Detecting Cliques Using Degree and Connectivity Constraints

Abhay Avinash Bhamaikar[1]and Pralhad Ramchandra Rao[2]

[1]Department of Information Technology, Shree Rayeshwar Institute of Engineering and Information Technology, Shiroda, Goa, India
abhay_bhamaikar@rediffmail.com
[2]Department of Computer Science and Technology, Goa University, Taleigao Plateau, Goa, India
pralhadrrao@gmail.com

Abstract

In graph mining determining clique is np complete problem. This paper introduces pruning strategies, by which linear time algorithm for detecting clique is obtained. Clique determination is widely applicable in social network analysis. In social network analysis cliques signifies that each person in the network knows every other person in the group. Here pruning is done using edge connectivity and degree constraints. Initially the graph (g) is checked for a bridge, if it is detected, then graph (g) is disconnected. Then minimum and maximum degree criteria are used to determine a clique. The algorithm also has wide application in bioinformatics.

Keywords

Cliques, Maximum Degree, Minimum Degree, Connectivity.

1. INTRODUCTION

Due to the application of graph data in numerous applications graph mining is gaining tremendous importance. A graph is composed of set of vertices V_G and a set of edges E_G. Each vertex has a label (or a identifier) and each edge $e_{i,j} \subseteq E_G$ connects the vertices v_i and v_j .In some application the labels are unique and the graph is termed as relational graph whereas if the labels are not unique then the graph is said to be non relational graph. In this we have restricted ourselves to relational graph.

The clique problem refers to finding a complete subgraphs ("cliques") in a given graph, i.e., sets of elements where each pair of elements is connected. For example, the maximum clique problem arises in the following real-world setting. Consider a social network, where the graph's vertices represent people, and the graph's edges represent mutual acquaintance. To find a largest subset of people who all know each other, one can systematically inspect all subsets, a process that is too time-consuming to be practical for social networks comprising more than a few dozen people. Although this brute-force search can be improved by more efficient algorithms, all of these algorithms take exponential time to solve the problem. Therefore, much of the theory about the clique problem is devoted to identifying special types of graph that admit more efficient algorithms, or to establishing the computational difficulty of the general problem in various models of computation. Along with its applications in social networks, the clique problem also has many applications in bioinformatics and computational chemistry.

Clique problems include: finding the maximum clique (a clique with the largest number of vertices),finding a maximum weight clique in a weighted graph, listing all maximal cliques (cliques that cannot be enlarged),solving the decision problem of testing whether a graph contains a clique larger than a given size.

These problems are all hard: the clique decision problem is NP-complete, the problem of finding the maximum clique is both fixed-parameter intractable and hard to approximate, and listing all maximal cliques may require exponential time as there exist graphs with exponentially many maximal cliques. Nevertheless, there are algorithms for these problems that run in exponential time or that handle certain more specialized input graphs in polynomial time.

We propose an algorithm to determine cliques in the given graph in linear time.

2. Related Works and Contribution

Pardalos et al. review the development of algorithms for enumerating the maximal cliques of a graph. In 1970, Augustin and Minker found the algorithm of Bierstone to be the most efficient among those tested. In 1973, Bron and Kerbosch, proposed a backtracking algorithm including two clever heuristics for pruning the search tree. Bron and Kerbosch tested their algorithm along with that of Bierstone on random graphs of 10 to 50 nodes and edge densities ρ = 10% - 95%. They found that their algorithm was markedly faster than the Bierstone algorithm.

In 1976, Johnston [Johnston] tested variations of the Bron-Kerbosch algorithm on small graphs of up to 48 vertices with edge-densities of 0% to 100%, large graphs of up to 1000 vertices with edge-density of 5%, and graphs of 100 vertices and densities ranging from 5% to 50%. The variation that was most efficient for large graphs with density less than 25% was called the *simplified Bron-Kerbosch* algorithm. In general, this version was several times faster than the original Bron-Kerbosch algorithm.

In 1977, Tsukiyama *et al.* introduced an algorithm for enumerating all of the maximal independent sets of a graph, based on Bierstone's vertex sequence method and a new backtracking method. Tsukiyama et al. showed that their algorithm had a theoretical time bound of O(nmμ), where μ is the number of MISs, n is the number of vertices, and m is the number of edges. In 1981, Loukakis and Tsouros presented a new backtracking algorithm for listing maximal independent sets. In 1985, Chiba and Nishizeki developed an algorithm for listing all the maximal cliques of a graph in O(a(G)m) time per clique, where a(G) is the *arboricity* of graph G. The most recent improvement on the time complexity of listing cliques is by Tomita et al. The algorithm which Tomita et al. discuss is basically the Bron-Kerbosch algorithm, with an interesting change in the way that the cliques are reported. They calculate the complexity of this modified Bron-Kerbosch algorithm to be O(3n/3), which is optimal.

The problem we study in this work is formally stated as follows: Given a Relational Graph G, we determine whether the induced subgraph G' is Clique over Connectivity and Degree Constraint.

3. Basic Definitions

Definition 1 Graphs: A graph G = (V,E) is a pair in which V is a (non-empty) set of vertices or nodes and E is either a set of edges E $\subseteq$ {{v,w} | v, w ε v, v ε w} or a set of arcs E $\subseteq$ {(v,w) | v, w ε v, v ε w}. In the latter case we call the graph directed.

Definition 2 A **clique** in an undirected graph $G = (V, E)$ is a subset of the vertex set $C \subseteq V$, such that for every two vertices in C, there exists an edge connecting the two. This is equivalent to saying that the subgraph induced by C is complete.

Definition 3 Maximum Degree (Δ) of a given Graph (G) is defined as the largest degree over all the vertices.

Definition 4 Minimum degree (δ) of a given Graph (G) is defined as the smallest degree over all the vertices.

Definition 5 The edge connectivity, λ *(G)*, of a connected Graph G is the minimum number of edges whose removal results in two connected subgraph.

Lemma One

Consider a Graph G' be the induced subgraph having maximum degree = Δ, Minimum Degree = δ and Number of nodes (n).If maximum degree is equal to Minimum Degree ($\Delta = \delta$) and number of nodes = $\delta + 1$ then the given graph is induced subgraph is clique.

Proof: Let G' be the induced subgraph obtained from graph (G).
Let Maximum Degree = Δ, Minimum Degree = δ and Number of nodes = n of the induced graph G'.
Since Maximum Degree = Minimum Degree ($\Delta = \delta$) It implies the degree of all the nodes of the induced subgraph G' is same (i.e. $\delta = \Delta$).
For clique to exist, each node in the graph must be connected to every other in the Graph.
Since Number of Nodes = $\delta + 1$, this implies that each node in the graph is connected to every other node in the graph.
Hence the induced subgraph is clique.

Lemma Two

If Given Graph Contains a Bridge then the Edge forming the bridge will not form the Clique.

Proof: Consider a Graph G, Let G' be the induced clique having n vertices obtained after removal of Bridge. The Edge connectivity of the Graph G Containing Bridge is one. The Edge Connectivity of Clique having number of vertices equal to n is (n – 1). Hence the edge containing bridge will never form edge of an clique.

4. Algorithm and Explanation

Algorithm Clique Detection (G')

Input: G, Initial Input Graph
Output: Clique (G') having degree ≥ 3

```
1. Initialize queue Q;
2. Insert connected components of G into Q;
3. If Graph (G) contains one or more bridges
4.         Then disconnect G by removal of Bridges
5.        Insert Graph G' into Q
6. Endif
7. If Minimum Degree == 2
```

```
8.      Then remove node with degree == 2
9.      Calculate Maximum Degree (Δ) and Minimum Degree (δ).
10.           If Maximum Degree = = Minimum Degree && Number of nodes = δ + 1
11.              Then G' is Clique
12.          Endif
13. Else if go to step 15
14. Endif
15. Initialize Graph (G')
16. Label Node from (1 to n)
17. Initialize pointer to Node labelled 1
18. Determine all nodes connected to Node Labelled 1
19. Check if all nodes connected to Node labelled 1 are interconnected
20. If Yes then given graph has "Clique" && Degree = Degree of Node Labelled 1
21.       Else Increment the pointer && go to step 5
22. Endif
```

Explanation

Line 3 to Line 6: Here the given Graph (G) is checked for bridge. If a bridge is detected then the graph is decomposed so as to obtain subgraph G'.

Line 7 to Line 8: Here if the minimum Degree of the Graph (G) equals two then the node having degree equal to two is deleted.

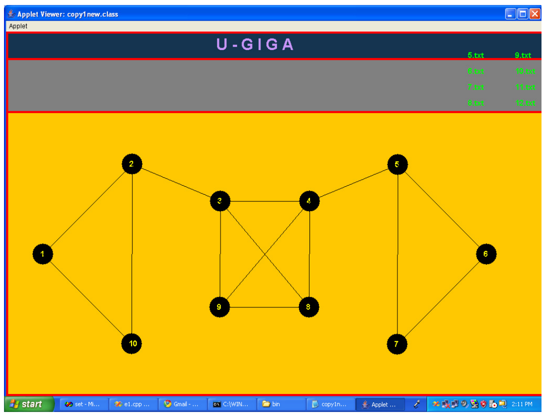
Line 9 to Line 12: Here Maximum and Minimum Degree of the graph is calculated. If the Maximum degree equals Minimum degree and the number of nodes equals maximum degree + 1, then the given graph is Clique.

Line 13 to Line 14: If the above steps does not detect the Clique, then the induced subgraph is initialised to step 15.

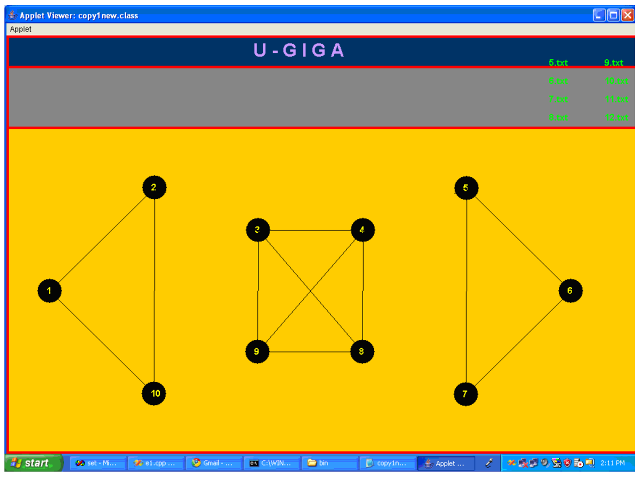
Line 15 to Line 22: Here each node is labelled, The node labelled 1 is initialised and all the neighbouring nodes to which the node 1 is connected is checked. Then each node is checked for connectivity with the connected nodes.

Snapshots

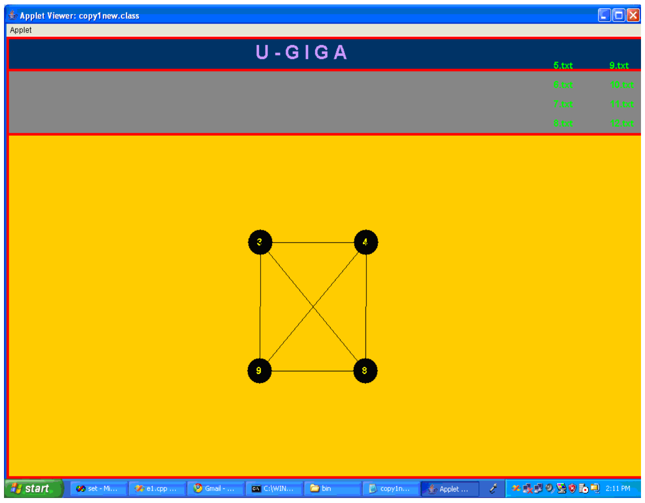
Initial Graph G containing 10 nodes



STEP 1: Remove the bridges. In the above graph 2-3 and 4-5 are bridges.



STEP 2: Remove the nodes with degree < 3.Remove the nodes: 1,2,5,6,7,10 having degree 2.After removing these nodes we get the following graph.



Step 3: Now for the resultant graph check for the condition If (number of nodes = = degree of node +1) {Nodes that are present form a clique}

Performance Evaluation

The software "U-GIGA" project aims at providing Graphical User Interface for various graph mining algorithms. The software has been implemented in JAVA (jdk1.5.0 or jdk1.6.0_12) and the graphs and their corresponding attributes are displayed to the user using the JAVA applet-viewer. All the graph algorithms have been performed on an Intel Core i3 processor at 3.20GHz, with 3GB RAM running Windows XP. The software basically checks for graph parameters such as order, size, etc. that are stored or saved in a foreign source (i.e. in a text file) and using this information about the graph, various graph attributes such as vertex/edge connectivity, clique detection, time complexity, etc. are calculated and displayed on the user interface. The software aims at investigating performance by varying the graph parameters and also keeps a track on the time taken to evaluate a graph (i.e. the time complexity of the software).

The various phases of the software for evaluating and displaying a particular graph scanned are as follows: Scan the text file, which the user chooses on the interface (by clicking the corresponding file link) and store the graph data to the corresponding variables. Using these variables, perform the following task and calculate the following graph attributes:
Define and store a particular path in variables, so as to draw and display the graph, using the information from these variables, on the user interface.

Calculate the number of bridges present in the graph
1. Calculate the edge connectivity.
2. Calculate the cliques detected.
3. Check if the given graph is a tree or not.
4. Calculate the minimum and maximum degree of the graph.
5. Check if the graph is monotonic, using information of edge connectivity, cliques, tree information.

These graph attributes are then displayed to the user.

This software is further improved so as to detect the clique using additional pruning techniques.

The software also Displays Computational time.

4. Conclusions

In this paper we proposed a novel way to detect if the given graph satisfies monotonic property over edge connectivity constraint. We developed an efficient algorithm Clique Detection G' which detects clique in the given graph G. Software U – GIGA using Java Programming Language has been designed, which assists in detecting the Clique. It also helps in determining the value of Maximum Degree, Minimum Degree and number of nodes of the obtained induced subgraph (G'). It also determines a bridge and disconnects the given Graph by removal of bridges.

References

[1] A N.Papadopoulos, A. Lyritsis, and Y Manalopoulos, "SkyGraph: an algorithm for important subgraph discovery in relational graphs", DMKD (2008), Springer, 23 Jun 2008, pp. 57-76.

[2] Cook DJ, Holder L B (eds) (2007) Mining graph data. Wiley, London.

[3] Hartuv E, Shamir R (2000), A Clustering algorithm based on Graph Connectivity. Inform Process Lett 76: 175-181.

[4] Wu Z, Leahy R (1993) An optimal graphs theoretic approach to data clustering: theory and its application to image segmentation, IEEE Trans Pattern Anal Machine Intell 15(11): 1101-1113.

[5] Augustin, J.G. and Minker, J. (1970) An analysis of some graph theoretical cluster techniques. *J. Assoc Comput. Mach*:, 17: 571-588.

[6] Booth, K.S. and Lueker, G.S. (1976) Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-tree Algorithms. *Journal of Computer and System Sciences*, 13: 333-379.

[7] Bron,C. and Kerbosch,J. (1973) Finding All Cliques of an Undirected Graph. *Communications of the ACM*, 16: 575-577.

[8] Chiba,N. and Nishizeki,T. (1985) Arboricity and Subgraph Listing Algorithms. *SIAM J. Comput.*, 14:210-223.

[9] Garey,M.R. and Johnson,D.S. (1979) *Computers and Intractability*. Freeman and Company, San Francisco. Harary, F. (1969)

[10] *Graph Theory*. Addison-Wesley Publishing Company, Reading, Mass.Harley, E., Bonner, A.J., and Goodman, N. (2001) Uniform Integration of Genome Mapping Data Using Intersection Graphs. *Bioinformatics* 17: 487-494.

[11] Johnston, H.C. (1976) Cliques of a Graph --- Variations on the Bron-Kerbosch Algorithm. *International Journal of Computer and Information Sciences* 5: 209-238.

[12] Loukakis, E. (1983) A New Backtracking Algorithm for Generating the Family of Maximal Independent Sets of a Graph. *Comp. and Maths. With Appls.*, 9: 583-589.

[13] Loukakis,E. and Tsouros,C. (1981) A Depth First Search Algorithm to Generate the Family of Maximal Independent Sets of a Graph Lexicographically. *Computing*, 27: 249-266.

[14] Moon, J.W. and Moser,L. (1965) On cliques in graphs. *Israel J. Math* 3: 23-28.

[15] Pardalos, P.M. and Xue,J. (1994) The maximum clique problem. *Journal of Global Optimization* 4:301-328.

[16] Pardalos, P.M., Bomze,I.M., Budinich,M. and Pelillo,M. (1999) The Maximum Clique Problem. in *Handbook of Combinatorial Optimization, Supplement, Vol. A* (Eds: Du,D.and Pardalos,P.M.), Kluwer Academic Publishers: 1-74.

[17] Tomita, E., Tanaka, A., and Takahashi, H. (1989) An Optimal Algorithm for Finding All the Cliques.*Proceedings of the International Workshop on Discrete Algorithms and Complexity. Japan*: 91-98.

[18] Tsukiyama, S., Ide, M., Ariyoshi, H. and Shirakawa, I. (1977) A New Algorithm for Generating All the Maximal Independent Sets. *Siam J. on Computing* 6: 505-517.

Authors

1) **Abhay Bhamaikar**

Working as Assistant Professor, Department of Information technology, Shree Rayeshwar Institute of Engineering and Information technology, Shivshail, Karai, Shiroda – Goa.

Areas of Interest: Data Mining, Graph Mining, Design and Analysis of Algorithm.

2) **Pralhad Ramchandra Rao**

Professor and Head, Department of Computer Science and Technology, Goa University, Taleigao Plateau, Goa.