SIAM J. COMPUT.
Vol. 15, No. 4, November 1986

FINDING A MAXIMUM CLIQUE IN AN ARBITRARY GRAPH*

EGON BALAS† AND CHANG SUNG YU‡

Abstract. We describe a new type of branch and bound procedure for finding a maximum clique in an arbitrary graph $G=(V,E)$. The two main ingredients, both of $O(|V|+|E|)$ time complexity, are (i) an algorithm for finding a maximal triangulated induced subgraph of G; and (ii) an algorithm for finding a maximal k-chromatic induced subgraph of G. We discuss computational experience on randomly generated graphs with up to 400 vertices and 30,000 edges.

Key words. maximum clique, vertex packing, triangulated subgraphs, k-chromatic subgraphs, implicit enumeration

AMS(MOS) subject classifications. 05, 90, 68

1. Introduction. In this paper we address the problem of finding a maximum cardinality clique in an arbitrary undirected graph. Given a graph $G=(V,E)$ with vertex set V and edge set E, a *complete subgraph* of G is one whose vertices are pairwise adjacent. A *clique* is a maximal complete subgraph. For the sake of brevity, the vertex set of a clique is also called a clique. A *vertex packing* (or *stable set* or *independent set* of vertices) is a vertex set whose elements are pairwise nonadjacent. A *vertex cover* is a set of vertices that cover (i.e., meet) all the edges. The *complement* of a graph $G=(V,E)$ is the graph $\bar{G}:=(V,\bar{E})$, where $\bar{E}:=\{(i,j)\notin E\colon i,j\in V,\ i\neq j\}$. For $S\subset V$, the subgraph of G *induced* by S is $G(S):=(S,(S\times S)\cap E)$.

The following three statements concerning any $S\subset V$ are easily seen (and well known) to be equivalent:

(1) S is the vertex set of a maximum clique in G,
(2) S is a maximum vertex packing in $\bar{G}$,
(3) $V\backslash S$ is a minimum vertex cover in $\bar{G}$.

Accordingly, the three problems of

(1) finding a maximum clique in G,
(2) finding a maximum vertex packing in $\bar{G}$,
(3) finding a minimum vertex cover in $\bar{G}$,

are equivalent. They are also known to be NP-complete. These problems have many applications, a few of which are listed below.

Information retrieval. If the vertices of a graph represent stored pieces of information and the edges describe the interrelations between the pieces, then the problem of retrieving from storage a maximum subset of totally interrelated pieces of information is a maximum clique problem.

Experimental design. If the vertices of a graph represent subsets (blocks) of size n of a set S of treatments in a statistical experiment and the edges designate those pairs of subsets having at most k elements in common, then the problem of finding a

* Received by the editors February 12, 1985, and in revised form September 15, 1985. The research underlying this report was supported by the National Science Foundation under grant ECS-8218181 and the U.S. Office of Naval Research under contract N00014-82-K-0329 NR047-607.

† Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

‡ AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

maximum collection of blocks with at most k common elements between any two blocks is a maximum clique problem.

Signal transmission. If the vertices of a graph represent signals and the edges mark those pairs of signals that can be clearly distinguished from each other, then the problem of selecting a maximum set of clearly distinguishable signals is again a maximum clique problem.

Computer vision. A relational structure is a set of elements partitioned into color classes, along with a binary relation between the elements. Given two relational structures, an input A and a reference B, one wants to optimally "match" A to B, i.e., find a substructure of B, to which A comes as "close" as possible. To this end one constructs an association graph G that has a vertex (a, b) for every pair of elements $a \in A$, $b \in B$, belonging to the same color class, and an edge for every pair of vertices (a, b), (c, d) such that the relation between a and c in A is the same as that between b and d in B. Then optimally "matching" A to B amounts to finding a maximum clique in the association graph G.

In each of the above cases, if we adopt definitions for the edges complementary to the above ones, we obtain of course a maximum vertex packing problem.

Earlier work on vertex packing or vertex covering algorithms includes the papers by Nemhauser and Trotter [1975], Balas and Samuelsson [1977], Tarjan and Trojanowski [1977]. The first two of these present branch and bound methods that use information from a partial polyhedral description of the convex hull of feasible points, and were tested on randomly generated graphs with up to 50 vertices and 50% density. The third one describes a recursive algorithm that depends on a somewhat complicated case analysis, but has a worst case bound of $O(2^{n/3})$. For the maximum clique problem itself, we mention the procedures of Bron and Kerbosch [1973], Gerhards and Lindenberg [1979], and Loukakis and Tsouros [1982]. These are implicit enumeration algorithms that use some logical tests to eliminate subproblems. We implemented and tested two of them and used them as benchmarks for the testing of our own procedure.

The approach that we describe in this paper uses the properties of a special class of perfect graphs, known as triangulated or chordal, in which it is easy to find a maximum clique. It also uses the relationship between cliques and vertex colorings of a graph in order to derive bounds on the clique size. It tends to create fewer branches than any of the above methods and is consequently more efficient.

The paper is organized as follows. Section 2 gives the necessary background information on triangulated graphs and outlines our general approach. Sections 3 and 4 describe the two main ingredients of our procedure: algorithms for finding (i) a maximal triangulated induced subgraph, and (ii) a maximal k-chromatic induced subgraph, of a given graph. Section 5 states the procedure in its entirety, while § 6 discusses computational experience.

2. Cliques, colorings and triangulated graphs.

A (vertex) *coloring* of a graph G assigns colors to the vertices of G in such a way that no two adjacent vertices get the same color. Thus a coloring is a partitioning of the vertex set of G into independent sets (color classes). The cardinality of a minimum (vertex) coloring is called the *chromatic number* of G. G is *k-chromatic* if its chromatic number is k.

If $G = (V, E)$ is a graph and A is the incidence matrix of maximal vertex packings (independent sets) versus vertices of G, i.e., $A = (a_{ij})$ with $a_{ij} = 1$ if independent set i contains vertex j, $a_{ij} = 0$ otherwise, then the maximum clique and the minimum coloring problems can be stated as

$$\text{(P1)}\qquad \begin{aligned} \omega(G) = \max\ & 1x \\ & Ax \leqq 1 \\ & x_j \in \{0, 1\} \quad \forall j \end{aligned}$$

and

$$\text{(P2)}\qquad \begin{aligned} \gamma(G) = \min\ & 1y \\ & A^T y \geqq 1 \\ & y \in \{0, 1\} \quad \forall i, \end{aligned}$$

respectively. Since the relaxation of the integrality conditions on x and y turns (P1) and (P2) into a pair of dual linear programs, clearly $\omega(G) \leqq \gamma(G)$ always holds. G is called *perfect* if $\omega(G') = \gamma(G')$ for all induced subgraphs G' of G. The perfect graph theorem (conjectured by Berge [1961] and proved by Lovász [1972]) asserts that G is perfect if and only if $\bar{G}$ is.

When G is perfect, the linear programming relaxations of (P1) and (P2) have integer solutions for arbitrary objective functions (Chvátal [1975]); hence they can be solved in time polynomial in the input. Although this does not imply polynomiality in $n = |V|$, since the matrix A may have a number of rows exponential in n, it has recently been shown by Grötschel, Lovász and Schrijver [1984] that if G is perfect, problem (P1) is solvable in time polynomial in n.

A special class of perfect graphs whose properties make them particularly well suited for our purposes, is the class of triangulated graphs. A graph G is *triangulated* (or *chordal*) if every cycle of G of length at least 4 has a chord (or, equivalently, if G has no holes, i.e., no chordless cycles of length $\geqq 4$). Triangulated graphs have been studied by Dirac [1961], Fulkerson and Gross [1965], Gavril [1972], Rose, Tarjan and Lueker [1976] and others. For an up-to-date survey of their properties see Golumbic [1980].

A vertex v of an arbitrary graph G is called *simplicial* if all vertices adjacent to v are pairwise adjacent among each other, i.e., if v belongs to exactly one clique. An ordering $v_1, \cdots, v_n$ of the vertices of G is called *perfect* if for $k = 1, \cdots, n$, v_k is simplicial in $G(\{v_k, \cdots, v_n\})$. The basic properties of triangulated graphs that we will be using can now be stated as follows:

(i) every triangulated graph has at least two simplicial vertices;
(ii) $G = (V, E)$ is triangulated if and only if V admits a perfect ordering; and
(iii) determining whether G is triangulated requires $O(|V| + |E|)$ steps.

If G is triangulated, then

(iv) every induced subgraph of G is triangulated;
(v) the cardinality of a maximum clique is equal to that of a minimum coloring;
(vi) G has at most $|V|$ cliques; and
(vii) finding a maximum clique and a minimum coloring in G requires $O(|V| + |E|)$ steps.

We use the above properties to give an $O(|V| + |E|)$ algorithm for generating a maximal triangulated induced subgraph (MTIS) $G(T)$ of an arbitrary graph G, and finding a maximum clique of $G(T)$. This algorithm, called TRIANG, is one of the two main ingredients of our procedure. The second ingredient, called COLOR, finds a minimum coloring of $G(T)$, which is of the same cardinality k as the maximum clique; then it extends $G(T)$ by appending vertices to it while maintaining its chromatic number, until it becomes a maximal k-chromatic induced subgraph of G.

Let this subgraph be $G(W)$. If $G(W)=G$, we are done: the maximum clique found in $G(T)$ is maximum in G. Otherwise we branch, based on the principle that any clique larger than the current largest one must contain one of the vertices in $V\setminus W$. The branching thus replaces the current problem by a set of new subproblems, each one defined on a vertex set contained in the neighbor set of some $v\in V\setminus W$. The procedure, or part of it, is then applied to each new subproblem, with variations to be discussed after we have described the main subroutines.

3. Finding a MTIS and its maximum clique.

Determining whether a graph $G=(V,E)$ is triangulated can be done most efficiently by checking whether G admits a perfect ordering. This can be done by applying n times, for $k=1,\cdots,n$, the following step:

Find a simplicial vertex in $G(V\setminus\{w_1,\cdots,w_{k-1}\})$. If none exists, stop: G is not triangulated. Otherwise call the new vertex w_k and set $k\leftarrow k+1$. If $k+1=n$, stop: $(w_1,\cdots,w_n)$ is a perfect ordering. Otherwise repeat.

The complexity of this procedure is $O(|V|^3)$; but one can do better, as shown by Rose, Tarjan and Lueker [1976], whose procedure has a complexity of $O(|V|+|E|)$. Their approach essentially uses the following property.

Given an arbitrary ordering $\sigma=(v_1,\cdots,v_n)$ of V, call the elements of the set

$$N^+(v_i):=\{v_j\in V\setminus\{v_i\}\,|\,(v_i,v_j)\in E \text{ and } j>i\}$$

the *successors* of v_i, and call the successor of v_i with smallest index, the ***first successor*** of v_i. We say that v_i is *quasi-simplicial* in $\sigma=(v_1,\cdots,v_n)$ if the first successor of v_i is adjacent to every other successor of v_i.

THEOREM 1. *Given a graph $G=(V,E)$ and an ordering $\sigma=(v_1,\cdots,v_n)$, the following two statements are equivalent*;

(i) *for $i=1,\cdots,n$, v_i is simplicial in $G(\{v_i,\cdots,v_n\})$*;

(ii) *for $i=1,\cdots,n$, v_i is quasi-simplicial in σ.*

Proof. (i) obviously implies (ii). Now suppose (ii) holds; then v_n is simplicial in $G(\{v_n\})$. Assume v_i is simplicial in $G(\{v_i,\cdots,v_n\})$ for $i=n,n-1,\cdots,k+1$, and let $i=k$. If v_k has less than two successors, we are done; otherwise let $w_1,\cdots,w_p$ be the successors of v_k, with $w_1=v_l$ the first successor. Since v_k is quasi-simplicial, $w_2,\cdots,w_p$ are adjacent to $w_1=v_l$. Since v_l by assumption is simplicial, $w_2,\cdots,w_p$ are pairwise adjacent. Hence v_k is simplicial, and the induction is complete. □

COROLLARY 1.1. *An ordering $\sigma=(v_1,\cdots,v_n)$ is perfect if and only if for $i=1,\cdots,n$, v_i is quasi-simplicial in σ.*

While the complexity of checking whether a vertex is simplicial is $O(|V|^2)$, checking for quasi-simpliciality requires only $O(|V|)$ adjacency tests.

Rose, Tarjan and Lueker [1976] use a lexicographic labeling algorithm (LEX P) for generating an ordering of V that is perfect if and only if G is triangulated; they then use an adjacency testing algorithm (FILL) to check whether the ordering produced by LEX P is perfect. The adjacency testing algorithm is a straightforward application of Corollary 1.1.

As to LEX P, it starts by assigning the label $\varnothing$ to every vertex. It proceeds by numbering the vertices $n,n-1,\cdots,1$ in an order based on the lexicographic labels which are updated at every iteration. For $i=n,n-1,\cdots,1$, it thus performs the following steps:

1. Choose a vertex v with lexicographically largest label, and assign the number i to v.
2. Append i to the label of every unnumbered vertex adjacent to v, and set $i\leftarrow i-1$.

THEOREM 2 (Rose, Tarjan and Lueker [1976]). *G is triangulated if and only if the ordering generated by algorithm* LEX P *is perfect.*

Our procedure combines this lexicographic labeling algorithm with a modified version of the adjacency testing algorithm, which screens the vertices to be included into the MTIS, rejecting all those whose addition would make the ordering imperfect. Thus our algorithm orders V into a sequence σ, but also generates a subsequence $\sigma(T)$ whose elements T induce a maximal triangulated subgraph in G.

ALGORITHM TRIANG. Let i denote the current number (rank in σ) to be assigned, T the set of vertices currently included into the triangulated subgraph, $\sigma(T)$ the ordering defined on T.

0. *Initialization.* Assign the label $\varnothing$ to every vertex. Set $i \leftarrow n$, $T \leftarrow \varnothing$, $\sigma(T) \leftarrow \varnothing$.

1. *Choosing v.* If $i = 0$, stop: $G(T)$ is a maximal triangulated subgraph. Otherwise, choose an unnumbered vertex v with lexicographically largest label, assign number i to v, i.e., set $v_i \leftarrow v$, and go to 2.

2. *Checking whether v can be added to T.* Let σ^0 be the sequence obtained from $\sigma(T)$ by adding v as the first element. If v is quasi-simplicial in σ^0, set $T \leftarrow T \cup \{v\}$, $\sigma(T) \leftarrow \sigma^0$, and go to 3. Otherwise set $i \leftarrow i-1$ and go to 1.

3. *Labeling.* Append i to the label of each unnumbered vertex w adjacent to v, set $i \leftarrow i-1$, and go to 1.

THEOREM 3. *Algorithm* TRIANG *finds a* MTIS *of G in* $O(|V|+|E|)$ *time.*

Proof. (i) Let $\sigma(T) = (w_1, \cdots, w_{|T|})$ be the ordering generated by TRIANG. Since step 2 guarantees that for $i = 1, \cdots, |T|$, w_i is quasi-simplicial in $G(\{w_i, \cdots, w_{|T|}\})$, from Theorem 1 the ordering $\sigma(T)$ is perfect. Hence the subgraph $G(T)$ is triangulated.

(ii) We claim that $G(T)$ is maximal. For if not, then there exists a vertex $v \in V \setminus T$ such that $G(T \cup \{v\})$ is triangulated. Since $v \notin T$, v was rejected at some iteration as not being quasi-simplicial in the sequence σ^0 formed in step 2. Let this particular sequence be σ^{00} and let T^0 be the vertex set ordered by σ^{00} (containing v as its first vertex). Then $G(T^0)$ is not triangulated. But $T^0 \subset T \cup \{v\}$, and $G(T \cup \{v\})$ is triangulated, a contradiction. Thus $G(T)$ is a MTIS of G.

(iii) Next we establish the complexity of TRIANG. One way of carrying out the steps of TRIANG efficiently is to keep a list of subsets of vertices with the same label, where the subsets are ordered according to lexicographically decreasing labels. Whenever we append an entry to the label of a vertex v in step 3, we take v out of its subset S. It is easy to see that the new label of v cannot be greater than that of the vertices in the subset S' preceding S. If it is the same as the latter, we put v into S'; otherwise we create a new subset $S'' = \{v\}$, placed between S' and S. This operation requires constant time for each vertex whose label is updated, and it keeps the vertex with lexicographically largest label at the head of the list. Thus each application of step 3 requires $O(\deg(v))$ time (where v is the vertex whose neighbors are having their labels updated and $\deg(v)$ is the degree of v), and each application of step 1 requires a constant amount of time (choose the vertex at the head of the list). Finally, checking in step 2 whether v is quasi-simplicial in σ^0 involves checking for each successor of v in σ^0, other than the first one, whether it is adjacent to the latter; and this again requires $O(\deg(v))$ time. To obtain the complexity of the entire sequence of steps we have to sum over all $v \in V$, which yields $O(|V|+|E|)$.

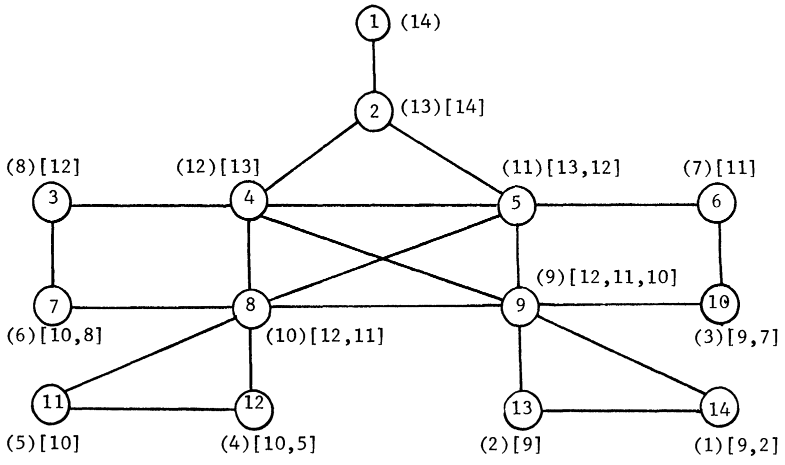
Notice that, as a byproduct of applying TRIANG, we also identify the (or a) largest clique of $G(T)$: it consists of the vertex v with the largest number of successors in $\sigma(T)$ (the perfect ordering produced by TRIANG), together with the successors of v. A minor modification of step 2 provides for counting the successors of v and storing the new largest clique whenever the earlier largest clique size is exceeded.

Next we illustrate TRIANG on an example.

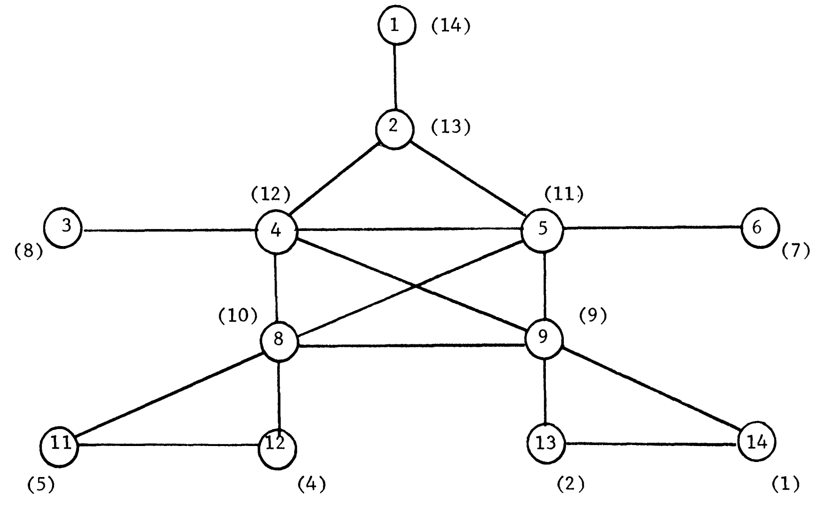
Example 1. Let G be the graph of Fig. 1(a), where the number (rank in σ) of each vertex is shown in parentheses, while its label is shown in square brackets.

After initializing the label of each vertex with $\emptyset$ (not shown in the figure), we choose vertex 1 (breaking ties arbitrarily) and assign to it number 14. Clearly, vertex 1 is quasi-simplicial in $G(\{1\})$, and so we set $T=\{1\}$, $\sigma(T)=(1)$. We then append 14 to the label of the unique vertex adjacent to 1, namely 2. At the next iteration we choose vertex 2, assign to it number 13, set $T=\{1,2\}$, $\sigma(T)=(2,1)$, and append 13 to the labels of 4 and 5. Next we choose vertex 4 (tied with 5 for the lexicographically largest label), number it 12, set $T=\{1,2,4\}$, $\sigma(T)=(4,2,1)$, and add 12 to the labels of 3, 5, 8, 9.

At the next iteration, vertex 5 becomes number 11, and 11 is added to the labels of 6, 8, 9. Then vertex 8 becomes number 10, and 10 is appended to the labels of 7,



(a)



(b)

FIG. 1

9, 11, 12. Vertex 9 becomes number 9, and 9 is appended to the labels of 10, 13 and 14. Vertex 3 becomes number 8, and 8 is appended to the label of 7. Vertex 6 becomes 7, with 7 added to the label of 10.

At this point, the next vertex with lexicographically largest label is 7; it gets the number 6, but setting $\sigma^0 = (7, 6, 3, 9, 8, 5, 4, 2, 1)$ in step 2 shows that 7 is not quasi-simplicial in σ^0: the first successor of 7, which is 3, is not adjacent to 8, also a successor of 7. Thus 7 is rejected (not included in T).

As the process continues, vertex 10 is also rejected (upon becoming number 3), and all the remaining vertices are added to T. The final sequence $\sigma(T)$ is (14, 13, 12, 11, 6, 3, 9, 8, 5, 4, 2, 1), and the MTIS $G(T)$ is shown in Fig. 1(b). The vertex with the largest number of successors in $\sigma(T)$ is 9, and the corresponding maximum clique has vertex set $\{9, 8, 5, 4\}$.

The MTIS found by TRIANG is maximal with respect to set inclusion, but not necessarily of maximum cardinality. The problem of finding a maximum cardinality triangulated induced subgraph of an arbitrary graph is NP-complete (Yannakakis [1978]). A graph may have many MTIS's, and they may be of different sizes. More importantly, they may have maximum cliques of different sizes.

Note that in Step 1 of TRIANG one is free to choose any unnumbered vertex with a lexicographically largest label, and different rules for breaking ties produce different MTIS's. We implemented and tested four tie breaking rules, based on the following criteria: (1) random choice; (2) largest number of adjacent vertices in the current set T; (3) largest number of adjacent vertices in $T \cup C$, where C is the set of candidates; (4) largest degree in G. The comparative results are discussed in § 6.

4. Finding a maximal k-chromatic induced subgraph.

Having generated a MTIS $G(T)$, next we extend $G(T)$ to a maximal induced subgraph $G(W)$ of G with the same maximum clique size as $G(T)$. This is done as follows. Since $G(T)$ is triangulated, its maximum clique size is the same as its chromatic number, i.e., the cardinality of a minimum coloring of (the vertices of) $G(T)$. Furthermore, given a perfect ordering $\sigma = (v_1, \cdots, v_{|T|})$ of the vertices of $G(T)$, finding a minimum coloring in $G(T)$ is quite straightforward: initialize k color classes (independent sets), where k is the size of a maximum clique in $G(T)$, and examine the vertices of $G(T)$ in the reverse order of σ (i.e., starting with $v_{|T|}$), putting each vertex in the first color class where it fits (i.e., where it has no neighbors). Since each v_i has at most $k-1$ successors in $G(v_i, \cdots, v_{|T|})$, each vertex fits into some color class. Expanding $G(T)$ while preserving k color classes can then be done simply by examining each vertex $v \in V \setminus T$ in turn, and putting it into the first color class where it fits, or leaving it out if it does not fit into any class. Here is a formal statement of the procedure.

Algorithm COLOR.

0. Initialize k color classes $C_1, \cdots, C_k$, $C_j = \emptyset$, $j = 1, \cdots, k$, set $V \leftarrow V \setminus T$, and go to 1.

1. If $T = \emptyset$, go to 2. Otherwise, choose $v \in T$ with the highest rank in σ, and a color class C_j that contains no vertex adjacent to v, set $C_j \leftarrow C_j \cup \{v\}$, $T \leftarrow T \setminus \{v\}$, and go to 1.

2. Choose any $v \in V$. If there exists a color class that contains no vertex adjacent to v, choose such a color class C_j and set $C_j \leftarrow C_j \cup \{v\}$. In any case, set $V \leftarrow V \setminus \{v\}$. If $V \neq \emptyset$, go to 2. If $V = \emptyset$, stop: $G(W)$, where $W = \bigcup_{i=1}^{k} C_i$, is a maximal k-chromatic induced subgraph of G.

The above algorithm extends a MTIS of G with maximum clique size k to a maximal k-chromatic induced subgraph of G. In the branch and bound procedure to be described below, we sometimes need to find a maximal k-chromatic induced subgraph of some graph for which we do not have a MTIS. In these cases we use a modified version of the above algorithm, called COLOR 2, in which the instruction "set $V \leftarrow V \backslash T$ and go to 1" of the Initialization Step is replaced by "go to 2," and Step 1 is skipped.

Since checking whether a vertex v fits into any of the k color classes requires $\deg(v)$ adjacency tests, the complexity of COLOR is again $O(|V|+|E|)$.

A more sophisticated version of the coloring procedure, called COLINT, amends the above algorithm with an interchange heuristic; i.e., having found a maximal k-chromatic induced subgraph $G(W)$ by steps 0–1–2 (or 0–2) above, it then tries to increase the cardinality of W by performing the following step for each $v \in V \backslash W$ in turn (here V is again the vertex set of G):

3. Put v into the color class C_j that contains the smallest number of vertices adjacent to v, and try to transfer each $w \in C_j$ adjacent to v into some other color class that contains no vertex adjacent to w.

If Step 3 is successful for α vertices $v \in V \backslash W$, the cardinality of W increases by α.

Step 3 is computationally more expensive than the procedure 0–1–2. Its calculations can at best be organized so as to keep its complexity within $O(\deg(v))$ for each vertex v whose coloring is attempted without success, and $O(|E|)$ for each vertex colored in Step 3. To attain this performance, it is recommendable to amend the adjacency list of each $v \in V$, say $L_1(v)$, with a second list of the form

$$L_2(v) = (j_1, \cdots, j_{|L_1|}; j_*; n(j_*)),$$

where for $i = 1, \cdots, |L_1|$, j_i is the index of the color class containing the ith element of $L_1(v)$; j_* is the index of the color class containing the fewest vertices adjacent to v; and $n(j_*)$ is the number of vertices adjacent to v in color class C_{j*}. Then identifying the color class C_{j*} containing the smallest number of vertices adjacent to v requires looking up the next to last entry of $L_2(v)$; and checking for every $w \in C_{j*}$ adjacent to v whether it can be transferred into some other color class where it has no neighbors amounts to checking whether the last entry of $L_2(w)$ is 0. However, upon the successful transfer of some $w \in C_{j*}$ to a new color class, the lists $L_2(u)$ have to be updated for every u adjacent to w, a task which for all $w \in C_{j*}$ adjacent to v requires a total effort of $O(|E|)$. Hence the total effort involved in the interchange heuristic is $O(|V| \cdot |E|)$.

Thus the worst case bound on the computational effort required by COLINT is higher than for COLOR, and it is not clear a priori whether the increase in the size of W justifies the increased effort. Therefore, pending some computational testing that we intend to carry out in the near future, we have not incorporated COLINT into the current version of our algorithm.

5. The algorithm as a whole.

Our algorithm starts by finding a MTIS $G(T)$ of G, and then extends $G(T)$ to a maximal k-chromatic induced subgraph $G(W)$ of G, where k is the size of a maximum clique in $G(T)$ (hence in $G(W)$). If $W = V$, we are done; otherwise we branch, based on the following considerations. For $v \in V$, we denote $N(v) = \{w \in V \backslash \{v\} \mid (v, w) \in E\}$.

THEOREM 4. *Let k be the cardinality of a maximum clique of $G(W)$, and let $(v_1, \cdots, v_m)$ be an arbitrary ordering of the set $V \backslash W$. If G has a clique K^* such that $|K^*| > k$, then K^* is contained in one of the m sets*

$$V_i := \{v_i\} \cup N(v_i) \backslash \{v_1, \cdots, v_{i-1}\}, \qquad i = 1, \cdots, m,$$

where for $i = 1$ we define $\{v_1, \cdots, v_{i-1}\} = \varnothing$.

Proof. Since $|K^*| > k$, $K^* \not\subseteq W$ and thus $v_i \in K^*$ for some $i \in \{1, \cdots, m\}$. Since K^* is a clique, this implies $K^* \subseteq \{v_i\} \cup N(v_i)$ for some $i \in \{1, \cdots, m\}$. Thus either $K^* \subseteq \{v_1\} \cup N(v_1)$, or else K^* is contained in one of the sets $\{v_i\} \cup N(v_i) \backslash \{v_1\}$, $i = 2, \cdots, m$. In the latter case, either $K^* \subseteq \{v_2\} \cup N(v_2) \backslash \{v_1\}$, or else K^* is contained in one of the sets $\{v_i\} \cup N(v_i) \backslash \{v_1, v_2\}$. The result follows by induction. □

Theorem 4 leads to the following *branching rule.* Node t of the search tree (where t is a string) is characterized by the pair $[I_t, E_t]$, where I_t and E_t are the sets of vertices forcibly included into, and excluded from, the graph on which the current subproblem is defined. In the language of implicit enumeration, the variables corresponding to I_t and E_t are fixed at 1 and 0, respectively, while the variables corresponding to $S_t = V \backslash (I_t \cup E_t)$ are free. Let $G(W_t)$ be a maximal k-chromatic induced subgraph of $G(S_t)$, and let $S_t \backslash W_t = \{v_1, \cdots, v_m\}$. We then generate m new nodes (subproblems) $t1, \cdots, tm$, by setting

$$\begin{aligned} &I_{ti} = I_t \cup \{v_i\}, \\ &E_{ti} = E_t \cup (S_t \backslash (\{v_i\} \cup N(v_i))) \cup \{v_1, \cdots, v_{i-1}\}, \qquad i = 1, \cdots, m. \end{aligned} \tag{5.1}$$

By construction, for any t the set $S_t = V \backslash (I_t \cup E_t)$ is contained in $\bigcap_{v \in I_t} N(v)$. Therefore I_t consists of pairwise adjacent vertices, and the cardinality of I_t (which is the same as the length of the string t) is equal to the level of node t in the search tree (where the level of the root of the tree is defined to be 0). The subproblem associated with node t of the search tree consists of finding a maximum clique in $G(S_t)$: if K_t is a clique in $G(S_t)$, then $K_t^* = K_t \cup I_t$ is a clique in G.

The problems $G(S_t)$ could be solved by applying to them recursively the two subroutines TRIANG and COLOR. However, computational testing has convinced us that this is not the best way to proceed: finding a MTIS in each subgraph $G(S_t)$ is relatively expensive, and the occasions when $G(S_t)$ contains a vertex set K_t such that $K_t \cup I_t$ is a clique larger than the current largest one, are not too frequent. It is considerably cheaper to simply use the algorithm COLOR to find a maximal $(k - |I_t|)$-chromatic induced subgraph of $G(S_t)$ (where k is the size of the largest clique in G found so far), except in cases where there is serious indication that $G(S_t)$ may contain a clique larger than $k - |I_t|$. Such is the case, for instance, whenever $|I_t| = k$ and $S_t \neq \emptyset$. The motivation for choosing $k - |I_t|$ as the chromatic member of the maximal subgraph we construct, is that if $G(S_t)$ itself turns out to be such a subgraph, then P_t can be eliminated.

We are now in a position to state our algorithm formally. We denote by L the list of live (active) nodes of the search tree, and by P_t the subproblem at node t.

Step 0 (*Initialization*). Put into L the problem P_0 of finding a maximum clique in $G = (V, E)$. Set $t = 0$, $I_t = E_t = \emptyset$, $S_t = V$, $k = 0$, and go to 1.

Step 1 (*Subproblem selection*). If $L = \emptyset$, stop: the current clique is maximum in G. Otherwise choose an element P_t of L and remove it from L.

If $|V \backslash E_t| \leqq k$, eliminate P_t and go to 1.

If $|V \backslash E_t| > k$ but $|I_t| = k$, go to 2.

Otherwise go to 4.

Step 2 (*TRIANG*). Find a MTIS $G(T_t)$ of $G(S_t)$, and a maximum clique K_t in $G(T_t)$. Store $I_t \cup K_t$ as the current largest clique, and set $k \leftarrow |I_t \cup K_t|$. If $T_t = S_t$, eliminate P_t and go to 1. Otherwise go to 3.

Step 3 (*COLOR*). Find a minimum coloring of $G(T_t)$, and expand its color classes to find a maximal $|K_t|$-chromatic induced subgraph $G(W_t)$ of $G(S_t)$. If $W_t = S_t$, eliminate P_t and go to 1. Otherwise go to 5.

Step 4 (*COLOR* 2). Find a maximal $(k-|I_t|)$-chromatic subgraph $G(W_t)$ of $G(S_t)$. If $W_t = S_t$, eliminate P_t and go to 1. Otherwise go to 5.

Step 5 (*Branching*). Order the set $S_t \setminus W_t$ into a sequence $(v_1, \cdots, v_m)$. Generate m new subproblems defined by (5.1), place them into L, and go to 1.

The strategy of this algorithm differs from that of most branch and bound procedures in that no attempt is made to derive an upper bound on the objective function value of the subproblems P_t generated during the procedure. Instead, the focus is on finding maximal subgraphs (i.e., subproblems) for which the current lower bound is also an upper bound, and which can thus be eliminated. Since we have no known upper bound on the value of each P_t, the search strategy used is depth first.

Our approach is based on the idea of finding a maximal induced subgraph G' of G, of a type for which it is easy to find a maximum clique in G'; and we chose G' to be triangulated. But triangulated subgraphs are not the only ones in which it is easy to locate a maximum clique. Another such class is that of *cotriangulated* graphs, i.e., complements of triangulated graphs. Finding a maximum clique in a cotriangulated graph G'' amounts to finding a maximum vertex packing in the corresponding triangulated graph $\bar{G}''$ (the complement of G''), and this can also be done in $O(|V|+|E|)$ time (see, for instance, Golumbic [1980]). In particular, when the graph G is dense, there are good chances that a maximal cotriangulated induced subgraph (MCIS) is larger, and contains a larger clique, than a MTIS of the same graph. Thus Step 2 of our algorithm can be replaced by the following more expensive, but also more efficient version.

Step 2′ (*TRIANG*). Find a MTIS $G(T'_t)$ of $G(S_t)$ and a maximum clique K'_t in $G(T'_t)$. Find a MCIS $G(T''_t)$ of $G(S_t)$ and a maximum clique with vertex set K''_t in $G(T''_t)$. Choose $|K_t| = \max\{|K'_t|, |K''_t|\}$, and set $T = T'$ if the maximum is attained for $|K'_t|$, $T = T''$ otherwise. Break ties by choosing $|T| = \max\{|T'|, |T''|\}$. Store $I_t \cup K_t$ as the current clique and set $k \leftarrow |I_t \cup K_t|$. If $T_t = S_t$, eliminate P_t and go to 1. Otherwise go to 3.

Another version, which avoids the effort involved in finding both a MTIS and a MCIS of G, makes a decision as to which one to use, based on the density of the given subgraph: if the number of edges of $G(S_t)$ is $\leqq 1/4|S_t|(|S_t|-1)$, it finds a MTIS; otherwise, it finds a MCIS in $G(S_t)$. Then it finds a maximum clique K_t in the resulting graph $G(T_t)$.

These amended versions of the algorithm have not yet been implemented: our computational experience is limited to the basic version stated above.

Next we illustrate this basic version of the algorithm on a numerical example.

Example 2. Consider the graph G shown in Fig. 2(a). We initialize and go to Step 1. We choose and remove from L the problem P_0 defined on G. Since $|V \setminus E_t| = |V| > k$ and $|I_t| = k = 0$, we go to Step 2, i.e., apply TRIANG. Suppose we generate the ordering $\sigma = (2, 12, 8, 5, 9, 1, 4, 11, 10, 7, 3, 6)$ and the maximal triangulated subgraph $G(T_0)$, with $T_0 = \{1, 3, 5, 6, 7, 8, 10, 11\}$ shown in Fig. 2(b). A maximum clique in $G(T_0)$ has vertex set $K_0 = \{5, 7, 8\}$. We store K_0, set $k = 3$ and go to Step 3.

COLOR finds the minimum coloring $(\{6, 1, 5\}, \{3, 7, 10, 11\}, \{8\})$ in $G(T_0)$, and extends it by including vertex 12 into the first, and vertex 2 into the second color class. Thus our maximal 3-chromatic induced subgraph is $G(W_0)$, with all but 2 vertices (4 and 9) contained in W_0.

The Branching Step leads to the creation of two subproblems, P_1 and P_2, defined by

$$I_1 = \{4\}, \qquad E_1 = \{2, 6, 10, 11, 12\}$$

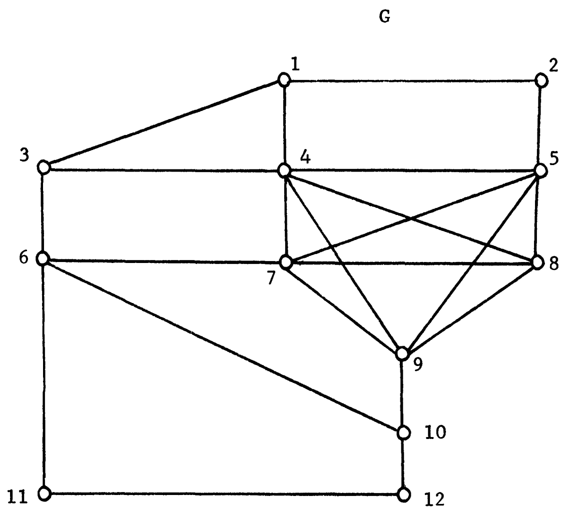


FIG. 2(a)

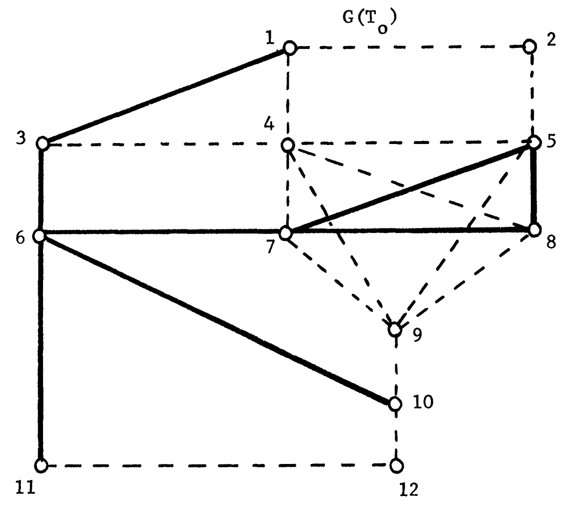


FIG. 2(b)

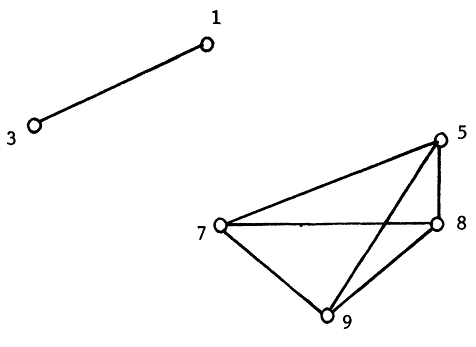


FIG. 2(c)

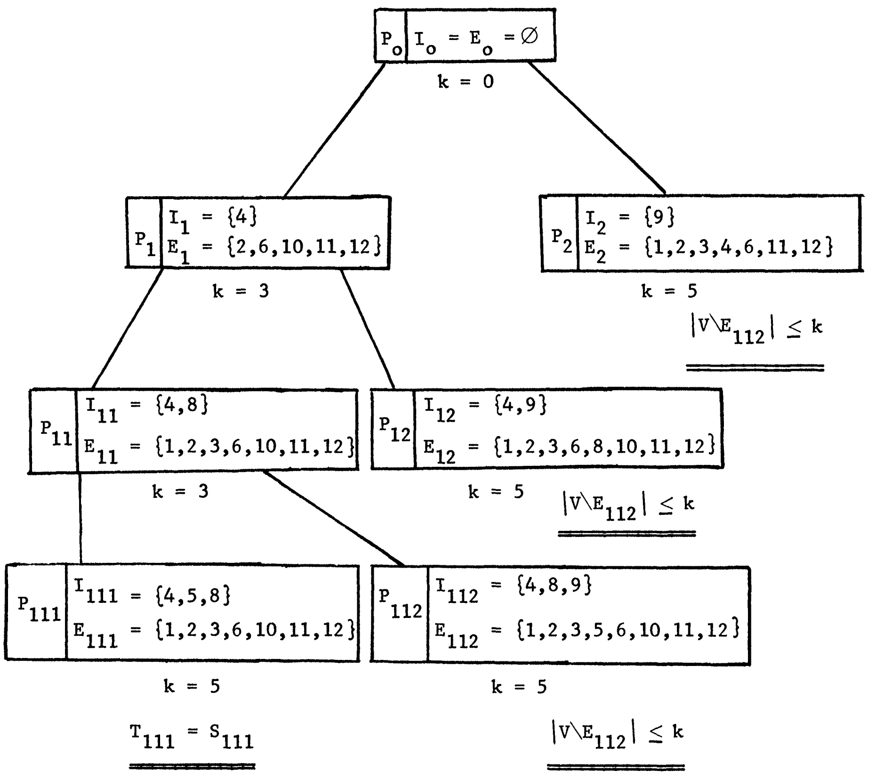

P0 I0 = E0 = ∅
k = 0
P1 I1 = {4} E1 = {2,6,10,11,12}
k = 3
P2 I2 = {9} E2 = {1,2,3,4,6,11,12}
k = 5
|V\E112| ≤ k
P11 I11 = {4,8} E11 = {1,2,3,6,10,11,12}
k = 3
P12 I12 = {4,9} E12 = {1,2,3,6,8,10,11,12}
k = 5
|V\E112| ≤ k
P111 I111 = {4,5,8} E111 = {1,2,3,6,10,11,12}
k = 5
T111 = S111
P112 I112 = {4,8,9} E112 = {1,2,3,5,6,10,11,12}
k = 5
|V\E112| ≤ k

FIG. 2(d)

and

$$I_2 = \{9\}, \qquad E_2 = \{1, 2, 3, 4, 6, 11, 12\},$$

respectively.

We choose P_1 first. Since $|V\backslash E_1| = 7 > k = 3$, and $|I_1| = 1 < k$, we go to Step 4. Since $k - |I_1| = 2$, we use COLOR 2 to find a maximal 2-chromatic induced subgraph of $G(S_1) = G(\{1, 3, 5, 7, 8, 9\})$. This is $G(W_1)$, shown in Fig. 2(c), with the coloring ($\{1, 7\}$, $\{3, 5\}$) and $W_1 = \{1, 3, 5, 7\}$. Since $S_1 \backslash W_1 = \{8, 9\} \neq \emptyset$, we go to Step 5.

Branching creates two successors of P_1, the subproblems P_{11} and P_{12}, defined by

$$I_{11} = I_1 \cup \{8\} = \{4, 8\}, \qquad E_{11} = E_1 \cup \{1, 3\} = \{1, 2, 3, 6, 10, 11, 12\}$$

and

$$I_{12} = I_1 \cup \{9\} = \{4, 9\}, \qquad E_{12} = E_1 \cup \{1, 3, 8\} = \{1, 2, 3, 6, 8, 10, 11, 12\}$$

respectively.

We choose problem P_{11} and since $|V\backslash E_{11}| = 5 > k = 3$ and $|I_{11}| = 2 < k$, we go to Step 4. Since $k - |I_{11}| = 1$, we construct a maximal 1-chromatic induced subgraph $G(W_{11})$ of $G(S_{11}) = G(\{5, 7, 9\})$. We have $W_{11} = \{7\}$ and since $S_{11} \backslash W_{11} = \{5, 9\} \neq \emptyset$, we go to the Branching Step. P_{11} has two successors, P_{111}, and P_{112}, defined by

$$I_{111} = I_{11} \cup \{5\} = \{4, 5, 8\}, \qquad E_{111} = E_{11},$$

and

$$I_{112} = I_{11} \cup \{9\} = \{4, 8, 9\}, \qquad E_{112} = E_{11} \cup \{5\} = \{1, 2, 3, 5, 6, 10, 11, 12\},$$

respectively.

We choose P_{111} and since $|I_{111}| = k = 3$, we go to Step 2. Applying TRIANG to $G(S_{111}) = G(\{7, 9\})$ yields the clique $\{7, 9\}$, which together with I_{111} forms the 5-clique with node set $\{4, 5, 7, 8, 9\}$ in G. Thus we store this as the current clique (in place of $\{5, 7, 8\}$), set $k = 5$ and since $T_{111} = S_{111}$, eliminate P_{111}.

Next we choose P_{112}. Since $|V \backslash E_{112}| = 4 \leqq k = 5$, we eliminate P_{112}. Then choosing P_{12}, we find that $|V \backslash E_{12}| = 4 \leqq k$ and eliminate P_{12}. The only subproblem remaining in L is now P_2. Since $|V \backslash E_2| = 5 = k$, we eliminate P_2 and terminate with the maximum clique induced by $\{4, 5, 7, 8, 9\}$ found at P_{111}.

The search tree is shown in Fig. 2(d).

6. Computational experience. Several variants of the basic version of our algorithm were implemented and tested. Four of the variants (TC1-TC4) correspond to the tie breaking rules used in TRIANG, as defined at the end of § 3. The variant TC* does not have Step 4; instead, Step 1 is always followed by 2 and 3. In other words, this variant finds a MTIS in every subgraph generated by the procedure. Finally, the variant C does not use TRIANG at all, and from Step 1 either goes to Step 4 or again to Step 1 (thus both Steps 2 and 3 are eliminated).

For purposes of comparison, we have also implemented Algorithm CACM457 of Bron and Kerbosch [1973], described by Gerhards and Lindenberg [1979] as the currently known most efficient clique finding procedure. Since CACM457 was originally designed to list all the cliques of a graph, we modified it to just find a largest clique,

TABLE 5.1
Computational results with PASCAL *codes on a* DEC 20-60, *on random graphs with* 50 *vertices.*

		CPU time (ms)						
D	Q	CACM457*	TC1	TC2	TC3	TC4	TC*	C
5	3	100	49	45	92	53	91	32
10	4	118	54	51	83	52	130	43
15	4	147	67	64	113	60	184	51
20	4	200	85	69	129	76	282	66
25	6	218	89	87	108	87	365	75
30	6	263	84	112	121	78	354	96
35	6	372	69	116	117	128	761	100
40	6	615	130	168	202	149	2731	119
45	7	630	146	147	173	162	+	153
50	7	969	249	262	234	210	+	207
55	8	1346	291	223	274	269	+	287
60	9	1838	293	406	346	309	+	336
65	9	3333	675	514	479	510	+	478
70	11	4809	543	634	750	463	+	568
80	13	19548	3212	2170	1732	1476	+	2023
85	18	18458	638	1350	753	1030	+	1430
90	23	25763	832	966	462	571	+	1674
95	27	45542	712	358	490	277	+	4086

D = density of the graph (%). Q = size of maximum clique. + runs not attempted in view of the obvious inferiority of this version.

by adding tests to eliminate those nodes of the search tree that cannot contain a clique larger than the current largest one. We call this modified algorithm CACM457*. We also implemented a more recent algorithm, due to Loukakis and Tsouros [1982], originally designed for finding a maximum vertex packing. Our implementation, which we call LT, finds a maximum clique in G by essentially applying the Loukakis-Tsouros algorithm to $\bar{G}$. Both CACM457* and LT are rather straightforward implicit enumeration procedures, very similar to each other. CACM457* has one more logical test (exclusion rule), which makes it on the average more efficient than the LT algorithm. (Although the Loukakis-Tsouros paper is more recent, its authors do not seem to have been aware of the existence of the Bron-Kerbosch algorithm of 1973.)

TABLE 5.2
Computational results with C *codes on a* VAX 11-780, *on random graphs with* 50-400 *vertices.*

Graph characteristics			No. of search tree nodes			CPU time (ms)		
Vertices	Density (%)	Size of maximum clique	CACM	L-T	B-Y	CACM	L-T	B-Y
50	10	3	28	21	7	116	133	50
50	20	4	62	59	25	183	200	67
50	30	5	87	84	40	233	283	67
50	40	6	175	208	82	467	617	116
50	50	8	244	265	139	700	983	200
50	60	9	547	968	284	1,283	3,450	383
50	70	12	1,310	4,076	551	3,100	20,199	883
50	80	15	4,334	16,143	825	10,600	104,096	1,917
50	90	22	9,907	85,869	421	27,499	778,335	1,983
100	10	4	73	62	27	500	567	200
100	20	5	225	234	92	999	1,067	267
100	30	6	623	670	327	2,233	2,499	599
100	40	7	1,459	1,600	643	4,966	5,816	1,083
100	50	9	4,186	5,290	1,938	13,416	22,716	3,217
100	60	12	14,304	20,600	7,798	50,648	126,645	15,883
100	70	15	93,392	335,011	53,074	308,254	1,073,020	112,695
200	10	4	291	277	160	2,799	2,750	983
200	20	5	1,368	1,341	700	7,233	6,283	1,899
200	30	7	5,247	5,400	2,464	26,732	26,832	5,599
200	40	9	19,118	22,445	9,490	94,730	128,045	19,782
200	50	11	—	—	61,374	—	—	123,628
200	60	14	—	—	526,852	—	—	1,164,170
300	10	5	538	—	354	8,482	—	2,416
300	20	6	4,420	—	1,775	30,382	—	5,783
300	30	8	17,409	—	11,587	127,611	—	25,965
300	40	10	114,416	—	55,417	654,790	—	118,495
300	50	13	—	—	526,078	—	—	1,156,020
400	10	5	1,266	—	619	17,056	—	4,066
400	20	7	9,391	—	3,575	78,463	—	13,383
400	30	8	63,753	—	32,092	435,916	—	80,997
400	40	10	—	—	238,790	—	—	562,627

CACM = CACM457*; L-T = Loukakis-Tsouros; B-Y = Balas-Yu.

The computational results listed below are of two kinds. The six variants of our algorithm, as well as the algorithm CACM457*, were first coded in PASCAL and run on a DEC 20-60 computer at CMU, on 18 problems defined on randomly generated graphs with 50 vertices, of density (i.e., probability of presence of a given edge) ranging from 5% to 95%. The results are shown in Table 5.1.

A comparison of the 6 variants of our algorithm shows TC*, i.e., the variant that finds a MTIS at every iteration, clearly inferior to the other five. Of the remaining five variants, TC4, i.e., the one that uses the degree in G as a tie breaking rule for choosing the next vertex to be numbered in TRIANG, seems slightly superior to the others on the harder problems, i.e., on graphs with largest clique size of at least 7, whereas on the easier problems TC1, the variant that breaks ties by random choice, performs best. Surprisingly, the variant C that uses only the coloring routine without looking for a MTIS, does equally well or even better than the others on problems with largest clique size $\leqq 7$, although its relative performance rapidly deteriorates for largest clique sizes greater than 11.

In the above discussion we used the largest clique size rather than the density of the graph as a criterion for distinguishing between easy and hard problems. Both criteria are of course relevant, but the size of the largest clique seems to be a better measure of problem difficulty.

Variant TC4 of our algorithm, along with CACM457* and the Loukakis-Tsouros algorithm, were subsequently coded in C and run on a VAX 11-780 at Bell Labs, on a set of larger random graphs, with up to 400 vertices and 30,000 edges. The results are shown in Table 5.2.

REFERENCES

[1] E. BALAS AND H. SAMUELSSON, *A node covering algorithm*, Naval Res. Log. Quart., 24(2) (1977), pp. 213-233.

[2] C. BERGE, *Farbung von Graphen, deren sämtliche bzw. deren ungerade Kreise Starr sind*, Wiss. Z. Martin-Luther-Universität, Halle-Wittenberg, Math-Natur. Reihe (1961), pp. 114-115.

[3] C. BRON AND J. KERBOSCH, *Finding all cliques of an undirected graph*, Comm. ACM, 16(9) (1973), pp. 575-577.

[4] V. CHVÁTAL, *On certain polytopes associated with graphs*, J. Combin. Theory, B, 18 (1975), pp. 138-154.

[5] G. A. DIRAC, *On rigid circuit graphs*, Abh. Math. Sem. Univ., Hamburg, 25 (1961), pp. 71-76.

[6] D. R. FULKERSON AND O. A. GROSS, *Incidence matrices and interval graphs*, Pacific J. Math., 15 (1965), pp. 835-855.

[7] F. GAVRIL, *Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of chordal graph*, this Journal, 1 (1972), pp. 180-187.

[8] L. GERHARDS AND W. LINDENBERG, *Clique detection for nondirected graphs: Two new algorithms*, Computing, 21 (1979), pp. 295-322.

[9] M. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[10] M. GRÖTSCHEL, L. LOVÁSZ AND A. SCHRIJVER, *Polynomial Algorithms for Perfect Graphs*, C. Berge and V. Chvátal, eds., Topics on Perfect Graphs, North-Holland, Amsterdam, 1984, pp. 325-356.

[11] E. LOUKAKIS AND C. TSOUROS, *Determining the number of internal stability of a graph*, Intern. J. Computer Math., 11 (1982), pp. 207-220.

[12] L. LOVÁSZ, *Normal hypergraphs and the perfect graph conjecture*, Discrete Math., 2 (1972), pp. 253-267.

[13] G. L. NEMHAUSER AND L. E. TROTTER, JR., *Vertex packings: structural properties and algorithms*, Math. Programming, 8 (1975), pp. 232-248.

[14] D. J. ROSE, R. E. TARJAN AND G. S. LUEKER, *Algorithmic aspects of vertex elimination on graphs*, this Journal, 5 (1976), pp. 266-283.

[15] R. E. TARJAN AND A. E. TROJANOWSKI, *Finding a maximum independent set*, this Journal, 6 (1977), pp. 537-546.

[16] M. YANNAKAKIS, *Node and edge deletion of NP-complete problems*, Proc. 10th Annual ACM Symposium on Theory of Computing, ACM, New York, 1978, pp. 253-264.