SIAM J. Comput.
Vol. 2, No. 1, March 1973

THE ENUMERATION OF MAXIMAL CLIQUES OF LARGE GRAPHS*

E. A. AKKOYUNLU†

Abstract. An algorithm for enumerating maximal cliques (complete subgraphs) is proposed. The aim is to deal with the difficulties caused by the size of the problem.

Key words. Algorithms, clique, enumeration, graph.

1. Introduction. The problem considered here is best formulated in terms of an undirected graph G (Fig. 1). If S_0 is the set of nodes and E the set of edges, the goal is to identify completely connected subgraphs (or cliques) which are maximal.

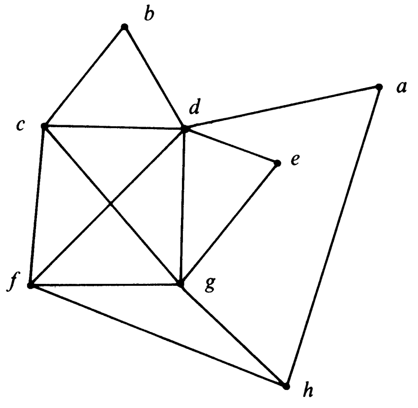

FIG. 1

More generally, S_0 is a set of elements, and there is defined over pairs in S_0 a symmetric, nontransitive binary relation R. (E is a set of pairs which are in relation R.) A subset S of S_0 is a *maximal subset* (ms) if

(i) every pair in S is in relation R, *and*

(ii) S is not a proper subset of any set with property (i).

The ms are of interest in many contexts: graph theory (the coloring problem), switching theory (state minimization [1]), operations research (scheduling [2], [3]), information systems, etc. There are several known algorithms [1], [4], [5], [6] for enumerating these sets. As a rule, however, these algorithms cannot handle large problems [3] efficiently. The difficulty arises because the terms generated include duplications, or even submaximal sets. To avoid this, a list of all the terms has to be kept and repeatedly scanned. In the worst cases, the list cannot be accommodated in core.

This paper proposes a new algorithm which is especially efficient in dealing with large problems, and has none of the shortcomings mentioned above. The set generated corresponds precisely (and without duplications) to the set desired.

* Received by the editors March 4, 1972, and in revised form November 20, 1972.

† Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11790. This research was supported in part by the National Science Foundation under Grants GK-1241 and GJ-243.

	a	b	c	d	e	f	g
b							
c		x					
d	x	x	x				
e				x			
f			x	x			
g			x	x	x	x	
h	x					x	x

FIG. 2

There is thus no need to refer to previously generated terms, and no need to maintain a list in core. Even for the largest problems memory size is not a critical limitation.

Let $\mathscr{M}$ denote the set of ms. Certain subsets of $\mathscr{M}$ are also important. In the state minimization problem where the relation R is mutual compatibility, what is wanted is *minimal cover*, i.e., the smallest number of ms in which each element occurs at least once. In scheduling, on the other hand, R is mutual exclusion, and one is typically interested in finding a *minimal pair-cover*, i.e., the smallest subset of $\mathscr{M}$ such that every pair in relation R occurs together in some term. This paper is mainly concerned with an algorithm for enumerating the set $\mathscr{M}$.

The information contained in the graph G is normally represented in the form of the table in Fig. 2. The somewhat redundant representation of Fig. 3 will be used instead, for it displays certain features more saliently. In particular, each element x divides the remaining elements into two sets: the set C_x of elements which are in relation R to x, and the set D_x of elements which are not. This is made explicit in Fig. 3.

2. Preliminaries. In Fig. 3, consider the row associated with c. For every ms, exactly one of the following statements must be true:

1. The ms includes $\{c\}$.
2. The ms includes at least one of $\{a, e, h\}$.

D_x	x	C_x
$b\ c\ e\ f\ g$	a	$d\ h$
$a\ e\ f\ g\ h$	b	$c\ d$
$a\ e\ h$	c	$b\ d\ f\ g$
h	d	$a\ b\ c\ e\ f\ g$
$a\ b\ c\ f\ h$	e	$d\ g$
$a\ b\ e$	f	$c\ d\ g\ h$
$a\ b$	g	$c\ d\ e\ f\ h$
$b\ b\ d\ e$	h	$a\ f\ g$

FIG. 3

To verify: if both statements were true, the ms would include one of the disallowed pairs, *ac*, *ce*, *ch*. If neither were true, the addition of c to this term could not possibly violate a constraint, so that the term could not have been maximal.

Consider next row b. An ms which includes b cannot include any of a, e, or h since $\{a, e, h\} \subset D_b$. This, together with the above, means that every ms which includes b must also include c. To generalize, we state:

If $D_x \subseteq D_y$, then x occurs in every ms where y occurs.

Further notation is introduced at this point in order to facilitate the manipulation of subsets of $\mathscr{M}$ according to certain characteristics. For $S \subseteq S_0$, let $L(S)$ denote "the set of all ms which include *at least one* element of S." Similarly, let $E(S)$ denote "the set of all ms which include *every* element of S." Thus,

$$L(S) = \{M | M \in \mathscr{M}, S \cap M \neq \varnothing\},$$

$$E(S) = \{M | M \in \mathscr{M}, S \subseteq M\}.$$

The following relations are direct consequences of these definitions:

$$L(\{x\}) = E(\{x\}), \tag{1}$$

$$E(S_1) \cap E(S_2) = E(S_1 \cup S_2), \tag{2}$$

$$L(S_1) \cap L(S_2) = L(S_1) \quad \text{if } S_1 \subseteq S_2, \tag{3}$$

$$L(\varnothing) = \varnothing. \tag{4}$$

Also, since any ms which includes x can only include elements in C_x, we can write

$$E(\{x\}) \cap L(S) = \begin{cases} E(\{x\}) & \text{if } x \in S, \\ E(\{x\}) \cap L(S \cap C_x) & \text{otherwise.} \end{cases}$$

More generally,

$$E(\{x\}) \cap \left(\bigcap_i L(S_i) \right) = E(\{x\}) \cap \left(\bigcap_{i : x \notin S_i} L(S_i \cap C_x) \right). \tag{5}$$

3. Enumeration through disjoint subproblems.

In the previous example, where $S_0 = \{a, b, c, d, e, f, g, h\}$, the set of all ms can be written

$$\mathscr{M} = L(S_0) = L(\{a, b, c, d, e, f, g, h\}),$$

which simply states that $\mathscr{M}$ is the set of all ms which include at least one element. As discussed above, $\mathscr{M}$ has two disjoint subsets,

(i) all ms which include $\{c\}$,

(ii) all ms which include at least one of $\{a, e, h\}$,

and this is expressed by writing,

$$\mathscr{M} = L(S_0) = E(\{c\}) \cup L(\{a, e, h\}).$$

To formalize this notion, we write

$$L(S \cup \{x\}) = E(\{x\}) \cup (L(S) \cap L(D_x)). \tag{6}$$

Here, the right-hand side is the union of disjoint sets: all ms which include x, and all which exclude x but include some element in S.

In what follows, the notation will be simplified by writing $L(a, b, \cdots, x)$ instead of the formally correct $L(\{a, b, \cdots, x\})$, etc. Equations (1) through (6) can be used to reduce the problem of enumerating $\mathscr{M}$ into a series of smaller problems. The key step is the substitution specified by (6) which is applied repeatedly. For the problem in Fig. 3, we have

$$\begin{aligned}\mathscr{M} &= L(a, b, c, d, e, f, g, h) = E(c) \cup L(a, e, h)\\ &= E(c) \cup E(a) \cup (L(e, h) \cap L(b, c, e, f, g))\\ &= E(c) \cup E(a) \cup ((E(e) \cup L(h) \cap L(a, b, c, f, h)) \cap L(b, c, e, f, g)).\end{aligned}$$

Since $L(h) = E(h)$, equation (5) allows the reductions

$$\begin{aligned}E(e) \cap L(b, c, e, f, g) &= E(e),\\ E(h) \cap L(a, b, c, f, h) \cap L(b, c, e, f, g) &= E(h) \cap L(f, g),\\ \mathscr{M} &= E(c) \cup E(a) \cup E(e) \cup (E(h) \cap L(f, g)).\end{aligned}$$

Now $L(f, g) = E(f) \cup (L(g) \cap L(a, b, e))$, so that we write,

$$\begin{aligned}E(h) \cap L(f, g) &= (E(h) \cap E(f)) \cup (E(h) \cap L(g) \cap L(a, b, e)),\\ \mathscr{M} &= E(c) \cup E(a) \cup E(e) \cup (E(h, f) \cup (E(h) \cap L(g) \cap L(a, b, e))).\end{aligned}$$

Using equation (5), we have

$$E(h) \cap L(g) \cap L(a, b, e) = E(h) \cap L(g) \cap L(a).$$

Finally, since $L(g) \cap L(a) = E(g) \cap L(a) = \varnothing$, we obtain

$$\mathscr{M} = E(c) \cup E(a) \cup E(e) \cup E(h, f).$$

The problem is thus reduced to four smaller problems whose solutions correspond to disjoint subsets of $\mathscr{M}$. Each of these can now be solved separately and the complete solution $\mathscr{M}$ obtained.

In evaluating $E(a)$, for example, only the set C_a need be considered. This corresponds to Fig. 4 whose ms set is $\{d, h\}$. The addition of a to every member of this set yields $E(a) = \{ad, ah\}$. Similarly, $E(h, f)$ is obtained by adding h and f to each term of the ms set derived from Fig. 5 which consists of variables in $C_h \cap C_f$ alone. This ms set is simply $\{g\}$ so that $E(h, f) = \{fgh\}$.

To generalize, let S be a set for which $x, y \in S$ implies $x \in C_y$. Then,

$$E(S) = \begin{cases} S & \text{if } \bigcap_{x \in S} C_x = \varnothing, \\ E(S) \cap L(\bigcap_{x \in S} C_x) & \text{otherwise.} \end{cases} \tag{7}$$

4. Enumeration algorithm. This section formalizes the approach outlined above by giving an algorithm for enumerating $\mathscr{M}$. The algorithm manipulates

D_x	x	C_x
h	d	—
d	h	—

FIG. 4

D_x	x	C_x
—	g	—

FIG. 5

symbolic expressions composed of E- and L-sets. The data base consists of the sets S_0, C_x and D_x, for $x \in S_0$. The algorithm is organized around a pushdown stack which serves to store partially formulated disjoint subproblems. Items on the stack are expressions involving sets of the form $E(S)$ and $L(S)$. A possible item on the stack could be

$$E(h) \cap L(g) \cap L(a, b, e).$$

Specifically, each item on the stack is a multiple set intersection between one or more L-sets and at most one E-set.

Starting with the expression $L(S_0)$, the algorithm consists of repeated applications of equations (6) and (7), supplemented by the reduction equations (1) through (5). Equation (6) is used to split an expression into two others which correspond to disjoint subsets of the original set; these are then pushed on the stack, and the process is repeated until an expression of the form $E(S)$ is obtained. Equation (7) is then applied once.

ALGORITHM A.

Input: S_0, $\{C_x | x \in S_0\}$, $\{D_x | x \in S_0\}$.

Output: $\mathscr{M}$.

Step 1 (Initialize). Place $L(S_0)$ on the stack.

Step 2 (Prepare to split). (a) If the stack is empty, stop.

(b) Unload the top expression from the stack and call it T. T is a multiple intersection whose form is either

$$E(S') \cap \left(\bigcap_{i \in I} L(S_i) \right)$$

or simply

$$\bigcap_{i \in I} L(S_i).$$

In the first case set $V = S'$, in the second case set $V = \varnothing$.

Step 3 (Choose the L-term to be split). (a) Select $k \in I$ so that S_k has the fewest elements possible. Also choose $x \in S_k$, and let $S = S_k - \{x\}$.

(b) If $S = \varnothing$ go to Step 5. (Equation (1) can be applied instead of equation (6).)

Step 4 (Formulate the second subproblem). (a) Compute

$$Q = \begin{cases} D_x & \text{if } V = \varnothing, \\ D_x \cap (\bigcap_{y \in V} C_y) & \text{otherwise.} \end{cases}$$

(b) If $Q = \varnothing$ go to Step 5. (Every ms which covers V also covers $\{x\}$.)

(c) Make a copy of T with $L(S) \cap L(Q)$ substituted for $L(S_k)$, and load this copy on the stack.

Step 5 (Formulate the first subproblem). (a) Let $J = \{j | j \in I, x \in S_j\}$. If $J = \emptyset$ go to Step 6. (Equation (7) is applicable.)

(b) For all $j \in J$ compute $W_j = S_j \cap C_x$. If $W_j = \emptyset$ for any $j \in J$ go to Step 2. (No ms covers both V and $\{x\}$.)

(c) Place the expression

$$E(V \cup \{x\}) \cap \left(\bigcap_{j \in J} L(W_j) \right)$$

on the stack and go to Step 2.

Step 6 (Apply equation (7).) (a) Compute

$$P = \bigcap_{y \in V \cup \{x\}} C_y.$$

(b) If $P = \emptyset$ output $V \cup \{x\}$, then go to Step 2.

(c) Load the stack with $E(V \cup \{x\}) \cap L(P)$, and go to Step 2. (End Algorithm A.)

In practice, rather than compute the set

$$\bigcap_{y \in V} C_y$$

each time, it is more convenient to store it along with the associated item on the stack.

5. Conclusion.

Techniques for generating maximal subgraphs were discussed with reference to the difficulties caused by the size of the problem. An effective procedure was proposed for systematically reducing these problems into smaller ones whose ms sets are disjoint. An important feature of the enumeration algorithm is that it is organized around a stack, so that its core requirements are minimal.

Acknowledgments. I would like to thank Professor S. H. Unger of Columbia University for the long discussions which helped focus this work.

REFERENCES

[1] M. C. Paull and S. H. Unger, *Minimizing the number of states in incompletely specified sequential switching functions*, IRE Trans. Electronic Computers, EC-8 (1959), pp. 356–367.

[2] E. A. Akkoyunlu, *Allocating facilities to interdependent activities*, Proc. Fifth Annual Princeton Conference on Information Sciences and Systems, Princeton, N.J., 1971, pp. 86–87.

[3] A. D. Hall, Jr. and F. S. Acton, *Scheduling university course examinations by computer*, Comm. ACM, 10 (1967), pp. 235–238.

[4] P. M. Marcus, *Derivation of maximal compatibles using Boolean algebra*, IBM J. Res. Develop., 8 (1964), pp. 537–538.

[5] G. D. Mulligan and D. G. Corneil, *Corrections to Bierstone's algorithm for generating cliques*, J. Assoc. Comput. Mach., 19 (1972), pp. 244–247.

[6] R. M. Burstall, *Tree searching methods with an application to a network design problem*, Machine Intelligence, 1 (1967), pp. 65–85.