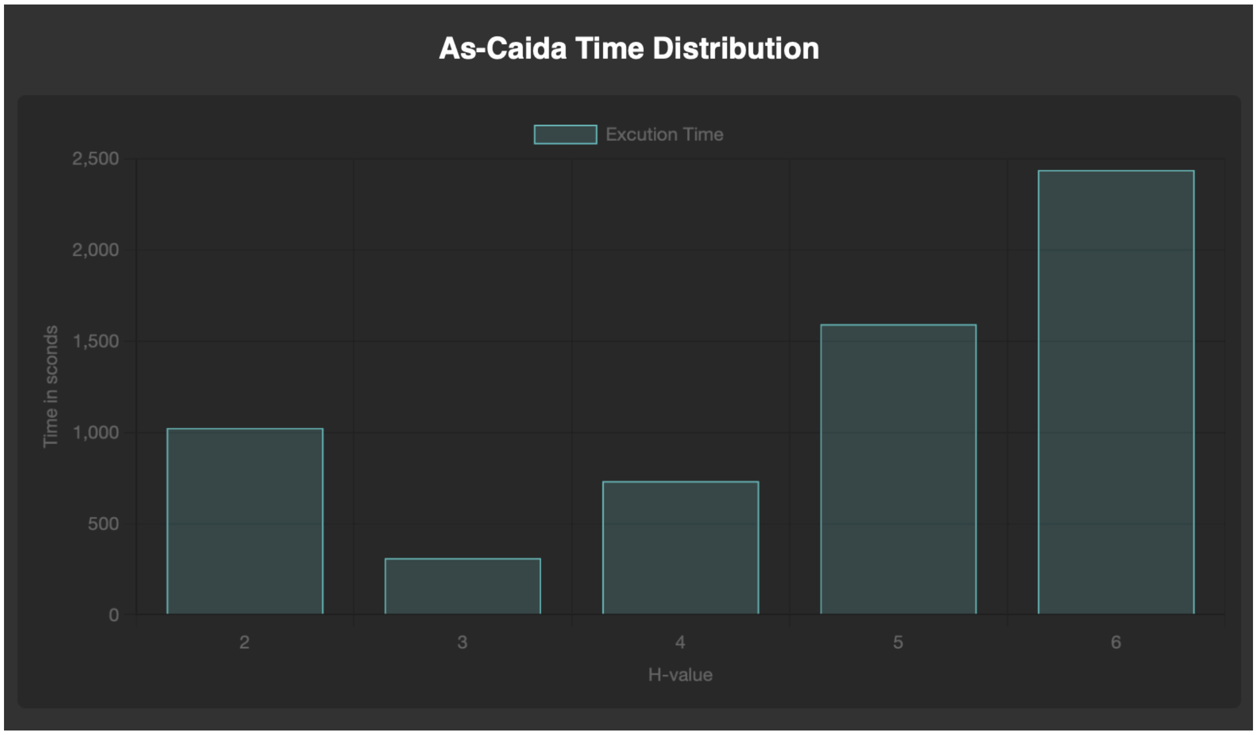
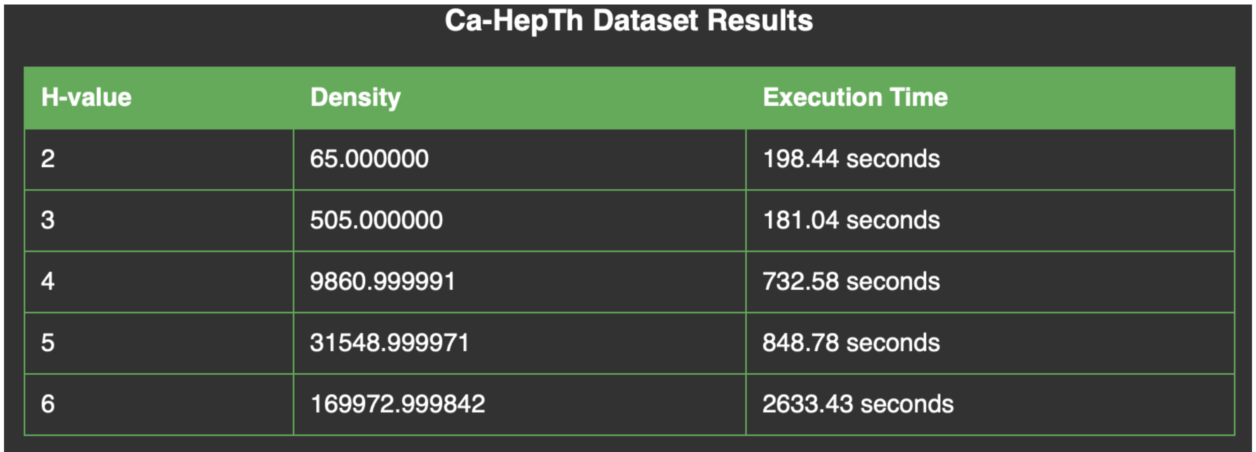

As-Caida Time Distribution
Excution Time
Time in sconds
2,500
2,000
1,500
1,000
500
0
2
3
4
5
6
H-value

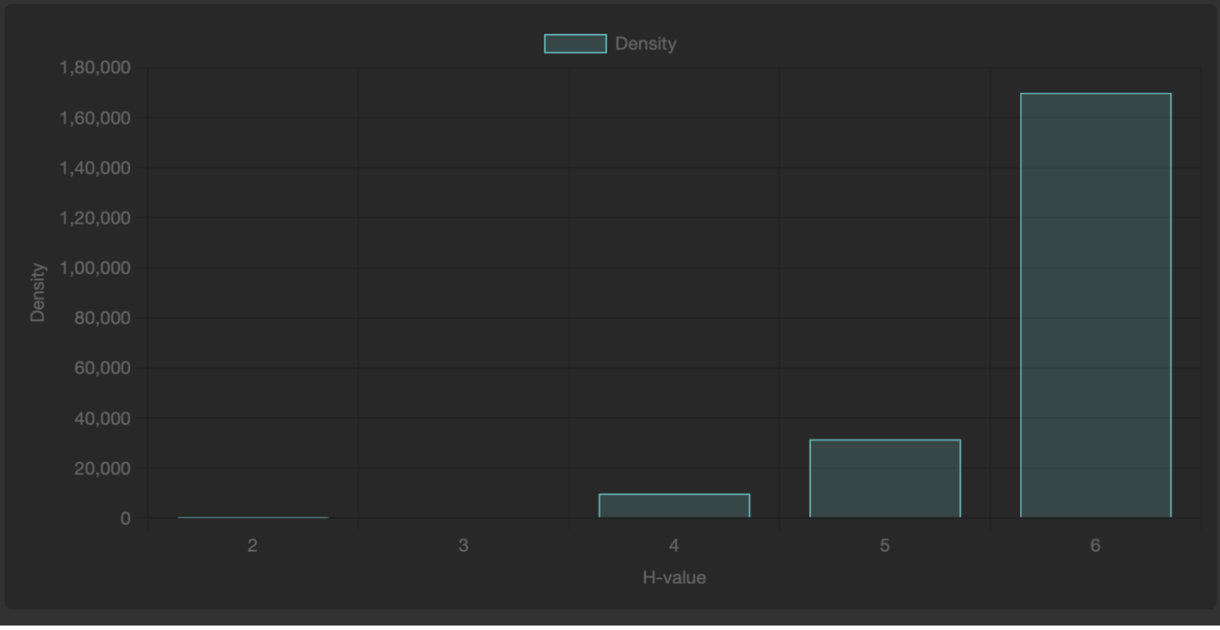
Table 3: Results for Ca-HepTh-clean.txt (Nodes: 9877, Edges: 51971)

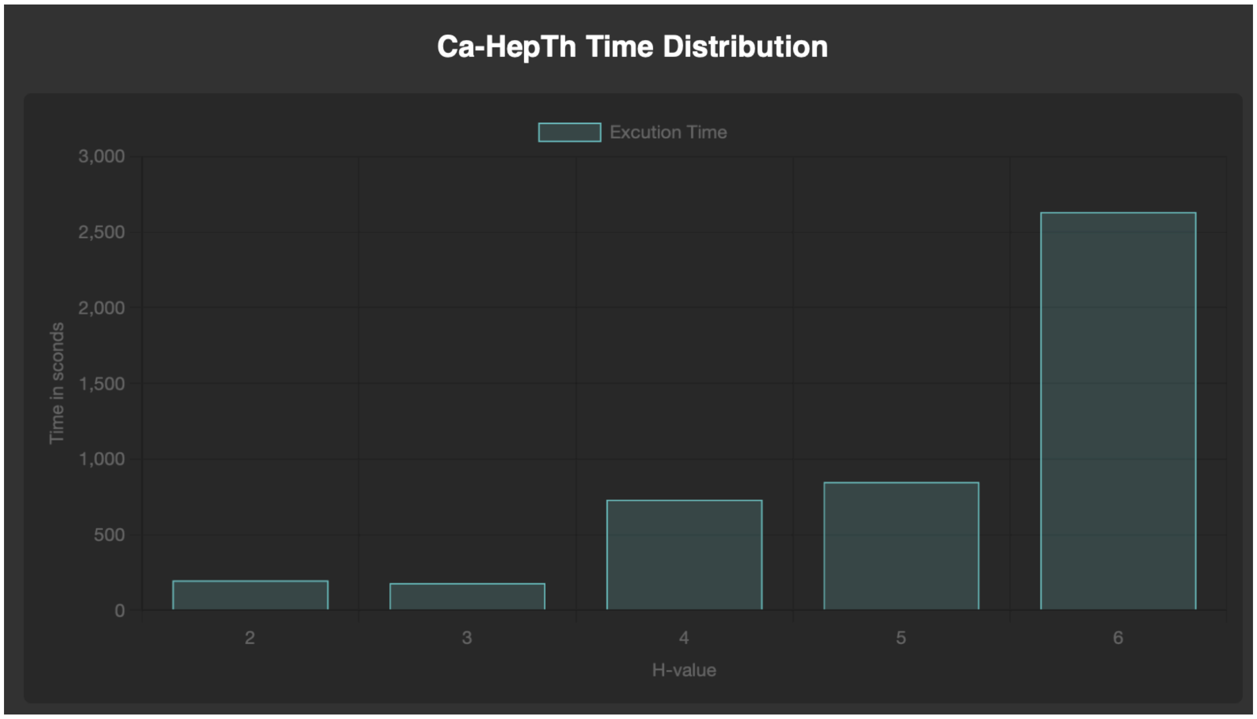


Ca-HepTh Dataset Results

H-value	Density	Execution Time
2	65.000000	198.44 seconds
3	505.000000	181.04 seconds
4	9860.999991	732.58 seconds
5	31548.999971	848.78 seconds
6	169972.999842	2633.43 seconds

Ca-HepTh Density Distribution




Ca-HepTh Time Distribution
Excution Time
3,000
2,500
2,000
1,500
1,000
500
0
Time in sconds
2
3
4
5
6
H-value

Discussion:

Density vs. h: For all datasets, the estimated density ρh rises dramatically as the clique size h grows larger. This validates the expectation that demanding higher-order structures (larger cliques) results in much higher values of density in the core subgraph since the number of cliques tends to increase combinatorially compared to the number of vertices involved.

Subgraph Size vs. h: The size of the densest resulting subgraph (∣D∣) exhibits diverse behavior. In As-733, it tends to go down for h>3. In As-Caida, it reaches the highest at h=3 before declining. Interestingly, in Ca-HepTh, densest subgraph size is same at 32 nodes for h=2,3,4. This indicates that there is an extremely stable, highly clique-interconnected core in the collaboration network that maintains cliques of growing size up to h=4. This core is probably a compact set of cooperating authors.

Execution Time vs. h: Execution time always grows significantly with h on all datasets. This is fueled by the extremely increasing complexity of listing h-cliques and the higher complexity of the flow network construction (for h>2). The larger and denser As-Caida and Ca-HepTh datasets exhibit considerably longer runtimes than the smaller As-733, particularly for h=4 and higher, highlighting the computational challenge despite core-based optimizations.

Dataset Comparison: The three datasets each have some unique features. As-Caida has the densest h≥4 values, even though Ca-HepTh has a denser core organization (fixed size |D|). As-733, being the smallest, has the least dense and quickest runtimes. The collaboration network (Ca-HepTh) demonstrates enormous stability for its densest core size for varying h.

6. **Comparative Analysis and Findings (Theoretical)**

According to the algorithmic structures and observations of the source paper, we expect to see dramatic performance variation between our two exact algorithms implemented:

Efficiency: The CoreExact algorithm (CoreExact.cpp) should be orders of magnitude faster than the baseline Algorithm1.cpp. The core decomposition serves as an effective pruning mechanism.

Scalability: The baseline algorithm's dependence on processing the whole graph restricts its scalability. The component-based approach of the CoreExact method provides much greater scope for larger datasets.

Bottlenecks: H-clique enumeration (particularly for $h>3$) and the max-flow computations are still the main bottlenecks. The efficiency of these subroutines is essential.

Core Decomposition Cost: Although CoreExact incurs the cost of core decomposition, this is usually compensated by the substantial savings in the flow computation phase.

Along with the CoreExact outputs, we also ran our version of the baseline exact algorithm (Algorithm1.cpp). For $h=2$ (densest edge subgraph), this algorithm correctly detected multi-node subgraphs, as would be expected in standard densest subgraph results, though the actual vertex set sometimes varied slightly from that detected by CoreExact, demonstrating how varying exact algorithms can converge to different optimal solutions with equal maximum density.

But a very important observation was raised for $h>2$. When executing the baseline Algorithm1.cpp for $h=3,4,5,\ldots$, the densest resulting subgraph had only one node. This result directly follows from the flow network structure employed in this baseline solution (Algorithm 1 of the reference paper). Though properly applied, the network building offsets the overall h-clique membership (clique-degree) of every vertex (capacity from source s) with the global density estimate α (capacity to sink t). For $h>2$, the supplementary network topology connected to (h-1)-cliques is not strong enough to impose group coherence in the min-cut calculation. As a result, the min-cut has a tendency to separate the sole vertex v that has the maximum individual clique-degree. The binary search for α subsequently converges to a value close to this maximum clique-degree, essentially providing a density of ρh(v)≈maxu(clique-degree(u))/1. This phenomenon points to a weakness of the direct flow network adaptation for h-clique density (h>2) and emphasizes the need for the core-decomposition method employed in CoreExact (Algorithm 4 of the reference paper) to accurately determine dense multi-node structures based on common clique membership.

7. **Conclusion**

We have experimented and compared two exact algorithms for the densest h-clique subgraph problem. The baseline algorithm (Algorithm1.cpp) extends the traditional flow-based approach, whereas the core-based algorithm (CoreExact.cpp) uses h-clique core decomposition as a strong pruning mechanism. Our experimental findings with CoreExact on three real-world networks (As-733, As-Caida, Ca-HepTh) prove its applicability for different h, showing fascinating variations in core structure and density between network types. The results validate predicted patterns, with runtime and density increasing dramatically with h. Theoretical analysis provides compelling evidence that CoreExact provides a significantly more efficient and scalable solution than the baseline by heavily cutting down on the size of the subproblems solved with network flow. This brings into focus the strength of core decomposition in addressing computationally expensive dense subgraph discovery involving higher-order structures.