DETERMINING CLIQUES OF A GRAPH

BY

LeRoy F. Johnson

TR75-008, August 1975

DETERMINING CLIQUES OF A GRAPH

LeRoy F. Johnson

School of Computer Science
University of New Brunswick
Fredericton, New Brunswick

Abstract

A generalization of the neighborhood of a vertex [7] is used to prove several results on the relation of a set of vertices of a graph to cliques of that graph. These results lead to an efficient adaptive algorithm for enumerating the cliques of a graph. The storage requirement of the algorithm is of order $0(n^2)$.

Presented At

The Fifth Manitoba Conference On
Numerical Mathematics and Computing

October 1975

TR75-008, August 1975

1. INTRODUCTION

It is well known that the determination of the cliques of a graph is an important problem. It occurs primarily in information retrieval systems but also appears in many diverse applications. For instance, Tucker [8] used cliques to analyze the strong Perfect Graph Conjecture and applied the theory to a refuse disposal system.

Since the number of cliques in a graph can be exponential, the speed of an algorithm on G must be related to the cliques of G. Because of this exponential possibility, the testing for potential cliques must be efficient for a good algorithm. The algorithm presented here eliminates the testing of subgraphs that cannot lead to cliques based on the cliques found. In this sense we say it is an adaptive algorithm, since it uses its information about the cliques discovered in a nontrivial way.

2. BACKGROUND

A graph $G = \langle V,E \rangle$ consists of a set of vertices V and a set of edges E where $E \subseteq V \times V$. A subgraph G' of G is a graph $G' = \langle V',E' \rangle$ where $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. A spanning subgraph denoted by $G[V'] = \langle V',E' \rangle$ is a subgraph where $E' = E \cap (V' \times V')$. A complete graph is one in which each vertex is connected to every other vertex. The valance of a vertex is the number of edges incident to that vertex, loops being counted twice.

Definition 1 - A clique of a graph G is a maximal complete subgraph.

Definition 2 - The neighborhood of a vertex v, N(v), is the set of vertices that are adjacent to v (i.e. connected by an edge) and v itself.

3. NEW RESULTS

Generalizing the concept of a neighborhood of a vertex to the neighborhood of a set Q of vertices leads to some interesting conditions that determine if Q can be a clique and indeed if it is a clique.

Definition 3 - The neighborhood of a set of vertices Q is the set N(Q) which consists of the vertices of G that are adjacent to every vertex in Q. (Note: we allow a vertex to be adjacent to itself.)

If $N(Q) = \emptyset$ we say that Q has an empty (null) neighborhood.

Our first result shows the relation of the set neighborhood to the neighborhoods of the set elements.

Lemma 1 $N(Q) = \cap_{v_i \in Q} N(v_i)$.

Proof: If x is adjacent to each vertex $v_i \in Q$ then $x \in N(v_i)$ for all v_i and $N(Q) \subseteq \cap_{v_i \in Q} N(v_i)$

If $x \in \cap N(v_i)$ then x is adjacent to every $v_i \in Q$, hence $N(Q) \supseteq \cap_{v_i \in Q} N(v_i)$ and the lemma is true.

⊕

If Q lies in a clique then G[Q] must be a complete subgraph, the following theorem gives a necessary and sufficient condition for this.

Theorem 1 G[Q] is a complete subgraph if and only if $Q \subseteq N(Q)$.

Proof: Assume $Q \subseteq N(Q)$ and suppose $G[Q]$ is not a complete graph. If $G[Q]$ is not complete, then there exists v_i and v_j in Q that are not adjacent thus v_i or $v_j \notin N(Q)$. But this implies a contradiction. If $G[Q]$ is complete then $v_i \in Q$ implies that $v_i \in N(Q)$.

⊕

Corollary If $Q \subseteq N(Q)$, then there exists a clique K such that $G[Q] \subseteq K$.

Theorem 2 If $Q = N(Q)$ then $G[Q]$ is a clique.

Proof: Assume $Q = N(Q)$. Suppose $G[Q]$ is not a maximal complete subgraph, then there exists $x \in N(Q)$ and $x \notin Q$ but this contradicts the assumption that $Q = N(Q)$. ⊕

Theorem 3 If $Q \supseteq N(Q)$ then $G[N(Q)]$ is a complete graph.

Proof: By definition every $v \in N(Q)$ is connected to every vertex in Q, thus, by the condition of the theorem, to every vertex in $N(Q)$. Therefore, $G[N(Q)]$ is complete. ⊕

Corollary If $Q \supseteq N(Q)$ then there exists a clique K such that $G[N(Q)] \subseteq K \subset G[Q]$.

These results form the basis of an algorithm to find cliques. Given a set, we find its neighborhood, then we test to see if the set should be increased or decreased and recording it if it is a clique.

Whether or not this leads to a good algorithm depends on how we select the sets of vertices for consideration. Examination of all 2^n subsets is, of course, disasterous. Theorem 2 indicates a bottom up approach to subset generation for if $Q = N(Q)$ then we know that there is no need to test sets S where $Q \subseteq S$.

4. AN ALGORITHM

Implementation of the algorithm suggested requires a method of selecting subsets of V to be tested. If the algorithm is to be efficient, then a set must only be generated once and only sets that are potentially cliques should be generated. The algorithm implemented here is an adaptive one in that the sets considered depend upon the previous results.

One method of selecting subsets to apply our theorems to, is to generate all subsets on a rooted tree such that for any tree vertex x all decendents will represent only (and all) super sets of that set represented by the vertex x. The following algorithm SUBSETS generates the subsets of V with the desired property. Its root represents the null set and upon return to the root the algorithm is complete.

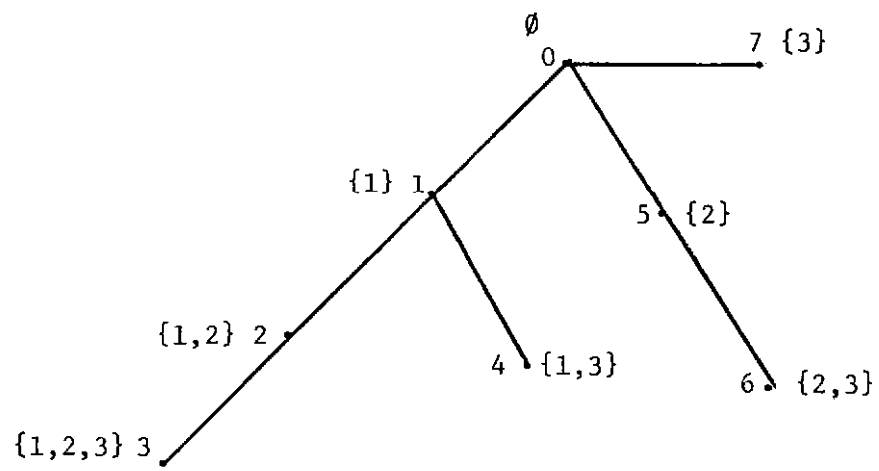

Figure 1

In figure 1 each vertex is numbered in order of generation and labeled with the set represented by the path from the root to that vertex which is the only portion of the tree explicitly recorded at that point in the algorithm. The following algorithm SUBSETS is used in the main algorithm which attempts to reduce the actual traversal of the rooted tree.

ALGORITHM: SUBSETS

Comment: Let V be a set, VS a pushdown stack and TOP the current top element of VS. Zero represents a null element for convenience.

```
1: Initialize
     VS ← 0
2: Stack
     2.1  VS ← (TOP + 1), VS
     2.2  Report subset represented by VS
     2.3  IF TOP ≠ N THEN GO TO 2:Stack
     2.4  Delete TOP from stack VS
     2.5  IF TOP = 0 THEN STOP
     2.6  TOP ← TOP + 1
     2.7  GO TO 2:Stack
```

ALGORITHM: CLIQUES

Comment: Let G = <V,E>. Let VS be a pushdown stack and TOP the top element of VS, T_k the neighborhood of the first k vertices in VS.

0: Preprocess

Number the vertices of G in order of ascending valence.

1: Initialize

VS ← 0

k ← 0; T_0 ← V; BIG ← N

2: Stack

VS ← (TOP + 1), VS

k ← k + 1

3: Test

3.1 T_k ← N(TOP) ∩ T_{k-1}

3.2 IF VS $\not\subseteq$ T_k, THEN GO TO Step 5:Update

3.3 IF Q = T_k, THEN GO TO Step 4:Report

3.4 IF TOP = N GO TO Step 6:Retreat

3.5 GO TO Step 2:STACK

4: Report

4.1 Output clique VS.

4.2 BIG ← Largest vertex not in VS ELSE 0.

5: Update

5.1 IF TOP $\neq$ N GO TO Step 7:Rotate

6: Retreat

6.1 Delete top element of stack.

$k \leftarrow k-1$

6.2 IF TOP = 0 STOP

6.3 IF TOP > BIG GO TO Retreat

7. Rotate

7.1 Replace TOP by TOP + 1

7.2 BIT $\leftarrow$ N

7.3 GO TO Step 3:Test

There are several features of the algorithm worth noting. When a set of vertices in VS is not in a clique then no superset will ever be tested. Thus, there are at most n-1 additions to VS that can fail to be a complete graph and the test for each addition requires only n operations because of iterative calculations of the set neighborhood. Once a clique is found, no set that is not a subset of a potential clique is tested. By ordering the vertices on valance, we first test those vertices that should fail the fastest and thus reduce the number of sets that must be tested. Since these vertices are more quickly examined and removed from consideration, the resulting search is vastly decreased. Large cliques are examined early. For instance, if G is a clique, the result is immediate.

Theorem 3. CLIQUES finds the set of cliques of a graph.

Proof: The procedure of SUBSETS generates all subsets of the vertices exactly once. A subset is not examined by CLIQUES only if it contains as a subset, a set of vertices that define a clique, thus any subset not examined by the procedure cannot define a clique. Application of Theorem 2 to subsets examined, ensures that each clique is determined.

Theorem 4. The storage requirement of CLIQUES is $O(n^2)$.

Proof: VS is at most n storage cells. T_k is at most n storage cells and there are at most n T_k's required.

The evaluation of the speed of an algorithm is often difficult to do theoretically because analytical results are not known; it is dangerous to do by computer since you are evaluating the implementation of the algorithm and not the algorithm. Any algorithms to find cliques must ask the question of a set of vertices: does this set generate a clique? The ratio of the yes to no answers to this question is related to the speed of the algorithm.

We then have the following figure of merit:

$$\alpha = \frac{\text{number of subsets examined}}{\text{number of cliques}}$$

Clearly $1 < \alpha < 2^n$ if G has at least one clique q. The lower bound for our algorithm is

$$\frac{\sum_{i=1}^{p} |q_i|}{p}$$

where $|q_i|$ is the order of clique q_i and p the number of cliques. A low value for α is desirable but not sufficient for a good algorithm, however, the other factors that determine speed are more readily determined analytically.

Future work will examine expected values for α and an upper bound for the algorithm presented here or perhaps a modification. At that time this algorithm will be compared to Bron and Kerbosch [3]. As well, other possibilities of applying these results will be examined further, such as using Theorem 3 in conjunction with Theorem 1. Present experimental results are promising in both regards.

REFERENCES

1. Aho, A.V., Hopcroft, J.E. and Ullman, J.D.
The Design and Analysis of Computer Algorithms. Addison Wesley, Reading, Mass. 1974, 384-387.

2. Auguston, J.G. and Minker, J.
An Analysis of Some Graph-Theoretical Cluster Techniques.
J A C M, Vol. 17, No. 4, (1970), 571-588.

3. Bron, C. and Kerbosch, J.
Finding All the Cliques in an Undirected Graph.
Comm. A C M, Vol. 16, No. 9, (1973), 575-577.

4. Johnson, L.F.
A Search Algorithm for Finding the Simple Cycles of a Directed Graph. University of New Brunswick Technical Report TR74-001 (February, 1974).

5. Moon, J.W. and Moser, L.
On Cliques in Graphs. Israel Journal Math, 3 (1965), 23-28.

6. Mulligan, G.D.
Algorithms for Finding Cliques of a Graph.
University of Toronto Technical Report No. 41 (May, 1972).

7. Osteen, R.E. and Tou, J.T.
A Clique Detection Algorithm Based on Neighborhoods in Graphs.
International Journal of Computer and Information Science
2(4) (1973), 257-268.

8. Tucker, A.
The Strong Perfect Graph Conjecture and an Application to a Municipal Routing Problem.
Graph Theory and Applications: Proceedings of the Conference at Western Michigan University May 10-13, 1972.
Springer-Verlag, Berlin-Herdelberg, New York 1972.