Sanjita Salunkhe
(#774512756)

CPR E 525
Project Report

Finding the number of cliques in a graph and reporting the speed up achieved using MPI.

Abstract

Due to the recent advances in technology and the ever-increasing demand for data, coming up with new effective models for data collection has gained a lot of attention. High-throughput data collection for different applications such as social network analysis, text analysis, and internet research is essential as data mining consists of summarizing and processing large data sets for extracting important knowledge from the data using advanced mathematical techniques [6].
This is when cliques come into play as they can identify the “tightly knit” clusters. In this project, I will find the number of cliques in a graph based on the input graph that I give. The input will be an undirected binary graph. I will be parallelizing the code and reporting the speed up I achieved after implementing it using MPI. For this project, I find cliques of sizes 3 and 4. Further research could be finding cliques of larger sizes where the speed up would be more evident or finding all the maximal cliques.

Introduction

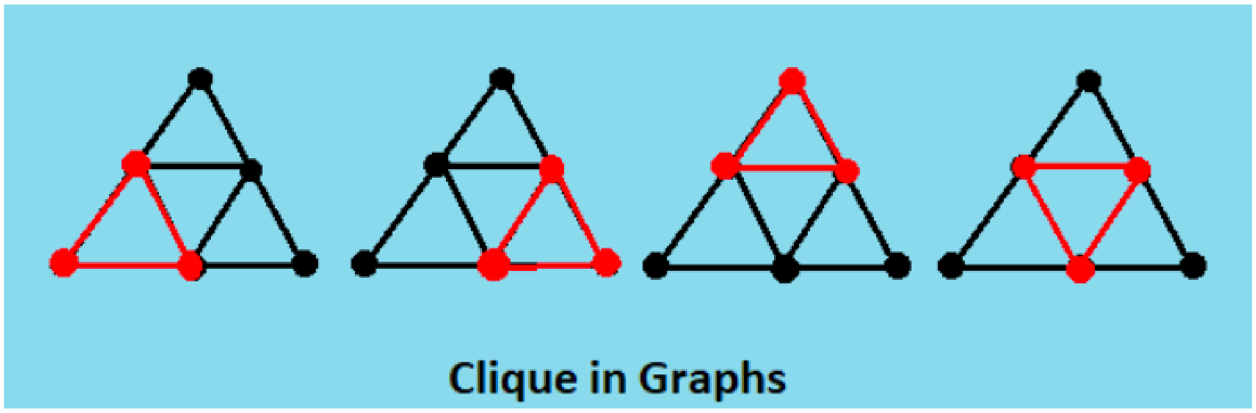


Fig. 1 Cliques in graphs

A clique is defined as the subset of vertices in an undirected graph G, given that every 2 distinct vertices are adjacent [6]. It can also be defined as a subgraph of a graph. The only condition for a subgraph to be a clique is every vertex should be connected to every other vertex by an edge of any weight. A maximum clique exists in every graph, which is a clique of the largest size. There might arise some confusion as a maximum clique might simply be referred to as a “clique.” A maximum clique, on the other hand, cannot be enlarged by simply adding one more adjacent vertex [6]. As a result, maximum cliques are maximal cliques, but this may not always be the case the other way around.

The equation for a clique polynomial [2] for a graph G is given by,

$$C_G(x) = \sum_{k=0}^{\omega(G)} c_k x^k$$

Where, c_k is the number of cliques of size k with $c_0 = 1,\ c_1 = |G|$ equal to the vertex count of G, $c_2 = m(G)$ equal to the edge count of G, etc.

Cliques find their applications in various fields, from data mining to spam detection to mathematical problems [5]. With the ever-increasing demand for data, developing new and advanced data collection techniques is necessary. Advanced mathematical tools help in designing such data collecting tools. However, certain methods can deal almost exclusively with numerical data, which is not always the case. This is where graph-based data mining proves advantageous because it is used for studying unstructured or partially structured data. Cliques play a major role in identifying the "tightly knit clusters" that help provide valuable insights into various application settings. The different application settings are – Internet graphs, stock market graphs, call graphs, social network analysis, text analytics, bioinformatics, computational finance, and telecommunication, among others [5].

Finding cohesive subgroups in contact networks has several practical uses in social network analysis. Criminal network analysis, for example, can discover organized criminal activities like money laundering and terrorism. Disease contagiousness studies have also employed the method of detecting coherent clusters in interaction networks. For advertising and marketing reasons, mining friendship networks and locating dense clusters can give important information. Cluster detection in protein interaction networks aids in the identification of protein complexes and functional modules that have an impact on cellular activities. Finding cliques in graph models of electroencephalogram time series data has also been used to aid in the detection of epileptic episodes.

The computational problem of detecting cliques in a graph is known as a clique problem [6]. Popular versions of the clique issues include finding a maximum clique, finding a maximum weight clique in a weighted graph, listing all the maximal cliques, and addressing the decision question of verifying if a graph includes a clique higher than a certain size. Because there exist graphs with an infinite number of maximal cliques, listing them all might take an infinite amount of time.

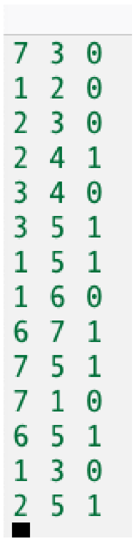
Theorems regarding clique [6]

1. Turán's theorem states that "if a graph has sufficiently many edges, it must contain a large clique."
2. Ramsey's theorem states that "every graph or its complement graph contains a clique with at least a logarithmic number of vertices."
3. Moon and Moser (1965) proved that a network with 3n vertices could only contain 3n maximum cliques. The Moon–Moser graphs are those that satisfy this constraint.

Method/Description of code

First, I give it an input file with the nodes and vertices, as shown in the figure below (Fig. 2). The program for finding 3 cliques then starts checking for all the combinations (e.g., 1 2 3, 1 2 4, 1 2 5, …), after which it checks for an edge between the vertices to see if it forms a clique. If there is a connection, the program will return the value of its weight, and if there is no connection, it will return a -1. It performs similarly for a clique of size 4.

I have stored all the edges in adjacency format (2D matrix). If there is no edge between 2 vertices, it will be -1; else, the weight of the edge is stored. After storing all the edges, I will be checking for 3 vertices clique using for loops, and according to their weights, I will be updating the count at each process. After that, I will be doing that similar for the 4 vertices clique. Now I will be reducing the count matrix, and the 0th rank process will update the final output.



```
7 3 0
1 2 0
2 3 0
2 4 1
3 4 0
3 5 1
1 5 1
1 6 0
6 7 1
7 5 1
7 1 0
6 5 1
1 3 0
2 5 1
```

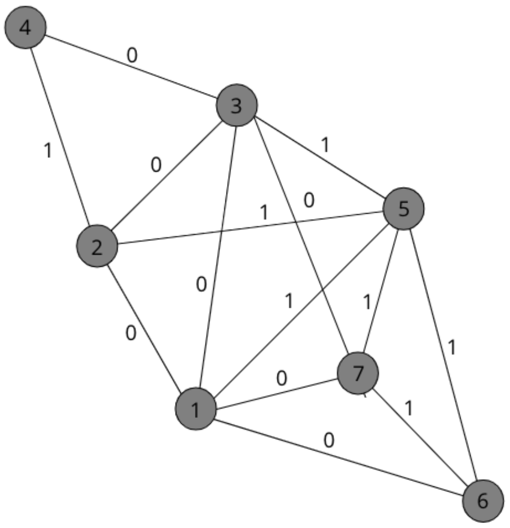


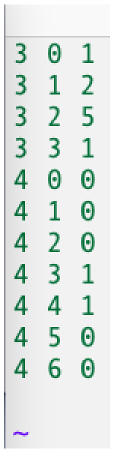
Fig. 2 Input graph network

Algorithm used

Clique of fixed size

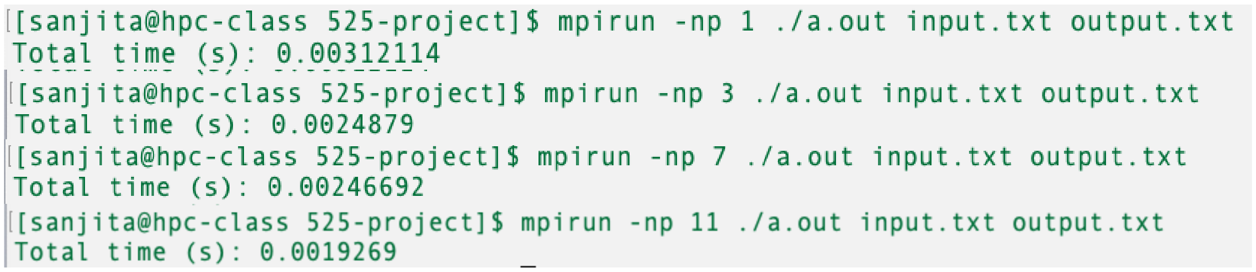
Using a brute force technique, we can locate any k-vertex clique in a network G. This method evaluates each k-vertices subgraph and determines if it constitutes a clique. It takes O(nkk2). This is because there are O(nk) subgraphs to examine, each with O(k2) edges whose presence in G must be verified. When k is a fixed constant, the issue can be solved in polynomial time. On the other hand, when k does not have a constant value and instead varies as part of the problem's input, the time is exponential [6].

Results



```
3 0 1
3 1 2
3 2 5
3 3 1
4 0 0
4 1 0
4 2 0
4 3 1
4 4 1
4 5 0
4 6 0
~
```

As seen on the left, the first column of the output is the clique size, the second column is the weight of the clique, and the third column reports the number of such cliques.
Example - If we refer to the graph above, we observe that there is one clique of size 4, which has a weight of 3 formed by the vertices - 3 5 7 1 and the weights of 3 5 (1), 5 7 (1), and 1 5 (1) are summed together. Another example is a clique of size 4 with a weight of 4 formed by the vertices – 1 5 6 7 and the weights of 1 5 (1), 5 7 (1), 5 6 (1), and 6 7 (1) are added together. After getting the output for cliques of sizes 3 and 4, I also checked for the speedup by changing the number of processors. As seen below, there is a minute difference in the total time taken to execute the code for 1, 3, 7, and 11 processors. Since I have considered very small clique sizes, the difference in time is not much, but when it comes to finding cliques of 100 or 1000, there is a noticeable difference in the time taken.



```
[sanjita@hpc-class 525-project]$ mpirun -np 1 ./a.out input.txt output.txt
Total time (s): 0.00312114
[sanjita@hpc-class 525-project]$ mpirun -np 3 ./a.out input.txt output.txt
Total time (s): 0.0024879
[sanjita@hpc-class 525-project]$ mpirun -np 7 ./a.out input.txt output.txt
Total time (s): 0.00246692
[sanjita@hpc-class 525-project]$ mpirun -np 11 ./a.out input.txt output.txt
Total time (s): 0.0019269
```

Fig. 3 Output reporting the number of cliques of sizes 3 and 4 & speed up for a different number of processors

Conclusion and Future Work

In this project, I implemented a program to find the cliques of sizes 3 and 4. To parallelize the code for better efficiency, I used MPI. Although the difference in the total time taken for the program to execute for a different number of processors was not very significant, it could be a part of future research. Since I implemented the code to find cliques of very small sizes (3 and 4), the speed up was not evident. However, when performing a similar operation to find larger clique sizes, for instance, 1000, the speedup will be evident, and there will be more need for parallelization.
Future work could aim at finding all the maximal cliques. Because locating a single maximum or large clique is a more difficult algorithmic challenge, it receives more attention than locating a single maximal clique. We can use an easy greedy algorithm to discover a single largest clique. We can use a brute force technique to determine the greatest clique by inspecting all of the subsets, but this is time-consuming if the network contains more than a dozen vertices. The Bron-Kerbosch algorithm, which lists all the maximum cliques in polynomial time per clique, is another option [6].

References

[1] Balasundaram B., Pajouh F.M. (2013) Graph Theoretic Clique Relaxations and Applications. In: Pardalos P., Du DZ., Graham R. (eds) Handbook of Combinatorial Optimization. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_9
[2] https://mathworld.wolfram.com/Clique.html
[3] https://en.wikipedia.org/wiki/Clique_(graph_theory)
[4] http://maheshgadgilsblog.blogspot.com/2011/11/why-clique-problem-is-important-think.html
[5] https://en.wikipedia.org/wiki/Clique_problem
[6] https://iq.opengenus.org/clique-in-graphs/
[6] Gianinazzi, Lukas, et al. “Parallel Algorithms for Finding Large Cliques in Sparse Graphs.” Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures. 2021.
[7] Szabó, S. “Parallel algorithms for finding cliques in a graph.” Journal of Physics: Conference Series. Vol. 268. No. 1. IOP Publishing, 2011.
[8] Cheng, James, et al. “Finding maximal cliques in massive networks.” ACM Transactions on Database Systems (TODS) 36.4 (2011): 1-34.
[9] Wang, Junjie, Shuigeng Zhou, and Jihong Guan. “detecting potential collusive cliques in futures markets based on trading behaviors from real data.” Neurocomputing 92 (2012): 44-53.
[10] G. Hua, H. Liao, H. Zhang, D. Ye and J. Ma, “Robust ENF Estimation Based on Harmonic Enhancement and Maximum Weight Clique,” in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3874-3887, 2021, doi: 10.1109/TIFS.2021.3099697.
[11] Hahn, Klaus, Peter R. Massopust, and Sergei Prigarin. “new method to measure complexity in binary or weighted networks and applications to functional connectivity in the human brain.”BMC bioinformatics 17.1 (2016): 1-18.